

.REM @

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

IDENTIFICATION

PRODUCT CODE: AC-S069B-MC
PRODUCT NAME: CZRMUB0 RM05/3/2 PERFORMANCE EXERCISER
PRODUCT DATE: APRIL 1981
MAINTAINER: CX DIAGNOSTIC GROUP
ALTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION

CONTENTS

- 1
- 2
- 3
- 4 1. ABSTRACT
- 5
- 6 1.1 GENERAL DOCUMENT NOTES
- 7
- 8
- 9 2. REQUIREMENTS
- 10 2.1 EQUIPMENT
- 11 2.2 MEDIA
- 12 2.3 PRELIMINARY PROGRAMS
- 13
- 14 3. OPERATING PROCEDURE
- 15 3.1 LOADING THE PROGRAM
- 16 3.2 STARTING ADDRESSES
- 17 3.3 PROGRAM CONTROL
- 18 3.4 SWITCH OPTIONS
- 19 3.5 PASS/TEST TERMINATION
- 20 3.5.1 PASS TERMINATION
- 21 3.5.2 TEST TERMINATION
- 22
- 23 3.6 RUN TIME
- 24 3.6.1 DATA TRANSFER MODE
- 25 3.6.2 SEEK VERIFICATION MODE
- 26 3.7 DUAL PORT OPERATION
- 27 3.8 XXDP, ACT11, APT11
- 28 3.9 APT ENVIRONMENTAL TABLE DEFINITIONS
- 29
- 30 4. CONTROLLING THE PROGRAM
- 31 4.1 PARAMETERS
- 32 4.1.1 PROGRAM CONTROL PARAMETERS
- 33 4.1.2 CHANGE DEVICE ADDRESS
- 34 4.2 KEYBOARD COMMANDS
- 35 4.2.1 'T' COMMAND
- 36 4.2.2 'D' COMMAND
- 37 4.2.3 'S' COMMAND
- 38 4.2.4 'W' COMMAND
- 39 4.2.5 'R' COMMAND
- 40 4.2.6 'WT' COMMAND
- 41
- 42 5. PERFORMANCE SUMMARY TYPEOUT
- 43 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
- 44 5.2 HARD/SOFT ERROR DEFINITIONS
- 45 5.2.1 HARD ERRORS
- 46 5.2.2 SOFT ERRORS
- 47
- 48 6. DATA CHECKING & ERROR RECOVERY
- 49 6.1 DATA BUFFER COMPARISON
- 50 6.2 VERIFICATION OF DATA WRITTEN
- 51 6.3 BAD ADDRESS FLAGGING
- 52
- 53 7. ERROR MESSAGES
- 54
- 55
- 56
- 57

58
59
60
61
62
63
64
65
66
67
68

7.1 ERROR DESCRIPTION LINES
7.2 DETAIL ERROR LINES

8. PROGRAM DESCRIPTION

8.1 HOW THE PROGRAM OPERATES
8.2 DUAL PORT OPERATION
8.3 SELECTION OF OPERATION VARIABLES
8.4 DATA PATTERNS

9. RM SOFTWARE DRIVER DOCUMENT

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. ABSTRACT

THE RM05/3/2 PERFORMANCE EXERCISER PROGRAM IS DESIGNED TO PERFORM AN INTERACTIVE TEST ON RM DISK DRIVES CONNECTED TO A MASSBUS SUBSYSTEM. THE DRIVES MAY BE CONTROLLED BY AN RH70 CONTROLLER. IN ADDITION TO PERFORMING AN INTERACTIVE TEST OF THE DISK DRIVES ON THE SUBSYSTEM, THE PROGRAM IS INTENDED TO BE USED TO VERIFY THAT THE DRIVES UNDER TEST ARE PERFORMING TO THEIR DATA ERROR RATE AND SEEK ERROR RATE (SEE ERROR RATE SPECIFICATIONS).

THE PERFORMANCE EXERCISER PROGRAM WILL EXERCISE DRIVES CONNECTED AS EITHER SINGLE OR DUAL PORT UNITS. DUAL PORT DRIVES ARE TESTED BY LOADING AND RUNNING THE PROGRAM FROM BOTH CONTROLLING SYSTEMS. THE PROGRAM WILL EXERCISE A MIXED SYSTEM OF DUAL PORT AND SINGLE PORT DRIVES.

TO OBTAIN INTERACTIVE TESTING, OPERATIONS ON THE MULTI-DRIVE CONFIGURATIONS ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER). OPERATIONS AMONG THE DRIVES ARE OPTIMIZED SO THAT A HIGH SUBSYSTEM DATA TRANSFER RATE OR A HIGH POSITIONING OPERATION RATE IS MAINTAINED.

THE PERFORMANCE OF EACH DRIVE IS MONITORED BY THE PROGRAM. IF A DRIVE EXCEEDS A PRESET NUMBER OF ERRORS IN ANY OF SEVERAL CATEGORIES, THAT DRIVE IS AUTOMATICALLY DEASSIGNED. (THE OPERATOR MAY OVERRIDE THE AUTOMATIC DEASSIGNMENT FEATURE.) THE PROGRAM REPORTS PERFORMANCE STATISTICS FOR EACH DRIVE BEING EXERCISED ON REQUEST FROM THE OPERATOR OR AUTOMATICALLY AT AN INTERVAL DETERMINED BY THE OPERATOR.

ALL DATA TRANSFER COMMANDS EXCEPT WRITE HEADFR & DATA AND WRITE CHECK HEADER & DATA ARE USED. RECALIBRATE AND READ-IN PRESET COMMANDS ARE USED AT STARTUP AND DRIVE INITIALIZATION. RECALIBRATE, OFFSET, AND RETURN-TO-CENTERLINE COMMANDS ARE USED DURING ERROR RECOVERY.

THE DATA TRANSFER COMMANDS ARE SELECTED RANDOMLY EXCEPT FOR THE WRITE CHECK COMMANDS. THE WRITE CHECK COMMANDS ARE USED TO VERIFY A PREVIOUS WRITE OPERATION. THUS, WHEN A WRITE COMMAND IS SELECTED, THE DATA WRITTEN IS VERIFIED BY THE APPROPRIATE WRITE CHECK COMMAND.

DEPENDING UPON WHETHER THE PROGRAM HAS BEEN LOADED VIA APT AUTOMATIC MODE OR APT DUMP MODE WILL DETERMINE WHETHER; PROGRAM/OPERATOR COMMUNICATIONS ARE THROUGH THE KEYBOARD. DYNAMIC PROGRAM OPTIONS ARE SELECTED VIA SWITCH REGISTER SETTINGS AND ERRORS ARE REPORTED ON THE CONSOLE TERMINAL.

1.1 GENERAL DOCUMENT NOTES

A. IN REFERENCE TO ALL NUMBERS IN THIS DOCUMENTION. TO INDICATE THE BASE OF A NUMBER LARGER THAN SEVEN. A PERIOD(.) WILL FOLLOW THE NUMBER TO INDICATE DECIMAL OR NO PERIOD WILL FOLLOW THE NUMBER TO INDICATE OCTAL. IF THE NUMBER OCCURS AT THE END OF A SENTENCE, A DOUBLE PERIOD(. .) INDICATES DECIMAL AND A SINGLE PERIOD(.) INDICATES OCTAL. ALSO, ANY REFERENCES TO TIME ARE ALWAYS IN DECIMAL.

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. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
20K 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)

2.2 MEDIA

THE PERFORMANCE EXERCISER PROGRAM REQUIRES FORMATTED DISK
PACKS GENERATED BY THE RM05/3/2 FORMATTER PROGRAM (CZRML).
THE PACKS MUST BE FORMATTED IN 32. SECTOR (16 BIT) MODE; THE
ALTERNATE (30. SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

RM05/3/2 DISKLESS TESTS, PART 1 & 2
RM05/3/2 FUNCTIONAL TESTS, PART 1, 2 & 3

3. OPERATING PROCEDURE

3.1 LOADING THE PROGRAM

THE PROGRAM MAY BE LOADED BY EITHER OF THE FOLLOWING MEDIA:
.PAPER TAPE, USING THE STANDARD PAPER TAPE PROCEDURE
.XXDP MEDIA, USING ANY XXDP DEVICE

3.2 STARTING ADDRESSES

200 - START ADDRESS, ALL SWITCHES CLEAR (SEE SECTION 3.4)

WHEN THE PROGRAM IS STARTED, A DATA PATTERN WILL BE WRITTEN TO
ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF
THE WRITE, THE PROGRAM GOES INTO A TESTING MODE.

204 - RESTART ADDRESS, THE RESTART ADDRESS PROVIDES THE OPERATOR WITH
THE ABILITY TO CHANGE THE DEFAULT RM/RH ADDRESSES (SEE SECTION
4.1.2), ANY PROGRAM PARAMETERS (SEE SECTION 4.1) OR CHANGE
DRIVE LIMIT PARAMETERS (SEE SECTION 4.2).

3.3 PROGRAM CONTROL

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

PROVIDED THE PROGRAM HAS BEEN LOADED AND STARTED VIA THE APT DUMP MODE OR THE DIAGNOSTIC IS RUNNING IN STAND ALONE PROCESSOR/DRIVE OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

HOWEVER, IF THE PROGRAM IS LOADED VIA APT SCRIPT MODE ALL SETUP AND SWITCH REGISTER SETTINGS WILL BE PROVIDED THROUGH THE APT E TABLE. TYPEOUTS FROM THE USER DIAGNOSTIC MAY OR MAYNOT BE INHIBITED DEPENDING UPON WHETHER OR NOT THE APPROPRIATE BIT IN THE E TABLE HAS BEEN SET.

3.4 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS IN KEYBOARD ENTRY MODE, OR IS AT A HIGHER PRIORITY PROCESSING AN DRIVE INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTINGS ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED, IF THE PROGRAM FINDS ALL 1'S IN THE SWITCHES. ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1	HALT ON ERROR
SW<14>	NOT USED
SW<13>=1	INHIBIT ERROR TYPEOUT
SW<12>	NOT USED
SW<11>	NOT USED
SW<10>=1	BELL ON ERROR
SW<09>	NOT USED
SW<08>=1	INHIBIT END OF PASS MESSAGES
SW<07>=1	DISPLAY ALL DATA COMPARE ERRORS
SW<06>=1	DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW<05>=1	PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW<04>=1	INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN END OF TEST IS REACHED
SW<03>=1	DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH RETRY IF UNCORRECTABLE 'DCK' ERROR
	IF DATA COMPARE ERRORS & SW<07> SET, DISPLAY REST OF BUFFER
SW<02> 1	INHIBIT SUBSYSTEM STATUS TYPEOUT DURING STARTUP
	INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME

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

SW<01>=1 INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
SW<00>=1 'READ ONLY' MODE

SW<00>

WHEN THE SWITCH IS SET(1), THE PROGRAM WILL OPERATE IN 'READ ONLY' MODE. IF THE SWITCH IS CLEARED(0), THE PROGRAM WILL RETURN TO READ/WRITE MODE DURING TESTING. THIS SWITCH ONLY EFFECTS THE TESTING PORTION OF THE PROGRAM.

FOR EXAMPLE, IF THE PROGRAM IS STARTED AT ADDRESS 200. A DATA PATTERN WILL BE WRITTEN TO ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF THE WRITE, THE PROGRAM GOES INTO A TESTING MODE. HOWEVER, IF THE OPERATOR SWITCHES TO 'READ ONLY' MODE (SW0=1) JUST PRIOR TO OR DURING THE SEQUENTIAL WRITING OF THE DISK, THE PROGRAM WILL CONTINUE WRITING UNTIL THE SEQUENTIAL WRITE IS COMPLETED. UPON COMPLETION OF THE SEQUENTIAL WRITE, THE PROGRAM WILL SWITCH TO A 'READ ONLY' TESTING MODE UNTIL SW0 IS RESET TO ZERO BY THE OPERATOR.

3.5 PASS/TEST TERMINATION

A PASS IN RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE IS DETERMINED BY EITHER BITS READ OR SEEKS PERFORMED. THE NUMBER OF BITS OR SEEKS REQUIRED FOR A PASS IS DERIVED FROM EITHER THE SOFT ERROR RATE SPECIFICATION OR THE SEEK ERROR RATE SPECIFICATION.

THE SOFT ERROR SPECIFICATION FOR THE RM DRIVE IS NO MORE THAN 1 SOFT ERROR (NON-DISK RELATED) IN 1×10^{10} BITS READ. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

THE SEEK ERROR SPECIFICATION FOR THE RM DRIVE IS NO MORE THAN 1 SEEK ERROR IN 1×10^6 SEEKS. (SEE SECTION 3.5.1 FOR THE 90% CONFIDENCE LEVEL)

A PASS IN 'W' OR 'R' COMMAND MODE IS DETERMINED BY THE MAXIMUM DISK ADDRESS LIMITS SETUP BY THE OPERATOR.

3.5.1 PASS TERMINATION

END OF PASS FOR A SINGLE DRIVE IN THE RANDOM 'T' COMMAND MODE OR SEQUENTIAL 'T' COMMAND MODE, IS DETERMINED BY ONE OF THE FOLLOWING CONDITIONS.

- A. IF PARAMETER 'ENDING' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 3×10^9 BITS (1.875×10^8 WORDS). IT WILL TAKE APPROXIMATELY 3.33 PASSES TO REACH THE SOFT ERROR RATE OF 1×10^{10} BITS (6.25×10^8 WORDS) READ. HOWEVER, IT WILL TAKE 10. PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3×10^{10} BITS (1.875×10^9 WORDS) READ.
- B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 1×10^6 SEEKS. IT WILL TAKE 1 PASS TO REACH THE SEEK ERROR RATE OF 1×10^6 SEEKS. HOWEVER, IT WILL TAKE 3 PASSES TO REACH THE 90% CONFIDENCE LEVEL OF 3×10^6 SEEKS.

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

END OF PASS FOR A SINGLE DRIVE IN 'W' OR 'R' COMMAND MODE, IS DETERMINED AS FOLLOWS.

A. WHEN A SEQUENTIAL SEEK IS MADE BEYOND THE MAXIMUM DISK ADDRESS LIMITS SET BY THE OPERATOR, THE PASS IS CONSIDERED ENDED.

3.5.2 TEST TERMINATION

IF SW04 IS CLEAR, THE TEST FOR A DRIVE IS TERMINATED WHEN:

- A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASSES'.
- B. THE TOTAL ERRORS ACCUMULATED EXCEED 25. .
- C. A FATAL ERROR OCCURS: EM12 OR EM14.
- D. OPERATOR DEASSIGNS THE DRIVE
- E. THE NUMBER OF PASSES SPECIFIED BY THE MONITOR HAVE BEEN REACHED, WHEN RUNNING IN 'XXDP' CHAIN MODE, 'ACT11' CHAIN MODE OR 'APT' SCRIPT MODE (ANY AUTO MODE).

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. (SEE SECTION 3.5.1) THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION MODE SELECTED, THE READ/WRITE RATIO PARAMETER ('RATIO'), AND BY SWR SWITCHES 0, 1, AND 2.

3.6.1 DATA TRANSFER MODE (DEFAULT)

1 DRIVE - APPROX. 1 HR. 45 MIN. (TO REACH 3×10^9 BITS OR 1.875×10^8 WORDS)
WITH SW<00> =1 AND SW<01> =1, THE PROGRAM WILL RUN APPROX. 20% FASTER

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'WRDCNT' = 256. (1 SECTOR)
PARAMETER 'MAXTRK' = 'MINTRK' (SAME VALUES)
PARAMETER 'MAXSEC' = 'MINSEC' (SAME VALUES)
SW<01> =1 (NO DATA COMPARE)
SW<00> =1 (READ ONLY MODE)

1 DRIVE - APPROX. 4.0 HRS (TO REACH 1×10^6 SEEKS)

3.7 DUAL PORT OPERATION

- A. LOAD THE PERFORMANCE EXERCISER PROGRAM INTO BOTH PROCESSORS.
- B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH DRIVE WHICH IS TO BE TESTED AS A DUAL PORT DRIVE AND CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THOUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

3.8 XXDP, ACT11, APT11 COMPATIBILITY

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

THIS PROGRAM IS COMPATIBLE WITH ACT11 AND APT11 IN BOTH DUMP AND AUTOMATIC MODES.

THIS PROGRAM IS ALSO, 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.

AUTOMATIC MODE OR CHAIN MODE (MONITOR)

1. THE BUS ADDRESS AND CONTROLLER INTERRUPT VECTOR ARE DEFAULTED TO 176700 AND 254 RESPECTIVELY.

DUMP MODE (NO MONITOR)

1. INPUT DIALOGUE PROMPTED AFTER PROGRAM STARTS

3.9 APT ETABLE DEFINITIONS

THE FOLLOWING DEFINITIONS ARE VALID FOR SPECIFYING APT ENVIRONMENTAL TABLE (ETABLE) ENTRIES, VIA RUNNING THE APT UTILITY PROGRAM 'TSP':

1. SOFTWARE ENVIRONMENT:
 - = 1 IF APT SCRIPT MODE
 - = 0 IF STANDLONE MODE
2. ENVIRONMENT MODE:
 - BIT 7 = 1 ETABLE DOES SIZING
 - = 0 PROGRAM DOES SIZING
 - BIT 6 = 1 SPOOL MESSAGES TO APT IF SCRIPT MODE
 - = 0 DON'T SPOOL TO APT
 - BIT 5 = 1 SUPPRESS TTY CONSOLE OUTPUT
 - = 0 ALLOW TTY CONSOLE OUTPUT
 - BIT 4 TO BIT 0 ARE NOT USED
3. SWITCH 1 (SOFTWARE SWITCH REGISTER)
IF ENVIRONMENT MODE BIT 7 (SIZING BIT) IS SET TO 1, THE SOFTWARE SWITCH REGISTER WILL BE USED, INSTEAD OF THE HARDWARE TTY CONSOLE SWITCH REGISTER.
4. SWITCH 2 (USER SWITCH REGISTER)
NOT USED
5. CPU OPTIONS
NOT USED
6. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
NOT USED
7. INTERRUPT VECTOR 1:
USED WHEN ENVIRONMENT MODE BIT 7 = 1; DEFAULT - 254

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

- 8. BUS PRIORITY 1:
NOT USED.
- 9. INTERRUPT VECTOR 2:
NOT USED
- 10. BUS PRIORITY 2:
NOT USED
- 11. BASE ADDRESS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700
- 12. DEVICE MAP:
NOT USED
- 13. CONTROLLER DESCRIPTOR WORDS:
NOT USED
- 14. CONTROLLER DESCRIPTOR WORDS:
NOT USED

4. CONTROLLING THE PROGRAM

THE FOLLOWING KEYBOARD CONVENTIONS ARE USED BY THE KEYBOARD ENTRY ROUTINES IN THE PROGRAM:

- A. TO DELETE AN INCORRECT CHARACTER FROM AN ENTRY STRING, TYPE A 'RUBOUT'. TYPING A RUBOUT WILL DELETE SUCESSIVE CHARACTERS FROM THE INPUT.
- B. TO DELETE AN ENTIRE LINE, TYPE A 'CONTROL U' (^U).
- C. AN ENTRY MUST BE TERMINATED BY EITHER A 'CARRIAGE RETURN' OR A 'PERIOD'. THE 'PERIOD' TERMINATION IS RECOGNIZED BY THE PROGRAM AS A DEFAULT ENTRY REQUEST. WHEN A LINE IS TERMINATED BY A 'PERIOD' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL ACCEPT THE ENTERED VALUE AND WILL DEFAULT TO THE PRELOADED VALUES FOR ANY REMAINING ENTRIES.
- D. IF A 'CONTROL C' IS TYPED DURING DRIVE TESTING MODE, THE PROGRAM WILL ENTER THE COMMAND MODE. IF A 'CONTROL C' IS TYPED DURING 'ENTER COMMAND' SEQUENCE, WITH NO DRIVES ASSIGNED, THE PROGRAM WILL BE RESTARTED AT LOCATION 204 . OTHERWISE, THE PROGRAM WILL RETURN TO 'ENTER COMMAND' PROMPT AND WAIT FOR A CORRECT SEQUENCE OF CHARACTERS. IF 'CONTROL C' IS TYPED DURING ANY OTHER ENTRY SEQUENCE, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP SEQUENCE BEING ENTERED.

4.1 PARAMETERS

WHEN THE PROGRAM IS STARTED, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

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

'CHANGE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY PARAMETERS TO CHANGED AND WILL CONTINUE.

IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE PROGRAM PARAMETERS.

THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER AND WAIT FOR THE ENTRY. THE PROGRAM WILL TYPE 'INVALID ENTRY' IF THE ENTRY IS NOT CORRECT AND WAIT FOR A CORRECT ENTRY TO BE TYPED. (SEE SECTION 4.1.1)

IF THIS IS THE PROGRAM'S FIRST START, THE STATUS OF THE DRIVES ON THE SELECTED MASSBUS SUBSYSTEM WILL BE PRINTED. ON ALL SUBSEQUENT STARTS, THIS TYPEOUT MAY BE INHIBITED BY SETTING SW<02> =1.

THE FOLLOWING IS AN EXAMPLE UNIT STATUS PRINTOUT:

'UNIT STATUS:
0 ONLINE RM05
1 LOAD DEVICE
2 OFFLINE RM03
3 NOT PRESENT
4 NOT PRESENT
5 NOT AN RM05/3/2
6 NOT PRESENT
7 NOT PRESENT'

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

4.1.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
WRDCNT	10.	8192. (SEE NOTE)	6 - 8192.	CONTROLS THE MAXIMUM WORD COUNT USED FOR DATA TRANSFERS
				NOTE: THE PROGRAM WILL SELECT A MAXIMUM WORD COUNT, WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAX. WORD COUNT ASSIGNED BY THE PROGRAM IS 8192.(1 TRK) WORDS. THE OPERATOR MAY SPECIFY ANY OTHER MAX. WORD COUNT AS LONG AS THE VALUE SPECIFIED IS AT LEAST 6 WORDS BUT NO LARGER THAN 8192. WORDS OR MEMORY AVAILABLE. (WHICH EVER VALUE IS SMALLER)
INTRVL	10.	0	0 - 32767.	DETERMINES THE INTERVAL (IN

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

PASSES 10. 1 1 - 32767.

PATTERN 10. 0 0 - 15.

RANDWC 8 000000 0 OR 1

RATIO 8 000002 0 - 7

ENDING 8 000001 0 OR 1

WRTCHK 8 000001 0 OR 1

MESSAGE 8 000001 0 OR 1

MINUTES) BETWEEN AUTOMATIC
PERFORMANCE SUMMARY TYPEOUTS;
NO TYPEOUT IF THIS PARAMETER
IS 0 OR IF SW<02> =1

NUMBER OF PASSES TO END OF
TEST. (THIS PARAMETER IS NOT
USED WHEN THE PROGRAM IS
OPERATING IN AUTO RUN MODE)

IF PARAMETER=0, DATA PATTERN
IS RANDOMLY SELECTED.
IF PARAMETER>0, SPECIFIES
ONE OF THE 15. PATTERNS.
THE SELECTED DATA PATTERN
IS POINTED BY THE PARAMETER
'PATTERN'. (SEE SECTION 8.4)

IF PARAMETER = 0, THE WORD
COUNT IS RANDOMLY SELECTED
BETWEEN 6 AND THE VALUE
'WRDCNT'.
IF PARAMETER 1, THE WORD
COUNT WILL BE THE VALUE
'WRDCNT'.

CONTROLS THE APPROXIMATE
RATIO OF READ TO WRITE
COMMANDS.

VALUE	R/W RATIO
0	15/1
1	7/1
2	6/2
3	5/3
4	4/4
5	3/5
6	2/6
7	1/7

IF PARAMETER - 1, END OF PASS
DETERMINED BY THE 'WORDS READ'
COUNT.
IF PARAMETER 0, END OF PASS
IS DETERMINED BY THE NUMBER
OF SEEKS.

IF EQ 1, DO AN APPROPRIATE
WRITE CHECK AFTER EACH WRITE
COMMAND. IF EQ 0, SELECT WRITE
CHECK COMMAND RANDOMLY.

IF PARAMETER =1, DO NOT PRINT
ERROR MESSAGES FOR DATA ERRORS
OCCURRING AT LOCATIONS DEFINED
BY THE OPERATOR AS BAD PACK
LOCATION.

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

RANDOM 8 000000 0 OR 1

IF PARAMETER = 0, PRINT ERROR MESSAGES ASSOCIATED WITH BAD PACK LOCATIONS.

IF PARAMETER=0,RANDOM DATA BLOCK ADDRESS IS USED IN 'T' COMMAND
IF PARAMETER=1,SEQUENTIAL DATA BLOCK IS USED IN 'T' COMMAND.

BADBLK 8 000000 0 OR 1

IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS BE INITIALIZED WHEN ASSIGNING A DRIVE;
IF EQ TO 0, THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED IF THE PACK SERIAL NUMBER HAS CHANGED SINCE THE LAST TIME IT WAS READ.

NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE PACK OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

4.1.2 CHANGE DEVICE ADDRESS

THE RM/RH ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM IS STARTED AT ADDRESS 204 OR IF THE PROGRAM DOES NOT RECEIVE A RESPONSE WHEN IT ACCESSES THE DEFAULT RM/RH ADDRESS.

(DEFAULT ADDRESS = 176700, VECTOR = 254)

ADDRESS SELECTION EXAMPLES

EXAMPLE 1

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

EXAMPLE 2

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

4.2 KEYBOARD COMMANDS

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES FOR TEST ('T' COMMAND), WRITE SEQUENTIAL DATA ('W' COMMAND), PERFORM A SEQUENTIAL READ ('R' COMMAND), PERFORM WRITE DATA AND FOLLOWED BY TEST ('WT' COMMAND), REQUEST A DRIVE PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE ('D' COMMAND).

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

THE 'T', 'W', 'R' AND 'WT' COMMANDS ARE EXCLUSIVE TO ONE ANOTHER ON THE SAME DRIVE UNDER TEST. THE 'D' COMMAND MUST BE ENTERED IN ORDER TO ISSUE A DIFFERENT COMMAND TO THE SAME DRIVE UNDER TEST. EXCEPT FOR THE 'S' COMMAND, WHICH CAN BE ENTERED AT ANY TIME DURING THE TEST.

IF THE PROGRAM WAS STARTED AT ADDRESS 204 OR IF NO DRIVES ARE ASSIGNED FOR TESTING, THE FOLLOWING MESSAGE WILL BE TYPE BEFORE ENTERING THE COMMAND MODE. HOWEVER, IF A 'CONTROL C' IS TYPED WHILE TESTING IS IN PROGRESS, THE FOLLOWING MESSAGE WILL BE OMITTED AND THE PROGRAM WILL ENTER COMMAND MODE.

'NO DRIVES ASSIGNED'

WHEN THE PROGRAM ENTERS THE COMMAND MODE, THE FOLLOWING PROMPT WILL BE TYPED:

'HH:MM:SS
ENTER COMMANDS:'

THE PROGRAM WILL THEN ACCEPT ANY OF THE VALID COMMANDS. AT THE COMPLETION OF A COMMAND, THE PROGRAM WILL EXIT COMMAND MODE AND TRY TO ASSIGN THE DRIVE(S) THAT WERE REQUESTED. IF THE DRIVE(S) CANNOT BE ASSIGNED, ONE OF THE FOLLOWING ERROR MESSAGES WILL BE REPORTED AND THE PROCESS CONTINUES FOR EACH DRIVE.

RESPONSE	COMMAND(S)
?DRIVE N LOAD DEVICE	T, W, R, WT
?DRIVE N OFFLINE	T, W, R, WT
?DRIVE N NOT ASSIGNED	D, S
?DRIVE N ALREADY ASSIGNED	T, W, R, WT
?DRIVE N NOT PRESENT	T, W, R, WT
?DRIVE N UNSAFE	T, W, R, WT
?DRIVE N NOT AN RM05/3/2	T, W, R, WT

NEXT, THE PROGRAM WILL PROCESS ALL THE ASSIGNED DRIVES AS FOLLOWS:

WHEN THE PROGRAM IS ASSIGNING THE DRIVES, THE OPERATOR WILL BE ASKED TO CHANGE THE DRIVE PARAMETERS WITH THE FOLLOWING PROMPT:

'CHANGE DRIVE PARAMETERS (L) N ?'

IF THE ENTRY IS A 'N' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT), THE PROGRAM WILL NOT ALLOW ANY DRIVE PARAMETERS TO BE CHANGED AND WILL PROCEED TO TEST THE DRIVES AS COMMANDED. IF THE ENTRY IS A 'Y' FOLLOWED BY CARRIAGE RETURN, THE OPERATOR WILL BE ALLOWED TO CHANGE THE DRIVE PARAMETERS AS FOLLOWS.

THE PROGRAM WILL FIRST TELL THE OPERATOR WHICH DRIVE IS BEING REFERENCED FOR CHANGES.

'***** DRIVE # N'

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

THE PROGRAM WILL THEN INFORM THE OPERATOR WHAT THE HARD WIRED MBA SERIAL NUMBER IS AND THE PACK SERIAL NUMBER IS, IN THE FOLLOWING FORMAT:

'MBA S/N: X PACK S/N: Y'

WHERE 'X' IS THE HARD WIRED DECIMAL SERIAL NUMBER CONTAINED IN THE RMSN REGISTER OF THE MBA. IF THE MBA SERIAL NUMBER IS NOT JUMPERED IN THE RMSN REGISTER, 'X' WILL APPEAR AS '????'.

WHERE 'Y' IS THE OCTAL SERIAL NUMBER READ FROM THE DEC144 BAD SECTOR FILE ON THE PACK. IF THE DEC144 FILE WAS UNABLE TO BE READ SUCCESSFULLY, 'Y' WILL APPEAR AS 'NONE'.

THE PROGRAM WILL THEN ASK FOR ADDRESS LIMIT CHANGES WITH THE FOLLOWING TYPEOUT:

'ENTER ADDRESS LIMITS:'

THE PROGRAM WILL REQUEST VALUES FOR THE FOLLOWING ADDRESS LIMIT PARAMETERS.

NAME	DEFAULT VALUE	VALUE RANGE	FUNCTION
MINCYL	0	0 - 821.	THE MINIMUM CYLINDER ADDRESS
MAXCYL	821.	0 - 821.	THE MAXIMUM CYLINDER ADDRESS
MINTRK	0	0 - NN	THE MINIMUM TRACK ADDRESS
MAXTRK	NN	0 - NN	THE MAXIMUM TRACK ADDRESS
MINSEC	0	0 - 31.	THE MINIMUM SECTOR ADDRESS
MAXSEC	31.	0 - 31.	THE MAXIMUM SECTOR ADDRESS

WHERE 'NN' IS 4. FOR AN RM03/02 AND 18. FOR AN RM05.

THE PROGRAM WILL THEN ASK FOR BAD SECTOR ADDRESSES WITH THE FOLLOWING TYPEOUT:

'ENTER BAD SECTR ADRS:'

THE FORMATS USED TO ENTER BAD SECTOR ADDRESS LOCATIONS ARE AS FOLLOWS:

EXAMPLE 1: CYL,TRK,SEC= C,T,S<CR>

- A. THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES FOR ERRORS OCCURRING AT THE SPECIFIED ADDRESS.
- B. LEADING ZEROS ARE NOT REQUIRED.

EXAMPLE 2: CYL,TRK,SEC= C,T<CR>

- A. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE CONSIDERED BAD.
- B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.

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

EXAMPLE 3: (YL,TRK,SEC- C<CR>

A. WHEN THIS FORMAT IS USED, THE ENTIRE CYLINDER WILL BE CONSIDERED BAD

B. DATA ERRORS WILL BE HANDLED AS IN 'EXAMPLE 1'.

IF CONTROL C (^C) IS TYPED AS AN ENTRY, ALL CURRENT BAD SECTOR ENTRIES WILL BE LOST AND THE FOLLOWING MESSAGE WILL BE TYPED.

'* ALL CURRENT ENTRIES LOST *'

AFTER TYPING THE PREVIOUS MESSAGE, THE PROGRAM WILL WAIT FOR THE OPERATOR TO ENTER ANOTHER BAD SECTOR AS IN THE PREVIOUS EXAMPLES.

IF 'L' IS TYPED FOR AN INPUT CHARACTER, THE PROGRAM WILL TYPE A LIST OF DEC144 BAD SECTORS, AND THE MANUALLY ENTERED BAD SECTORS, WHICH ARE STORED IN THE DRIVE PARAMETER TABLE (DPB) FOR THAT PARTICULAR DRIVE.

IF THERE ARE NO BAD SECTORS IN THE DPB TABLE, THE FOLLOWING MESSAGE WILL BE TYPED:

'DEC144 AND MANUAL BAD SECTOR LIST
* NO ENTRIES *'

HOWEVER, IF THERE ARE ENTRIES IN DPB TABLE, THE LIST WILL BE TYPED IN THE FOLLOWING FORMAT:

'DEC144 AND MANUAL BAD SECTOR LIST
8,8,3
16,13
256
500,1,29'

THE ABOVE LIST OF BAD SECTORS, ENTRY 1 INDICATES THAT CYLINDER 8., TRACK 8., SECTOR 3 IS THE BAD SECTOR. ENTRY 2 INDICATES THAT ON CYLINDER 16., TRACK 13., ALL THE SECTORS ARE BAD (ENTIRE TRACK IS BAD). ENTRY 3 INDICATES THAT ON CYLINDER 256., ALL TRACKS AND SECTORS ARE BAD (ENTIRE CYLINDER IS BAD). ENTRY 4 INDICATES THAT CYLINDER 500., TRACK 1, SECTOR 29. IS THE BAD SECTOR.

AFTER TYPING EITHER OF THE TWO PREVIOUS MESSAGES, THE PROGRAM WILL RETURN TO WAIT FOR MORE ENTRIES TO BE MADE INTO THE BAD SECTOR TABLE, AS IN EXAMPLES 1, 2 AND 3.

TO TERMINATE THE BAD SECTOR ADDRESS ENTRY, TYPE A 'CARRIAGE RETURN' IN RESPONSE TO THE ENTRY REQUEST OR TERMINATE THE ENTRY WITH A 'PERIOD' FOLLOWED BY A 'CARRIAGE RETURN'.

4.2.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR A TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S).

FORMAT: TN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE

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

TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: TO<CR> - ASSIGN DRIVE 0 FOR TEST
TA<CR> - ASSIGN ALL AVAILABLE DRIVES FOR TEST

4.2.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: DO<CR> - DEASSIGN DRIVE 0
DA<CR> - DEASSIGN ALL DRIVES BEING TESTED.

4.2.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED DRIVE(S). AFTER THE 'S' COMMAND HAS BEEN PERFORMED, THE PROGRAM WILL AUTOMATICALLY RESUME TESTING THE DRIVE(S) WHICH WERE UNDER TEST.

FORMAT: SN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: SO<CR> - TYPEOUT PERFORMANCE SUMMARY FOR DRIVE 0
SA<CR> - TYPEOUT PERFORMANCE SUMMARY FOR ALL DRIVES BEING TESTED.

4.2.4 'W' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE OF THE DISK, WITH DATA ACCEPTABLE TO THE PERFORMANCE EXERCISER PROGRAM.

FORMAT: WN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: WO<CR> - WRITE A DATA PATTERN ON DRIVE 0.
WA<CR> - WRITE A DATA PATTERN ON ALL AVAILABLE DRIVES.

4.2.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE DISK.

FORMAT: RN<CR>

N = DRIVE NUMBER. MAY BE 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: RO<CR> - READ THE DATA ON DRIVE 0.
RA<CR> - READ THE DATA ON ALL AVAILABLE DRIVES.

4.2.6 'WT' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE DATA, FOLLOWED BY A 'T' COMMAND.

FORMAT: WTN<CR>

N = DRIVE NUMBER 0 TO 7 OR 'A'. ENTRY MUST BE TERMINATED BY A CARRIAGE RETURN <CR>.

EXAMPLE: WT0<CR> - WRITE A DATA PATTERN AND TEST DRIVE 0
WTA<CR> - WRITE A DATA PATTERN AND TEST ALL DRIVES

5. PERFORMANCE SUMMARY TYPEOUT

5.1 THE PROGRAM WILL DISPLAY A PERFORMANCE SUMMARY FOR THE DRIVES BEING EXERCISED. THIS SUMMARY WILL BE DISPLAYED AUTOMATICALLY IF THE PARAMETER 'INTRVL' IS NOT ZERO AND SW<02>=0, OR IF THE DRIVE HAS REACHED THE DEFINED NUMBER OF PASSES AND SW<08>=0, OR IF THE OPERATOR REQUESTS TO DO SO BY USE OF THE 'S' COMMAND.

THE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'TIME'	ELAPSED TIME OF PROGRAM
'DRIVE'	DRIVE NUMBER - DRIVE TYPE
'PASS'	PRESENT PASS COUNT FOR THE DRIVE
'MBA S/N'	HARD WIRED MASSBUS ADAPTER SERIAL NUMBER(RMSN)
'PACK S/N'	SERIAL NUMBER READ FROM THE DEC144 FILE ON THE PACK
'WT OFLOW'	NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
'WRDS WRITN'	TOTAL NUMBER OF WORDS WRITTEN BY THE DRIVE
'RD OFLOW'	NUMBER OF TIMES 'WRDS READ' HAS OVERFLOWED
'WRDS READ'	TOTAL NUMBER OF WORDS READ BY THE DRIVE
'SEEKS'	NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
'SOFT'	NUMBER OF SOFT DATA ERRORS
'HARD'	NUMBER OF HARD DATA ERRORS
'SKI'	NUMBER OF 'SKI' ERRORS
'MISP'	NUMBER OF PROGRAM DETECTED POSITIONING ERRORS
'OTHER'	TOTAL ERRORS OF OTHER TYPES

ALL DATA TRANSFER COUNTS, SEEK COUNTS AND ERROR COUNTS ARE ACCUMULATIVE AND WILL NOT BE CLEARED AFTER EACH PASS.

TO CALCULATE THE TOTAL NUMBER WORDS READ OR WRITTEN, TAKE THE OVERFLOW COUNT (RD OFLOW OR WT OFLOW), MULTIPLY IT BY 2,147,483,647., THEN ADD THAT NUMBER TO THE WORDS READ OR WRITTEN (WRDS READ OR WRDS WRITN).

NOTE: ERRORS EM1, EM2, EM3, EM4, EM5, & EM10 ARE NOT INCLUDED IN THE 'OTHER' ERROR TOTAL.

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR) WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION

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

AND IS NOT CORRECTABLE OR DOES NOT BECOMF CORRECTABLE AFTER THE PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR.

THE RETRY SEQUENCE IS 16. RE-READS AT TRACK CENTER AND 2 ATTEMPS BOTH AT POSITIVE AND NEGATIVE OFFSETS.

5.2.2 SOFT ERRORS

- A. ECC CORRECTABLE 'DCK' ERRORS.
- B. 'DCK' & 'ECH' ERRORS WHICH BECOME ECC CORRECTABLE DURING RETRY OR WHICH ARE READ CORRECTLY DURING RETRY.
- C. HEADER READ ERRORS - READ DATA, READ HEADER & DATA, OR WRITE DATA COMMANDS
- D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE 'DCK' ERROR DURING THE RETRY SEQUENCE.

6. DATA CHECKING & ERROR RECOVERY

6.1 DATA COMPARISON

DATA COMPARISON OCCURS AFTER EACH 'RDDAT' (READ DATA) OR 'RDHD' (READ HEADER AND DATA) OPERATION UNDER THE FOLLOWING CONDITIONS:

- A. THE COMMAND TERMINATED WITH NO ERRORS AND SW<01>-0
- B. THE OPERATION TERMINATED WITH 'DCK' SET AND THE ERROR IS ECC CORRECTABLE OR THE SECTOR IN ERROR IS READ CORRECTLY AFTER RETRY ATTEMPTS.

6.2 VERIFICATION OF DATA WRITTEN

DATA VERIFICATION IS DONE EITHER THROUGH READING THE DATA BACK AND MATCHING THE DATA WITH ONE OF THE 15. PATTERNS OR THROUGH ISSUING A WRITE CHECK COMMAND AFTER DOING A WRITE DATA COMMAND.

6.3 BAD ADDRESS FLAGGING

WHEN A DRIVE IS ASSIGNED TO BE TESTED, THE PROGRAM READS THE BAD SECTOR FILE (DEC144) FROM THE DISK AND THEN ALLOWS ADDITIONAL BAD SECTORS TO BE ENTERED MANUALLY.

A MAXIMUM OF 252. BAD SECTORS ARE ALLOWED FOR EACH DRIVE, BOTH READING FROM THE DEC144 FILE AND ENTERING FROM KEYBOARD.

THE MANUALLY ENTERED BAD SECTORS ARE NOT RECORDED TO THE BAD SECTOR FILE OF THE DISK CURRENT UNDER TESTING.

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED IN THE BAD SECTOR TABLE, THE PROGRAM WILL INHIBIT THE ERROR REPORT FOR THAT ERROR.

DATA CHECK ERRORS ('DCK')

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

WRITE CHECK ERRORS ('WCE')
OPERATION INCOMPLETE ERRORS ('OPI')
DRIVE TIMING ERRORS ('DTE')
HEADER READ ERRORS ('FER W/ HCRC', 'MCE W/ HCRC' OR 'HCRC')

7. ERROR MESSAGES

ERRORS ARE REPORTED ON THE TTY CONSOLE. THE PROGRAM CONTAINS NO CODED ERROR HALTS. IF THE PROGRAM HALTS (ASSUMING, OF COURSE, THAT SW<15> IS NOT SET), AN UNRECOVERABLE PROGRAM CONDITION HAS OCCURRED OR A CENTRAL PROCESSOR FAILURE HAS OCCURRED.

ERROR MESSAGES ARE MADE UP OF SEVERAL LINES. EACH TYPE OF ERROR HAS SEVERAL OPTIONAL LINES WHICH MAY APPEAR WITH IT. ALL OF THE POSSIBLE ERROR MESSAGE LINES WHICH MAY APPEAR ARE GIVEN IN THE SECTION DESCRIBING THE PARTICULAR ERROR HEADER.

7.1 ERROR DESCRIPTION LINES

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE TAG	TEXT
---	----
EM1	RH CONTROLLER INTERRUPT OCCURRED (RMAS=0) THE RH CONTROLLER INTERRUPTED AND THE ATTENTION SUMMARY REGISTER (RMAS) WAS CLEARED.
EM2	UNEXPECTED ATTENTION OCCURRED THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT PERFORMING AN OPERATION.
EM3	MASSBUS PARITY ERROR (MCPE=1) THE RH DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.
EM4	MASSBUS PARITY ERROR (PAR=1) THE INDICATED RM DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH LOADED THE SPECIFIED REGISTER.
EM5	ADDRESS PLUG CHANGE BIT SET THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
EM6	RH DIDN'T RESPOND TO ADDRESSING WHEN THE PROGRAM ADDRESSED THE RH, NO RESPONSE WAS RECEIVED FROM THE INDICATED ADDRESS.
EM10	UNCORRECTABLE MASSBUS PARITY ERROR

970		THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED REGISTER.
971		
972		
973		
974	EM11	FATAL MASSBUS PARITY ERROR
975		
976		A CONTROL BUS PARITY ERROR OCCURRED WHEN THE RM ATTEMPTED TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.
977		
978		
979	EM12	PERSISTENT DEVICE UNSAFE
980		
981		THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN CLEARED BY MANUAL INTERVENTION.
982		
983		
984		
985		
986		
987	EM13	OPERATION NOT COMPLETED WITHIN TIME LIMIT
988		
989		THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 10. SECONDS AFTER THE OPERATION WAS INITIATED.
990		
991		
992	EM14	UNIT WENT OFFLINE
993		
994		THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION. (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'T' COMMAND TO RE-INITIATE TESTING.
995		
996		
997		
998		
999	EM15	NO RESPONSE TO PORT REQUEST
1000		
1001		THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED TO THE REQUESTING PORT WITHIN 15. SECONDS AFTER PORT REQUEST TO THE DRIVE FROM THE REPORTING PORT.
1002		
1003		
1004		
1005	EM20	HEADER CRC ERROR
1006		
1007		A HEADER CRC ERROR WAS DETECTED AT THE INDICATED DISK ADDRESS. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.
1008		
1009		
1010		
1011	EM21	DATA CHECK ('DCK') ERROR
1012		
1013		A DATA CHECK ERROR WAS DETECTED AT THE INDICATED SECTOR. THE FULL RETRY SEQUENCE (INCLUDING OFFSET) WILL BE INITIATED FOR THE SECTOR IN ERROR IF THE ECC HARD ERROR ('ECH') BIT IS SET.
1014		
1015		
1016		
1017		
1018	EM22	WRITE CHECK ERROR - DATA CHECK ('DCK') SET
1019		
1020		A WRITE CHECK ERROR OCCURRED AND THE DATA CHECK ('DCK') BIT WAS SET. IF 'ECH' IS NOT SET, THE OPERATION WILL BE RETRIED UP TO 3 TIMES; IF THE 'ECH' BIT IS SET, THE OPERATION WILL BE RETRIED UP TO 10. TIMES.
1021		
1022		
1023		
1024		
1025	EM23	WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET
1026		

1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083

A WRITE CHECK ERROR OCCURRED AND 'DCK' WAS NOT SET. THE WORDS WHICH CAUSED THE ERROR ARE DISPLAYED IN THE ERROR MESSAGE. THE OPERATION WILL BE RETRIED 3 TIMES.

EM24 HEADER READ ERROR - 'FMT' BIT DROPPED

A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING PERFORMED AND A 'FMT' ERROR OCCURRED. THE PROGRAM RE-READ THE HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM25 HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM26 FORMAT ERROR ('FER')

FORMAT ERROR OCCURRED. WHEN THE HEADER WAS RE-READ, THE 'HCRC' BIT WAS NOT SET. THE CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM27 HEADER COMPARE ('HCE') ERROR

SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.

EM30 MISCELLANEOUS DRIVE ERROR

THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS - 'AOE', 'RMR', 'ILF', OR 'ILR'

EM31 OPERATION INCOMPLETE ('OPI') ERROR

AN OPERATION INCOMPLETE ERROR OCCURRED AT THE INDICATED SECTOR.

EM32 DRIVE TIMING ('DTE') ERROR

DRIVE TIMING ERROR OCCURRED ON THE INDICATED SECTOR. THE OPERATION WILL BE RETRIED 3 TIMES.

EM33 PARITY ('PAR') ERROR AFTER OPERATION STARTED

THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE OPERATION WILL BE RETRIED 3 TIMES.

EM34 WRITE CLOCK FAILURE ('WCF')

A WRITE CLOCK FAILURE OCCURRED DURING THE OPERATION. THE OPERATION WILL BE RETRIED 3 TIMES.

EM35 INVALID ADDRESS ('IAE') ERROR

AN INVALID ADDRESS ERROR OCCURRED DURING THE OPERATION.

1084	EM36	WRITE LOCK ('WLE') ERROR
1085		A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE
1086		LOCKED.
1087		
1088	EM40	RH CONTROLLER OR UNIBUS TRANSFER ERROR
1089		'TRE' IS SET IN THE RH CONTROL REGISTER AND NO DRIVE
1090		ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 3
1091		TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF',
1092		OR 'MDPE'.
1093		
1094	EM41	BUS ADDRESS OR WORD COUNT INCORRECT
1095		NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES
1096		THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE
1097		WORD COUNT REGISTER IS NOT ZERO.
1098		
1099	EM42	DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED
1100		NO SUBSYSTEM ERROR WAS SIGNALLED; HOWEVER, THE DATA DOES NOT
1101		COMPARE.
1102		
1103	EM43	CAN'T MATCH DATA READ WITH A PATTERN
1104		THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD
1105		PATTERNS.
1106		
1107	EM44	ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER
1108		THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM
1109		FOUND EITHER ERROR BITS IN THE RM SET OR ERROR BITS IN
1110		THE RH CONTROLLER SET.
1111		
1112	EM45	ECC LOGIC FAILURE
1113		DURING 'DCK' ERROR PROCESSING, THE CONTENTS OF THE ECC
1114		POSITION REGISTER (RMEC1) OR THE CONTENTS OF ECC PATTERN
1115		REGISTER (RMEC2) WERE NOT VALID. THE POSITION REGISTER WAS
1116		EITHER 0 OR GREATER THAN 010040, OR THE PATTERN REGISTER
1117		CONTAINED ZEROS.
1118		
1119	EM46	BUS ADDRESS OR WORD COUNT NOT CONSISTENT
1120		THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE
1121		NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS
1122		REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE
1123		WORD COUNT REGISTER.
1124		
1125		
1126	EM50	SEEK INCOMPLETE ERROR
1127		THE DRIVE SIGNALLED EITHER 'SKI' ERROR.
1128		
1129	EM51	NOT USED
1130		
1131		
1132	FM60	DEVICE UNSAFE
1133		
1134		
1135		
1136		
1137		
1138		
1139		
1140		

1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197

THE INDICATED DRIVE UNSAFE ERROR OCCURRED; THE ERROR WAS
CLEARED BY A 'DRIVE CLEAR' INSTRUCTION.

7.2 DETAIL ERROR LINES

THE LINE NUMBERS GIVEN BELOW ARE FOR REFERENCE ONLY.

LINE 1

HH:MM:SS

'HH:MM:SS' IS THE TIME SINCE THE PROGRAM WAS STARTED.
(HOURS, MINUTES, SECONDS)

LINE 2

'PRSNT COMMAND= XXXX PREV COMMAND= YYYY'

MNEMONICS USED FOR THE COMMANDS ARE DEFINED BELOW:

- SEEK - SEEK (OCTAL 5)
- RECAL - RECALIBRATE (OCTAL 7)
- DRVCLR - DRIVE CLEAR (OCTAL 11)
- RELSE - RELEASE (OCTAL 13)
- OFFSET - OFFSET (OCTAL 15)
- RTC - RETURN TO CENTERLINE (OCTAL 17)
- READIN - READIN PRESET (OCTAL 21)
- PACK - PACK ACKNOWLEDGE (OCTAL 23)
- SEARCH - SEARCH (OCTAL 31)
- *GETREG - GET REGISTERS (OCTAL 41)
- *SETFMT - SET FORMAT (ECI OR HCI) (OCTAL 43)
- *SELDRV - SELECT DRIVE (OCTAL 45)
- WCKD - WRITE CHECK DATA (OCTAL 51)
- WCKHD - WRITE CHECK HEADER & DATA (OCTAL 53)
- WRDAT - WRITE DATA (OCTAL 61)
- WRTHD - WRITE CHECK HEADER & DATA (OCTAL 63)
- RDDAT - READ DATA (OCTAL 71)
- RDHD - READ HEADER & DATA (OCTAL 73)

* SPECIAL RM DRIVER COMMAND (NOT A CONTROLLER COMMAND)

(DISPLAY OF THE RH/RM REGISTERS IN TWO GROUPS:
RMCS1, RMCS2, RMD5, RMER1, RMER2, RMEC1 AND RMEC2 FORM THE FIRST
GROUP; ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST GROUP WILL BE
DISPLAYED.)

THE ABOVE LINE WILL BE TYPED IF THE ERROR OCCURRED DURING
THE NON-DATA TRANSFER PART OF THE OPERATION.

'* ERROR AT BAD TRACK/SECTOR'

1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254

THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURES AT AN ADDRESS ON THE DISK WHICH THE OPERATOR HAS IDENTIFIED AS BEING BAD. PARAMETER 'NOTPRT' MUST BE 0 FOR THE ERROR TO BE REPORTED.

A WORD CALLED 'STATUS' IS DISPLAYED WITH THE RM REGISTERS. THE CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE RM DRIVE HANDLER ROUTINE. (SEE SECTION 9.7)

LINE 3

ERROR AT CXXX TYY SZZ PREV ADDR= CUUU TVV SWW

THE ACTUAL ADDRESS OF THE ERROR SECTOR AND THE PREVIOUS DISK ADDRESS ARE GIVEN IN THIS LINE. CYLINDER, TRACK, & SECTOR ADDRESSES ARE IN DECIMAL.

LINE 4

PRSNR ADDR= CXXX TYY SZZ PREV ADDR= CUUU TVV SWW

THIS LINE IDENTIFIES THE ADDRESS WHEN THE ERROR WAS DETECTED; THE PREVIOUS ADDRESS IS ALSO GIVEN. CYLINDER, TRACK, & SECTOR ADDRESSES ARE GIVEN IN DECIMAL.

LINE 5

START CYL= XXX END CYL= YYY

THIS LINE IDENTIFIES THE STARTING CYLINDER OR A SEEK (IMPLIED) AND THE DESTINATION CYLINDER. CYLINDER ADDRESSES ARE IN DECIMAL.

LINE 6

START CYL= XXX END CYL= YYY ACTUAL CYL= ZZZ

THIS LINE IDENTIFIES THE STARTING CYLINDER OF AN IMPLIED SEEK, THE DESTINATION CYLINDER, AND THE CYLINDER THE DISK ACTUALLY STOPPED AT. CYLINDER ADDRESSES ARE IN DECIMAL.

LINE 7

RMBA= XXXX RMWC= YYYY

THIS LINE GIVES THE CONTENTS OF THE RH CONTROLLER BUFFER ADDRESS REGISTER AND THE RH CONTROLLER WORD COUNT REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT SET.

LINE 8

START CYL= XXX START TRK= YY START SECTOR= ZZ

1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311

THIS LINE IDENTIFIES THE STARTING DISK ADDRESS OF THE PRESENT OPERATION. CYLINDER, TRACK, AND SECTOR VALUES ARE DECIMAL.

LINE 9

RMDA= XXXX RMCA= YYYY

THIS LINE GIVES THE CONTENTS OF THE RM TRACK AND SECTOR ADDRESS REGISTER AND THE CONTENTS OF THE DESIRED CYLINDER ADDRESS REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT SET.

LINE 10

BUFFER ADDR= XXXX WRD CNT= YYYY ACTUAL NUMBR WRDS XFRD- ZZZZ

THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE CURRENT DATA TRANSFER OPERATION, ITS SIZE (WORD COUNT), AND THE ACTUAL NUMBER OF WORD TRANSFERED. THE STARTING ADDRESS OF THE BUFFER IS IN OCTAL, THE WORD COUNT AND WORDS TRANSFERED VALUE ARE IN DECIMAL.

LINE 11

GOOD DATA= XXXX BAD DATA= YYYY SECT POS= ZZZ

THIS LINE GIVES THE GOOD DATA, THE ACTUAL DATA FROM THE DISK, AND THE LOCATION IN THE SECTOR OF THE ACTUAL DATA. THE SECTOR POSITION IS IN DECIMAL.

LINE 12

HEADER CONTENTS OF ERROR SECTOR= XXXX XXXX XXXX XXXX

THIS LINE GIVES THE CONTENTS OF THE HEADER OF THE SECTOR WHICH GAVE THE ERROR.

LINE 13

RMEC1= XXXX RMEC2= YYYY

THIS LINE WILL BE PRINTED AFTER A SUCESSFUL RETRY OF A SECTOR WHICH BECAME ECC CORRECTABLE DURING RETRY.

LINE 14

ECC CORRECTABLE WITHOUT OFFSET

THE SECTOR IN ERROR IS ECC CORRECTABLE; NO RETRY ATTEMPTS ARE NECESSARY.

1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368

LINE 15

READ CORRECTLY AT (NEG OR POS) OFFSETTHE SECTOR IN ERROR WAS READ WITHOUT ERROR AT THE INDICATED
OFFSET VALUE.

LINE 16

ECC CORRECTABLE AT (NEG OR POS) OFFSETTHE SECTOR IN ERROR BECAME ECC CORRECTABLE AT THE INDICATED
OFFSET.

LINE 17

CORRECTED ON X RETRYTHE OPERATION WAS PERFORMED ERROR FREE ON THE INDICATED RETRY
ATTEMPT.

LINE 18

UNCORRECTABLE AFTER X RETRIESTHE OPERATION COULD NOT BE PERFORMED CORRECTLY AFTER THE
INDICATED NUMBER OF RETRY ATTEMPTS.

LINE 19

DIFFERENT ERROR DURING RETRYWHILE THE PROGRAM WAS RETRYING THE ERROR, A DIFFERENT OCCURRED.
IF THIS LINE IS PRINTED, THE RH/RM REGISTERS WILL ALSO BE
PRINTED (SEE LINE 2).

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS= XXXXTHIS LINE GIVES THE TOTAL DATA COMPARISON ERROR COUNT. THE
VALUE GIVEN IS IN DECIMAL.

1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425

LINE 22

THE DATA COMPARED OK

THIS LINE INDICATES THE RESULTS OF THE DATA COMPARISON FOLLOWING
ECC CORRECTION.

LINE 23

ECC CORRECTION RESULTS

THE PROGRAM PERFORMED ECC CORRECTION AND THE RESULTS ARE REPORTED.
THE ADDRESS IN MEMORY OF THE WORD(S) IN ERROR ARE GIVEN, THE WORD(S)
BEFORE CORRECTION AND THE WORD(S) AFTER CORRECTION ARE PRINTED.

LINE 24

ERROR BURST BEGINS AT WORD XXX IN DATA FIELD OF ERROR SECTOR

THIS IS AN INFORMATIONAL LINE WHICH WILL BE PRINTED FOR 'DCK' ERRORS
WHICH ARE ECC CORRECTABLE OR WHICH BECOME ECC CORRECTABLE DURING
RETRY. 'XXX' IS THE WORD OFFSET VALUE FROM 'RMEC1' AND IS IN
DECIMAL.

LINE 25

ERROR WAS NOT IN THE DATA READ -
ECC CORRECTION CAN'T BE PERFORMED

THE DATA ERROR WAS NOT IN DATA TRANSFERED TO MEMORY.

LINE 26

CONTENTS OF THE ERROR SECTOR (REPORTED ABOVE)

IF SW<03> IS SET, THE SECTOR WHICH GAVE THE 'DCK', 'DTE' OR,
'WCF' ERROR OR 'HARD' DATA CHECK ERROR IS PRINTED. THE
CONTENTS OF THE SECTOR FOLLOW THIS LINE.

LINE 27

TOTAL ERRORS:X WOFL:N WRDS WRITN: YYYY ROFL:N WRDS READ: ZZZZ

THIS IS THE LAST LINE PRINTED FOR ALL NON-POSITIONING
TYPE ERRORS.

'ERRORS' IS THE TOTAL ERROR COUNT FOR THE DRIVE AND INCLUDES
EVERY ERROR DETECTED, REGARDLESS OF TYPE.

'WOFL' NUMBER OF TIMES 'WRDS WRITN' HAS OVERFLOWED
'WRDS WRITN' IS THE TOTAL NUMBER OF WORDS WRITTEN THE DRIVE.

1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482

'ROFL' NUMBER OF TIMES 'WRDS READ' HAS OVERFLOWED
'WRDS READ' IS THE TOTAL NUMBER OF WORD READ BY THE DRIVE.

LINE 28

TOTAL SEEKS: XXX TOTAL POS ERR= YYY TOTAL SKI ERR- Z

THIS IS THE LAST LINE PRINTED FOR ALL POSITIONING TYPE ERRORS.

'TOTAL SEEKS' IS THE TOTAL NUMBER OF SEEK OPERATIONS PERFORMED BY THE DRIVE.

'TOTAL POS ERR' IS THE TOTAL NUMBER OF PROGRAM DETECTED POSITIONING ERROR BY THE DRIVE.

'TOTAL SKI ERR' IS THE TOTAL NUMBER OF 'SKI' ERRORS SIGNALLED BY THE DRIVE.

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED, PROVIDING APT TTY ENABLE BIT IS SET OR DIAGNOSTIC LOADED BY OTHER THAN APT SCRIPT MODE, ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH CONTROLLER INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. COMMAND ENTRIES WILL NOW BE ACCEPTED BY THE PROGRAM.

THE PROGRAM SCANS ITS INTERNAL ASSIGNMENT TABLES, LOOKING FOR:

- 1) DRIVES TO ASSIGN/DEASSIGN
- 2) PERFORMANCE SUMMARY TYPEOUT REQUESTS
- 3) DRIVES REQUIRING COMMAND INITIATION, BUFFER ASSIGNMENT, OR PARAMETER SELECTION.
- 4) DRIVES COMPLETING CURRENT OPERATIONS.

THE PROGRAM CONTINUES SCANNING ITS TABLES UNTIL AN ENTRY IS FOUND. IN THE CASE OF THE PROGRAM AT INITIAL START, THE FIRST ENTRY WILL BE MADE BY THE OPERATOR WHEN A DRIVE IS ASSIGNED ('T' COMMAND).

WHEN A DRIVE IS ASSIGNED, THE KEYBOARD ENTRY ROUTINE VERIFIES THAT THE DRIVE IS PRESENT, IS AN RM05/3/2, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT16', AND ISSUES A 'RECALIBRATE' INSTRUCTION.

PARAMETERS FOR THE OPERATION ARE SELECTED AND A BUFFER IS ASSIGNED. IF THE OPERATION IS A WRITE OR WRITE CHECK COMMAND, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK COMMANDS ARE ISSUED AFTER EACH WRITE COMMAND. THE WRITE CHECK COMMAND USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE COMMAND.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539

THE COMMAND INITIATION ROUTINE FIRST LOOKS AT THE CYLINDER ADDRESS OF THE REQUESTED OPERATION. IF THE DRIVE MUST SEEK TO ANOTHER CYLINDER TO PERFORM THE OPERATION, THE PROGRAM ISSUES A SEARCH INSTRUCTION TO THE DRIVE WITH A 'TARGET' SECTOR WHICH IS 5 SECTORS EARLIER THAN THE 'TRANSFER' SECTOR. (THIS ALLOWS THE PROGRAM TO INITIATE OPERATIONS ON ANOTHER DRIVE WHILE THE PRESENT DRIVE, OR OTHER DRIVES, ARE SEARCHING FOR 'TARGET' SECTORS. ALL SEEKS ISSUED BY THE PROGRAM ARE IMPLIED SEEK SEARCH OPERATIONS.) WHEN A SEARCHING DRIVE FINDS THE 'TARGET' SECTOR AND INTERRUPTS, THE PROGRAM THEN ISSUES THE REQUESTED COMMAND TO THE DRIVE THAT INTERRUPTED.

WHEN THE DATA TRANSFER OPERATION IS COMPLETE, THE DRIVE REGISTERS ARE STORED AND A DATA TRANSFER IS INITIATED FOR A WAITING DRIVE.

IF THE OPERATION HAS BEEN COMPLETED NORMALLY, THE SAVED DRIVE REGISTERS ARE CHECKED TO VERIFY THAT NO ERROR BITS ARE SET; THE RH CONTROLLER BUS ADDRESS AND WORD COUNT ADDRESS REGISTERS ARE CHECKED TO VERIFY THAT THE CORRECT NUMBER OF WORDS HAVE BEEN TRANSFERRED AND THAT THE TWO REGISTERS ARE CONSISTENT WITH EACH OTHER; AND IF THE COMMAND WAS A READ COMMAND, THE DATA BUFFER IS COMPARED. WHEN THIS SEQUENCE IS COMPLETED, THE DRIVE IS RETURNED TO THE ASSIGNED, INACTIVE LIST. THE PROGRAM THEN INITIATES A DATA TRANSFER ON A WAITING DRIVE AND RESELECTS AND REINITIATES ANOTHER OPERATION ON THE RELEASED DRIVE.

ERRORS WHICH OCCUR ARE PROCESSED IN THE FOLLOWING ORDER. MULTIPLE ERRORS WILL BE REPORTED AS THE FIRST ERROR TYPE CHECKED.

A. ERRORS REPORTED FOR OPERATIONS WHICH HAVE NOT COMPLETED NORMALLY.

OPERATION NOT COMPLETED WITHIN TIME LIMIT - EM13
UNIT WENT OFFLINE - EM14

B. ERRORS REPORTED FOR OPERATIONS WHICH COMPLETE NORMALLY.

CORRECTABLE UNSAFE - EM60
DRIVE TIMING ERROR - EM32
DATA CHECK ERROR - EM21
WRITE CHECK WITH DCK SET - EM22
HEADER CRC ERRORS - EM20
FORMAT ERRORS - EM24, EM26
HEADER COMPARE ERRORS - EM25, EM27
PROGRAM DETECTED POSITIONING ERROR - EM51
SEEK INCOMPLETE ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH CONTROLLER OR UNIBUS TRANSFER ERROR - EM40
'OPI' ERROR - EM31
'PAR' ERROR - EM33
'WCF' ERROR - EM34
'IAE' ERROR - EM35
'WLE' ERROR - EM36
MISCELLANEOUS DRIVE ERROR - EM30

C. ERRORS NOT FLAGGED BY THE HARDWARE ERROR DETECTION LOGIC.

BUS ADDRESS OR WORD COUNT INCORRECT - EM41
DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42

1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596

CAN'T MATCH DATA READ WITH A PATTERN - EM43
ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER - EM44
ECC LOGIC FAILURE - EM45
BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46

8.2 DUAL PORT OPERATION

DUAL PORT OPERATION IS NEARLY IDENTICAL TO THE OPERATION DESCRIBED IN SECTION 8.1. THE DIFFERENCES ARE IN COMMAND SEQUENCE INITIATION AND COMMAND TERMINATION.

WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND THE RMCS1 REGISTER IS READ TO TEST THE 'DVA' BIT. IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, A DRIVE CLEAR COMMAND IS ISSUED TO THE DRIVE TO SET 'PORT REQUEST'. THE PROGRAM THEN CHECKS 'DVA' IN 'RMCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 15. SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 15. SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

AFTER A DRIVE HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE; IF THE OPERATION TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL ERROR PROCESSING HAS BEEN COMPLETED.

SINGLE PORT DRIVES, DRIVES WHICH ARE IN NEUTRAL BUT NOT BEING EXERCISED BY THE OPPOSITE PORT ARE STILL TREATED AS DUAL PORT DRIVES IN THAT A RELEASE COMMAND IS ISSUED AT THE END OF NORMAL COMMAND PROCESSING OR AT THE END OF ERROR PROCESSING. A RELEASE COMMAND ISSUED UNDER THESE CONDITIONS HAS NO FUNCTIONAL EFFECT ON THE OPERATION OF THE DRIVE.

8.3 SELECTION OF OPERATION VARIABLES

- A. SECTOR ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINSEC' AND 'MAXSEC'. TRACK ADDRESS SELECTION IS RANDOM BETWEEN THE VALUES IN 'MINTRK' AND 'MAXTRK'. CYLINDER ADDRESS SELECTION IS RANDOM BETWEEN 'MINCYL' AND 'MAXCYL'. IF A MINIMUM ADDRESS IS GREATER THAN THE CORRESPONDING MAXIMUM ADDRESS, THE PROGRAM WILL SWAP 'MAX' AND 'MIN' ADDRESSES AND CONTINUE.
- B. THE WORD COUNT IS RANDOMLY SELECTED BETWEEN 6 AND THE VALUE 'WRDCNT'. THIS IS NECESSARY AS THE PROGRAM REQUIRES 4 LOCATIONS IN THE DATA PORTION OF THE SECTOR TO BE ABLE TO MATCH THE DATA TO A PATTERN FOR DATA COMPARISON PURPOSES AND NEEDS 2 MORE

1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653

LOCATIONS IF A READ HEADER & DATA COMMAND IS ISSUED.

- C. THE DATA WRITTEN IS RANDOMLY SELECTED AMONG THE 15. STANDARD PATTERNS. THE PARAMETER 'PATTERN' ENABLES THE RANDOM PATTERN SELECTION, IF THIS PARAMETER IS 0.
- D. THE COMMANDS ARE SELECTED RANDOMLY. WRITE CHECK DATA COMMAND IS PERFORMED ONLY IF THE PREVIOUS COMMAND WAS THE APPROPRIATE WRITE DATA COMMAND.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE COMMAND IS SELECTED. THE ENTIRE BUFFER IS FILLED WITH THE SELECTED PATTERN. WHEN DATA IS READ FROM THE DISK, THE PROGRAM COMPARES DATA ON A SECTOR BASIS. IF THE PARAMETER 'PATTERN' IS 0 THE PROGRAM WILL ATTEMPT TO MATCH THE FIRST 4 DATA WORDS OF EACH SECTOR, TO ONE OF THE FOLLOWING PATTERNS. HOWEVER, IF THE PARAMETER 'PATTERN' IS NOT 0, THE PROGRAM WILL ASSUME THAT THE DESIRED DATA PATTERN IN LOCATION 'PATTERN' IS THE DATA TO LOOK FOR AND WILL NOT TRY TO MATCH ANY PATTERNS. THIS ALLOWS THE OPERATOR TO SCAN THE DISK FOR ANY SPECIFIC PATTERN.

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	*PAT 8
000001	177776	000000	133331	052525	155555	026455	155555
000003	177774	000000	133331	052525	155555	026455	133333
000007	177770	000000	133331	052525	155555	026455	155555
000017	177760	177777	133331	125252	155555	151322	133333
000037	177740	177777	133331	125252	155555	151322	155555
000077	177700	177777	133331	125252	155555	151322	133333
000177	177600	000000	133331	052525	155555	026455	155555
000377	177400	000000	133331	052525	155555	026455	133333
000777	177000	177777	133331	125252	155555	151322	155555
001777	176000	177777	133331	125252	155555	151322	133333
003777	174000	000000	133331	052525	155555	026455	155555
007777	170000	177777	133331	125252	155555	151322	133333
017777	160000	000000	133331	052525	155555	026455	155555
037777	140000	177777	133331	125252	155555	151322	133333
077777	100000	000000	133331	052525	155555	026455	155555
177777	000000	177777	133331	125252	155555	151322	133333
PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15	
000001	177776	172666	077777	153333	000000	177777	
000002	177775	155555	137777	066667	177777	000000	
000004	177773	172666	157777	153333	177777	000000	
000010	177767	155555	167777	066667	177777	000000	
000020	177757	172666	173777	153333	177777	000000	
000040	177737	155555	175777	066667	177777	000000	
000100	177677	172666	176777	153333	177777	000000	
000200	177577	155555	177377	066667	177777	000000	
000400	177377	172666	177577	153333	177777	000000	
001000	176777	155555	177677	066667	177777	000000	
002000	175777	172666	177737	153333	177777	000000	

1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710

```
004000 173777 155555 177757 066667 177777 000000
010000 167777 172666 177767 153333 177777 000000
020000 157777 155555 177773 066667 177777 000000
040000 137777 172666 177775 153333 177777 000000
100000 077777 155555 177776 066667 177777 000000
```

* WORST CASE PATTERN

9.1 RH/RM DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH/RM DRIVER.

9.2 TO INITIALIZE THE DRIVER:

```
JSR    PC,RMINIT
RETURN
```

UPON RETURN YOU MUST EXAMINE THE 'DRVSTA' TABLE TO DETERMINE THE DRIVES THAT ARE ONLINE FOR TESTING. THE 'DRVSTA' TABLE IS EIGHT BYTES; ONE BYTE PER DRIVE. THE STATE OF EACH DRIVE WILL BE INDICATED AS FOLLOWS:

DRVSTA	DRIVE STATE
-----	-----
>0	ONLINE
=0	OFFLINE, DRIVE IS NOT AN RM05/3/2, OR NONEXISTENT DRIVE
<0	UNSAFE

THE DRIVE TYPE IS DEFINED IN AN 8 BYTE LONG TABLE TAGGED 'DRVTYP'. THE TABLE CONTAINS ONE BYTE FOR EACH DRIVE AND IS INDEXED BY THE DRIVE NUMBER. ENTRIES ARE ENCODED AS FOLLOWS:

DRVTYP	CONDITION
-----	-----
0	NONEXISTENT DRIVE
4	RM03
5	RM02
7	RM05
-1	NOT AN RM05/3/2

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT16.

9.3 AFTER THE DRIVER HAS BEEN INITIALIZED, IT IS CALLED USING THE FOLLOWING SEQUENCE.

```
CALL:
JSR    RO,RM05           :MAKE THE CALL
PNTDPB :ADDRESS OF DPB*
RETURN1 :RETURN IF QUEUE IS FULL
RETURN2 :RETURN IF REQUEST IS IN
        :QUEUE OR THERE IS AN
        :ERROR CONDITION
```

1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767

*DPB (DATA PARAMETER BLOCK)

```
PNTDPB: .BYTE 0      ;(0) DRIVE NUMBER
        .BYTE 0      ;(1) OFFSET VALUE OR FMT16, ECT, AND HCI
        .BYTE 0      ;(2) COMMAND
        .BYTE 0      ;(3) PSEL AND A17 AND A16
        .WORD 0      ;(4) WORD COUNT (MUST BE NEG.)
        .WORD 0      ;(6) BUFFER ADDRESS OR
                    ;REGISTER TABLE POINTER
        .BYTE 0      ;(10) SECTOR ADDRESS OR
                    ;FIRST REG. INDEX
        .BYTE 0      ;(11) TRACK ADDRESS OR
                    ;LAST REG. INDEX
        .WORD 0      ;(12) CYLINDER ADDRESS
        .WORD 0      ;(14) ERROR TABLE POINTER
                    ;POINTS TO THE FIRST OF TWENTY
                    ;LOCATIONS OF WHERE THE DRIVER
                    ;IS TO STORE THE RH/RM
                    ;REGISTERS ON AN ERROR. IF LEFT
                    ;ZERO REGISTERS ARE NOT SAVED.
        .WORD 0      ;(16) STATUS/ERROR INDICATOR
                    ;BIT15=1=ERROR OCCURRED
                    ;BIT07=1=DONE
                    ;BIT14-BIT09 AND BIT06-BIT03
                    ;INDICATE TYPE OF ERROR
```

9.4 THE DRIVER PROVIDES A SOFTWARE TIMEOUT CAPABILITY.
TO UTILIZE THIS CAPABILITY YOU MUST SUPPLY THE 'RM TIMER' ROUTINE
WITH THE ELAPSED TIME IN THE FOLLOWING MANNER:

```
MOV     #16.,-(SP)    ;16. MILLISECONDS BETWEEN
                    ;CLOCK TICKS
JSR     PC,RMTMR     ;CALL THE TIMER ROUTINE
```

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE
CLOCK AND THE ELAPSED TIME MUST BE IN MILLISECONDS.

9.4.1 EXAMPLE - WRITE 1000. WORDS

```
1$:     JSR     RD,RM05    ;CALL THE DRIVER
        WRTDPB    ;DPB ADDRESS
BR      1$              ;WAIT FOR QUEUE IF FULL
2$:     TST     WRTDPB+16  ;WAIT FOR COMMAND TO COMPLETE
        BEQ     2$
        BMI     ERROR1    ;ERROR OCCURRED
        .
        .
        .
WRTDPB: .BYTE 5          ;DRIVE #5
        .BYTE 0
        .BYTE 161       ;WRITE COMMAND
        .BYTE 0
        .WORD -1000.    ;WORD COUNT
        .WORD WRTBUF    ;BUFFER ADDRESS
```

1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824

```
.BYTE 3 ;SECTOR
.BYTE 5 ;TRACK
.WORD 400 ;CYLINDER
.WORD ERRTB5 ;ERROR TABLE
.WORD 0 ;STATUS/ERROR INDICATOR
```

ALTERNATE DPB SETUP

```
WRTPB: .WORD 5 ;THIS SETUP ACHIEVED
        .WORD WRITE ;EVERYTHING THE
        .WORD -1000. ;ABOVE TABLE DID, BUT
        .WORD WRTPBF ;IN A CLEANER FORMAT
        .BYTE 3,5
        .WORD 400,ERRTB5,0
```

9.5 RH/RM REGISTERS

MNEMONIC	INDEX
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMDS	12
RMER1	14
RMAS	16
RMLA	20
RMDB	22
RMMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMHR	36
RMMR2	40
RMER2	42
RMEC1	44
RMEC2	46
RMBAE	50*
RMCS3	52*

* RH70 CONTROLLER REGISTERS

9.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N

1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881

SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHK HEADER & DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

N = HOUSEKEEPING
P = POSITIONING
D = DATA TRANSFER
S = SPECIAL PROVIDED BY THE DRIVER

9.7 DPB STATUS/ERROR INDICATOR WORD

THIS INDICATOR WILL INFORM THE USER OF THE RESULTS OF THE REQUEST. THIS IS ACCOMPLISHED BY SETTING VARIES BITS OF THE INDICATOR TO A ONE.

BIT NO.	MEANING IF ON A '1'
-----	-----
15	ERROR OCCURRED DONE (BIT07=0); BITS 14-9 SPECIFIES TYPE DONE (BIT07=1); BITS 6-3 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION
5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O
4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED

1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938

- 3(2) DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC 'RECALIBRATE' SEQUENCE
- 2 PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 15. SECONDS.
- 1 NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.

NOTES FOR ABOVE

- (1) = REQUEST WASN'T PUT IN QUEUE. (RH/RM REGISTERS WERE NOT SAVED)
- (2) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A 'DRIVE CLEAR' TO THE DRIVE. NOTE: ALL RH/RM REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE 'DRIVE CLEAR'.
- (3) = REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RM REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
- (4) = A 'RECALIBRATE' SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.

9.8 ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB. WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE AN ERROR CALL OF THE FORM 'ERROR N', WHERE 'N' IS THE ERROR NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.

N	TYPE	DATA AVAILABLE
---	----	-----
1	RH/RM INTERRUPT OCCURRED (RMAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ

1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954

4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMS RMERRS+2=RMR1 RMERRS+4=RMR2 RMERRS+6=RMR2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

a

1
55
56

;*LAST REVISION 04-APR-81

.TITLE CZRMUBO RM05/3/2 PERF EXER
;*COPYRIGHT (C) 1981
;*DIGITAL EQUIPMENT CORPORATION
;*COLORADO SPGS., CO. 80919

;*PROGRAM BY MIKE LEAVITT

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

57

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
13	INHIBIT ERROR TYPEOUTS
10	BELL ON ERROR
8	INHIBIT END OF PASS MESSAGES
7	DISPLAY ALL DATA COMPARE ERRORS
6	DON'T CHANGE PARAMETERS (LOOP ON PRESENT VALUES)
5	A. PARTIAL REGISTER DISPLAY IF ERROR B. NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
4	A. DO NOT CHECK FOR MAXIMUM ERROR COUNTS B. DO NOT DROP DRIVE AT END OF TEST
3	A. DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR B. DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER 28TH RETRY C. IF DATA COMPARE ERROR & SW07 SET, DISPLAY REMAINDER OF BUFFER
2	A. DO NOT TYPE UNIT STATUS AT PROGRAM START B. DO NOT TYPE PERFORMANCE REPORT AFTER SPECIFIED TIME
1	INHIBIT DATA COMPARE AFTER READ COMMAND, W/O ERROR
0	READ ONLY MODE

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004STACK = 1100
ERROR = EMT ;;BASIC DEFINITION OF ERROR CALL
SCOPE = IOT ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

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

;*GENERAL PURPOSE REGISTER DEFINITIONS

000000

R0 = %0 ;;GENERAL REGISTER

```

000001 R1 = %1 ;; GENERAL REGISTER
000002 R2 = %2 ;; GENERAL REGISTER
000003 R3 = %3 ;; GENERAL REGISTER
000004 R4 = %4 ;; GENERAL REGISTER
000005 R5 = %5 ;; GENERAL REGISTER
000006 R6 = %6 ;; GENERAL REGISTER
000007 R7 = %7 ;; GENERAL REGISTER
000006 SP = %6 ;; STACK POINTER
000007 PC = %7 ;; PROGRAM COUNTER
  
```

;*PRIORITY LEVEL DEFINITIONS

```

000000 PRO = 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
    
```

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

.SBTTL RM CONTROLLER REGISTERS

;CONTROL AND STATUS REGISTER 1 (RMCS1)

```

000100 IE = 100 ;:INTERRUPT ENABLE (BIT #6)
000200 RDY = 200 ;:READY (BIT #7)
000400 A16 = 400 ;:HIGH ORDER BUS ADDRESS BIT (BIT #8)
001000 A17 = 1000 ;:HIGH ORDER BUS ADDRESS BIT (BIT #9)
002000 PSEL = 2000 ;:PORT SELECT (BIT #10)
020000 MCPE = 20000 ;:MASSBUS PARITY ERROR (BIT #13)
040000 TRE = 40000 ;:TRANSFER ERROR (BIT #14)
;SC = 100000 ;:SPECIAL CONDITION (BIT #15)
    
```

;WORD COUNT REGISTER (RMWC)
 ;(EACH BIT IS CALLED BY BIT NUMBER)

;BUS ADDRESS REGISTER (RMBA)
 ;(EACH BIT IS CALLED BY BIT NUMBER)

;CONTROL AND STATUS REGISTER 2 (RMCS2)

```

000001 US1 = 1 ;:UNIT SELECT (BIT #0)
000002 US2 = 2 ;:UNIT SELECT (BIT #1)
000004 US4 = 4 ;:UNIT SELECT (BIT #2)
    
```

100	000010	BAI	= 10	:BUS ADDRESS INCREMENT INHIBIT (BIT #3)
101	000020	PAT	= 20	:MASSBUS PARITY TEST (BIT #4)
102	000040	CLR	= 40	:CLEAR (BIT #5)
103	000100	IR	= 100	:INPUT READY (BIT #6)
104	000200	OR	= 200	:OUTPUT READY (BIT #7)
105	000400	MDPE	= 400	:MASS BUS PARITY ERROR (BIT #8)
106	001000	MXF	= 1000	:MISSED TRANSFER ERROR (BIT #9)
107	002000	PGE	= 2000	:PROGRAM ERROR (BIT #10)
108	004000	NEM	= 4000	:NON EXISTENT MEMORY (BIT #11)
109	010000	NED	= 10000	:NON EXISTENT DRIVE (BIT #12)
110	020000	UPE	= 20000	:UNIBUS PARITY ERROR (BIT #13)
111	040000	WCE	= 40000	:WRITE CHECK ERROR (BIT #14)
112	100000	DLT	= 100000	:DATA LATE (BIT #15)
113				
114		:DATA BUFFER REGISTER (RMDB)		
115		:(EACH BIT IS CALLED BY BIT NUMBER)		
116				
117		.SBTTL RM REGISTERS		
118				
119		:CONTROL AND STATUS 1 REGISTER. (#00)		
120				
121	000001	GO	= 1	:GO BIT (BIT #0)
122	000002	F0	= 2	:FUNCTION CODE BIT #1
123	000004	F1	= 4	:FUNCTION CODE BIT #2
124	000010	F2	= 10	:FUNCTION CODE BIT #3
125	000020	F3	= 20	:FUNCTION CODE BIT #4
126	000040	F4	= 40	:FUNCTION CODE BIT #5
127	004000	DVA	= 4000	:DEVICE AVAILABLE (BIT #11)
128				
129		:DRIVE STATUS REGISTER (RMD51) (#01)		
130				
131	000001	OFFON	= 1	:OFFSET ON (BIT #0)
132	000100	VV	= 100	:VOLUME VALID (BIT #6)
133	000200	DRY	= 200	:DRIVE READY (BIT #7)
134	000400	DPR	= 400	:DRIVE PRESENT (BIT #8)
135	001000	PGM	= 1000	:PROGRAMABLE (BIT #9)
136	002000	LBT	= 2000	:LAST BLOCK TRANSFERRED (BIT #10)
137	004000	WRL	= 4000	:WRITE LOCK (BIT #11)
138	010000	MOL	= 10000	:MEDIUM ON-LINE (BIT #12)
139	020000	PIP	= 20000	:POSITIONING OPERATION IN PROGRESS (BIT #13)
140	040000	ERR	= 40000	:COMPOSITE ERROR (BIT #14)
141	100000	ATA	= 100000	:ATTENTION ACTIVE (BIT #15)
142				
143		:ERROR REGISTER #01 (RMER1) (#02)		
144				
145	000001	ILF	= 1	:ILLEGAL FUNCTION (BIT #0)
146	000002	ILR	= 2	:ILLEGAL REGISTER (BIT #1)
147	000004	RMR	= 4	:REGISTER MODIFICATION REFUSED (BIT #2)
148	000010	PAR	= 10	:PARITY ERROR (BIT #3)
149	000020	FER	= 20	:FORMAT ERROR (BIT #4)
150	000040	WCF	= 40	:WRITE CLOCK FAIL (BIT #5)
151	000100	ECH	= 100	:ECC HARD ERROR (BIT #6)
152	000200	HCE	= 200	:HEADER COMPARE ERROR (BIT #7)
153	000400	HCRC	= 400	:HEADER CRC ERROR (BIT #8)
154	001000	AOE	= 1000	:ADDRESS OVERFLOW ERROR (BIT #9)
155	002000	IAE	= 2000	:INVALID ADDRESS ERROR (BIT #10)
156	004000	WLE	= 4000	:WRITE LOCK ERROR (BIT #11)

```

157          010000          DTE      = 10000          ;DRIVE TIMING ERROR (BIT #12)
158          020000          OPI      = 20000          ;OPERATION INCOMPLETE (BIT #13)
159          040000          UNS      = 40000          ;DRIVE UNSAFE (BIT #14)
160          100000          DCK      = 100000         ;DATA CHECK ERROR (BIT 15)
161
162          ;MAINTAINABILITY REGISTER (RMMR1) (#03)
163
164
165          ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
166
167          000001          AT0      = 1          ;DEVICE 0 (BIT #0)
168          000002          AT1      = 2          ;DEVICE 1 (BIT #1)
169          000004          AT2      = 4          ;DEVICE 2 (BIT #2)
170          000010          AT3      = 10         ;DEVICE 3 (BIT #3)
171          000020          AT4      = 20         ;DEVICE 4 (BIT #4)
172          000040          AT5      = 40         ;DEVICE 5 (BIT #5)
173          000100          AT6      = 100        ;DEVICE 6 (BIT #6)
174          000200          AT7      = 200        ;DEVICE 7 (BIT #7)
175
176          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
177
178
179          ;DRIVE TYPE REGISTER (RMDT) (#06)
180
181          000001          DT00     = 1          ;DRIVE TYPE NUMBER BIT 1
182          000002          DT01     = 2          ;DRIVE TYPE NUMBER BIT 2
183          000004          DT02     = 4          ;DRIVE TYPE NUMBER BIT 3
184          000010          DT03     = 10         ;DRIVE TYPE NUMBER BIT 4
185          000020          DT04     = 20         ;DRIVE TYPE NUMBER BIT 5
186          000040          DT05     = 40         ;DRIVE TYPE NUMBER BIT 6
187          000100          DT06     = 100        ;DRIVE TYPE NUMBER BIT 7
188          000200          DT07     = 200        ;DRIVE TYPE NUMBER BIT 8
189          000400          DT08     = 400        ;DRIVE TYPE NUMBER BIT 9
190          004000          DRQ      = 4000       ;DRIVE REQUEST REQUIRED (BIT #11)
191          020000          MOH      = 20000      ;MOVING HEAD (BIT #13)
192          040000          TAP      = 40000      ;TAPE DRIVE (BIT #14)
193          100000          NSA      = 100000     ;NOT SECTOR ADDRESSED (BIT #15)
194
195          ;LOOK-AHEAD REGISTER (RMLA) (#07)
196
197          000100          SC1      = 100        ;SECTOR COUNT FIELD 0 (BIT #6)
198          000200          SC2      = 200        ;SECTOR COUNT FIELD 1 (BIT #7)
199          000400          SC04     = 400        ;SECTOR COUNT FIELD 2 (BIT #8)
200          001000          SC10     = 1000       ;SECTOR COUNT FIELD 3 (BIT #9)
201          002000          SC20     = 2000       ;SECTOR COUNT FIELD 4 (BIT #10)
202
203          ;SERIAL NUMBER REGISTER (RMSN) (#10)
204          ;(EACH IS CALLED BY BIT NUMBER)
205
206          ;OFFSET REGISTER (PMOF) (#11)
207
208          000001          OFFDIR   = 1          ;OFFSET DIRECTION
209          002000          HCI      = 2000       ;HEADER COMPARE INHIBIT (BIT #10)
210          004000          ECI      = 4000       ;ERROR CORRECTION CODE INHIBIT (BIT #11)
211          010000          FMT16    = 10000      ;FORMAT BIT (BIT #12)
212
213          ;DESIRED CYLINDER ADDRESS (RMDC) (#12)

```

```

214      ;(EACH BIT IS CALLED BY BIT NUMBER)
215
216      ;CURRENT CYLINDER ADDRESS (RMCC) (#13)
217      ;(REGISTER CURRENTLY NOT USED)
218
219      ;RM ERROR REGISTER #02 (RMER2) (#15)
220
221      000010      DPE      = 10      ;DATA PARITY ERROR (BIT #3)
222      000200      DVC      = 200     ;DEVICE CHECK (BIT #7)
223      002000      LBC      = 2000    ;LOSS OF BIT CLOCK (BIT #10)
224      004000      LSC      = 4000    ;LOSS OF SYSTEM CLOCK (BIT #11)
225      010000      IVC      = 10000   ;INVLAID COMMAND ERROR (BIT #12)
226      020000      OPE      = 20000   ;OPERATOR PLUG ERROR (BIT #13)
227      040000      SKI      = 40000   ;SEEK INCOMPLETE (BIT #14)
228      100000      BSE      = 100000  ;BAD SECTOR ERROR (BIT #15)
229
230      ;ECC POSITION REGISTER (RME(1) (#16)
231      ;(EACH BIT IS CALLED BY BIT NUMBER)
232
233      ;ECC PATTERN REGISTER (RME(2) (#17)
234      ;(EACH BIT IS CALLED BY BIT NUMBER)
235
236      .SBITL  RM DRIVER COMMANDS
237
238      000101      RNOP     = 101      ;NO OPERATION
239      000105      SEEK     = 105      ;SEEK
240      000107      RECAL    = 107      ;RECALIBRATE
241      000111      DRVCLR   = 111      ;DRIVE CLEAR
242      000113      RELSE    = 113      ;RELEASE
243      000115      OFFSET   = 115      ;OFFSET
244      000117      RTC      = 117      ;RETURN TO CENTER LINE
245      000121      READIN   = 121      ;READ IN PRESET
246      000123      ACK      = 123      ;PACK ACKNOWLEDGE
247      000131      SEARCH   = 131      ;SEARCH
248      000141      GETREG   = 141      ;GET REGISTERS
249      000143      SETFMT   = 143      ;SET FORMAT (& ECI OR HCI)
250      000145      SELDRV   = 145      ;SELECT DRIVE
251      000151      WCKD     = 151      ;WRITE CHECK DATA
252      000153      WCKHD    = 153      ;WRITE CHECK HEADER & DATA
253      000161      WRTDAT   = 161      ;WRITE DATA
254      000163      WRTHD    = 163      ;WRITE HEADER & DATA
255      000171      RDDAT    = 171      ;READ DATA
256      000173      RDHD     = 173      ;READ HEADER & DATA
257
258      176700      ABASE     = 176700
259      000254      AVECT1    = 254
260

```

```

.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      .=174
000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
000176      000000      SWREG:   .WORD 0          ;;SOFTWARE SWITCH REGISTER

.SBTTL STARTING ADDRESS(ES)

000200      000137      003644      JMP      @#START1          ;;JUMP TO STARTING ADDRESS OF PROGRAM
2
3 000204      000137      003634      JMP      @#START          ;CHANGE THE RH/RM ADDRESS
4
5
.SBTTL ACT11 HOOKS

;*****
;HOOKS REQUIRED BY ACT11
000210      $SVPC=.          ;SAVE PC
000046      000046      .=46
000046      031622      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
000052      000052      .=52
000052      040000      .WORD 40000      ;;2)SET LOC.52 TO 40000
000210      000210      .=$SVPC          ;; RESTORE PC

6
7      001100      .=1100
8
.SBTTL APT PARAMETER BLOCK

;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
000024      001100      .SX=.          ;;SAVE CURRENT LOCATION
000024      000024      .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000044      000200      200          ;;FOR APT START UP
000044      000044      .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044      001100      $APTHDR      ;;POINT TO APT HEADER BLOCK
001100      001100      .=$X          ;;RESET LOCATION COUNTER

;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDPT1 DIAGNOSTIC
;INTERFACE SPEC.

001100      $APTHD:
001100      000000      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102      001206      $MBADR: .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104      014234      $STMT:  .WORD 6300.      ;;RUN TIME OF LONGEST TEST
001106      014234      $PASTM: .WORD 6300.      ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110      014234      $UNITM: .WORD 6300.      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112      000032      .WORD $ETEND-$MAIL/2    ;;LENGTH MAILBOX-ETABLE (WORDS)
9
10      001114      TAB.XY:   .WORD $CMTAGSTARING ADDRESS

```

.SBTTL COMMON TAGS

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

001114	001114			SCMTAG: .=TAB.XY	::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			\$WR: .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	207	377	377	\$BELL: .ASCII <207><377><377>	::CODE FOR BELL
001202	077			\$QUES: .ASCII /?/	::QUESTION MARK
001203	015			\$CRLF: .ASCII <15>	::CARRIAGE RETURN
001204	012	000		\$LF: .ASCII <12>	::LINE FEED

.SBTTL APT MAILBOX-ETABLE

001206				.EVEN	
001206	000000			\$MAIL: .WORD	::APT MAILBOX
001210	000000			\$MSGTY: .WORD	::MESSAGE TYPE CODE
001212	000000			\$FATAL: .WORD	::FATAL ERROR NUMBER
001214	000000			\$TESTN: .WORD	::TEST NUMBER
001216	000000			\$PASS: .WORD	::PASS COUNT
001220	000000			\$DEVCT: .WORD	::DEVICE COUNT
001222	000000			\$UNIT: .WORD	::I/O UNIT NUMBER
001224	000000			\$MSGAD: .WORD	::MESSAGE ADDRESS
001226	000000			\$MSGLG: .WORD	::MESSAGE LENGTH
				\$ETABLE: .WORD	::APT ENVIRONMENT TABLE

001226 000
001227 000
001230 000000
001232 000000
001234 000000

\$ENV: .BYTE AENV
\$ENVM: .BYTE AENVM
\$SWREG: .WORD ASWREG
\$USWR: .WORD AUSWR
\$CPUOP: .WORD ACPUOP

::ENVIRONMENT BYTE
::ENVIRONMENT MODE BITS
::APT SWITCH REGISTER
::USER SWITCHES
::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

001236 000
001237 000

\$MAMS1: .BYTE AMAMS1
\$MTYP1: .BYTE AMTYP1

::HIGH ADDRESS, M.S. BYTE
::MEM. TYPE, BLK#1
MEM. TYPE BYTE -- (HIGH BYTE)
900 NSEC CORE=001
300 NSEC BIPOLAR=002
500 NSEC MOS=003

001240 000000

\$MADR1: .WORD AMADR1

::HIGH ADDRESS, BLK#1
MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE

001242 000
001243 000
001244 000000
001246 000
001247 000
001250 000000
001252 000
001253 000
001254 000000
001256 000254
001260 000000
001262 176700
001264 000000
001266 000000
001270 000000
001272

\$MAMS2: .BYTE AMAMS2
\$MTYP2: .BYTE AMTYP2
\$MADR2: .WORD AMADR2
\$MAMS3: .BYTE AMAMS3
\$MTYP3: .BYTE AMTYP3
\$MADR3: .WORD AMADR3
\$MAMS4: .BYTE AMAMS4
\$MTYP4: .BYTE AMTYP4
\$MADR4: .WORD AMADR4
\$VECT1: .WORD AVECT1
\$VECT2: .WORD AVECT2
\$BASE: .WORD ABASE
\$DEVN: .WORD ADEVN
\$CDW1: .WORD ACDW1
\$CDW2: .WORD ACDW2
\$ETEND:
.EXIT

::HIGH ADDRESS, M.S. BYTE
::MEM. TYPE, BLK#2
::MEM. LAST ADDRESS, BLK#2
::HIGH ADDRESS, M.S. BYTE
::MEM. TYPE, BLK#3
::MEM. LAST ADDRESS, BLK#3
::HIGH ADDRESS, M.S. BYTE
::MEM. TYPE, BLK#4
::MEM. LAST ADDRESS, BLK#4
:: INTERRUPT VECTOR#1, BUS PRIORITY#1
:: INTERRUPT VECTOR#2, BUS PRIORITY#2
:: BASE ADDRESS OF EQUIPMENT UNDER TEST
:: DEVICE MAP
:: CONTROLLER DESCRIPTION WORD#1
:: CONTROLLER DESCRIPTION WORD#2

.SBTTL USER DEFINED TAGS

001272	176700	\$RMADR:	.WORD	176700	;FIRST ADDRESS OF RH/RM REGISTERS
001274	000254	\$RMVEC:	.WORD	254	;VECTOR ADDRESS
001276	172540	\$LKCSR:	.WORD	172540	;ADDR OF KW11-P STATUS REGISTER
001300	172542	\$LKCSB:	.WORD	172542	;ADDR OF KW11-P COUNTER BUFFER
001302	000104	\$LPVEC:	.WORD	104	;ADDR OF KW11-P VECTOR
001304	177546	\$LKS:	.WORD	177546	;ADDR OF KW11-L STATUS REGISTER
001306	000100	\$LLVEC:	.WORD	100	;ADDR OF KW11-L VECTOR
001310	177777	PCLOCK:	.WORD	-1	; '0' IF KW11-P IS ON SYSTEM
001312	177777	CLKFLG:	.WORD	-1	; '0' IF A CLOCK IS AVAILABLE
001314	000074	HZ:	.WORD	60.	; 60. IF 60HZ SYSTEM, 50. IF 50HZ SYSTEM
001316	000000	STATIN:	.WORD	0	; 'TYPE STATISTICS' INDICATOR
001320	000000	PACK:	.WORD	0	; 'W' COMMAND INDICATOR
	001220	DRIVE	= \$UNIT		;DRIVE # STORAGE: ERRORS 1-5 & 10
					;SAME AS USED IN APT
001322	000000	ATTN:	.WORD	0	;ATTN REG STORAGE: ERRORS 1-5 & 10
001324	000000	UNIT:	.WORD	0	;DRIVE # STORAGE FOR PRINTOUT
001326	000000	MASK:	.WORD	0	;ERROR RETRY REGISTER MASK
001330	000	RETRY:	.BYTE	0,0	;ERROR RETRY LIMIT IN THE LOWER BYTE
					;RETRY COUNT IN THE UPPER BYTE
001332	000003	FAIRNS:	.WORD	3	;MAXIMUM TIME IN QUEUE VALUE
001334	000000	LSTAD:	.WORD	0	;STORE LAST MEMORY ADDRESS HERE
001336	000000	CHGADR:	.WORD	0	;CHANGE RH/RM UNIBUS ADDRESS FLAG
001340	000000	CFLAG:	.WORD	0	;IF = -1, 'CONTROL C' WAS FLAGGED
001342	000000	BADSEC:	.WORD	0	;BAD TRACK/SECTOR FLAG
001344	000000	HOUR:	.WORD	0	;HOUR COUNT STORED HERE
001346	000000	MINUTE:	.WORD	0	;MINUTE'S COUNT STORED HERE
001350	000000	SECOND:	.WORD	0	;SECOND'S COUNT STORED HERE
001352	000000	ONESEC:	.WORD	0	;TIMER ROUTINE COUNTER (FOR ONE SECOND)
001354	177777	ZROIND:	.WORD	-1	;ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001356	000	FRSTER:	.BYTE	0	;DATA COMPARE ERROR FLAG
					;IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
					;IF < 0, MISCOMPARISON FOUND
001357	000		.BYTE	0	;MISCOMPARISON OR CAN'T MATCH PATTERN FLAG
					;IF < 0, ERROR IN BUFFER
001360	000000	SAVER1:	.WORD	0	;SAVE R1 HERE
001362	000000	SAVER5:	.WORD	0	;SAVE R5 HERE
001364	000000	ERCTR:	.WORD	0	;NUMBER OF ERRORS
001366	000000	LIMIT:	.WORD	0	;DISPLAY LIMIT
001370	000000	CMCNT:	.WORD	0	;WORD COUNT
001372	000000	CMCYL:	.WORD	0	;CYLINDER ADDRESS
001374	000	CMSEC:	.BYTE	0	;SECTOR ADDRESS
001375	000	CMTRK:	.BYTE	0	;TRACK ADDRESS
001376	000000	ECBIT:	.WORD	0	;ERROR BURST BIT OFFSET
001400	000000	ECSEC:	.WORD	0	;ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001402	000000	ECMSK0:	.WORD	0	;CORRECTION MASK FOR FIRST ERROR WORD
001404	000000	ECMSK1:	.WORD	0	;CORRECTION MASK FOR SECOND ERROR WORD
001406	000000	ECWRD:	.WORD	0	;LOCATION OF FIRST ERROR WORD
001410	000000	ECGD:	.WORD	0	;GOOD DATA, FIRST WORD
001412	000000	ECBADO:	.WORD	0	;BAD DATA, FIRST WORD
001414	000000	ECWRD1:	.WORD	0	;LOCATION OF SECOND ERROR WORD
001416	000000	ECGD1:	.WORD	0	;GOOD DATA, SECOND WORD
001420	000000	ECBAD1:	.WORD	0	;BAD DATA, SECOND WORD
001422	001055	CYLIMT:	.WORD	557.	;CYLINDER ADDRESS LIMIT
001424	000036	SEFLMT:	.WORD	30.	;SECTOR ADDRESS LIMIT


```

001426 000015      TRKLMT: .WORD 13.      ;TRACK ADDRESS LIMIT
001430 000000      DRVPAR: .WORD 0        ;WHEN DRIVES ARE BEING ASSIGNED,
                                ;0=CHANGE DRIVE PARAMETERS
                                ;1=DO NOT CHANGE DRIVE PARAMETERS
001432 000000      XXDP:  .WORD 0        ;THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH
                                ;THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE
                                ;'XXDP' DEVICE CODE FOR THE RM05/3/2.
    
```

.SBTTL COMMON PARAMETERS

```

;THE FOLLOWING TWO LOCATIONS CONTAIN THE SOFT ERROR RATE WORDS USED TO
;DETERMINE END OF PASS WHEN THE PROGRAM IS DATA BIASED.
;IT WILL TAKE APPROXIMATELY 3.33 PASSES TO REACH THE SOFT ERROR RATE OF
;1 X 10^10 BITS (6.25 X 10^8 WORDS) READ OR 10. PASSES TO REACH THE 90%
;CONFIDENCE LEVEL OF 3 X 10^10 BITS (1.875 X 10^9 WORDS) READ.
;ENDCON= LSB AND ENDCON+2= MSB
    
```

```

001434 002740      ENDCON: .WORD 002740 ;(1.875 X 10^8 WORDS) OR (3 X 10^9 BITS) READ
001436 005455      .WORD 005455
    
```

```

;THE FOLLOWING TWO LOCATIONS CONTAIN THE SEEK ERROR RATE WORDS USED TO
;DETERMINE END OF PASS WHEN THE PROGRAM IS SEEK BIASED.
;IT WILL TAKE 1 PASS TO REACH A SEEK ERROR RATE OF 1 X 10^6 SEEKS OR 3
;PASSES TO REACH A 90% CONFIDENCE LEVEL OF 3 X 10^6 SEEKS.
;ENDSEK= LSB AND ENDSEK+2= MSB
    
```

```

001440 015200      ENDSEK: .WORD 015200 ;(1 X 10^6 SEEKS)
001442 000006      .WORD 000006
    
```

```

001444 000031      MAXER:  .WORD 25.      ;MAXIMUM ERRORS ALLOWED PER DRIVE
001446 000004      CMPLMT: .WORD 4        ;NUMBER OF COMPARE ERRORS TYPED OUT
001450 020000      WRDCNT: .WORD 8192.     ;MAXIMUM WORD COUNT (32. SECTORS)
001452 000000 000000 INTRVL: .WORD 0,0      ;FIRST WORD IS THE PERFORMANCE TIMEOUT INTERVAL
                                ;(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
001456 000001      PASSES: .WORD 1        ;NUMBER OF PASSES TO END OF TEST [THIS PARAMETER IS
                                ;NOT USED WHEN PROGRAM IS OPERATING IN AUTO RUN(CHAIN)
                                ;MODE].
    
```

```

001460 000000      PATTERN: .WORD 0        ;IF EQ 0,RANDOMLY SELECT DATA PATTERN
                                ;IF NOT EQ 0,SELECT ONE SET OF PATTERN
                                ;POINTED BY THE 'PATTERN'.
    
```

```

001462 000000      RANDWC: .WORD 0        ;IF EQ TO 0, GENERATE A RANDOM WORD COUNT
                                ;FOR THE OPERATION.
                                ;IF NOT EQ TO 0, USE THE VALUE IN 'WRDCNT' FOR
                                ;THE WORD COUNT
    
```

```

001464 000003      RATIO:  .WORD 3        ;READ/WRITE RATIO [RANGE 0 - 7]
                                ;0 - 15/1      (READ/WRITE)
                                ;1 - 7/1
                                ;2 - 6/2
                                ;3 - 5/3
                                ;4 - 4/4
                                ;5 - 3/5
                                ;6 - 2/6
                                ;7 - 1/7
    
```

```

001466 000001      ENDING:  .WORD 1        ;IF NOT EQ 0, END OF PASS DETERMINED
                                ;BY THE 'WORDS READ' COUNT. (2.5 X 10^8 WORDS)
                                ;IF EQ 0, END OF PASS DETERMINED
                                ;BY THE SEEK COUNT. (4 X 10^5 SEEKS)
    
```

```

001470 000001      WRTCHK: .WORD 1      ;IF NOT EQ 0, DO AN APPROPRIATE WRITE
                    ;CHECK AFTER EACH WRITE COMMAND.
                    ;IF EQ 0, SELECT WRITE CHECK COMMANDS
                    ;RANDOMLY.
001472 000001      MESSAGE: .WORD 1      ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
                    ;ASSOCIATED WITH OPERATOR SPECIFIED
                    ;BAD SECTOR AREAS.
                    ;IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO
                    ;THESE AREAS.
001474 000000      RANDOM: .WORD 0      ;IF EQ TO 0, RANDOMLY SELECT DATA BLOCK
                    ;ADDRESS. IF NOT EQU 0, SEQUENTIALLY
                    ;SELECT DATA BLOCK ADDRESS
001476 000000      BADBLK: .WORD 0      ;IF EQ TO 1, THE BAD SECTOR ENTRY TABLE WILL ALWAYS
                    ;BE INITIALIZED WHEN ASSIGNING A DRIVE; IF EQ TO 0,
                    ;THE BAD SECTOR ENTRY TABLE WILL ONLY BE INITIALIZED
                    ;IF THE PACK SERIAL NUMBER HAS CHANGED SINCE THE
                    ;LAST TIME IT WAS READ. (NOTE: IF THE SERIAL NO. HAS
                    ;CHANGED, THIS MOST LIKELY MEANS THAT THE PACK OR DRIVE
                    ;HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED)

```

.SBTTL VALUES FOR FIRST OPERATION

```

001500 000010      BEGPAT: .WORD 8.      ;STARTING PATTERN CODE [RANGE 1 - 15.]
001502 000004      BEGCOD: .WORD 4      ;STARTING COMMAND CODE [RANGE 0 - 5]
                    ;0 = WRITE CHECK DATA ('WCKD')
                    ;1 = WRITE CHECK HEADER & DATA ('WCHKHD' - NOT USED)
                    ;2 = WRITE DATA ('WRDAT')
                    ;3 = WRITE HEADER & DATA ('WRTHD' - NOT USED)
                    ;4 = READ DATA ('RDDAT')
                    ;5 = READ HEADER & DATA ('RDHD')
001504 000400      BEGWC: .WORD 256.      ;STARTING WRD CNT [RANGE 6 - WRDCNT]

```

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

```

;LIST OF DRIVES PERFORMING COMMANDS
001506 000000      ORDERQ: .WORD 0
001510 000000      .WORD 0
001512 000000      .WORD 0
001514 000000      .WORD 0
001516 000000      .WORD 0
001520 000000      .WORD 0
001522 000000      .WORD 0
001524 000000      .WORD 0
001526 000000      .WORD 0

001530 000000      ASNLST: .WORD 0      ;A BIT SET IS AN ASSIGNED DRIVE

;ADDRESSES OF DRIVES TO BE DEASSIGNED
001532 000000      DUNIT: .WORD 0
001534 000000      .WORD 0
001536 000000      .WORD 0
001540 000000      .WORD 0
001542 000000      .WORD 0
001544 000000      .WORD 0
001546 000000      .WORD 0
001550 000000      .WORD 0
001552 000000      .WORD 0

```


001764	000000	000000	.WORD	0,0
001770	000000	000000	.WORD	0,0
001774	000000	000000	.WORD	0,0
002000	000000	000000	.WORD	0,0
002004	000000	000000	.WORD	0,0
002010	000000	000000	.WORD	0,0
002014	000000	000000	.WORD	0,0
002020	000000	000000	.WORD	0,0
002024	000000	000000	.WORD	0,0
002030	000000	000000	.WORD	0,0
002034	000000	000000	.WORD	0,0
002040	000000	000000	.WORD	0,0

:DRIVE PARAMETER BLOCK(DPB) POINTER TABLE

002044	046202	BLKADR:	.WORD	DRIVE0	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
002046	050414		.WORD	DRIVE1	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
002050	052626		.WORD	DRIVE2	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
002052	055040		.WORD	DRIVE3	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
002054	057252		.WORD	DRIVE4	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
002056	061464		.WORD	DRIVE5	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
002060	063676		.WORD	DRIVE6	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
002062	066110		.WORD	DRIVE7	:ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7

:DRIVER COMMAND CONTROL TABLE (USED IN RM DRIVER)

002064	151	COMTBL:	.BYTE	WCKD	:WRITE CHECK DATA
002065	377		.BYTE	-1	:WRITE CHECK HEADER AND DATA (NOT USED)
002066	161		.BYTE	WRTDAT	:WRITE DATA
002067	377		.BYTE	-1	:WRITE HEADER AND DATA (NOT USED)
002070	171		.BYTE	RDDAT	:READ DATA
002071	173		.BYTE	RDHD	:READ HEADER AND DATA

:FUNCTION(COMMAND) CODE CONTROL TABLE

002072	004	OPTBL:	.BYTE	4	:SEEK
002073	006		.BYTE	6	:RECAL
002074	010		.BYTE	10	:DRIVE CLEAR
002075	012		.BYTE	12	:RELEASE
002076	014		.BYTE	14	:OFFSET
002077	016		.BYTE	16	:RETURN TO CENTERLINE
002100	020		.BYTE	20	:READIN PRESET
002101	022		.BYTE	22	:PACK ACKNOWLEDGE
002102	030		.BYTE	30	:SEARCH
002103	050		.BYTE	50	:WRITE CHECK DATA
002104	052		.BYTE	52	:WRITE CHECK HEADER AND DATA
002105	060		.BYTE	60	:WRITE DATA
002106	062		.BYTE	62	:WRITE HEADER AND DATA
002107	070		.BYTE	70	:READ DATA
002110	072		.BYTE	72	:READ HEADER AND DATA
002111	377		.BYTE	-1	:TERMINATOR

.EVEN

:MESSAGE CONTROL TABLE FOR 'OPTBL' TABLE

002112	123	105	105	MNTBL:	.ASCIZ	/SEEK	/
002122	122	105	103		.ASCIZ	/RECAL	/
002132	104	122	126		.ASCIZ	/DRVCLR	/
002142	122	105	114		.ASCIZ	/RELSE	/
002152	117	106	106		.ASCIZ	/OFFSET	/
002162	122	124	103		.ASCIZ	/RTC	/

002172	122	105	101	.ASCIZ	/READIN /	
002202	120	101	103	.ASCIZ	/PACK /	
002212	123	105	101	.ASCIZ	/SEARCH /	
002222	127	103	113	.ASCIZ	/WCKD /	
002232	127	103	113	.ASCIZ	/WCKHD /	
002242	127	122	124	.ASCIZ	/WRDAT /	
002252	127	122	124	.ASCIZ	/WRTHD /	
002262	122	104	104	.ASCIZ	/RDDAT /	
002272	122	104	110	.ASCIZ	/RDHD /	
002302	116	117	116	.ASCIZ	/NONE /	

002312	101	106	124	OFMSG0:	.ASCIZ	/AFTER RETRY WITHOUT OFFSET/
002345	101	124	040	OFMSG1:	.ASCIZ	/AT NEGATIVE OFFSET/
002370	101	124	040	OFMSG2:	.ASCIZ	/AT POSITIVE OFFSET/

.EVEN

002414	000			OFFCOD:	.BYTE	0	:OFFSET CODE TABLE
002415	001				.BYTE	1	:NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
002416	000				.BYTE	0	:NUMBER FOR POSITIVE OFFSET (DIR = IN)

.EVEN

002420	002312			OFMTBL:	.WORD	OFMSG0	:1ST OFFSET MESSAGE
002422	002345				.WORD	OFMSG1	:2ND OFFSET MESSAGE
002424	002370				.WORD	OFMSG2	:3RD OFFSET MESSAGE

.SBTTL DATA PATTERNS

:STANDARD DATA PATTERN POINTER TABLE

002426	002472	STNDAT:	.WORD	DATA0	:STANDARD DATA PATTERN 0
002430	002532		.WORD	DATA1	:STANDARD DATA PATTERN 1
002432	002572		.WORD	DATA2	:STANDARD DATA PATTERN 2
002434	002632		.WORD	DATA3	:STANDARD DATA PATTERN 3
002436	002672		.WORD	DATA4	:STANDARD DATA PATTERN 4
002440	002732		.WORD	DATA5	:STANDARD DATA PATTERN 5
002442	002772		.WORD	DATA6	:STANDARD DATA PATTERN 6
002444	003032		.WORD	DATA7	:STANDARD DATA PATTERN 7
002446	003072		.WORD	DATA8	:STANDARD DATA PATTERN 8
002450	003132		.WORD	DATA9	:STANDARD DATA PATTERN 9
002452	003172		.WORD	DATA10	:STANDARD DATA PATTERN 10
002454	003232		.WORD	DATA11	:STANDARD DATA PATTERN 11
002456	003272		.WORD	DATA12	:STANDARD DATA PATTERN 12
002460	003332		.WORD	DATA13	:STANDARD DATA PATTERN 13
002462	003372		.WORD	DATA14	:STANDARD DATA PATTERN 14
002464	003432		.WORD	DATA15	:STANDARD DATA PATTERN 15
002466	002472		.WORD	ZEROS	:ALL 0'S PATTERN
002470	003374		.WORD	ONES	:ALL 1'S PATTERN

002472		ZEROS:			
002472	000000	DATA0:	.WORD	0	:ALL 0'S DATA PATTERN
002474	000000		.WORD	0	
002476	000000		.WORD	0	
002500	000000		.WORD	0	
002502	000000		.WORD	0	
002504	000000		.WORD	0	
002506	000000		.WORD	0	
002510	000000		.WORD	0	
002512	000000		.WORD	0	

002514 000000
002516 000000
002520 000000
002522 000000
002524 000000
002526 000000
002530 000000

.WORD 0
.WORD 0
.WORD 0
.WORD 0
.WORD 0
.WORD 0
.WORD 0

002532 000001
002534 000003
002536 000007
002540 000017
002542 000037
002544 000077
002546 000177
002550 000377
002552 000777
002554 001777
002556 003777
002560 007777
002562 017777
002564 037777
002566 077777
002570 177777

DATA1: .WORD 000001 ;STANDARD PATTERN 1
.WORD 000003
.WORD 000007
.WORD 000017
.WORD 000037
.WORD 000077
.WORD 000177
.WORD 000377
.WORD 000777
.WORD 001777
.WORD 003777
.WORD 007777
.WORD 017777
.WORD 037777
.WORD 077777
.WORD 177777

002572 177776
002574 177774
002576 177770
002600 177760
002602 177740
002604 177700
002606 177600
002610 177400
002612 177000
002614 176000
002616 174000
002620 170000
002622 160000
002624 140000
002626 100000
002630 000000

DATA2: .WORD 177776 ;STANDARD PATTERN 2
.WORD 177774
.WORD 177770
.WORD 177760
.WORD 177740
.WORD 177700
.WORD 177600
.WORD 177400
.WORD 177000
.WORD 176000
.WORD 174000
.WORD 170000
.WORD 160000
.WORD 140000
.WORD 100000
.WORD 000000

002632 000000
002634 000000
002636 000000
002640 177777
002642 177777
002644 177777
002646 000000
002650 000000
002652 177777
002654 177777
002656 000000
002660 177777
002662 000000
002664 177777
002666 000000

DATA3: .WORD 000000 ;STANDARD PATTERN 3
.WORD 000000
.WORD 000000
.WORD 177777
.WORD 177777
.WORD 177777
.WORD 000000
.WORD 000000
.WORD 177777
.WORD 177777
.WORD 000000
.WORD 177777
.WORD 000000
.WORD 177777
.WORD 000000

002670	177777		.WORD	177777	
002672	133331	DATA4:	.WORD	133331	:STANDARD PATTERN 4
002674	133331		.WORD	133331	
002676	133331		.WORD	133331	
002700	133331		.WORD	133331	
002702	133331		.WORD	133331	
002704	133331		.WORD	133331	
002706	133331		.WORD	133331	
002710	133331		.WORD	133331	
002712	133331		.WORD	133331	
002714	133331		.WORD	133331	
002716	133331		.WORD	133331	
002720	133331		.WORD	133331	
002722	133331		.WORD	133331	
002724	133331		.WORD	133331	
002726	133331		.WORD	133331	
002730	133331		.WORD	133331	
002732	052525	DATA5:	.WORD	052525	:STANDARD PATTERN 5
002734	052525		.WORD	052525	
002736	052525		.WORD	052525	
002740	125252		.WORD	125252	
002742	125252		.WORD	125252	
002744	125252		.WORD	125252	
002746	052525		.WORD	052525	
002750	052525		.WORD	052525	
002752	125252		.WORD	125252	
002754	125252		.WORD	125252	
002756	052525		.WORD	052525	
002760	125252		.WORD	125252	
002762	052525		.WORD	052525	
002764	125252		.WORD	125252	
002766	052525		.WORD	052525	
002770	125252		.WORD	125252	
002772	155555	DATA6:	.WORD	155555	:STANDARD PATTERN 6
002774	155555		.WORD	155555	
002776	155555		.WORD	155555	
003000	155555		.WORD	155555	
003002	155555		.WORD	155555	
003004	155555		.WORD	155555	
003006	155555		.WORD	155555	
003010	155555		.WORD	155555	
003012	155555		.WORD	155555	
003014	155555		.WORD	155555	
003016	155555		.WORD	155555	
003020	155555		.WORD	155555	
003022	155555		.WORD	155555	
003024	155555		.WORD	155555	
003026	155555		.WORD	155555	
003030	155555		.WORD	155555	
003032	026455	DATA7:	.WORD	026455	:STANDARD PATTERN 7
003034	026455		.WORD	026455	
003036	026455		.WORD	026455	
003040	151322		.WORD	151322	

003042	151322	.WORD	151322
003044	151322	.WORD	151322
003046	026455	.WORD	026455
003050	026455	.WORD	026455
003052	151322	.WORD	151322
003054	151322	.WORD	151322
003056	026455	.WORD	026455
003060	151322	.WORD	151322
003062	026455	.WORD	026455
003064	151322	.WORD	151322
003066	026455	.WORD	026455
003070	151322	.WORD	151322

003072	155555	DATA8: .WORD	155555	; STANDARD PATTERN 8 (WORST CASE)
003074	133333	.WORD	133333	
003076	155555	.WORD	155555	
003100	133333	.WORD	133333	
003102	155555	.WORD	155555	
003104	133333	.WORD	133333	
003106	155555	.WORD	155555	
003110	133333	.WORD	133333	
003112	155555	.WORD	155555	
003114	133333	.WORD	133333	
003116	155555	.WORD	155555	
003120	133333	.WORD	133333	
003122	155555	.WORD	155555	
003124	133333	.WORD	133333	
003126	155555	.WORD	155555	
003130	133333	.WORD	133333	

003132	000001	DATA9: .WORD	000001	; STANDARD PATTERN 9
003134	000002	.WORD	000002	
003136	000004	.WORD	000004	
003140	000010	.WORD	000010	
003142	000020	.WORD	000020	
003144	000040	.WORD	000040	
003146	000100	.WORD	000100	
003150	000200	.WORD	000200	
003152	000400	.WORD	000400	
003154	001000	.WORD	001000	
003156	002000	.WORD	002000	
003160	004000	.WORD	004000	
003162	010000	.WORD	010000	
003164	020000	.WORD	020000	
003166	040000	.WORD	040000	
003170	100000	.WORD	100000	

003172	177776	DATA10: .WORD	177776	; STANDARD PATTERN 10
003174	177775	.WORD	177775	
003176	177773	.WORD	177773	
003200	177767	.WORD	177767	
003202	177757	.WORD	177757	
003204	177737	.WORD	177737	
003206	177677	.WORD	177677	
003210	177577	.WORD	177577	
003212	177377	.WORD	177377	
003214	176777	.WORD	176777	

003216	175777	.WORD	175777	
003220	173777	.WORD	173777	
003222	167777	.WORD	167777	
003224	157777	.WORD	157777	
003226	137777	.WORD	137777	
003230	077777	.WORD	077777	
003232	172666	DATA11: .WORD	172666	: STANDARD PATTERN 11
003234	155555	.WORD	155555	
003236	172666	.WORD	172666	
003240	155555	.WORD	155555	
003242	172666	.WORD	172666	
003244	155555	.WORD	155555	
003246	172666	.WORD	172666	
003250	155555	.WORD	155555	
003252	172666	.WORD	172666	
003254	155555	.WORD	155555	
003256	172666	.WORD	172666	
003260	155555	.WORD	155555	
003262	172666	.WORD	172666	
003264	155555	.WORD	155555	
003266	172666	.WORD	172666	
003270	155555	.WORD	155555	
003272	077777	DATA12: .WORD	077777	: STANDARD PATTERN 12
003274	137777	.WORD	137777	
003276	157777	.WORD	157777	
003300	167777	.WORD	167777	
003302	173777	.WORD	173777	
003304	175777	.WORD	175777	
003306	176777	.WORD	176777	
003310	177377	.WORD	177377	
003312	177577	.WORD	177577	
003314	177677	.WORD	177677	
003316	177737	.WORD	177737	
003320	177757	.WORD	177757	
003322	177767	.WORD	177767	
003324	177773	.WORD	177773	
003326	177775	.WORD	177775	
003330	177776	.WORD	177776	
003332	153333	DATA13: .WORD	153333	: STANDARD PATTERN 13
003334	066667	.WORD	066667	
003336	153333	.WORD	153333	
003340	066667	.WORD	066667	
003342	153333	.WORD	153333	
003344	066667	.WORD	066667	
003346	153333	.WORD	153333	
003350	066667	.WORD	066667	
003352	153333	.WORD	153333	
003354	066667	.WORD	066667	
003356	153333	.WORD	153333	
003360	066667	.WORD	066667	
003362	153333	.WORD	153333	
003364	066667	.WORD	066667	
003366	153333	.WORD	153333	
003370	066667	.WORD	066667	

.SBTT: ERROR POINTER TABLE

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

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

1	003472			\$ERRTB:	
2				:ERROR 1	
3					
4	003472	070412		EM1	;RH CONTROLLER INTERRUPT OCCURRED (RMAS 0)
5	003474	073063		DH1	
6	003476	073530		DT1	
7	003500	000000		0	
8					
9				:ERROR 2	
10					
11	003502	070464		EM2	;UNEXPECTED ATTENTION OCCURRED
12	003504	073070		DH2	
13	003506	073534		DT2	
14	003510	000000		0	
15					
16				:ERROR 3	
17					
18	003512	070522		EM3	;MASSBUS PARITY ERROR (MCPE 1)
19	003514	073144		DH3	
20	003516	073552		DT3	
21	003520	000000		0	
22					
23				:ERROR 4	
24					
25	003522	070560		EM4	;MASSBUS PARITY ERROR (PAR=1)
26	003524	073172		DH4	
27	003526	073562		DT4	
28	003530	000000		0	
29					
30				:ERROR 5	
31					
32	003532	070615		EM5	;ADDRESS PLUG BIT CHANGED
33	003534	073070		DH2	
34	003536	073534		DT2	
35	003540	000000		0	
36					
37				:ERROR 6	
38					
39	003542	070651		EM6	;RH CONTROLLER DIDN'T RESPOND TO ADDRESSING
40	003544	073231		DH6	
41	003546	073574		DT6	
42	003550	000000		0	

```

1
2
3 ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
3 003552 011600
4 003554 005740
5 003556 022626
6 003560 104401 003566
   003564 000417
   003624
7 003624 010046
8 003626 104402
9 003630 000240
10
11 003632 000404
12
13
14
15
16 003634 012737 177777 001336
17 003642 000407
18 003644 012737 000400 001336
19
   003652 000240
   003654 012737 000001 001212
20
21 003662 005227 000000
22 003666 001375
23 003670 000005
24
25
   .SBTTL START OF PROGRAM
15 START: MOV    #-1,CHGADR ;SET RH/RM ADDRESS CHANGE FLAG
16 BR      START2 ;START THE PROGRAM
18 START1: MOV   #400,CHGADR ;CLEAR THE RH/RM ADDRESS CHANGE FLAG
19 *****
   TST1:  NOP
        MOV   #1,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
21 START2: INC   #0 ;TTY LOOP, WAIT FOR INCREMENT
22 BNE    -4 ;OF WORD
23 RESET  ;CLEAR THE WORLD

   .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  #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  #176543,$HINUM ;:PRIME THE RANDOM NUMBER GENERATOR
   MOV  #123456,$LONUM ;:BOTH HIGH AND LOW WORDS
   :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
   RTI

```

```

004040 012737 000176 001154 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
004046 012737 000174 001156 MOV #DISPREG,DISPLAY
004054 012637 000004 66$: MOV (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR

004060 005037 001214 CLR $PASS ;;CLEAR PASS COUNT
004064 132737 000200 001227 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
004072 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
004074 012737 001230 001154 MOV #SWREG,SWR ;;NO,USE APT SWITCH REGISTER
004102 67$:
26 ;SETUP 'TIMEOUT' TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
27 004102 012737 003552 000004 MOV #BADTMO,ERRVEC ;SETUP FOR UNEXPECTED TIMEOUT
28 004110 012737 000300 000006 MOV #PR6,ERRVEC+2 ;LEVEL 6
29
30 .SBTTL TYPE PROGRAM NAME
;:TYPE THE NAME OF THE PROGRAM IF FIRST PASS
004116 005227 177777 INC #-1 ;;FIRST TIME?
004122 001031 BNE 68$ ;;BRANCH IF NO
004124 104401 004132 TYPE ,69$ ;;TYPE ASCIZ STRING
004130 000426 BR 68$ ;;GET OVER THE ASCIZ
;:69$: .ASCIZ <CRLF>@CZRMUBO - RM05/3/2 PERFORMANCE EXERCISER@<CRLF>
68$:
.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
004206 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
004212 001012 BNE 70$ ;;BRANCH IF YES
004214 123727 001226 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
004222 001406 BEQ 70$ ;;BRANCH IF YES
004224 023727 001154 000176 CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
004232 001005 BNE 71$ ;;BRANCH IF NO
004234 104406 GTSWR ;;GET SOFT-SWR SETTINGS
004236 000403 BR 71$
004240 112737 000001 001150 70$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
004246 71$:

31 ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
32 ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
33
34
35 004246 005037 001432 CLR XXDP ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
36 004252 122737 000016 000041 CMPB #16,@#41 ;;LOADED FROM AN RM05/3/2 ?
37 004260 001121 BNE 33$ ;;BR IF NOT
38 004262 013737 000040 001432 MOV @#40,XXDP ;;GET DEVICE INDICATOR AND NUMBER
39 004270 122737 000007 001432 CMPB #7,XXDP ;;IS IT A VALID NUMBER ?
40 004276 103002 BHIS 1$ ;;YES
41 004300 105037 001432 CLRB XXDP ;;NO, DEFAULT TO DRIVE 0
42 004304 005737 000042 1$: TST @#42 ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
43 004310 001425 BEQ 2$ ;;BR IF NEITHER
44 004312 104401 004320 TYPE ,73$ ;;TYPE ASCIZ STRING
004316 000412 BR 72$ ;;GET OVER THE ASCIZ
;:73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
72$:

45 004344 005046 CLR -(SP) ;;CLEAR WORD ON STACK
46 004346 113716 001432 MOVB XXDP,(SP) ;;GET DRIVE ADDRESS
47 004352 104403 TYPOS ;;TYPE THE ADDRESS
48 004354 001 .BYTE 1 ;;ONLY 1 CHARACTER
49 004355 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
50 004356 104401 001203 TYPE ,%CRLF ;;CR-LF
51 004362 000450 BR 3$ ;;GET NUMBER OF DRIVES
52

```

```

53 004364 005227 177777      2$:   INC      #-1          ;FIRST TIME THRU HERE ?
54 004370 001055              BNE      3$           ;NO
55 004372 104401 004400      TYPE     75$         ;;TYPE ASCIZ STRING
    004376 000410          BR       74$         ;;GET OVER THE ASCIZ
    ;:75$: .ASCIZ <CRLF>/TO TEST DRIVE /
    74$:
56 004420 005046              CLR      -(SP)        ;CLEAR WORD ON STACK
57 004422 113716 001432      MOVB    XXDP,(SP)    ;GET DRIVE ADDRESS
58 004426 104403              TYPOS   ;TYPE DRIVE ADDRESS
59 004430          001      .BYTE    1           ;ONLY 1 CHARACTER
60 004431          000      .BYTE    0           ;SUPPRESS LEADING ZEROS
01 004432 104401 004440      TYPE     76$         ;;TYPE ASCIZ STRING
    004436 000432          BR       3$         ;;GET OVER THE ASCIZ
    ;:76$: .ASCIZ /, HALT PROGRAM, CLEAR LOC. 40 AND RESTART PROGRAM./<CRLF>
    3$:
65 004524 004737 033226      JSR     PC,$TKINT    ;TURN ON THE KEYBOARD INTERRUPT
66 004530 105737 001226      TSTB   $ENV         ;RUN UNDER APT MODE
67 004534 001415              BEQ     5$           ;NO,DO NOT BOTHER
68 004536 105737 001256      ^STB   $VECT1       ;NEW VECTOR ?
69 004542 001403              BEQ     4$           ;NOT LOAD IF = 0
70 004544 113737 001256 001274  MOVB    $VECT1,$RMVEC ;NEW VECTOR
71 004552 005737 001262      TST    $BASE        ;NEW BASE ADDRESS ?
72 004556 001411              BEQ     6$           ;NO
73 004560 013737 001262 001272  MOV     $BASE,$RMADR ;NEW BASE ADDRESS
74 004566 000405              BR      6$
75
76 004570 105737 001150      5$:   TSTB   $AUTOB      ;RUNNING IN AUTO MODE ?
77 004574 001002              BNE     6$           ;YES
78 004576 004737 100566      JSR     PC,BUSADR   ;CHECK R4/RM BUS ADDRESS
84
85 004602 013737 001272 040154 6$:   MOV     $RMADR,RMADR ;LOAD ADDRESS INTO DRIVER
86 004610 013737 001274 040156  MOV     $RMVEC,RMVEC ;LOAD VECTOR INTO DRIVER
87 004616 005037 001316      CLR     STATIN      ;CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
88 004622 012705 001506      MOV     #ORDERQ,R5 ;START OF AREA TO CLEAR
89 004626 005025              CLR     (R5)+
90 004630 022705 002044      CMP     #BLKADR,R5  ;LOOK FOR END OF CLEAR AREA
91 004634 001374              BNE     7$           ;BR IF NOT FINISHED
92 004636 012706 001100      MOV     #STACK,SP   ;SETUP THE STACK POINTER
93 004642 005037 177776      CLR     PS          ;CLEAR THE PROCESSOR STATUS WORD
94 004646 013737 001314 001352  MOV     HZ,ONESEC    ;RESTORE ONE SECOND COUNTER VALUE
95 004654 005037 001344      CLR     HOUR        ;CLEAR THE HOUR'S COUNTER
96 004660 005037 001346      CLR     MINUTE      ;CLEAR THE MINUTE'S COUNTER
97 004664 005037 001350      CLR     SECOND      ;CLEAR THE SECOND'S COUNTER
98 004670 005037 001454      CLR     INTRVL+2    ;CLEAR INTERVAL COUNTER
99 004674 005037 001320      CLR     PACK        ;SET 'T' & CLEAR 'R' OR 'W' COMMAND FLAG
100 004700 005037 001340      CLR     CFLAG       ;CLEAR THE 'CONTROL C' FLAG
103 004704 005037 046324      CLR     DRIVE0+$FIRST ;RESET $FIRST FLAG FOR DRIVE 0
    004710 005037 050536      CLR     DRIVE1+$FIRST ;RESET $FIRST FLAG FOR DRIVE 1
    004714 005037 052750      CLR     DRIVE2+$FIRST ;RESET $FIRST FLAG FOR DRIVE 2
    004720 005037 055162      CLR     DRIVE3+$FIRST ;RESET $FIRST FLAG FOR DRIVE 3
    004724 005037 057374      CLR     DRIVE4+$FIRST ;RESET $FIRST FLAG FOR DRIVE 4
    004730 005037 061606      CLR     DRIVE5+$FIRST ;RESET $FIRST FLAG FOR DRIVE 5
    004734 005037 064020      CLR     DRIVE6+$FIRST ;RESET $FIRST FLAG FOR DRIVE 6
    004740 005037 066232      CLR     DRIVE7+$FIRST ;RESET $FIRST FLAG FOR DRIVE 7

```

104
106
107

;AUTO SIZE FOR RH70 CONTROLLER AND DETERMINE IF IT IS
 ;JUMPED FOR 22 OR 32 REGISTERS

```

108
109 004744 005037 040162          SIZE70: CLR      RHEXT          ;CLEAR RMBAE OFFSET
110 004750 042737 174000 001234    BIC      #174000,$CPUOP ;CLEAR CPU TYPE REGISTER
111 004756 013746 000004          MOV      ERRVEC,-(SP)  ;SAVE CONTENTS OF ERROR VECTOR
112 004762 012737 005034 000004    MOV      #2$,ERRVEC   ;SETUP 'TRAP' RETURN ADDRESS
113 004770 013700 001272          MOV      $RMADR,R0    ;GET RMCS1 ADDRESS
114 004774 062700 000050          ADD      #50,R0       ;GET REGISTER OFFSET FOR RH70
115 005000 012701 000012          MOV      #10.,R1     ;GET NUMBER OF REGISTERS TO CHECK
116 005004 005720          TST      (R0)+        ;TRAP IF NOT A VALID RMBAE
117 005006 005720          TST      (R0)+        ;TRAP IF NOT A VALID RMCS3
118 005010 012737 000050 040162    MOV      #50,RHEXT    ;LOAD OFFSET FOR RMBAE (22 REGISTER RH)
119 005016 005720          TST      (R0)+        ;TRAP IF NOT A VALID REGISTER
120 005020 005301          DEC      R1           ;DONE WITH ALL 32 REGISTERS ?
121 005022 001375          BNE      1$          ;BR IF NO
122 005024 012737 000074 040162    MOV      #74,RHEXT   ;LOAD OFFSET FOR RMBAE (32 REGISTER RH)
123 005032 000403          BR       3$          ;
124 005034 012716 005042          MOV      #3$, (SP)   ;SETUP RETURN ADDRESS
125 005040 000002          RTI
126
127 005042 013700 001272          3$:      MOV      $RMADR,R0 ;GET RMCS1 REGISTER
128 005046 013701 040162          MOV      RHEXT,R1    ;GET RMBAE REGISTER OFFSET
129 005052 001415          BEQ      4$          ;BR IF NONE
130 005054 060001          ADD      R0,R1       ;GET RMBAE REGISTER
131 005056 052710 001400          BIS      #A17!A16,(R0) ;SET EXTENDED ADDRESS BITS IN RMCS1
132 005062 022711 000003          CMP      #3,(R1)     ;ARE THE EXTENDED BITS SET IN RMBAE ?
133 005066 001007          BNE      4$          ;BR IF NO
134 005070 005011          CLR      (R1)        ;CLEAR EXTENDED ADDRESS BITS IN RMBAE
135 005072 033710 001400          BIT      A17!A16,(R0) ;ARE THE EXTEND BITS CLEAR IN RMCS1 ?
136 005076 001003          BNE      4$          ;BR IF NO
137 005100 052737 030000 001234    BIS      #BIT13!BIT12,$CPUOP ;SET THE 11/70 CPU TYPE CODE
138 005106 012637 000004          MOV      (SP)+,ERRVEC ;RESTORE CONTENTS OF ERROR VECTOR
139
141          ;ROUTINE TO DETERMINE BUFFER MAX WORD COUNT AND FUDGE PACK SERIAL NUMBER
142          ;TO ALLOW BAD SECTOR FILE(DEC144) TO BE READ FROM EACH DRIVE AT
143          ;LEAST ONE TIME.
144
145 005112 005227 177777          SIZMEM: INC      #-1          ;FIRST TIME THRU HERE ?
146 005116 001027          BNE      1$          ;BR IF NO
147 005120 012700 177777          MOV      #-1,R0      ;FUDGE MSB'S FOR INITIALIZING PACK S/N
148 005124 010037 046344          MOV      R0,DRIVE0+$PSNM ;INIT. S/N FOR DRIVE 0
149 005130 010037 050556          MOV      R0,DRIVE1+$PSNM ;INIT. S/N FOR DRIVE 1
150 005134 010037 052770          MOV      R0,DRIVE2+$PSNM ;INIT. S/N FOR DRIVE 2
151 005140 010037 055202          MOV      R0,DRIVE3+$PSNM ;INIT. S/N FOR DRIVE 3
152 005144 010037 057414          MOV      R0,DRIVE4+$PSNM ;INIT. S/N FOR DRIVE 4
153 005150 010037 061626          MOV      R0,DRIVE5+$PSNM ;INIT. S/N FOR DRIVE 5
154 005154 010037 064040          MOV      R0,DRIVE6+$PSNM ;INIT. S/N FOR DRIVE 6
155 005160 010037 066252          MOV      R0,DRIVE7+$PSNM ;INIT. S/N FOR DRIVE 7
156 005164 004737 100434          JSR      PC,$$SIZE   ;SEE HOW MUCH MEMORY ON SYSTEM
157 005170 013737 100564 001334    MOV      $LSTAD,LSTAD ;SAVE THE LAST ADDRESS
158 005176 012737 000001 001642 1$:      MOV      #1,BUFTBL   ;LOAD NUMBER OF BUFFERS
159 005204 012737 102072 001644    MOV      #ENDPGM,BUFTBL+2 ;STARTING ADDRESS OF BUFFER
160 005212 013737 001334 001646    MOV      LSTAD,BUFTBL+4 ;LAST ADDR TO BUFFER ALLOCATION TABLE
161 005220 023727 001334 160000    CMP      LSTAD,#160000 ;OVER 28K ?
162 005226 101403          BLOS    2$          ;NO
163 005230 012737 160000 001646    MOV      #160000,BUFTBL+4 ;XXDP MAX MEMORY 28K
164 005236 162737 102072 001646 2$:      SUB      #ENDPGM,BUFTBL+4 ;SUBTRACT PROGRAM SPACE
165 005244 000241          CLC                ;CLEAR THE 'C' BIT
    
```

```

161 005246 006037 001646 ROR BUFTBL+4 ;CONVERT TO WORD COUNT
162 005252 162737 000144 001646 SUB #100.,BUFTBL+4 ;SAVE ROOM FOR THE 'ABS' LOADER
163 005260 105737 001150 TSTB $AUTOB ;RUNNING IN AUTO MODE ?
164 005264 001403 BEQ 3$ ;BR IF NO
165 005266 162737 003000 001646 SUB #1536.,BUFTBL+4 ;SUBTRACT 'XXDP' LOADER SIZE
166 005274 023737 001450 001646 3$: CMP WRDCNT,BUFTBL+4 ;IS MAX WORD COUNT TOO LARGE ?
167 005302 003406 BLE 4$ ;BR IF NO
168 005304 013737 001646 001450 MOV B'FTBL+4,WRDCNT ;USE MAX AVAIL MEMORY AS MAX WRD CNT
169 005312 013737 001450 077362 MOV WRDCNT,PARLST+2 ;VALUE FOR THE PARAMETER TABLE
170 005320 4$:
171
172 ;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
173
174 005320 005737 037616 LKPAR: TST PWRFLG ;RETURNING FROM POWER FAIL ?
175 005324 001045 BNE SETVEC ;BRANCH IF YES
176
177 005326 005037 001340 1$: CLR CFLAG ;CLEAR CONTROL C FLAG
178 005332 105737 001150 TSTB $AUTOB ;RUNNING IN AUTO MODE ?
179 005336 001040 BNE SETVEC ;BR IF YES
180 005340 104401 077470 TYPE ,ASKPAR ;TYPE 'CHANGE PARAMETERS ?'
181 005344 104411 RDLIN ;READ THE ENTRY
182 005346 012600 MOV (SP)+,RO ;SAVE ADDRESS OF RESPONSE
183 005350 005737 001340 TST CFLAG ;CONTROL C TYPED ?
184 005354 001364 BNE 1$ ;BR IF YES
185 005356 105710 TSTB (RO) ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
186 005360 001427 BEQ SETVEC ;BR IF YES
187 005362 105760 000001 TSTB 1(RO) ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
188 005366 001006 BNE 2$ ;BR IF NO
189 005370 122710 000131 (MPB #'Y,(RO) ;WAS IT A 'Y' RESPONSE ?
190 005374 001406 BEQ ENTPR ;BR IF YES
191 005376 122710 000116 (MPB #'N,(RO) ;WAS IT A 'N' RESPONSE ?
192 005402 001416 BEQ SETVEC ;BR IF YES
193 005404 104401 076677 2$: TYPE ,BADENT ;TYPE BAD ENTRY MESSAGE
194 005410 000746 BR 1$ ;TRY AGAIN
195
196 005412 012703 077360 ENTPR: MOV #PARLST,R3 ;PARAMETER TABLE ADDRESS
197 005416 004737 030700 JSR PC,PARENT ;GET THE PARAMETER ENTRY
198 005422 023727 001450 000006 CMP WRDCNT,#6 ;IS THE 'WRDCNT' VALUE OK ?
199 005430 103003 BHIS SETVEC ;BR IF IT IS
200 005432 012737 000006 001450 MOV #6,WRDCNT ;SET 'WRDCNT' TO THE MINIMUM VALUE
201
202 ;DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
203 ;THE PROGRAM WILL USE. PROGRAM RETURN HERE ON POWER FAIL
204
205 005440 004737 023344 SETVEC: JSR PC,CKCLK ;START THE CLOCK
206 005444 004737 040166 JSR PC,RMINIT ;INITIALIZE THE RM DRIVER
207 005450 012737 177777 040114 MOV #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
208 005456 013704 040154 MOV RMADR,R4 ;GET RM/RH BASE ADDRESS
209 005462 012764 000040 000010 MOV #CLR,RMCS2(R4) ;CLEAR MASSBUS CONTROLLER
210 005470 005227 177777 INC #-1 ;FIRST TIME THRU ?
211 005474 001407 BEQ 1$ ;BR IF YES
212 005476 005737 037616 TST PWRFLG ;RETURNING FROM POWER FAIL ?
213 005502 001004 BNE 1$ ;BRANCH IF YES
214 005504 032777 000004 173442 BIT #SW02,@SWR ;TYPEOUT THE DRIVE STATUS TABLE ?
215 005512 001106 BNE 12$ ;BR IF NOT
216 005514 005037 037616 1$: CLR PWRFLG ;CLEAR POWER FAIL FLAG
217 005520 005004 CLR R4 ;DRIVE TABLE POINTER
    
```


218	005522	104401	076026		TYPE	,SYSTAT		;TYPE STATUS HEADING
219	005526	104401	001203	2\$:	TYPE	, \$CRLF		;CR-LF
220	005532	010446			MOV	R4,-(SP)		::SAVE R4 FOR TYPEOUT
	005534	104403			TYPOS			::TYPE DRIVE NUMBER
	005536	002			.BYTE	2		::GO TYPE--OCTAL ASCII
	005537	000			.BYTE	0		::TYPE 2 DIGIT(S)
221	005540	104401	075552		TYPE	,BLNKS4		::SUPPRESS LEADING ZEROS
222	005544	105764	040036		TSTB	DRVSTA(R4)		;TYPE 4 BLANKS
223	005550	100416			BMI	5\$;CHECK DRIVE'S STATUS
224	005552	001020			BNE	6\$;BR IF UNSAFE
225								;BR IF ONLINE
226	005554	105764	040046		TSTB	DRVSTYP(R4)		;SEE IF OFFLINE OR NONEXISTENT
227	005560	001404			REQ	3\$;BR IF NONEXISTENT
228	005562	100006			BPL	4\$;BR IF OFFLINE
229	005564	104401	075724		TYPE	,NOTRM		;DRIVE NOT AN RM05/3/2
230	005570	000451			BR	11\$;CHECK NEXT DRIVE
231								
232	005572	104401	075745	3\$:	TYPE	,NOTPRS		;DRIVE NOT PRESENT
233	005576	000446			BR	11\$;CHECK NEXT DRIVE
234								
235	005600	104401	075633	4\$:	TYPE	,UNTOFF		;DRIVE OFFLINE
236	005604	000416			BR	8\$;PRINT DRIVE TYPE
237								
238	005606	104401	076001	5\$:	TYPE	,NOTSAF		;DRIVE UNSAFE
239	005612	000413			BR	8\$;PRINT DRIVE TYPE
240								
241	005614	005737	001432	6\$:	TST	XXDP		;LOADED FROM THIS DEVICE ?
242	005620	001406			BEQ	7\$;BR IF NO
243	005622	123704	001432		CMPB	XXDP,R4		;LOADED FROM THIS DRIVE ?
244	005626	001003			BNE	7\$;BR IF NO
245	005630	104401	076011		TYPE	,LODEV		;TYPE 'LOAD DEVICE'
246	005634	000427			BR	11\$		
247	005636	104401	075644	7\$:	TYPE	,UNTON		;DRIVE ONLINE
248	005642	104401	075554	8\$:	TYPE	,BLNKS2		;TYPE 2 BLANKS
249	005646	012737	076051	005712	MOV	#\$RM03,10\$;ADDRESS OF RM03 MESSAGE
250	005654	122764	000004	040046	CMPB	#\$4,DRVSTYP(R4)		;IS DEVICE AN RM03 ?
251	005662	001412			BEQ	9\$;BR IF YES
252	005664	012737	076044	005712	MOV	#\$RM02,10\$;ADDRESS OF RM02 MESSAGE
253	005672	122764	000005	040046	CMPB	#\$5,DRVSTYP(R4)		;IS DEVICE AN RM02 ?
254	005700	001403			BEQ	9\$;BR IF YES
255	005702	012737	076056	005712	MOV	#\$RM05,10\$;ADDRESS OF RM05 MESSAGE
256								
257	005710	104401		9\$:	TYPE			;TYPE THE DRIVE TYPE MESSAGE
258	005712	000000		10\$:	.WORD	0		;MESSAGE ADDRESS HERE
259								
260	005714	005204		11\$:	INC	R4		;INCREMENT DRIVE NUMBER/TABLE POINTER
261	005716	020427	000010		CMP	R4,#8.		;FINISHED ?
262	005722	001301			BNE	2\$;BR IF NOT
263	005724	104401	001203		TYPE	, \$CRLF		;CR-LF
264	005730			12\$:				

```

1                                     ;INITIALIZE PROGRAM PARAMETERS FOR STARTUP
2
3 005730 004737 033226 STA: JSR PC,$TKINT ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
4 005734 012737 002740 MOV #002740,ENDCON ;INITIALIZE XFER COUNT(LSB)
5 005742 012737 005455 MOV #005455,ENDCON+2
6 005750 105737 001226 TSTB $ENV ;APT SCRIPT MODE, THEN MAKE IT RUN 2 MIN.
7 005754 001411 BEQ 1$ ;NO
8 005756 012737 000001 MOV #1,RATIO ;SPEED UP TEST
9 005764 012737 077777 MOV #77777,ENDCON ;INITIALIZE QUICK XFER COUNT(LSB)
10 005772 012737 000027 MOV #27,ENDCON+2
11
12 006000 105737 001150 1$: TSTB $AUTOB ;RUNNING IN AUTO MODE ?
13 006004 001003 BNE 2$ ;BR IF YES
14 006006 005737 001336 TST CHGADR ;START AT 200 ?
15 006012 00345C BLE 8$ ;NO
16
17 006014 005001 2$: CLR R1 ;DRIVE #
18 006016 005002 CLR R2 ;AVAIL TABLE INDEX
19 006020 005003 CLR R3 ;DRIVE# * 2
20 006022 005737 0C1432 3$: TST XXDP ;LOADED FROM THIS DEVICE ?
21 006026 001403 BEQ 4$ ;BR IF NO
22 006030 123701 001432 CMPB XXDP,R1 ;LOADED FROM THIS DRIVE ?
23 006034 001426 BEQ 7$ ;BR IF YES
24 006036 105761 040036 4$: TSTB DRVSTA(R1) ;DRIVE ON LINE ?
25 006042 003423 BLE 7$ ;NO
26 006044 016300 002044 MOV BLKADR(R3),R0 ;LOAD DPB ADDRESS
27 006050 004737 026602 JSR PC,CLRDPB ;CLEAR DPB BLOCK
28 006054 004737 027420 JSR PC,GETID ;GET DRIVE (MBA) SERIAL NUMBER
29 006060 004537 027520 JSR R5,GETADR ;RETRIEVE BAD SECTOR FILE
30 006064 010062 001554 MOV R0,NEWUNT(R2) ;LOAD DPB ADDRESS TO ABAIL QUEUE
31 006070 004737 027020 JSR PC,DRVPRM ;SETUP DRIVE PARAMETER LIMITS
32 006074 005060 000122 CLR $FIRST(R0) ;RESET $FIRST FLAG FOR FIRST 204 START
33 006100 112760 177776 0C0026 MOVB #-2,$PACK(R0) ;SETUP COMMAND 'WT' (WRITE DATA AND TEST)
34 006106 004737 017140 JSR PC,WRTPK ;SETUP INITIAL PARAMETERS
35
36 006112 022322 7$: CMP (R3)+,(R2)+ ;INCREMENT INDEX
37 006114 005201 INCL R1 ;NEXT DRIVE
38 006116 020127 000007 CMP R1,#7 ;ALL DRIVES ASSIGNED ?
39 006122 003737 BLE 3$ ;NO
40 006124 005037 001336 CLR CHGADR ;CLEAR START FLAG
41 006130 000137 006306 JMP MAIN1 ;JUMP TO WAIT PARAMETER AND BUFFER LOOP
42
43 006134 012737 000001 001340 8$: MOV #1,CFLAG ;DUMMY 'CONTROL C' FLAG
    
```

```

1          .SBTTL  MAIN PROGRAM
2
3 006142  005737  001340  MAIN:  TST      CFLAG      ;KEYBOARD INTERRUPTED ?
4 005146  001407                BEQ      3$           ;BR IF NOT
5 006150  005737  001506  1$:   TST      ORDERQ     ;ANY DRIVES IN ORDER QJE ?
6 006154  001402                BEQ      2$           ;BR IF NO, ELSE
7 006156  000137  007006  JMP      IDLE          ;LET ALL DRIVES FINISH ORDER
8 006162  004737  025240  2$:   JSR      PC,KSR    ;SERVICE THE KEYBCARD
9 006166  000240                3$:   NOP              ; !! FOR DEBUGGING !!
10
11          ;CHECK FOR DRIVES TO BE DROPPED
12
13 006170  012703  000010  MAINDA: MOV     #8.,R3     ;DRIVE COUNTER
14 006174  012705  001532  MOV     #DUNIT,R5     ;ADDRESS OF 'DROP DRIVE' TABLE
15 006200  005715  1$:   TST      (R5)      ;SEE IF ENTRY AT PRESENT POSITION
16 006202  001004                BNE     3$           ;BR IF THERE IS ONE
17 006204  005725  2$:   TST      (R5)+     ;INCREMENT TO NEXT TABLE POSITION
18 006206  005303                DEC     R3           ;DECREMENT DRIVE COUNTER
19 006210  001373                BNE     1$          ;BR IF MORE TO CHECK
20 006212  000435                BR     MAIN1        ;GO CHECK FOR NEW ASSIGNED DRIVES
21
22 006214  012701  001576  3$:   MOV     #AVAIL,R1   ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
23 006220  005711  4$:   TST      (R1)      ;IF AT END OF 'AVAIL' TABLE ?
24 006222  001404                BEQ     5$           ;BR IF YES
25 006224  021115                CMP     (R1),(R5)   ;IS DRIVE IN 'AVAIL' THE TABLE ?
26 006226  001412                BEQ     7$           ;BR IF YES
27 006230  005721                TST    (R1)+       ;NO, INCREMENT 'AVAIL' TABLE ADDRESS
28 006232  000772                BR     4$          ;AND CONTINUE LOOKING
29
30 006234  012701  001620  5$:   MOV     #WAIT,R1     ;ADDRESS OF THE 'WAIT' BUFFER TABLE
31 006240  005711  6$:   TST      (R1)      ;AT THE END OF 'WAIT' TABLE ?
32 006242  001760                BEQ     2$           ;BR IF YES
33 006244  021115                CMP     (R1),(R5)   ;IS DRIVE IN THE 'WAIT' TABLE ?
34 006246  001402                BEQ     7$           ;BR IF YES
35 006250  005721                TST    (R1)+       ;NO, INCREMENT 'WAIT' TABLE ADDRESS
36 006252  000772                BR     6$          ;AND CONTINUE LOOKING
37
38 006254  011100  7$:   MOV     (R1),R0      ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
39 006256  104401  001203  TYPE   .$CRLF         ;CR-LF
40 006262  104401  076341  TYPE   ,DEASSG       ;TYPE 'DRIVE DEASSIGNED'
41 006266  004737  023636  JSR    PC,SUMMARY    ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
42 006272  104401  076155  TYPE   ,STARS        ;TYPE '****...ETC'
43 006276  005015                CLR    (R5)         ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
44 006300  004737  020700  JSR    PC,COMPRES    ;COMPRESS THE RESPECTIVE TABLE
45 006304  000737                BR     2$          ;SEE IF ANY MORE DRIVES
46
47          ;LOOK FOR DRIVES TO BE ASSIGNED
48
49 006306  012703  000010  MAIN1: MOV     #8.,R3     ;DRIVE COUNT
50 006312  005001                CLR    R1           ;ASSIGN LIST INDEX
51 006314  005002                CLR    R2           ;'AVAIL' INDEX
52 006316  005005                CLR    R5           ;NEW DRIVE INDEX
53 006320  005765  001554  1$:   TST      NEWUNT(R5)   ;NEW DRIVE IN THIS POSITION
54 006324  001005                BNE     3$           ;BR IF THERE IS
55 006326  005725  2$:   TST      (R5)+     ;INCREMENT R5
56 006330  005201                INC    R1           ;INCREMENT ASSIGN INDEX
57 006332  005303                DEC    R3           ;DECREMENT DRIVE COUNT

```

```

58 006334 001371          BNE 1$          ;BR IF MORE DRIVES
59 006336 000432          BR  MAIN2        ;START OPERATIONS FOR THE AVAILABLE DRIVES
60
61 006340 1044J1 001203    3$:  TYPE  ,SCLF      ;CR-LF
62 006344 104401 075625    TYPE  ,UN*MSG     ;'DRIVE'
63 006350 010146          MOV   R1,-(SP)    ;;SAVE R1 FOR TYPEOUT
                                ;;TYPE DRIVE NUMBER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 2 DIGIT(S)
                                ;;SUPPRESS LEADING ZEROS
                                ;'STARTED'
        006352 104403          TYPOS
        006354 002          .BYTE 2
        006355 000          .BYTE 0
64 006356 104401 076411    TYPE  ,ASGND
65 006362 005762 001576    4$:  TST  AVAIL(R2) ;AT END OF AVAILABLE TABLE
66 006366 001402          BEQ  5$          ;BR IF YES
67 006370 005722          TST  (R2)+       ;INCREMENT AVAILABLE TABLE INDEX
68 006372 000773          BR   4$          ;CONTINUE LOOKING FOR END OF TABLE
69 006374 016562 001554 001576 5$:  MOV  NEWUNT(R5),AVAIL(R2) ;MOVE ADDR OF DRIVE INTO AVAIL LST
70 006402 005065 001554          CLR  NEWUNT(R5)  ;TAKE DRIVE OUT OF NEW DRIVE TABLE
71 006406 156137 040142 001530  BISB ATABIT(R1),ASNLST ;SET DRIVE ASSIGNED INDICATOR
72 006414 005037 031662          CLR  AUTLST     ;CLEAR AUTO ASSIGN
73 006420 005722          TST  (R2)+       ;INCREMENT AVAILABLE TABLE POINTER
74 006422 000741          BR   2$          ;LOOK FOR MORE DRIVES
75
76                          ;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
77                          ;THE 'AVAILABLE' QUEUE
78
79 006424 005002          MAIN2: CLR  R2          ;START FROM THE FIRST LOCATION
80 006426 105737 001530          TSTB ASNLST      ;ANY DRIVES ACTIVE ?
81 006432 001025          BNE  2$          ;BR IF YES
82 006434 105737 001150          TSTB $AUTOB     ;RUNNING IN AUTO MODE ?
83 006440 001020          BNE  1$          ;BR IF YES
84 006442 012737 000001 001340  MOV  #1,CFLAG    ;DUMMY 'CONTROL C' FLAG
85 006450 013737 001314 001352  MOV  HZ,ONESEC   ;RESTORE ONE SECOND COUNTER VALUE
86 006456 005037 001344          CLR  HOUR        ;CLEAR THE HOUR'S COUNTER
87 006462 005037 001346          CLR  MINUTE      ;CLEAR THE MINUTE'S COUNTER
88 006466 005037 001350          CLR  SECOND      ;CLEAR THE SECOND'S COUNTER
89 006472 005037 001454          CLR  INTRVL+2    ;CLEAR INTERVAL COUNTER
90 006476 104401 077120          TYPE  ,NODRVS    ;TYPE 'NO DRIVES ASSIGNED'
91 006502 000137 031612          1$:  JMP  $GET42     ;GIVE CONTROL TO MONITOR
92
93 006506 005762 001620          2$:  TST  WAIT(R2)   ;ANY DRIVES WAITING FOR THE BUFFER ?
94 006512 001435          BEQ  5$          ;BR IF NO
95 006514 016200 001620          MOV  WAIT(R2),R0 ;LOAD R0 WITH THE DPB ADDRESS
96 006520 005046          CLR  -(SP)      ;CLEAR THE STACK FOR BUFFER REQ
97 006522 004737 016402          JSR  PC,GETBUF  ;CALL TO GET THE BUFFER RT.
98 006526 012660 000006          MOV  (SP)+,$BUF(R0) ;IF 0,BUFFER IS STILL NOT AVAILABLE
99 006532 001423          BEQ  4$          ;BRANCH IF NO BUFFER AVAILABLE
100 006534 005060 000120          CLR  $NEXT(R0)   ;CLEAR PARAMETER SELECT FLAG
101 006540 005060 000106          CLR  $FAIR(R0)  ;CLEAR THE FAIR FLAG
102 006544 004737 016772          JSR  PC,FILBUF  ;FILL THE BUFFER
103 006550 004737 017050          JSR  PC,GODRIV  ;SET COMMAND AND GO
104 006554 012705 001506          MOV  #ORDERQ,R5 ;PUT THE WAIT QUE INTO ORDER QUE
105 006560 005725          3$:  TST  (R5)+       ;QUE AVAILABLE ?
106 006562 001376          BNE  3$          ;BR IF NO
107 006564 010045          MOV  R0,-(R5)   ;LOAD THE DPB ADDRESS INTO THE ORDER QUE
108 006566 012701 001620          MOV  #WAIT,R1   ;REMOVE THE DRIVE FROM THE 'WAIT' QUE
109 006572 060201          ADD  R2,R1      ;OFFSET THE QUE POSITION
110 006574 004737 020700          JSR  PC,COMPRES ;COMPRESS THE QUE

```

```

111 006600 000742          BR      2$          ;BRANCH IF DONE
112 006602 005722      4$:  TST      (R2)+        ;CHECK THE NEXT QUE
113 006604 000740          BR      2$          ;LOOPING BACK
114
115 006606 005737 001506  5$:  TST      ORDERQ        ;ANY OUTSTANDING ORDERS ?
116 006612 001075          BNE     IDLE         ;BR IF YES
117
118 006614 005002          CLR     R2           ;CLEAR DRIVE TABLE POINTER
119 006616 005762 001576  6$:  TST      AVAIL(R2)        ;ANY DRIVES WAITING FOR PARAMETERS
120 006622 001002          BNE     7$          ;BRANCH IF ANY
121 006624 000137 007006          JMP     IDLE         ;BRANCH IF NONE
122 006630 015200 001576  7$:  MOV     AVAIL(R2),R0    ;CONTROL BLOCK ADDR IN R0
123 006634 005760 000120          TST     $N:*(R0)     ;PARAMETERS BEEN SELECTED ?
124 006640 001010          BNE     9$          ;BR IF THEY HAVE
125 006642 105760 000026          TSTB   $PACK(R0)    ;'R' OR 'W' COMMAND FOR THIS DRIVE ?
126 006646 001403          BEQ    8$          ;BR IF NO
127 006650 004737 017140          JSR    PC,WRTPK     ;GET DATA PARAMETERS
128 006654 000404          BR     10$         ;GET THE BUFFER
129
130 006656 004737 017470      8$:  JSR    PC,GENPAR    ;GO GENERATE THE PARAMETERS
131 006662 004737 020430      9$:  JSR    PC,LOLPAR    ;LOAD THE PARAMETERS JUST GENERATED
132 006666 005046      10$:  CLR     -(SP)        ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
133 006670 004737 016402          JSR    PC,GETBUF    ;GET BUFFER
134 006674 012660 000006          MOV     (SP)+,$BUF(R0) ;MOVE BUFFER ADDR TO DPB
135 006700 001416          BEQ    12$         ;BR IF '0' ADDR (NO BUFFER)
136 006702 004737 016772          JSR    PC,FILBUF    ;FILL THE BUFFER
137 006706 005060 000120          CLR     $NEXT(R0)   ;CLEAR PARAMETER SELECT FLAG
138 006712 005060 000106          CLR     $FAIR(R0)   ;CLEAR THE 'FAIRNESS' COUNT
139 006716 004737 017050          JSR    PC,GODRIV    ;PUT CURRENT DPB IN DRIVER
140 006722 012705 001506          MOV     #ORDERQ,R5 ;ADDRESS OF ORDER QUE IN R5
141 006726 005725      11$:  TST     (R5)+        ;END OF QUE ?
142 006730 001376          BNE    11$        ;BR IF NOT
143 006732 010045          MOV     R0,-(R5)    ;PUT BLOCK ADDRESS INTO QUE
144 006734 000416          BR     15$        ;CONTINUE LOOKING
145
146 006736 005260 000106      12$:  INC     $FAIR(R0)    ;INCREMENT THE FAIR COUNT
147 006742 022760 000003 000106  CMP     #3,$FAIR(R0) ;THREE TIMES,BUFFER IS NOT AVAILABLE?
148 006750 101006          BHI    14$        ;BRANCH IF NOT OVER THREE TIMES
149 006752 012705 001620          MOV     #WAIT,R5    ;LOAD INTO THE WAIT QUE
150 006756 005725      13$:  TST     (R5)+        ;AN AVAILABLE LOCATION ?
151 006760 001376          BNE    13$        ;BRANCH IF NOT
152 006762 010045          MOV     R0,-(R5)    ;LOAD INTO WAIT QUE
153 006764 000402          BR     15$        ;REMOVE THE DPB FROM AVAILABLE QUE
154
155 006766 005722      14$:  TST     (R2)+        ;INCREMENT INDEX
156 006770 000712          BR     6$          ;BRANCH BACK TO FIRE NEXT DRIVE
157 006772 012701 001576  15$:  MOV     #AVAIL,R1   ;'AVAILABLE' TABLE ADDRESS
158 006776 060201          ADD     R2,R1       ;FORM ADDRESS OF LAST ENTRY
159 007000 004737 020700          JSR    PC,COMPRES   ;COMPRESS THE TABLE
160 007004 000704          BR     6$          ;CONTINUE LOOKING
161
162          ;WAIT FOR A COMMAND TO FINISH
163
164 007006 012701 001506  IDLE:  MOV     #ORDERQ,R1 ;ADDRESS OF THE ORDER QUE IN R1
165 007012 012100      1$:  MOV     (R1)+,R0    ;PUT BLOCK ADDRESS INTO R0
166 007014 001431          BEQ    4$          ;BR IF END OF QUE
167 007016 005760      2$:  TST     $STATUS(R0) ;SEE IF DRIVE FINISHED

```

168	007022	001775				BEQ	2\$;BR IF DRIVE NOT FINISHED
169	007024	005741				TST	-(R1)		;BACKUP THE QUE POINTER
170	007026	010146				MOV	R1,-(SP)		;SAVE THE QUE ADDRESS
171	007030	004737	016232			JSR	PC,STATIS		;ACCUMULATE STATISTICS FOR DRIVE IN RO
172	007034	004737	007150			JSR	PC,PROCES		;PROCESS END OF COMMAND
173	007040	005037	001342			CLR	BADSEC		;CLEAR THE BAD TRK/SEC ERROR INDICATOR
174	007044	004737	031134			JSR	PC,ABNRML		;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
175	007050	004737	031162			JSR	PC,\$EOP		;IS IT END OF PASS ?
176	007054	012601				MOV	(SP)+,R1		;RESTORE THE ORDER TABLE INDEX
177	007056	012705	001576			MOV	#AVAIL,R5		;ADDRESS OF THE 'AVAIL' TABLE
178	007062	005725		3\$:		TST	(R5)+		;IS IT THE END OF THE 'AVAIL' TABLE ?
179	007064	001376				BNE	3\$;BR IF NO
180	007066	011145				MOV	(R1),-(R5)		;MOVE THE DPB INTO THE 'AVAIL' TABLE
181	007070	004737	020700			JSR	PC,COMPRES		;COMPRESS THE ORDER QUE
182	007074	004737	015536			JSR	PC,RELBUF		;RESTORE BUFFER
183									
184	007100	032777	000004	172046	4\$:	BIT	#SW02,@SWR		;TYPE PERFORMANCE SUMMARY
185	007106	001016				BNE	5\$;BR IF NOT
186	007110	005737	001316			TST	STATIN		;TIME TO TYPE THE PERFORMANCE SUMMARY
187	007114	001413				BEQ	5\$;BR IF NOT
188	007116	005037	001316			CLR	STATIN		;CLEAR THE INDICATOR
189	007122	005737	001530			TST	ASNLST		;ANY DRIVES ASSIGNED ?
190	007126	001406				BEQ	5\$;BR IF NO
191	007130	104401	076063			TYPE	,REPHD		;TYPE PERFORMANCE REPORT HEADING
192	007134	004737	023550			JSR	PC,STATPR		;TYPE THE SUMMARY
193	007140	104401	076124			TYPE	,STAR30		;TYPE '*****.ETC'
194	007144	000137	006142		5\$:	JMP	MAIN		;CONTINUE THE LOOP

```

1
2
3 007150 111037 001324 PROCES: MOVB (R0),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
4 007154 005760 000016 TST $STATUS(R0) ;SEE IF DRIVER SIGNALLED AN ERROR
5 007160 100427 BMI ERPROC ;BR IF ERROR
6 007162 032760 100000 002136 BIT #BIT15,$RMCS1(R0) ;SEE IF 'SC' SET
7 007170 001410 BEQ 1$ ;BR IF NOT SET
8 007172 032760 040000 002136 BIT #BIT14,$RMCS1(R0) ;SEE IF 'TRE' SET
9 007200 001017 BNE ERPROC ;BR IF SET
10 007202 032760 040000 002150 BIT #BIT14,$RMDS(R0) ;SEE IF 'ERR' SET
11 007210 001013 BNE ERPROC ;BR IF SET
12
13 ;NO ERRORS DETECTED IN REGISTERS, DO SOME CHECKING ANYWAY
14
15 007212 004737 013176 1$: JSR PC,CKERR ;CHECK ERROR BITS
16 007216 004737 013270 JSR PC,CKBUS ;CHECK BUS ADDRESS & WORD COUNT
17 007222 032777 000002 171724 BIT #SW01,@SWR ;DO DATA COMPARE ?
18 007230 001002 BNE 2$ ;BR IF NO
19 007232 004737 013354 JSR PC,CMPAR ;COMPARE DATA W/O ERROR
20 007236 000207 RTS PC ;RETURN
21
22 ;COMMAND TERMINATED WITH AN ERROR - PROCESS THE ERROR
23
24 007240 032760 000200 000016 ERPROC: BIT #BIT07,$STATUS(R0) ;DONE BIT SET ?
25 007246 001402 BEQ ERPRC1 ;BR IF COMMAND DIDN'T COMPLETE NORMALLY
26 007250 000137 007634 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
27
28 ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
29
30 ERPRC1:
31 007254 032760 010000 000016 BIT #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
32 007262 001025 BNE PUNSAF ;BR IF YES
33 007264 032760 004000 000016 BIT #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED
34 007272 001043 BNE UCPAR ;BR IF IT DID
35 007274 032760 002000 000016 BIT #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
36 007302 001046 BNE FALPAR ;BR IF THERE IS ONE
37 007304 032760 001000 000016 BIT #BIT09,$STATUS(R0) ;TIMEOUT?
38 007312 001070 BNE SWTIM ;BR IF YES
39 007314 032760 040002 000016 BIT #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
40 007322 001103 BNE OFLIN ;BR IF IT DID
41 007324 032760 000004 000016 BIT #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
42 007332 001121 BNE PRTIM ;BR IF IT DID
43 007334 000207 RTS PC ;ERROR. RETURN
44
45 ;DRIVE IS PERSISTENTLY UNSAFE
46
47 007336 104401 001203 PUNSAF: TYPE $CRLF ;CR-LF
48 007342 004737 021046 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
49 007346 104414 071022 DISPLY $EM12 ;PERSISTENT DEVICE UNSAFE MESSAGE
50 007352 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
51 007356 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
52 007362 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
53 007366 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
54 007372 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
55 007376 000137 031054 JMP DROP ;DROP THE DRIVE
56
57 ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED

```

```

58
59 007402 104401 001203      UCPAR: TYPE      ,SCLF      :CR-LF
60 007406 104401 001203      TYPE      ,SCLF      :CR-LF
61 007412 104401 070724      TYPE      ,EM10      :'UNCORRECTABLE PARITY ERROR' MESSAGE
62 007416 000406              BR          FALPR1     :FINISH PROCESSING THE ERROR
63
64      ;'FATAL' MASSBUS PARITY ERROR OCCURRED
65
66 007420 104401 001203      FALPAR: TYPE      ,SCLF      :CR-LF
67 007424 104401 001203      TYPE      ,SCLF      :CR-LF
68 007430 104401 070767      TYPE      ,EM11      :'FATAL PARITY ERROR' MESSAGE
69
70 007434 104401 076311      FALPR1: TYPE      ,MSGON     :TYPE 'ON'
71 007440 104401 075625      TYPE      ,UNTMSG     :TYPE 'DRIVE'
72 007444 013746 001324      MOV        UNIT,-(SP)  ;;SAVE UNIT FOR TYPEOUT
                                ;;TYPE DRIVE NUMBER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 2 DIGIT(S)
                                ;;SUPPRESS LEADING ZEROS
                                ;;INCREMENT TOTAL ERROR COUNT
                                ;;HALT ON ERROR ?
                                ;;BR IF NOT
                                ;;ERROR HALT
                                1$:      RTS          PC
007450 104403              TYPOS
007452      002              .BYTE      2
007453      000              .BYTE      0
73 007454 004737 024724      JSR        PC,INCTOT   :INCREM TOTAL ERROR COUNT
74 007460 032777 100000 171466  BIT        #SW15,@SWR   :HALT ON ERROR ?
75 007466 001401              BEQ        1$
76 007470 000000              HALT
77 007472 000207              1$:      RTS          PC
78
79      ;SOFTWARE TIMEOUT OCCURRED
80
81 007474 004737 021046      SWTIM:  JSR        PC,LINE1  :PRINT LINE 1 OF ERROR MESSAGE
82 007500 104414 071053      DISPLY    ,EM13        :PRINT THE TIME OUT MESSAGE
83 007504 004737 021126      JSR        PC,LINE2    :PRINT LINE 2 OF ERROR MESSAGE
84 007510 004737 021566      JSR        PC,LINE3    :PRINT LINE 3 OF ERROR MESSAGE
85 007514 004737 022236      JSR        PC,LINE4    :PRINT LINE 4 OF ERROR MESSAGE
86 007520 004737 024724      JSR        PC,INCTOT   :INCREM TOTAL ERROR COUNT
87 007524 004737 022710      JSR        PC,LINE7    :PRINT LINE 7 OF ERROR MESSAGE
88 007530 000207              RTS          PC          :RETURN
89
90      ;DRIVE WENT OFFLINE
91
92 007532 104401 001203      OFLIN:  TYPE      ,SCLF      :CR-LF
93 007536 004737 021046      JSR        PC,LINE1    :PRINT LINE 1 OF THE ERROR MESSAGE
94 007542 104414 071125      DISPLY    ,EM14        :PRINT OFFLINE MESSAGE
95 007546 004737 021126      JSR        PC,LINE2    :PRINT LINE 2 OF THE ERROR MESSAGE
96 007552 004737 021566      JSR        PC,LINE3    :PRINT LINE 3 OF THE ERROR MESSAGE
97 007556 004737 022236      JSR        PC,LINE4    :PRINT LINE 4 OF THE ERROR MESSAGE
98 007562 004737 024724      JSR        PC,INCTOT   :INCREM TOTAL ERROR COUNT
99 007566 004737 022710      JSR        PC,LINE7    :PRINT LINE 7 OF THE ERROR MESSAGE
100 007572 000137 031054      JMP        DROP        :DROP THE DRIVE
101
102      ;PORT REQUEST TIMEOUT ERROR
103
104 007576 004737 021046      PRTIM:  JSR        PC,LINE1  :TYPE LINE 1 OF THE ERROR MESSAGE
105 007602 104414 071150      DISPLY    ,EM15        :PRINT PORT TIME OUT MESSAGE
108 007606 004737 021126      JSR        PC,LINE2    :TYPE LINE 2 OF THE ERROR MESSAGE
                                JSR        PC,LINE3    :TYPE LINE 3 OF THE ERROR MESSAGE
                                JSR        PC,LINE4    :TYPE LINE 4 OF THE ERROR MESSAGE
109 007622 004737 024724      JSR        PC,INCTOT   :INCREM TOTAL ERROR COUNT
110 007626 004737 022710      JSR        PC,LINE7    :TYPE LINE 7 OF THE ERROR MESSAGE

```



```

11 007632 000207          RTS      PC          ;RETURN
112
113          ;PROCESS COMMAND COMPLETION WITH 'ERROR' & 'DONE' BITS SET
114
119 007634 032760 000030 000016 DONE:  BIT      #BIT04,BIT03,$STATUS(R0) ;UNSAFE OCCURRED
120 007642 001402          BEQ      1$          ;BR IF NOT
121 007644 000137 012610          JMP      UNSAF      ;REPORT UNSAFE
122
123 007650 032760 040000 002152 1$:   BIT      #BIT14,$RMER1(R0) ;UNSAFE OCCURRED
124 007656 001402          BEQ      2$          ;BR IF NOT
125 007660 000137 012610          JMP      UNSAF      ;REPORT UNSAFE
126
127 007664 032760 040000 002146 2$:   BIT      #BIT14,$RMCS2(R0) ;IS 'WCE' SET ?
128 007672 001402          BEQ      3$          ;BRANCH IF NOT SET
129 007674 000137 010624          JMP      WCKER      ;WRITE CHECK ERROR
130
131 007700 032760 040000 002150 3$:   BIT      #BIT14,$RMDS(R0) ;CHECK 'ERR'
132 007706 001002          BNE      4$          ;BR IF SET
133 007710 000137 012350          JMP      TRFER      ;PROCESS 'TRE'
134
135 007714 032760 000400 002152 4$:   BIT      #BIT08,$RMER1(R0) ;'HCRC' SET?
136 007722 001402          BEQ      5$          ;BR IF NOT
137 007724 000137 011132          JMP      HRCER      ;PROCESS 'HCRC'
138
139 007730 032760 000020 002152 5$:   BIT      #BIT04,$RMER1(R0) ;'FMT' SET?
140 007736 001402          BEQ      6$          ;BR IF NOT SET
141 007740 000137 011310          JMP      CKFMT      ;CHECK FORMAT ERROR
142
143 007744 032760 000200 002152 6$:   BIT      #BIT07,$RMER1(R0) ;'HCE' SET?
144 007752 001402          BEQ      7$          ;BR IF NOT SET
145 007754 000137 011464          JMP      CKHCE      ;CHECK 'HCE' ERROR
146
147 007760 032760 020000 002152 7$:   BIT      #BIT13,$RMER1(R0) ;'OPI' SET?
148 007766 001402          BEQ      8$          ;BR IF NOT SET
149 007770 000137 011744          JMP      OPIER      ;REPORT 'OPI'
150
151 007774 032760 000010 002152 8$:   BIT      #BIT3,$RMER1(R0) ;'PAR' SET?
152 010002 001402          BEQ      9$          ;BR IF NOT SET.
153 010004 000137 012072          JMP      PARER      ;REPORT 'PAR'
154
155 010010 032760 000040 002152 9$:   BIT      #BIT5,$RMER1(R0) ;'WCF' SET?
156 010016 001402          BEQ     10$         ;BR IF NOT SET
157 010020 000137 012512          JMP      WCFER      ;REPORT 'WCF'
158
159 010024 032760 002000 002152 10$:  BIT      #BIT10,$RMER1(R0) ;'IAE' SET?
160 010032 001402          BEQ     11$         ;BR IF NOT SET
161 010034 000137 012164          JMP      IAEER      ;REPORT 'IAE'
162
163 010040 032760 004000 002152 11$:  BIT      #BIT11,$RMER1(R0) ;'WLE' SET?
164 010046 001402          BEQ     12$         ;BR IF NOT SET
165 010050 000137 012216          JMP      WLEER      ;REPORT 'WLE'
166
167 010054 032760 001000 002152 12$:  BIT      #BIT9,$RMER1(R0) ;'AOE' SET?
168 010062 001405          BEQ     13$         ;BR IF NOT SET
169 010064 032760 002000 002150          BIT      #BIT10,$RMDS(R0) ;'LBT' SET?
170 010072 001401          BEQ     13$         ;BR IF NOT SET
171 010074 000207          RTS      PC          ;'AOE' & 'LBT' SET, EXIT

```

```

172
173 010076 032760 010000 002152 13$: BIT #BIT12,$RMER1(R0) ;SEE IF 'DTE' SET
174 010104 001402 BEQ 14$ ;BR IF NOT
175 010106 000137 012050 JMP DTEER ;REPORT 'DTE' ERROR
176
177 010112 005760 002152 14$: TST $RMER1(R0) ;SEE IF 'DCK' SET
178 010116 100002 BPL 15$ ;BR IF NOT
179 010120 000137 010156 JMP DCKER ;PROCESS 'DCK'
180
181 010124 032760 040000 002200 15$: BIT #BIT14,$RMER2(R0) ;'SKI' SET
182 010132 001006 BNE 16$ ;BRANCH IF SKI SET
183 010134 032760 100000 002200 BIT #BIT15,$RMER2(R0) ;'BSE' SET ?
184 010142 001004 BNE 17$ ;BRANCH IF SO (NO, OTHER ERROR)
185 010144 000137 011256 JMP DRIVER ;REPORT ERROR
186
187 010150 000137 012450 16$: JMP SKIER ;REPORT SKI ERROR
188 010154 000207 17$: RTS PC ;EXIT FROM ERROR ANALYSIS ROUT.
189
190 ;PROCESS DATA ('DCK') CHECK ERROR
191
192 010156 004737 020714 DCKER: JSR PC,SPOTCK ;SEE IF ERROR AT A BAD SECTOR ON THE DISK
193 010162 000207 RTS PC ;IT IS, DON'T REPORT IT
194 010164 026027 002202 010040 CMP $RMEC1(R0),#10040 ;IS POSITION COUNT OVER MAXIMUM ?
195 010172 101006 BHI 1$ ;BR IF YES
196 010174 005760 002202 TST $RMEC1(R0) ;POSITION COUNT 0 ?
197 010200 001403 BEQ 1$ ;BR IF YES
198 010202 005760 002204 TST $RMEC2(R0) ;VALUE IN PATTERN REGISTER ?
199 010206 001026 BNE 4$ ;BR IF YES
200 010210 004737 021046 1$: JSR PC,LINE1 ;TYPE FIRST LINE OF ERROR MESSAGE
201 010214 104414 072577 DISPLY ,EM45 ;TYPE 'ECC LOGIC ERROR'
202 010220 004737 021126 JSR PC,LINE2 ;TYPE LINE 2 OF ERROR MESSAGE
203 010224 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
204 010230 012737 000003 001330 MOV #3,RETRY ;RETRY COUNT
205 010236 004737 016104 JSR PC,$RETRY ;RETRY THE COMMAND
206 010242 000403 BR 2$ ;RETRY WAS NOT SUCCESSFUL
207 010244 004737 022626 JSR PC,LINE6C ;TYPE LINE 6C OF ERROR MESSAGE
208 010250 000402 BR 3$ ;FINISH THE ERROR REPORT
209 010252 004737 022634 2$: JSR PC,LINE6D ;TYPE LINE 6D OF ERROR MESSAGE
210 010256 004737 022710 3$: JSR PC,LINE7 ;TYPE LINE 7 OF ERROR MESSAGE
211 010262 000207 RTS PC ;RETURN
212
213 ;THE VALUES IN THE ECC REGISTERS ARE CORRECT, REPORT 'DCK' ERROR
214
215 010264 004737 021046 4$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
216 010270 104414 071225 DISPLY ,EM21 ;DATA CHECK ERROR
217
218 010274 004737 021126 DCKER1: JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
219 010300 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
220 010304 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
221 010310 004737 015352 JSR PC,PRTBAD ;SEE IF BAD SECTOR TO BE PRINTED
222 010314 012737 110100 001326 MOV #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
223 010322 032760 010100 002152 BIT #BIT12!BIT06,$RMER1(R0) ;CHECK 'DTE' & 'ECH'
224 010330 001003 BNE 1$ ;BR IF SET
225 010332 004737 022600 JSR PC,LINE6 ;PRINT LINE 6 OF ERROR MESSAGE
226 010336 000477 BR 8$ ;FINISH THE ERROR REPORT
227 010340 012737 000020 001330 1$: MOV #16.,RETRY ;RETRY COUNT
228 010346 005001 CLR R1 ;R1 IS OFFSET CODE POINTER

```

```

229
230 010350 004737 C17050      2$: JSR   PC,GODRIV      ;RETRY
231 010354 005760 000016      3$: TST   $STATUS(R0)   ;TEST FOR DONE
232 010360 001775              BEQ   3$              ;BR IF NOT DONE
233 010362 100075              BPL   10$             ;BR IF NOT ERROR
234 010364 032760 000200 000016 BIT   #BIT7,$STATUS(R0) ;SEE IF COMMAND TERMINATED NORMALLY
235 010372 001006              BNE   14$            ;BR IF NOT
236 010374 004737 024724      JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
237 010400 104414 074732      DISPLY ,LIN8M        ;'DIFFERENT ERROR DURING RETRY'
238 010404 000137 007254      JMP   ERPRC1         ;SEE WHICH ERROR
239 010410 033760 001326 002152 14$: BIT   MASK,$RMER1(R0) ;LOOK AT CURRENT ERROR
240 010416 001430              BEQ   5$              ;BR IF DIFFERENT ERROR
241 010420 032760 010100 002152 BIT   #BIT12:BIT6,$RMER1(R0) ;'ECH' OR 'DTE' STILL SET ?
242 010426 001437              BEQ   7$              ;BR IF NEITHER SET
243 010430 105237 001331      INCB  RETRY+1        ;INCREMENT RETRY COUNT
244 010434 123737 001330 001331 CMPB  RETRY,RETRY+1   ;DONE ?
245 010442 001342              BNE   2$              ;BR IF NOT
246 010444 005201              INC   R1              ;INCREMENT TABLE INDEX
247 010446 116137 002414 070323 MOVSB OFFCOD(R1),GENDPB+$FMT ;OFFSET CODE
248 010454 001435              BEC   9$              ;BR IF END OF OFFSET TABLE
249 010456 062737 000002 001330 ADD   #2,RETRY        ;NEW RETRY LIMIT
250 010464 004737 015766      JSR   PC,OFFST       ;OFFSET
251 010470 005737 070340      4$: TST   GENDPB+$STATUS ;SEE IF FINISHED WITH OFFSET
252 010474 001775              BEQ   4$              ;BR IF NOT
253 010476 100324              BPL   2$              ;BR IF NO ERROR PERFORMING OFFSET
254 010500 004737 023132      5$: JSR   PC,LINE8     ;PRINT LINE 8 OF ERROR MESSAGE
255 010504 004737 024630      6$: JSR   PC,INCHR     ;INCREMENT 'HARD' ERROR COUNT
256 010510 004737 024724      JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
257 010514 004737 022710      JSR   PC,LINE7       ;PRINT LINE 7 OF ERROR MESSAGE
258 010520 004737 015352      JSR   PC,PRTBAD      ;PRINT THE BAD SECTOR
259 010524 000436              BR    13$            ;CLEAN UP AND RETURN
260 010526 004737 022620      7$: JSR   PC,LINE6B    ;PRINT LINE 6B OF ERROR MESSAGE
261 010532 004737 022536      JSR   PC,LINE5B      ;PRINT LINE 5B OF THE ERROR MESSAGE
262 010536 004737 024604      8$: JSR   PC,INCSOF    ;INCREMENT 'SOFT' ERROR COUNT
263 010542 004737 014606      JSR   PC,ECC         ;CORRECT THE ERROR USING ECC AND CHECK IT
264 010546 000407              BR    11$            ;COMPARE THE BUFFER
265
266 010550 004737 022634      9$: JSR   PC,LINE6D    ;PRINT LINE 6D OF ERROR MESSAGE
267 010554 000753              BR    6$              ;INCREMENT ERROR COUNT
268 010556 004737 022612      10$: JSR  PC,LINE6A     ;PRINT LINE 6A OF ERROR MESSAGE
269 010562 004737 024604      JSR   PC,INCSOF      ;INCREMENT 'SOFT' ERROR COUNT
270 010566 012737 000001 001356 11$: MOV   #1,FRSTER     ;SET PROCESSING 'DCKER' INDICATOR
271 010574 004737 013372      JSR   PC,CMPARD      ;COMPARE THE BUFFER
272 010600 105737 001357      TSTB  FRSTER+1       ;ERROR IN COMPARE ?
273 010604 100406              BMI   13$            ;BRANCH IF ERROR
274 010606 004737 024724      JSR   PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
275 010612 104414 075126      DISPLY ,LIN9G        ;'DATA COMPARE OK' MESSAGE
276 010616 004737 022710      12$: JSR   PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
277 010622 000207      13$: RTS   PC         ;RETURN
278
279 ;WRITE CHECK ERROR PROCESSING
280
281 010624 032760 100000 002152 WCKER: BIT   #BIT15,$RMER1(R0) ;SEE IF 'DCK' SET ALSO
282 010632 001034              BNE   2$              ;BR IF IT IS
283 010634 004737 021046      JSR   PC,LINE1       ;PRINT LINE 1 OF ERROR MESSAGE
284 010640 104414 071331      DISPLY ,EM23         ;PRINT WCE & DCK NOT
285 010644 005037 001326      CLR   MASK           ;CLEAR ERROR MASK

```

```

288 010650 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
      010654 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
      010660 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
      010664 004737 022326 JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
289 010670 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
290 010674 012737 000003 001330 MOV #3,RETRY ;RETRY LIMIT
291 010702 004737 016104 JSR PC,$RETRY ;RETRY THE OPERATION
292 010706 000403 BR 1$ ;RETRY UNSUCCESSFUL
293 010710 004737 022626 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
294 010714 000501 BR 10$ ;FINISH PROCESSING THE ERROR
295 010716 004737 022634 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
296 010722 000476 BR 10$ ;FINISH PROCESSING THE ERROR
297 010724 004737 02071 2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR ON THE DISK
298 010730 000477 BR 11$ ;EXIT IF AT BAD SECTOR ON DISK
299 010732 004737 021046 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
300 010736 012737 071256 010764 MOV #EM22,4$ ;ASSUME THAT EM22 WILL BE PRINTED
301 010744 032760 040000 002146 BIT #BIT14,$RMCS2(RO) ;DID 'WCK' ALSO SET ?
302 010752 001003 BNE 3$ ;BR IF IT DID
303 010754 012737 072157 010764 MOV #EM37,4$ ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
304 ;WRITE CHECK
305 010762 104414 3$: DISPLY ;TYPE THE ERROR MESSAGE
306 010764 000000 4$: .WORD 0 ;MESSAGE ADDRESS GOES HERE
309 010766 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
      010772 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
      010776 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
      011002 004737 022326 JSR PC,LINE5 ;PRINT LINE 5 OF ERROR MESSAGE
310 011006 032760 000100 002152 BIT #BIT06,$RMER1(RO) ;ECH SET ALSO ?
311 011014 001441 BEQ 10$ ;FINISH PROCESSING THE ERROR
312 011016 012737 000020 001330 5$: MOV #16,,RETRY ;RETRY LIMIT - 16 (10)
313 011024 004737 017050 6$: JSR PC,GODRIV ;RETRY THE COMMAND
314 011030 005760 000016 7$: TST STATUS(RO) ;COMMAND FINISHED ?
315 011034 001775 BEQ 7$ ;BR IF NOT
316 011036 100405 BMI 8$ ;BR IF ERROR ON COMMAND
317 011040 105237 001331 INCB RETRY+1 ;INCREMENT RETRY COUNT
318 011044 004737 022626 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
319 011050 000423 BR 10$ ;FINISH ERROR PROCESSING
320 011052 105237 001331 8$: INCB RETRY+1 ;INCREMENT RETRY COUNT
321 011056 123737 001330 001331 CMPB RETRY,RETRY+1 ;DONE ?
322 011064 001714 BEQ 1$ ;BR IF AT RETRY LIMIT
323 011066 032760 100000 002152 BIT #BIT15,$RMER1(RO) ;'DCK' SET
324 011074 001407 BEQ 9$ ;BR IF NOT - DIFFERENT ERROR
325 011076 032760 000100 002152 BIT #BIT06,$RMER1(RO) ;'ECH' ALSO SET ?
326 011104 001347 BNE 6$ ;BR IF IT IS, RETRY COMMAND
327 011106 004737 022626 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
328 011112 000402 BR 10$ ;FINISH PROCESSING ERROR
329 011114 004737 023132 9$: JSR PC,LINE8 ;PRINT LINE 8 - 'DIFFERENT ERROR '
330 011120 004737 024724 10$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
331 011124 004737 022710 JSR PC,LINE7 ;FINISH THE ERROR MESSAGE
332 011130 000207 11$: RTS PC ;RETURN
333
334 ;REPORT 'HCRC' ERROR
335
336 011132 004737 020714 HCRCER: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
337 011136 000446 BR 3$ ;EXIT IF IT IS
338 011140 004737 023216 JSR PC,READDR ;READ ERROR SECTOR HEADER
339 011144 004737 016012 JSR PC,READHD ;GET THE HEAD INFORMATION
340 011150 004737 021046 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE

```

```

341 011154 104414 071204      DISPLY ,EM20      ;REPORT 'HCRC'
342 011160 004737 021126      JSR    PC,LINE2  ;PRINT LINE 2 OF ERROR MESSAGE
343 011164 004737 021566      JSR    PC,LINE3  ;PRINT LINE 3 OF ERROR MESSAGE
344 011170 004737 022236      JSR    PC,LINE4  ;PRINT LINE 4 OF ERROR MESSAGE
345 011174 004737 022470      JSR    PC,LINE5A ;PRINT THE HEADER INFORMATION
351 011200 004737 024604      1$: JSR    PC,INCSOF ;INCREMENT 'SOFT' ERROR COUNT
352 011204 004737 024724      JSR    PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
353 011210 012737 000400      001326 MOV    #BIT8,MASK ;SET ERROR MASK
354 011216 012737 000003      001330 MOV    #3,RETRY  ;RETRY LIMIT
355 011224 004737 016104      JSR    PC,$RETRY ;RETRY COMMAND
356 011230 000405                BR     2$        ;RETRY NOT SUCCESSFUL
357 011232 004737 022626      JSR    PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
358 011236 004737 022710      JSR    PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
359 011242 000404                BR     3$        ;EXIT
360 011244 004737 022634      2$: JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
361 011250 004737 022710      JSR    PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
362 011254 000207      3$: RTS    PC      ;RETURN
363
364      ;REPORT DRIVE ERROR
365
366 011256 004737 021046      DRIVER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
367 011262 104414 071621      DISPLY ,EM30      ;REPORT DRIVE ERROR
368 011266 004737 021126      JSR    PC,LINE2  ;PRINT LINE 2 OF ERROR MESSAGE
369 011272 004737 021566      JSR    PC,LINE3  ;PRINT LINE 3 OF ERROR MESSAGE
370 011276 004737 024724      JSR    PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
371 011302 004737 022710      JSR    PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
372 011306 000207      RTS    PC      ;RETURN
373
374      ;PROCESS FORMAT ('FER') ERROR
375
376 011310 032760 000400      002152 CKFMT: BIT    #BIT8,$RMER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
377 011316 001402                1$ BEQ    1$        ;BR IF NOT SET
378 011320 000137 011132      JMP    HRCRER    ;REPORT HCRC ERROR
379
380 011324 004737 023216      1$: JSR    PC,READDR ;GET CORRECTED TRACK & SECTOR ADDRSSES
381 011330 004737 016012      JSR    PC,READHD ;READ HEADER
382 011334 032737 000400      070356 BIT    #BIT8,GENREG+RMER1 ;'HCRC' SET WHEN HEADER READ?
383 011342 001002                2$ BNE    2$        ;BR IF 'HCRC' SET
384 011344 000137 012244      JMP    FMTER     ;NO. ERROR IS 'FMT' ONLY
385
386 011350 004737 020714      2$: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR ON THE DISK
387 011354 000442                BR     5$        ;EXIT IF IT IS
388 011356 004737 021046      JSR    PC,LINE1  ;PRINT LINE 1 OF ERROR MESSAGE
389 011362 104414 071410      DISPLY ,EM24      ;HEADER READ ERROR - FMT BIT DROPPED UP
390 011366 004737 021126      JSR    PC,LINE2  ;PRINT LINE 2 OF ERROR MESSAGE
391 011372 004737 021566      JSR    PC,LINE3  ;PRINT LINE 3 OF ERROR MESSAGE
392 011376 004737 022236      JSR    PC,LINE4  ;PRINT LINE 4 OF ERROR MESSAGE
398 011402 004737 022470      3$: JSR    PC,LINE5A ;DISPLAY HEADER
399 011406 004737 024604      JSR    PC,INCSOF ;INCREMENT SOFT ERROR COUNT
400 011412 004737 024724      JSR    PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
401 011416 012737 000020      001326 MOV    #BIT4,MASK ;SET ERROR MASK
402 011424 012737 000003      001330 MOV    #3,RETRY  ;RETRY LIMIT
403 011432 004737 016104      JSR    PC,$RETRY ;RETRY THE COMMAND
404 011436 000405                BR     4$        ;RETRY NOT SUCCESSFUL
405 011440 004737 022626      JSR    PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
406 011444 004737 022710      JSR    PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
407 011450 000404                BR     5$        ;EXIT

```

```

408 011452 004737 022634      4$:   JSR   PC,LINE6D      ;PRINT LINE 6D OF ERROR MESSAGE
409 011456 004737 022710      JSR   PC,LINE7          ;PRINT LINE 7 OF ERROR MESSAGE
410 011462 000207              5$:   RTS    PC           ;RETURN
411
412                               ;PROCESS HEADER COMPARE ('HCE') ERROR
413
414 011464 032760 000400 002152 CKHCF: BIT   #BIT8,$RMER1(R0)      ;HCRC SET ON ORIGINAL ERROR ?
415 011472 001402              BEQ   1$                ;BR IF NOT SET
416 011474 000137 011132              JMP   HRCRCR           ;REPORT HEADER CRC ERROR
417
418 011500 004737 023216      1$:   JSR   PC,READDR      ;GET CURRENT SECTOR & TRACK ADDR
419 011504 004737 016012      JSR   PC,READHD        ;READ HEADER OF CURRENT SECTOR
420 011510 032737 000400 070356 BIT   #BIT8,GENREG+RMER1 ;'HCRC' SET ?
421 011516 001017              BNE   3$                ;BR IF SET
422 011520 013746 101066      MOV   CYLNDR,-(SP)     ;;PUSH CYLNDR ON STACK
423 011524 042737 150000 101066 BIC   #150000,CYLNDR   ;CLEAR FORMAT, MFG AND USER BITS FROM HEADER
424 011532 026037 002172 101066 CMP   $RMDC(R0),CYLNDR ;CORRECT CYLINDER ?
425 011540 001402              BEQ   2$                ;BR IF IT IS
426 011542 000137 011672              JMP   POSER            ;REPORT POSITIONING ERROR
427 011546
428 011552 000137 012637 101066      2$:   MOV   (SP)+,CYLNDR  ;;POP STACK INTO CYLNDR
429 011552 000137 012306              JMP   HCEER           ;REPORT 'HCE' ERROR
430 011556 004737 020714      3$:   JSR   PC,SPOTCK     ;SEE IF ERROR AT BAD SECTOR
431 011562 000442              BR    6$                ;EXIT IF IT IS
432 011564 004737 021046      JSR   PC,LINE1         ;PRINT LINE 1 OF ERROR MESSAGE
433 011570 104414 071456      DISPLY ,EM25           ;HEADER READ ERROR - 'HCE' SET
434 011574 004737 021126      JSR   PC,LINE2         ;PRINT LINE 2 OF ERROR MESSAGE
435 011600 004737 021566      JSR   PC,LINE3         ;PRINT LINE 3 OF ERROR MESSAGE
436 011604 004737 022236      JSR   PC,LINE4         ;PRINT LINE 4 OF ERROR MESSAGE
442 011610 004737 022470      4$:   JSR   PC,LINE5A      ;PRINT LINE 5 OF ERROR MESSAGE
443 011614 004737 024604      JSR   PC,INCSOF        ;INCREMENT SOFT ERROR COUNT
444 011620 004737 024724      JSR   PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
445 011624 012737 000200 001326 MOV   #BIT7,MASK       ;SET ERROR MASK
446 011632 012737 000003 001330 MOV   #3,RETRY         ;RETRY LIMIT
447 011640 004737 016104      JSR   PC,$RETRY        ;RETRY THE COMMAND
448 011644 000405              BR    5$                ;RETRY NOT SUCCESSFUL
449 011646 004737 022626      JSR   PC,LINE6C        ;PRINT LINE 6C OF ERROR MESSAGE
450 011652 004737 022710      JSR   PC,LINE7         ;PRINT LINE 7 OF ERROR MESSAGE
451 011656 000404              BR    6$                ;EXIT
452 011660 004737 022634      5$:   JSR   PC,LINE6D        ;PRINT LINE 6D OF ERROR MESSAGE
453 011664 004737 022710      JSR   PC,LINE7         ;PRINT LINE 7 OF ERROR MESSAGE
454 011670 000207              6$:   RTS    PC           ;RETURN
455
456                               ;REPORT POSSIBLE POSITIONING ERROR
457
458 011672 004737 021046      POSER: JSR   PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
459 011676 104414 072775      DISPLY ,EM51           ;PROGRAM DETECTED POSITIONING ERROR
460 011702 004737 021126      JSR   PC,LINE2         ;PRINT LINE 2 OF ERROR MESSAGE
461 011706 004737 021614      JSR   PC,LINE3C        ;PRINT LINE 3C OF ERROR MESSAGE
462 011712 012637 101066      MOV   (SP)+,CYLNDR    ;;POP STACK INTO CYLNDR
463 011716 004737 022470      JSR   PC,LINE5A        ;PRINT LINE 5A OF THE ERROR MESSAGE
464 011722 004737 024700      JSR   PC,INCMIS        ;INCREMENT MISPOSITIONING COUNT
465 011726 004737 024724      JSR   PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
466 011732 004737 023040      JSR   PC,LINE7A        ;PRINT LINE 7A OF ERROR MESSAGE
467 011736 004737 015650      JSR   PC,RECALT        ;RECALIBRATE
468 011742 000207              RTS    PC               ;EXIT

```

```

469
470
471
472 011744 004737 020714 OPIER: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
473 011750 000207 RTS PC ;RETURN IF IT IS
474 011752 004737 021046 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
475 011756 104414 071653 DISPLY ,EM31 ;'OPI' ERROR
476 011762 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
477 011766 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
478 011772 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
479 011776 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
480 012002 012737 020000 001326 MOV #BIT13,MASK ;ERROR MASK
481 012010 012737 000003 001330 OPIER1: MOV #3,RETRY ;RETRY LIMIT
482 012016 004737 016104 JSR PC,$RETRY ;RETRY THE COMMAND
483 012022 000405 BR 1$ ;RETRY UNSUCCESSFUL
484 012024 004737 022626 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
485 012030 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
486 012034 000207 RTS PC ;EXIT
487 012036 004737 022634 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
488 012042 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
489 012046 000207 RTS PC ;RETURN
490
491 ;REPORT 'DTE' ERROR
492
493 012050 004737 020714 DTEER: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SECTOR
494 012054 000207 RTS PC ;RETURN IF IT IS
495 012056 004737 021046 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
496 012062 104414 071716 DISPLY ,EM32 ;'DTE' ERROR
497 012066 000137 010274 JMP DCKER1 ;FINISH PROCESSING THE 'DTE' ERROR
498
499 ;REPORT 'PAR' ERROR
500
501 012072 004737 021046 PARER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
502 012076 104414 071751 DISPLY ,EM33 ;REPORT 'PAR'
503 012102 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
504 012106 004737 021672 JSR PC,LINE3E ;PRINT LINE 3E OF ERROR MESSAGE
505 012112 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
506 012116 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
507 012122 012737 000010 001326 MOV #BIT03,MASK ;ERROR MASK
508 012130 012737 000003 001330 MOV #3,RETRY ;RETRY LIMIT
509 012136 004737 016104 JSR PC,$RETRY ;RETRY COMMAND
510 012142 000405 BR 2$ ;RETRY UNSUCCESSFUL
511 012144 004737 022626 JSR PC,LINE6C ;RETRY SUCCESSFUL
512 012150 004737 022710 1$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
513 012154 000207 RTS PC ;EXIT
514 012156 004737 022634 2$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
515 012162 000772 BR 1$ ;FINISH ERROR MESSAGE
516
517 ;REPORT 'IAE' ERROR
518
519 012164 004737 021046 IAEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
520 012170 104414 072070 DISPLY ,EM35 ;REPORT 'IAE'
521 012174 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
522 012200 004737 021760 JSR PC,LINE3F ;PRINT LINE 3F OF ERROR MESSAGE
523 012204 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
524 012210 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
525 012214 000207 RTS PC ;RETURN

```

```
526
527 ;REPORT 'WLE' ERROR
528
529 012216 004737 021046 WLEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
530 012222 104414 072126 DISPLY ,EM36 ;REPORT 'WLE'
531 012226 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
532 012232 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
533 012236 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
534 012242 000207 RTS PC ;RETURN
535
536 ;REPORT FORMAT ERROR
537
538 012244 004737 021046 FMTER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
539 012250 104414 071537 DISPLY ,EM26 ;FORMAT ERROR
540 012254 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
541 012260 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
542 012264 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
549 012270 004737 022470 JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
550 012274 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
551 012300 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
552 012304 000207 RTS PC
553
554 ;REPORT HEADER COMPARE ERROR
555
556 012306 004737 021046 HCEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
557 012312 104414 071564 DISPLY ,EM27 ;HEADER COMPARE ERROR
558 012316 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
559 012322 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
560 012326 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
567 012332 004737 022470 JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
568 012336 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
569 012342 004737 022710 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
570 012346 000207 RTS PC ;RETURN
571
572 ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
573
574 012350 004737 021046 TRFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
575 012354 104414 072241 DISPLY ,EP 0 ;RH/RM CONTROLLER OR UNIBUS TRANSFER ERROR
576 012360 004737 021126 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
577 012364 004737 021566 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
578 012370 004737 022236 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
579 012374 004737 024724 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
580 012400 032760 121400 002146 BIT #BIT15:BIT13:BIT9:BIT8,$RMCS2(RO) ;'DLT','UPE','MXF','MDPE' SET ?
581 012406 001415 BEQ 2$ ;BR IF NONE SET
582 012410 012737 000003 001330 MOV #3,$RETRY ;RETRY LIMIT
583 012416 005037 001326 CLR MASK ;CLEAR ERROR MASK
584 012422 004737 016104 JSR PC,$RETRY ;RETRY THE OPERATION
585 012426 000403 BR 1$ ;RETURN HERE IF RETRY UNSUCCESSFUL
586 012430 004737 022626 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
587 012434 000402 BR 2$ ;FINISH THE ERROR REPORT
588 012436 004737 022634 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
589 012442 004737 022710 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
590 012446 000207 RTS PC
591
592 ;PROCESS 'SKI' ERRORS
593
594 012450 004737 021046 SKIER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
```



```

595 012454 104414 072737          DISPLY ,EM50          ;'SKI' ERROR
596 012460 004737 021126          JSR   PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
597 012464 004737 021602          JSR   PC,LINE3B       ;PRINT LINE 3B OF ERROR MESSAGE
598 012470 004737 024724          JSR   PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
599 012474 004737 024654          JSR   PC,INCSKI       ;INCREMENT 'SKI' ERROR COUNT
600 012500 004737 023040          JSR   PC,LINE7A       ;PRINT LINE 7A OF ERROR MESSAGE
601 012504 004737 015650          JSR   PC,RECALT       ;RECALIBRATE
602 012510 000207          RTS   PC
603
604                                     ;REPORT WRITE CLOCK FAILURE ('WCF')
605
606 012512 004737 021046          WCFER: JSR   PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
607 012516 104414 072026          DISPLY ,EM34          ;REPORT WRITE CLOCK FAILURE
608 012522 004737 021126          JSR   PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
609 012526 004737 021574          JSR   PC,LINE3A       ;PRINT LINE 3A OF ERROR MESSAGE
610 012532 004737 022236          JSR   PC,LINE4        ;PRINT LINE 4 OF ERROR MESSAGE
611 012536 004737 024724          JSR   PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
612 012542 004737 015352          JSR   PC,PRTBAD       ;SEE IF BAD SECTOR TO BE PRINTED
613 012546 012737 000003 001330   MOV   #3,RETRY        ;RETRY COUNT
614 012554 012737 000040 001326   MOV   #BIT05,MASK     ;ERROR MASK
615 012562 004737 016104          JSR   PC,$RETRY       ;RETRY THE COMMAND
616 012566 000405          BR    2$              ;RETURN HERE IF RETRY UNSUCCESSFUL
617 012570 004737 022626          JSR   PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
618 012574 004737 022710          1$: JSR   PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
619 012600 000207          RTS   PC
620 012602 004737 022634          2$: JSR   PC,LINE6D   ;PRINT LINE 6D OF ERROR MESSAGE
621 012606 000772          BR    1$
622
623                                     ;PROCESS DRIVE UNSAFE ERROR
624
625 012610 004737 021046          UNSAF: JSR   PC,LINE1   ;PRINT LINE 1 OF ERROR MESSAGE
626 012614 104414 073040          DISPLY ,EM60          ;REPORT DRIVE UNSAFE
627 012620 004737 021126          JSR   PC,LINE2        ;PRINT LINE 2 OF ERROR MESSAGE
628 012624 004737 021566          JSR   PC,LINE3        ;PRINT LINE 3 OF ERROR MESSAGE
629 012630 004737 024724          JSR   PC,INCTOT       ;INCREMENT TOTAL ERROR COUNT
630 012634 032760 040000 002200   BIT   #BIT14,$RMR2(R0) ;IS 'SKI' ALSO SFT?
631 012642 001016          BNE   2$              ;BR IF YES
632 012644 012737 040000 001326   MOV   #BIT14,MASK     ;LOAD THE ERROR MASK
633 012652 012737 000003 001330   MOV   #3,RETRY        ;RETRY COUNT
634 012660 004737 016104          JSR   PC,$RETRY       ;RETRY THE COMMAND
635 012664 000403          BR    1$              ;RETRY WAS UNSUCCESSFUL
636 012666 004737 022626          JSR   PC,LINE6C       ;PRINT LINE 6C OF ERROR MESSAGE
637 012672 000402          BR    2$              ;CONTINUE WITH ERROR REPORT
638 012674 004737 022634          1$: JSR   PC,LINE6D   ;PRINT LINE 6D OF ERROR MESSAGE
639 012700 004737 022710          2$: JSR   PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
640 012704 032760 040000 002200   BIT   #BIT14,$RMR2(R0) ;CHECK 'SKI' AGAIN
641 012712 001001          BNE   3$              ;BR IF SET
642 012714 000207          RTS   PC              ;RETURN
643 012716 004737 015650          3$: JSR   PC,RECALT   ;RECALIBRATE
644 012722 000207          RTS   PC              ;RETURN
645
646                                     ;REPORT AN 'UNKNOWN' DATA PATTERN
647
648 012724 105737 001356          NOMTCH: TSTB  FRSTER   ;FIRST ERROR IN THE SECTOR?
649 012730 001013          BNE   1$              ;BR IF NOT OR IF PROCESSING 'DCKER'
650 012732 004737 021046          JSR   PC,LINE1        ;TYPE LINE 1 OF ERROR MESSAGE
651 012736 104414 072435          DISPLY ,EM43          ;'CAN'T MATCH DATA WITH PATTERN'

```

652	012742	004737	021126		JSR	PC,LINE2		;PRINT LINE 2 OF ERROR MESSAGE
653	012746	004737	021574		JSR	PC,LINE3A		;PRINT LINE 3A OF ERROR MESSAGE
654	012752	004737	022236		JSR	PC,LINE4		;PRINT LINE 4 OF ERROR MESSAGE
655	012756	000404			BR	2\$;CONTINUE PROCESSING ERROR
656	012760	104414	072435	1\$:	DISPLY	,EM43		; 'CAN'T MATCH DATA WITH PATTERN'
657	012764	104414	001203		DISPLY	,\$CRLF		;CR-LF
658	012770	104414	075060	2\$:	DISPLY	,LIN91		;HEADER FOR DATA PRINTOUT
666	012774	010146			MOV	R1,-(SP)		;ADDRESS OF WORD 1
	012776	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 1
	013002	104414	075554		DISPLY	,BLNKS2		;TYPE 2 BLANKS
	013005	012146			MOV	(R1)+,-(SP)		;ADDRESS OF WORD 1
	013010	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 1
	013014	104414	001203		DISPLY	,\$CRLF		;CR-LF
	013020	010146			MOV	R1,-(SP)		;ADDRESS OF WORD 2
	013022	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 2
	013026	104414	075554		DISPLY	,BLNKS2		;TYPE 2 BLANKS
	013032	012146			MOV	(R1)+,-(SP)		;ADDRESS OF WORD 2
	013034	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 2
	013040	104414	001203		DISPLY	,\$CRLF		;CR-LF
	013044	010146			MOV	R1,-(SP)		;ADDRESS OF WORD 3
	013046	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 3
	013052	104414	075554		DISPLY	,BLNKS2		;TYPE 2 BLANKS
	013056	012146			MOV	(R1)+,-(SP)		;ADDRESS OF WORD 3
	013060	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 3
	013064	104414	001203		DISPLY	,\$CRLF		;CR-LF
	013070	010146			MOV	R1,-(SP)		;ADDRESS OF WORD 4
	013072	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 4
	013076	104414	075554		DISPLY	,BLNKS2		;TYPE 2 BLANKS
	013102	012146			MOV	(R1)+,-(SP)		;ADDRESS OF WORD 4
	013104	004737	023144		JSR	PC,LIN0CT		;TYPE WORD 4
	013110	104414	001203		DISPLY	,\$CRLF		;CR-LF
667	013114	062701	000770		ADD	#<252.*2.>,R1		;INCREMENT BUFFER POINTER
668	013120	162737	000400	001370	SUB	#256.,CMCNT		;ADJUST WORD COUNT FOR MATCH
669	013126	132760	000001	000024	BITB	#1,\$CODE(R0)		;HEADER OPERATION INVOLVED ?
670	013134	001405			BEQ	3\$;NO
671	013136	062701	000004		ADD	#4,R1		;ADJUST BUFFER ADDRESS
672	013142	162737	000002	001370	SUB	#2,CMCNT		;ADJUST WORD COUNT
673	013150	005002			CLR	R2		;CLEAR 'WORDS TO COMPARE' COUNT IN R2
674	013152	112737	000001	001356	MOVB	#1,FRSTER		;SET 'NOT FIRST ERROR' INDICATOR
675	013160	112737	177777	001357	MOVB	#-1,FRSTER+1		;SET ERROR FOUND INDICATOR
676	013166	013737	001446	001366	MOV	CMPLMT,LIMIT		;RESET THE COMPARE ERROR TIMEOUT LIMIT
677	013174	000207			RTS	PC		;RETURN
678								
679								
680								;CHECK ERROR BITS IN THE RH/RM REGISTERS
681	013176	032760	060000	002136	CKERR: BIT	#60000,\$RMCS1(R0)		;SEE IF 'TRE' OR 'MCPE' SET
682	013204	001012			BNE	1\$;BR IF EITHER SET
683	013206	032760	177400	002146	BIT	#177400,\$RMCS2(R0)		;SEE IF ERROR BITS IN CS2 SET
684	013214	001006			BNE	1\$;BR IF ANY SET
685	013216	005760	002152		TST	\$RMER1(R0)		;ANY BITS SET IN ER1
686	013222	001003			BNE	1\$;BR IF ANY SET
687	013224	005760	002200		TST	\$RMER2(R0)		;ANY BITS SET IN ER2 ?
688	013230	001416			BEQ	2\$;BR IF NONE SET
689	013232	004737	021046	1\$:	JSR	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
690	013236	104414	072502		DISPLY	,EM44		;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
691	013242	004737	021126		JSR	PC,LINE2		;PRINT LINE 2 OF ERROR MESSAGE
692	013246	004737	021566		JSR	PC,LINE3		;PRINT LINE 3 OF ERROR MESSAGE

693	013252	004737	022236	JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
694	013256	004737	024724	JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
695	013262	004737	022710	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
696	013266	000207		2\$: RTS	PC	:RETURN
697						
698						:CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
699						
700	013270	005760	002140	CKBUS: TST	\$RMWC(RO)	:CHECK WORD COUNT
701	013274	001010		BNE	1\$:BR IF NOT ZERO
702	013276	016046	000020	MOV	\$WRDL(RO),-(SP)	:WORD LENGTH
703	013302	006316		ASL	(SP)	:CHANGE INTO BYTE COUNT
704	013304	066016	000006	ADD	\$BUF(RO),(SP)	:ADD THE STARTING LOCATION
705	013310	022660	002142	CMP	(SP)+,\$RMB(RO)	:BUFFER ADDRESS PROPER ?
706	013314	001416		BEQ	2\$:BR IF OK
707	013316	004737	021046	1\$: JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
708	013322	104414	072310	DISPLY	,EM41	:BUS ADDRESS OR WORD COUNT INCORRECT
709	013326	004737	021126	JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
710	013332	004737	021624	JSR	PC,LINE3D	:PRINT LINE 3D OF ERROR MESSAGE
711	013336	004737	022236	JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
712	013342	004737	024724	JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
713	013346	004737	022710	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
714	013352	000207		2\$: RTS	PC	

```

1                                     :COMPARE THE BUFFER
2
3 013354 132760 000004 000024 CMPAR: BITB #4,$CODE(40) :SEE IF READ COMMAND
4 013362 001001                                     BNE 1$ :BR IF IT IS
5 013364 000207                                     RTS PC :RETURN
6 013366 005037 001356 1$: CLR FRSTER :CLEAR 'FIRST ERROR' INDICATOR
7
8 013372 CMPARD:
9 013372 005037 001364 1$: CLR ERCTR :CLEAR THE ERROR COUNTER
10 013376 016001 000006 MOV $BJF(RO),R1 :BUFFER ADDRESS
11 013402 016037 000020 001370 MOV $WRDL(RO),CMCNT :WORD COUNT TO WORKING LOCATION
12 013410 066037 002140 001370 ADD $RMWC(RO),CMCNT :CALCULATE ACTUAL WORDS TRANSFERED
13 013416 001001 BNE 2$
14 013420 000207 RTS PC :EXIT--NO WORDS XFERED
15 013422 016037 000012 001372 2$: MOV $CYL(RO),CMCYL :CYLINDER ADDRESS WORKING LOCATION
16 013430 052737 150000 001372 BIS #150000,CMCYL :SET MFG, USER AND FMT BITS
17 013436 016C37 000010 001374 MOV $SECT(RO),CMSECT :SECTOR & TRACK ADDRESSES TO WORKING LOCNS
18 013444 013737 001446 001366 MOV CMPLMT,LIMIT :DISPLAY LIMIT
19 013452 005237 001366 INC LIMIT :CONVERT PARAMETER INTO LIMIT VALUE
20 013456 012737 177777 001354 CMSTR: MOV #-1,ZROIND :CLEAR THE 'ZERO'S' INDICATOR
21 013464 005037 001360 CLR SAVER1 :CLEAR THE R1 SAVE WORD
22 013470 005037 001362 CLR SAVER5 :CLEAR THE R5 SAVE WORD
23 013474 023760 001370 000022 CMP CMCNT,$SECT(RO) :IS BUFFER SIZE GREATER THAN ONE SECTOR ?
24 013502 101005 BHI 1$ :BR IF IT IS
25 013504 013702 001370 MOV CMCNT,R2 :LESS THAN, USE REMAINING BUFFER
26 013510 005037 001370 CLR CMCNT :SET COUNTER TO 0
27 013514 000405 BR 2$
28 013516 016002 000022 1$: MOV $SECT(RO),R2 :COMPARE SECTOR
29 013522 166037 000022 001370 SUB $SECT(RO),CMCNT :DECREMENT WORD COUNT
30 013530 126027 000024 000005 2$: CMPB $CODE(RO),#5 :READ HEADER & DATA?
31 013536 001026 BNE CMDAT :BR IF NOT
32
33                                     :COMPARE HEADER WORDS
34
35 013540 012705 001372 CMHED: MOV #CMCYL,R5 :ADDRESS OF COMPARING CYLINDER
36 013544 052711 150000 BIS #150000,(R1) :SET BITS INCASE BAD SECTOR ENCOUNTER
37 013550 022521 CMP (R5)+,(R1)+ :CHECK CYLINDER
38 013552 001402 BEQ 1$ :BR IF COMPARE OK
39 013554 004737 013602 JSR PC,CMSTR2 :REPORT ERROR
40 013560 022521 1$: CMP (R5)+,(R1)+ :COMPARE SECTOR/TRACK
41 013562 001402 BEQ 2$ :BR IF EQ
42 013564 004737 013602 JSR PC,CMSTR2 :REPORT ERROR
43 013570 162702 000002 2$: SUB #2,R2 :SUBTRACT HEADER LENGTH FROM SIZE
44 013574 003007 BGT CMDAT :BR IF NOT FINISHED
45 013576 000137 014166 JMP CMPRX :COMPARE THE DATA PORTION
46
47 013602 005237 001364 CMSTR2: INC ERCTR :INCREMENT THE ERROR COUNT
48 013606 004737 014174 JSR PC,CMPRT :REPORT THE COMPARISON ERROR
49 013612 000207 RTS PC :CHECK THE REST OF THE HEADER
50
51                                     :COMPARE DATA FIELD
52
53 013614 004737 027346 CMDAT: JSR PC,GETILMT :GET ADDRESS LIMITS
54 013620 004737 014516 JSR PC,MATCH :FIND THE PATTERN
55 013624 000403 BR 1$ :FOUND A PATTERN
56 013626 004737 012724 JSR PC,NOMTCH :RETURN HERE IF NO MATCH WITH PATTERN MADE
57 013632 000456 BR 7$ :BYPASS COMPARE ROUTINE

```

58	013634	011405		1\$:	MOV	(R4),R5	:ADDRESS OF PATTERN ADDRESS IN R4	
59	013636	012703	000020		MOV	#16.,R3	:R3 IS PATTERN POS COUNTER	
60	013642	022125		2\$:	CMP	(R1)+,(R5)+	:COMPARE BUFFER WITH PATTERN	
61	013644	001016			BNE	4\$:BR IF NOT EQUAL	
62	013646	005737	001364		TST	ERCTR	:ERRORS DETECTED ?	
63	013652	001406			BEQ	3\$:BR IF NO ERRORS	
64	013654	032777	000010	165272	BIT	#SW3,@SWR	:SWITCH 3 SET ?	
65	013662	001402			BEQ	3\$:BR IF NOT SET	
66	013664	004737	014174		JSR	PC,CMPRT	:DISPLAY THE WORD	
67	013670	005302		3\$:	DEC	R2	:DECREMENT SIZE COUNT	
68	013672	003436			BLE	7\$:BR WHEN AT END	
69	013674	005303			DEC	R3	:DECREMENT PATT POS COUNT	
70	013676	001361			BNE	2\$:BR IF NOT AT END OF PATT	
71	013700	000755			BR	1\$:RESTART THE PATTERN	
72	013702	005761	177776	4\$:	TST	-2(R1)	:IS MISCOMPARED CHARACTER-0	
73	013706	001410			BEQ	5\$:BR IF YES	
74	013710	012737	177777	001354	MOV	#-1,ZROIND	:SET NON-ZERO MISCOMPARED INDICATOR	
75	013716	005237	001364		INC	ERCTR	:INCREMENT THE ERROR COUNTER	
76	013722	004737	014174		JSR	PC,CMPRT	:REPORT ERROR	
77	013726	000760			BR	3\$:CONTINUE COMPARE	
78	013730	105737	001356	5\$:	TSTB	FRSTER	:FIRST ERROR?	
79	013734	106407			BMI	6\$:BR IF NOT	
80	013736	005037	001354		CLR	ZROIND	:SET THE ZERO INDICATOR	
81	013742	010137	001360		MOV	R1,SAVER1	:SAVE CURRENT R1	
82	013746	010537	001362		MOV	R5,SAVER5	:SAVE CURRENT R5	
83	013752	000746			BR	3\$:CONTINUE COMPARE	
84	013754	005737	001354	6\$:	TST	ZROIND	:ANY MISCOMPARISONS NOT ZEROS ?	
85	013760	001743			BEQ	3\$:BR IF NONE-ALL ERRORS-ZERO	
86	013762	004737	014174		JSR	PC,CMPRT	:REPORT ERROR	
87	013766	000740			BR	3\$:CONTINUE COMPARING	
88	013770	023727	001370	000003	7\$:	CMP	CMCNT,#3	:LAST 3 WORDS ?
89	013776	003473			BLE	CMPRX	:YES	
90	014000	126027	000024	000005	9\$:	CMPB	\$CODE(R0),#5	:READ HEAD AND DATA ?
91	014006	001414			BEQ	9\$:YES	
92	014010	013702	001370	8\$:	MOV	CMCNT,R2	:SET COUNTER - REMAIN BUFFER LENGTH	
93	014014	020227	000003		CMP	R2,#3	:LAST 3 WORDS ?	
94	014020	003462			BLE	CMPRX	:YES,EXIT	
95	014022	162737	000400	001370	SUB	#256.,CMCNT	:GREATER THAN A SECTOR ?	
96	014030	003671			BLE	CMDAT	:NO,RETURN TO COMPARE LOOP	
97	014032	012702	000400		MOV	#256.,R2	:SET COUNTER =SECTOR SIZE	
98	014036	000666			BR	CMDAT	:RETURN TO COMPARE LOOP	
99	014040	105237	001374	9\$:	INCB	CMSEC	:INCREMENT COUNTER	
100	014044	123737	001374	001424	CMPB	CMSEC,SECLMT	:MAX SECTOR # ?	
101	014052	101424			BLOS	10\$:NO	
102	014054	105037	001374		CLRB	CMSEC	:RESET SECTOR #	
103	014060	105237	001375		INCB	CMTRK	:INCREMENT TRACK #	
104	014064	123737	001375	001426	CMPB	CM'K,TRKLMT	:MAX TRACK # ?	
105	014072	101414			BLOS	10\$:NO	
106	014074	105037	001375		CLRB	CMTRK	:RESET TRACK #	
107	014100	005237	001372		INC	CMCYL	:INCREMENT CYLINDER NUMBER	
108	014104	013746	001372		MOV	CMCYL,-(SP)	:GET COMPARING CYLINDER	
109	014110	042716	150000		BIC	#150000,(SP)	:SAVE ONLY THE CYLINDER BITS	
110	014114	022637	001422		CMP	(SP)+,CYLIMT	:LAST CYLINDER ?	
111	014120	101401			BLOS	10\$:NO	
112	014122	000421			BR	CMPRX	:NORMAL RETURN,NOT WRAP AROUND	
113								
114	014124	012705	001372	10\$:	MOV	#CMCYL,R5	:ADDRESS OF COMPARING CYLINDER	

```

115 014130 052711 150000      BIS      #150000,(R1)      ;SET BITS INCASE BAD SECTOR ENCOUNTER
116 014134 022521             CMP      (R5)+,(R1)+    ;COMPARE 1ST HEADER WORD
117 014136 001402             BEQ     11$            ;MATCH
118 014140 004737 013602      JSR     PC,CMSTR2       ;NOT MATCH
119 014144 022521             11$:    CMP      (R5)+,(R1)+    ;SECOND WORD OF HEADER
120 014146 001402             BEQ     12$            ;MATCH
121 014150 004737 013602      JSR     PC,CMSTR2       ;NOT MATCH
122 014154 162737 000002 001370 12$:    SUB     #2,CMCNT        ;ADJUST WORD COUNT
123 014162 003401             BLE     CMPRX          ;COMPARE IS DONE
124 014164 000711             BR      8$            ;RETURN TO COMPARE LOOP
125
126 014166 004737 014450      CMPRX:  JSR     PC,ENDCMP   ;PRINT LAST LINE IF ERRORS
127 014172 000207             RTS     PC
128
129                               ;TYPE DATA COMPARE ERRORS
130
131 014174 005737 001360      CMPRT:  TST     SAVER1     ;PRINT SAVED VALUES ?
132 014200 001010             BNE     2$            ;BR IF YES
133 014202 105737 001356      TSTB   FRSTER         ;FIRST ERROR?
134 014206 100402             BMI     1$            ;BR IF NOT
135 014210 004737 014270      JSR     PC,4$         ;PRINT INITIAL MESSAGE INFO
136 014214 004737 014352      1$:    JSR     PC,8$         ;PRINT REMAINDER OF MESSAGE
137 014220 000422             BR      3$            ;EXIT
138 014222
139 014222 010146             MOV     R1,-(SP)      ;;PUSH R1 ON STACK
140 014224 010546             MOV     R5,-(SP)      ;;PUSH R5 ON STACK
141 014226 013701 001360      MOV     SAVER1,R1     ;DISPLAY SAVED R1
142 014232 013705 001362      MOV     SAVER5,R5     ;DISPLAY SAVED R5
143 014236 004737 014270      JSR     PC,4$         ;PRINT INITIAL MESSAGE INFO
144 014242 004737 014352      JSR     PC,8$         ;PRINT SAVED VALUES
145 014246 005037 001360      CLR     SAVER1        ;CLEAR SAVED REGISTER INDICATORS
146 014252 005037 001362      CLR     SAVER5        ;CLEAR THE OTHER ONE
147 014256 012605             MOV     (SP)+,R5     ;;POP STACK INTO R5
148 014260 012601             MOV     (SP)+,R1     ;;POP STACK INTO R1
149 014262 004737 014352      JSR     PC,8$         ;PRINT REMAINDER OF MESSAGE
150 014266 000207             3$:    RTS     PC            ;RETURN
151
152 014270 105737 001356      4$:    TSTB   FRSTER         ;FIRST ERROR ?
153 014274 100425             BMI     7$            ;BR IF NOT
154 014276 001013             BNE     5$            ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
155 014300 004737 021046      JSR     PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
156 014304 104414 072354      DISPLY  ,EM42         ;DATA COMPARE ERROR
157 014310 004737 021126      JSR     PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
158 014314 004737 021574      JSR     PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
159 014320 004737 022236      JSR     PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
160 014324 000404             BR      6$            ;GO TO TYPE HEADER
161 014326 104414 074755      5$:    DISPLY  ,LIN9B     ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
162 014332 104414 001203      DISPLY  ,$CRLF        ;CR-LF
163 014336 104414 075004      6$:    DISPLY  ,LIN9H     ;DISPLAY HEADER
164 014342 012737 177777 001356 7$:    MOV     #-1,FRSTER    ;SET FIRST ERROR FLAG
165 014350 000207             RTS     PC            ;RETURN
166
167 014352 005737 001366      8$:    TST     LIMIT        ;TIMEOUT LIMIT REACHED ?
168 014356 001403             BEQ     9$            ;BR IF IT HAS
169 014360 005337 001366      DEC     LIMIT         ;DECREMENT LIMIT COUNTER
170 014364 001005             BNE     10$           ;BR IF NOT AT LIMIT
171 014366 032777 000200 164560 9$:    BIT     #SW07,@SWR    ;PRINT ALL DATA COMPARE ERRORS ?

```

```

169 014374 001001          BNE      10$      ;BR IF YES
170 014376 000207          RTS       PC       ;RETURN
171
172 014400 010146          10$:    MOV      R1,-(SP) ;BUFFER ADDRESS
173 014402 162716 000002    SUB      #2,(SP)   ;ADJUST ADDRESS
174 014406 004737 023144    JSR      PC,LINOC  ;TYPE IT
175 014412 104414 075554    DISPLY  ,BLNKS2   ;TYPE 2 BLANKS
176 014416 016546 177776    MOV      -2(R5),-(SP) ;PUT GOOD DATA ON THE STACK
177 014422 004737 023144    JSR      PC,LINOC  ;TYPE IT
178 014426 104414 075554    DISPLY  ,BLNKS2   ;TYPE 2 BLANKS
179 014432 016146 177776    MOV      -2(R1),-(SP) ;BAD DATA
180 014436 004737 023144    JSR      PC,LINOC  ;TYPE IT
181 014442 104414 001203    DISPLY  ,%CRLF    ;CR-LF
182 014446 000207          RTS       PC       ;RETURN
183
184                          ;LAST LINE OF COMPARE ERROR REPORTING
185
186 014450 105737 001357    ENDCMP: TSTB     FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
187 014454 001417          BEQ      2$       ;BR IF NOT
188 014456 005737 001364    TST     ERCTR     ;SEE HOW MANY ERRORS
189 014462 001410          BEQ      1$       ;BR IF ONLY CAN'T MATCH PATTERN
190 014464 104414 075077    DISPLY  ,LINE9    ;'NUMBER OF ERRORS='
191 014470 013746 001364    MOV      ERCTR,-(SP) ;NUMBER OF ERRORS
192 014474 004737 023176    JSR      PC,LINDEC ;TYPE IT
193 014500 104414 001203    DISPLY  ,%CRLF    ;CR-LF
194 014504 004737 024724    1$:     JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
195 014510 004737 022710    JSR      PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
196 014514 000207          2$:     RTS       PC       ;RETURN
197
198
199                          ;ROUTINE TO MATCH THE DATA WITH A PATTERN, ONLY WHEN LOCATION 'PATTERN'
200                          ;IS EQUAL TO 0 (RANDOM DATA PATTERN MODE). OTHERWISE, THIS ROUTINE WILL
201                          ;RETURN THE ADDRESS OF THE EXPECTED FIXED DATA PATTERN IN R4.
202                          ;CALL:
203                          ;
204                          :     MOV      #BUFFER,R1      ;BUFFER ADDRESS
205                          :     JSR      PC,MATCH        ;PATTERN ADDRESS IN R4
206                          :     RETURN1
207                          :     RETURN2
208                          ;COULDN'T MATCH PATTERN
209
210
211
212
213
214 014516 010146          MATCH:  MOV      R1,-(SP) ;SAVE R1 ON THE STACK
215 014520 013704 001460    MOV      PATTERN,R4 ;WAS RANDOM PATTERN ENABLED ?
216 014524 001402          BEQ      1$       ;BR IF YES
217 014526 006304          ASL     R4        ;* 2
218 014530 000416          BR      4$       ;USE KNOWN PATTERN
219 014532 012704 000044    1$:     MOV      #4,R4  ;PATTERN TABLE INDEX
220 014536 011601          2$:     MOV      (SP),R1 ;RELOAD R1
221 014540 162704 000002    SUB      #2,R4    ;DECREMENT INDEX
222 014544 001413          BEQ     5$       ;BR IF PATTERN NOT MATCH
223 014546 016405 002426    MOV      STNDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
224 014552 012703 000004    MOV      #4,R3    ;NUMBER OF LOCATIONS TO CHECK
225 014556 022125          3$:     CMP      (R1)+,(R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN
226 014560 001366          BNE     2$       ;BR IF NOT EQUAL, TRY NEXT PATTERN
227 014562 005303          DEC     R3       ;FINISHED CHECKING?
228 014564 001374          BNE     3$       ;BR IF NOT FINISHED
229 014566 062704 002426    4$:     ADD      #STNDAT,R4 ;MAKE PATTERN ADDRESS ABSOLUTE
230 014572 000403          BR      6$       ;EXIT
231 014574 062766 000002 000002 5$:     ADD      #2,2(SP) ;INCREMENT RETURN ADDRESS

```

```

236 014602 012601          6$:  MOV      (SP)+,R1      ;RESTORE R1
237 014604 000207          RTS      PC              ;RETURN
238
239                          ;USE ECC TO CORRECT THE DATA ERROR
240
241 014606 016037 002142 001400 ECC:  MOV      $RMB(A(RO),ECSEC ;ADDRESS OF LAST LOCN XFERED
242 014614 016046 002140      MOV      $RMWC(RO),-(SP) ;ACT WORDS XFERED (2'S COMP)
243 014620 066016 000020      ADD      $WRDL(RO),-(SP) ;ADD WORDS REQUESTED
244 014624 001002          BNE      1$
245 014626 005726          TST      (SP)+          ;RESTORE STACK
246 014630 000207          RTS      PC              ;EXIT--NO WORDS XFERED
247 014632 005046          1$:  CLR      -(SP)          ;CLEAR NEXT STACK LOCN
248 014634 016046 000022      MOV      $SSEC(RO),-(SP) ;SECTOR SIZE
249 014640 004737 031710      JSR      PC,$DIV        ;DIVIDE WORDS XFERED BY SECTOR SIZE
250 014644 005716          TST      (SP)          ;PARTIAL SECTOR XFERED ?
251 014646 001413          BEQ      2$            ;BR IF NOT
252 014650 006316          ASL      (SP)          ;CONVERT INTO NUMBER OF BYTES
253 014652 161637 001400      SUB      (SP),ECSEC     ;SUBTRACT SECTOR RESIDUE
254 014656 122760 000005 000024 CMPB     #5,$CODE(RO)   ;WAS OPERATION, READ HEAD & DATA
255 014664 001007          BNE      3$            ;BR IF NOT
256 014666 062737 000004 001400      ADD      #4,ECSEC      ;ADD HEADER SIZE (IN BYTES) BACK IN
257 014674 000403          BR       3$            ;GO ADJUST THE STACK POINTER
258 014676 162737 001000 001400 2$:  SUB      #256,*2,ECSEC  ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
259 014704 062706 000004          3$:  ADD      #4,SP          ;ADJUST THE STACK POINTER
260 014710 016037 002202 001376      MOV      $RMEC1(RO),ECBIT ;ECC POSITION COUNT
261 014716 005337 001376          DEC      ECBIT         ;ADJUST BIT POSITION
262 014722 013737 001376 001406      MOV      ECBIT,ECWRD   ;LOAD THE WORD COUNT LOCATION
263 014730 042737 177760 001376      BIC      #*C17,ECBIT   ;SAVE THE BIT OFFSET COUNT
264 014736 042737 000017 001406      BIC      #17,ECWRD    ;CLEAR THE BIT OFFSET
265 014744 006237 001406          ASR      ECWRD         ;CHANGE TO BYTE COUNT(DIVIDE BY 2)
266 014750 006237 001406          ASR      ECWRD         ;CHANGE TO BYTE COUNT(DIVIDE BY 4)
267 014754 006237 001406          ASR      ECWRD         ;CHANGE TO BYTE COUNT(DIVIDE BY 8.)
268 014760 104414 075154          DISPLY   ,LINT0A      ;'ERROR BURST BEGINS AT '
269 014764 013746 001406          MOV      ECWRD,-(SP)   ;PUT THE WORD COUNT ON THE STACK
270 014770 006216          ASR      (SP)          ;GET STARTING WORD FOR MESSAGE(DIVIDE BY 16.)
271 014772 004737 033130          JSR      PC,$SB2D     ;CONVERT THE WORD COUNT TO DECIMAL
272 014776 004737 032234          JSR      PC,$SUPRL    ;AND PRINT IT
273 015002 104414 075210          DISPLY   ,LINT0B      ;' IN DATA FIELD OF ERROR SECTOR'
274 015006 063737 001400 001406      ADD      ECSEC,ECWRD   ;FIND THE BEGINNING OF THE ERROR BURST
275 015014 026037 002142 001406      CMP      $RMB(A(RO),ECWRD ;SEE IF BURST WAS IN DATA READ
276 015022 101002          BHI      4$            ;BR IF IN DATA READ
277 015024 000137 015340          JMP      ECC2         ;NOT IN DATA READ - REPORT IT
278
279 015030 016037 002204 001402 4$:  MOV      $RMEC2(RO),ECMSKO ;GET THE ERROR BIT MASK
280 015036 005037 001404          CLR      ECMSK1       ;CLEAR THE UPPER MASK WORD
281 015042 005337 001376          5$:  DEC      ECBIT         ;DECREMENT THE BIT OFFSET COUNT
282 015046 002405          BLT      6$            ;BR IF DONE
283 015050 006337 001402          ASL      ECMSKO       ;SHIFT THE ERROR MASK
284 015054 006137 001404          ROL      ECMSK1       ;SHIFT THE LOWER INTO THE UPPER
285 015060 000770          BR       5$            ;CONTINUE THE SHIFT
286
287 015062 017737 164320 001412 6$:  MOV      @ECWRD,ECBADO ;SAVE THE INCORRECT WORD
288 015070 013746 001402          MOV      ECMSKO,-(SP)  ;PUT LOWER MASK ON STACK
289 015074 047716 164306          BIC      @ECWRD,(SP)   ;CLEAR ERRONEOUS ONE BITS FROM MASK
290 015100 043777 001402 164300      BIC      ECMSKO,@ECWRD ;CLEAR ERRONEOUS ONE BITS FROM BAD WORD
291 015106 052677 164274          BIS      (SP)+,@ECWRD ;SET DROPPED BITS
292

```



```

293 015112 005737 001404      TST      ECMSK1      ;DOES ERROR GO INTO NEXT WORD ?
294 015116 001415      BEQ      7$          ;BR IF NO
295 015120 013737 001406 001414      MOV      ECWRD,ECWRD1 ;DUPLICATE ADDRESS
296 015126 062737 000002 001414      ADD      #2,ECWRD1   ;INCREMENT ERROR ADDRESS
297 015134 026037 002142 001414      CMP      $RMB(A(RO)),ECWRD1 ;IS NEXT WORD IN THE BUFFER ?
298 015142 101006      BHI      8$          ;BR IF YES, ELSE
299 015144 005737 001402      TST      ECMSKO      ;WAS ERROR IN FIRST WORD ?
300 015150 001473      BEQ      ECC2        ;BR IF NO
301 015152 005037 001414      7$:      CLR      ECWRD1    ;CLEAR 2ND WORD ADDRESS
302 015156 000414      BR       ECC1        ;PRINT WORD CORRECTED
303
304 015160 017737 164230 001420 8$:      MOV      @ECWRD1,ECBAD1 ;SAVE THE SECOND BAD WORD
305 015166 013746 001404      MOV      ECMSK1,-(SP) ;PUT THE UPPER MASK ON THE STACK
306 015172 047716 164216      BIC      @ECWRD1,(SP) ;CLEAR ERRONEOUS ONE BITS FROM UPPER MASK
307 015176 043777 001404 164210      BIC      ECMSK1,@ECWRD1 ;CLEAR ERRONEOUS ONE BITS FROM DATA WORD
308 015204 052677 164204      BIS      (SP)+,@ECWRD1 ;SET DROPPED BITS
309
310 015210 104414 075354      ECC1:    DISPLY   ,LIN10H      ;HEADER
311 015214 013746 001406      MOV      ECWRD,-(SP)  ;PUT ECWRD ON THE STACK
312 015220 004737 023144      JSR      PC,LIN0CT   ;TYPE ECWRD
313 015224 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
314 015230 013746 001412      MOV      ECBAD0,-(SP) ;PUT ECBAD0 ON THE STACK
315 015234 004737 023144      JSR      PC,LIN0CT   ;TYPE ECBAD0
316 015240 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
317 015244 017746 164136      MOV      @ECWRD,-(SP) ;PUT @ECWRD ON THE STACK
318 015250 004737 023144      JSR      PC,LIN0CT   ;TYPE @ECWRD
319 015254 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
320
321 015260 005737 001414      TST      ECWRD1      ;PRINT THE NEXT WORD ?
322 015264 001427      BEQ      ECCX        ;BR IF NOT
323 015266 104414 001203      DISPLY   ,$CRLF      ;CR-LF
324 015272 013746 001414      MOV      ECWRD1,-(SP) ;PUT ECWRD1 ON THE STACK
325 015276 004737 023144      JSR      PC,LIN0CT   ;TYPE ECWRD1
326 015302 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
327 015306 013746 001420      MOV      ECBAD1,-(SP) ;PUT ECBAD1 ON THE STACK
328 015312 004737 023144      JSR      PC,LIN0CT   ;TYPE ECBAD1
329 015316 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
330 015322 017746 164066      MOV      @ECWRD1,-(SP) ;PUT @ECWRD1 ON THE STACK
331 015326 004737 023144      JSR      PC,LIN0CT   ;TYPE @ECWRD1
332 015332 104414 075554      DISPLY   ,BLNKS2     ;TYPE 2 BLANKS
333 015336 000402      BR       ECCX        ;EXIT
334
335 015340 104414 075250      ECC2:    DISPLY   ,LIN10C      ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
336 015344 104414 001203      ECCX:    DISPLY   ,$CRLF      ;CR-LF
337 015350 000207      RTS      PC          ;RETURN
338
339 ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
340
341 015352 032777 000010 163574 PRTBAD: BIT      #SW3,@SWR      ;PRINT THE BAD SECTOR ?
342 015360 001520      BEQ      8$          ;BR IF NOT
343 015362 016001 002142      MOV      $RMB(A(RO)),R1 ;PUT THE END ADDRESS INTO R1
344 015366 016046 000020      MOV      $WRDL(RO),-(SP) ;FIND THE BEGINNING OF THE SECTOR
345 015372 066016 002140      ADD      $RMC(RO),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
346 015376 001002      BNE      1$          ;
347 015400 005726      TST      (SP)+       ;RESTORE STACK
348 015402 000207      RTS      PC          ;EXIT--NO WORDS XFERRED
349 015404 005046      1$:      CLR      -(SP)     ;MAKE THE UPPER DIVIDEND 0

```

```

342 015406 016046 000022      MOV    $SSEC(RO),-(SP) ;DIVIDE THE WORDS XFERED BY THE SECTOR SIZE
343 015412 004737 031710      JSR    PC,$DIV        ;DIVIDE
344 015416 005716              TST    (SP)          ;REMANDFR = 0 ?
345 015420 001403              BEQ    2$            ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
346 015422 006316              ASL    (SP)          ;CONVERT THE RESIDUAL SECTOR INTO BYTE COUNT
347 015424 161601              SUB    (SP),R1       ;SUBTRACT IT FROM THE END ADDRESS
348 015426 000410              BR     3$            ;FINISH THE SIZING
349 015430 162701 001000      SUB    #256,*2,R1    ;SUBTRACT FULL SECTOR FROM END ADDR (IN BYTES)
350 015434 122760 000005 000024 2$:  CMPB   #5,$CODE(RO)  ;WAS OPERATION READ HEADER & DATA ?
351 015442 001002              BNE    3$            ;BR IF NOT
352 015444 162701 000004      SUB    #4,R1         ;SUBTRACT HEADER SIZE FROM ADDR
353 015450 062706 000004      ADD    #4,SP         ;RESTORE THE STACK POINTER
354 015454 104414 001203      DISPLY , $CRLF       ;CR-LF
355 015460 104414 075437      DISPLY , LIN11H      ;PRINT THE HEADER
356 015464 122760 000005 000024 3$:  CMPB   #5,$CODE(RO)  ;WAS OPERATION READ HEADER & DATA ?
357 015472 001021              BNE    4$            ;BR IF NOT
358 015474 104414 075512      DISPLY , LIN11       ;TYPE 'ADDR  HEADER'
359 015500 101146              MOV    R1,-(SP)      ;PUT THE ADDRESS ON THE STACK
360 015502 004737 023144      JSR    PC,LIN0CT     ;TYPE THE ADDRESS
361 015506 104414 075553      DISPLY ,BLNKS3       ;TYPE 3 BLANKS
362 015512 012146              MOV    (R1)+,-(SP)   ;PUT WORD ON STACK
363 015514 004737 023144      JSR    PC,LIN0CT     ;TYPE THE 1ST HEADER WORD
364 015520 104414 075555      DISPLY ,BLNKS1       ;TYPE 1 BLANK
365 015524 012146              MOV    (R1)+,-(SP)   ;PUT WORD ON STACK
366 015526 004737 023144      JSR    PC,LIN0CT     ;TYPE THE 2ND HEADER WORD
367 015532 104414 001203      DISPLY , $CRLF       ;CR-LF
368
369 015536 104414 075533      4$:  DISPLY ,LIN11A      ;TYPE 'ADDR  DATA'
370 015542 012702 000010      5$:  MOV    #8.,R2        ;8. DATA WORDS PER LINE
371 015546 010146              MOV    R1,-(SP)      ;PUT THE ADDRESS ON THE STACK
372 015550 004737 023144      JSR    PC,LIN0CT     ;TYPE THE ADDRESS
373 015554 104414 075554      DISPLY ,BLNKS2       ;TYPE 2 BLANKS
374 015560 020160 002142      6$:  CMP    R1,$RMB(A(RO)) ;PRINTED ALL THE SECTOR ?
375 015564 001412              BEQ    7$            ;BR IF ALL PRINTED
376 015566 104414 075555      DISPLY ,BLNKS1       ;TYPE 1 BLANK
377 015572 012146              MOV    (R1)+,-(SP)   ;PUT THE DATA ON THE STACK
378 015574 004737 023144      JSR    PC,LIN0CT     ;TYPE THE DATA
379 015600 005302              DEC    R2            ;DECREMENT THE HORIZONTAL COUNT
380 015602 001366              BNE    6$            ;BR IF NOT AT THE END OF THE LINE
381 015604 104414 001203      DISPLY , $CRLF       ;CR-LF
382 015610 000754              BR     5$            ;RESTORE THE WORDS/LINE COUNT
383 015612 104414 001203      7$:  DISPLY , $CRLF       ;CR-LF
384 015616 104414 001203      DISPLY , $CRLF       ;CR-LF
385 015622 000207      8$:  RTS    PC          ;RETURN
386
387 ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN RO
388 ;CALL:
389 :     MOV    #DPB,RO    ;DPB ADDRESS
390 :     JSR    PC,RTNCTR
391 :     RETURN
392
393 015624 111037 070322      RTNCTR: MOVB   (RO),GENDPB ;MOVE THE DRIVE # TO THE GENERAL DPB
394 015626 112737 000117 070324 1$:  MOVB   #RTC,GENDPB+$COMMD ;COMMAND CODE
395 015630 004037 040714      JSR    RO,RM05       ;DRIVER ENTRANCE
396 015642 070322              GENDPB ;DPB ADDRESS FOR COMMAND
397 015644 000774              BR     1$            ;DRIVER DIDN'T ACCEPT COMMAND
398 015646 000207      RTS    PC          ;RETURN
    
```

```

399
400 ;ROUTINE TO DO A RECALIBRATE USING ACTIVE DPB
401 ;CALL:
402 ;     MOV     #DPB,RO           ;DPB ADDRESS
403 ;     JSR     PC,RECALT
404 ;     RETURN
405
406 015650 010037 015674          RECALT: MOV     RO,2$           ;LOAD THE DPB ADDRESS
407 015654 116060 002136 000027 MOVB    $RMCS1(RO),$PREV0(RO) ;SAVE THE PREVIOUS COMMAND
408 015662 112760 000107 000002 MOVB    #RECAL,$COMND(RO)    ;LOAD THE NEW COMMAND
409 015670 004037 040714          1$: JSR     RO,RM05           ;START THE RECALIBRATE
410 015674 000000                2$: .WORD 0                   ;DPB ADDRESS
411 015676 000774                BR     1$                   ;DRIVER DIDN'T ACCEPT THE COMMAND
412 015700 005760 000016          3$: TST     $STATUS(RO)      ;SEE IF FINISHED
413 015704 001775                BEQ     3$                   ;IF EQ NO
414 015706 004737 023216          JSR     PC,READDR          ;DECREMENT THE ADDRESSES
415 015712 012660 000034          MOV     (SP)+,$PREVA+2(RO) ;MOVE THE CYLINDER ADDRESS
416 015716 112660 000033          MOVB   (SP)+,$PREVA+1(RO)  ;MOVE THE TRACK ADDRESS
417 015722 112660 000032          MOVB   (SP)+,$PREVA(RO)   ;MOVE THE SECTOR ADDRESS
418 015726 005060 000012          CLR     $CYL(RO)          ;CLEAR THE CURRENT CYLINDER ADDRESS
419 015732 005060 000010          CLR     $SEC(RO)          ;CLEAR THE CURRENT TRK/SEC ADDRESS
420 015736 000207                RTS     PC                   ;RETURN
421
422 ;ROUTINE TO A RECAL WITH NO DPB ACTIVE
423 ;CALL:
424 ;     MOVB    #DRIVE,GENDPB    ;DRIVE ADDRESS
425 ;     JSR     PC,RECALO
426 ;     RETURN
427
428 015740 112737 000107 070324 RECALO: MOVB    #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
429 015746 004037 040714          1$: JSR     RO,RM05           ;DRIVER ENTRANCE
430 015752 070322                GENDPB ;DPB ADDRESS FOR COMMAND
431 015754 000774                BR     1$                   ;DRIVER DIDN'T ACCEPT THE COMMAND
432 015756 005737 070340          2$: TST     GENDPB+$STATUS    ;SEE IF FINISHED
433 015762 001775                BEQ     2$                   ;BR IF NOT FINISHED
434 015764 000207                RTS     PC
435
436 ;OFFSET THE DRIVE IN RO (OFFSET CODE PRELOADED INTO 'RMOF')
437 ;CALL:
438 ;     MOVB    #OFFSET,GENDPB+$FMT ;OFFSET CODE
439 ;     MOV     #DPB,RO           ;DPB ADDRESS
440 ;     JSR     PC,OFFST
441 ;     RETURN
442
443 015766 111037 070322          OFFST: MOVB    (RO),GENDPB    ;DRIVE # TO GENERAL DPB
444 015772 112737 000115 070324 MOVB    #OFFSET,GENDPB+$COMND ;COMMAND
445 016000 004037 040714          1$: JSR     RO,RM05           ;DRIVER ENTRANCE
446 016004 070322                GENDPB ;DPB ADDRESS FOR COMMAND
447 016006 000774                BR     1$                   ;DRIVER DIDN'T ACCEPT COMMAND
448 016010 000207                RTS     PC
449
450 ;UTILITY READ HEADER ROUTINE
451 ;CALL:
452 ;     MOV     #DPB,RO           ;DPB ADDRESS
453 ;     MOV     #SECTOR,-(SP)     ;SECTOR ADDRESS
454 ;     MOV     #TRACK,-(SP)      ;TRACK ADDRESS
455 ;     MOV     #CYLINDER,-(SP)  ;CYLINDER ADDRESS

```

```

456      :      JSR      PC,F.EADDR
457      :      RETURN
458
459 016012 116637 000004 070333 READHD: MOVB 4(SP),GENDPB+$TRK      ;TRACK ADDRESS
460 016020 116637 000006 070332      MOVB 6(SP),GENDPB+$SEC      ;SECTOR ADDRESS
461 016026 016637 000002 070334      MOV 2(SP),GENDPB+$CYL      ;CYLINDER ADDRESS
462 016034 111037 070322      MOVB (R0),GENDPB      ;DRIVE NUMBER
463 016040 112737 000173 070324      MOVB #RDHD,GENDPB+$COMND      ;COMMAND
464 016046 012737 177776 070326      MOV #-2,GENDPB+$WCNT      ;WORD CTR = 2
465 016054 04037 040714      1$: JSR RO,RM05      ;DRIVER ENTRANCE
466 016060 070322      GENDPB      ;DPB ADDRESS FOR COMMAND
467 016062 000774      BR 1$      ;DRIVER DIDN'T ACCEPT COMMAND
468 016064 005737 070340      2$: TST GENDPB+$STATUS      ;FINISHED?
469 016070 001775      BEQ 2$      ;BR IF NOT
470 016072 011666 000006      MOV (SP),6(SP)      ;ADJUST STACK FOR RETURN
471 016076 062706 000006      ADD #6,SP      ;ADJUST RETRUN POINTER
472 016102 000207      RTS PC      ;RETURN
473
474      :RETRY THE PRESENT OPERATION
475      :CALL:
476      :      MOV #COUNT,RETRY      ;RETRY COUNT
477      :      JSR PC,$RETRY
478      :      RETURN1      ;RETRY UNSUCCESSFUL
479      :      RETURN2      ;SUCCESSFUL RETRY
480      :      ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
481      :      ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
482
483 016104 004737 017050      $RETRY: JSR PC,GODRIV      ;RE-START COMMAND
484 016110 005760 000016      1$: TST $STATUS(R0)      ;COMMAND FINISHED?
485 016114 001775      BEQ 1$      ;BR IF NOT
486 016116 100405      BMI 2$      ;BR IF ERROR
487 016120 105237 001331      INCB RETRY+1      ;INCREMENT RETRY COUNT
488 016124 062716 000002      ADD #2,(SP)      ;INCREMENT RETURN
489 016130 000425      BR 5$      ;GO TO EXIT
490 016132 032760 000200 000016      2$: BIT #BIT7,$STATUS(R0)      ;DID COMMAND TERMINATE NORMALLY?
491 016140 001430      BEQ 7$      ;BR IF NOT
492 016142 005737 001326      TST MASK      ;IS ERROR MASK 0?
493 016146 001004      BNE 3$      ;BR IF NOT
494 016150 005760 002152      TST $RMER1(R0)      ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
495 016154 001014      BNE 6$      ;BR IF NOT
496 016156 000404      BR 4$      ;CONTINUE RETRY
497 016160 033760 001326 002152      3$: BIT MASK,$RMER1(R0)      ;SAME ERROR?
498 016166 001407      BEQ 6$      ;BR IF NOT
499 016170 105237 001331      INCB RETRY+1      ;INCREMENT RETRY COUNT
500 016174 123737 001330 001331      CMPB RETRY,RETRY+1      ;DONE?
501 016202 001340      BNE $RETRY      ;BR IF NOT DONE
502 016204 000207      5$: RTS PC      ;RETURN
503 016206 004737 023132      6$: JSR PC,LINE8      ;REPORT DIFFERENT ERROR
504 016212 004737 022710      JSR PC,LINE7      ;PRINT LINE 7
505 016216 005726      TST (SP)+      ;ADJUST STACK POINTER FOR DIRECT RETURN
506 016220 000207      RTS PC      ;RETURN
507 016222 104414 074732      7$: DISPLY ,LIN8M      ;'DIFFERENT ERROR DURING RETRY'
508 016226 000137 007254      JMP ERPRC1      ;REPORT THE ERROR

```

```

1          ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
2          ;CALL:
3          :
4          :     MOV     #DPB,RO      ;DPB ADDRESS
5          :     JSR     PC,STATIS
6          :     RETURN
7 016232 032760 000300 000016 STATIS: BIT     #BIT07:BIT06,$STATUS(RO) ;CHECK FOR DATA TERMINATION
8 016240 001456          BEQ     3$          ;BR IF NOT DATA TERMINATION
9 016242 016037 002142 016400 MOV     $RMB(A(RO),FACTOR) ;STORE THE FINAL BUFFER ADDRESS
10 016250 166037 000006 016400 SUB     $B(JF(RO),FACTOR) ;SUBTRACT THE INITIAL ADDRESS
11 016256 001447          BEQ     3$          ;BR IF NO DATA TRANSFER
12 016260 006237 016400 ASR     FACTOR          ;CONVERT TO A WORD COUNT
13
14 016264 122760 000002 000024 CMPB   #2,$CODE(RO) ;SEE IF COMMAND WAS A WRITE
15 016272 001404          BEQ     1$          ;BRANCH IF YES
16 016274 122760 000000 000024 CMPB   #0,$CODE(RO) ;PRESENT OPERATION AN AUTO WRITE CHECK ?
17 016302 001012          BNE     2$          ;BR IF NO
18 016304 063760 016400 000060 1$: ADD     FACTOR,$WRITN(RO) ;ADD WORDS WRITTEN DURING WRITE DATA
19 016312 005560 000062          ADC     $WRITN+2(RO) ;DID HIGH WORD OVFL0 AFTER ADDING CARRY ?
20 016316 102004          BVC     2$          ;BR IF NO
21 016320 005060 000062          CLR     $WRITN+2(RO) ;CLEAR HIGH WORD
22 016324 005260 000056          INC     $WTOFL(RO) ;AND COUNT WRITE OVERFLOW
23
24 016330 122760 000002 000024 2$: CMPB   #2,$CODE(RO) ;SEE IF COMMAND WAS A WRITE
25 016336 001417          BEQ     3$          ;BRANCH IF YES
26 016340 063760 016400 000036 ADD     FACTOR,$ENDAT(RO) ;END OF PASS DATA WORD COUNT
27 016346 005560 000040          ADC     $ENDAT+2(RO) ;ADD ANY CARRY
28 016352 063760 016400 000066 ADD     FACTOR,$READ(RO) ;UPDATE THE READ WORD COUNT
29 016360 005560 000070          ADC     $READ+2(RO) ;DID HIGH WORD OVFL0 AFTER ADDING CARRY ?
30 016364 102004          BVC     3$          ;BR IF NO
31 016366 005060 000070          CLR     $READ+2(RO) ;CLEAR HIGH WORD
32 016372 005260 000064          INC     $RDOFL(RO) ;AND COUNT READ OVERFLOW
33 016376 000207          3$: RTS     PC
34
35 016400 000000          FACTOR: .WORD 0 ;USED FOR WORDS TRANSFERED
36
37          ;ROUTINE TO GET A BUFFER
38          ;CALL:
39          :
40          :     MOV     #DPB,RO      ;DPB ADDRESS
41          :     CLR     -(SP)          ;CLEAR THE STACK
42          :     JSR     PC,GETBUF
43          :     RETURN
44          :
45          :     ;BUFFER ADDRESS WILL BE ON THE STACK
46          :     ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
47
48 GETBUF: MOV     R1,-(SP) ;SAVE R1
49 MOV     R2,-(SP) ;SAVE R2
50 MOV     R3,-(SP) ;SAVE R3
51 MOV     BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
52 BEQ     5$          ;BR IF NONE AVAILABLE
53 MOV     #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
54 016422 026061 000020 000002 1$: CMP     $WRDL(RO),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
55 016430 101405          BLOS   2$          ;BRANCH IF IT IS
56 016432 005302          DEC     R2          ;DECREMENT TABLE COUNT
57 016434 001434          BEQ     5$          ;BR IF THROUGH TABLE
58 016436 062701 000004          ADD     #4,R1      ;INCREMENT TABLE POINTER
59 016442 000767          BP     1$          ;CONTINUE LOOKING
60 016444 011166 000010          2$: MOV     (R1),10(SP) ;BUFFER ADDRESS TO STACK

```

```

58 016450 166061 000020 000002 SUB $WRDL(R0),2(R1) ;ADJUST BUFFER WRD CNT
59 016456 001407 BEQ 3$ ;BR IF DIFFERENCE IS ZERO
60 016460 006360 000020 ASL $WRDL(R0) ;CONVERT # WORDS TO BYTES
61 016464 066011 000020 ADD $WRDL(R0),(R1) ;MAKE NEW STARTING ADDRESS
62 016470 006260 000020 ASR $WRDL(R0) ;RETURN # BYTES TO WORDS
63 016474 000414 BR 5$ ;RETURN
64 016476 005337 001642 3$: DEC BUFTBL ;DECREMENT ENTRIES COUNT
65 016502 001411 BEQ 5$ ;BR IF ALLOCATION TABLE EMPTY
66 016504 005302 DEC R2 ;DECREMENT TABLE COUNT
67 016506 001407 BEQ 5$ ;BR IF ITEM WERE LAST ENTRY
68 016510 010103 MOV R1,R3 ;MOVE TABLE POINTER
69 016512 062703 000004 ADD #4,R3 ;POINT TO NEXT ENTRY
70 016516 012321 4$: MOV (R3)+,(R1)+ ;MOVE ITEMS
71 016520 012321 MOV (R3)+,(R1)+
72 016522 005302 DEC R2 ;DECREMENT TABLE COUNT
73 016524 001374 BNE 4$ ;CONTINUE IF NOT AT END OF TABLE
74 016526 012603 5$: MOV (SP)+,R3 ;RESTORE R3
75 016530 012602 MOV (SP)+,R2 ;RESTORE R2
76 016532 012601 MOV (SP)+,R1 ;RESTORE R1
77 016534 000207 RTS PC ;RETURN
78
79
80 ;ROUTINE TO PUT BUFFER BACK IN TABLE
81 ;CALL:
82 ; MOV #DPB,R0 ;DPB ADDRESS
83 ; JSR PC,RELBUF
84 ; RETURN
85
86 016536 010146 RELBUF: MOV R1,-(SP) ;SAVE R1
87 016540 010246 MOV R2,-(SP) ;SAVE R2
88 016542 010346 MOV R3,-(SP) ;SAVE R3
89 016544 010446 MOV R4,-(SP) ;SAVE R4
90 016546 010546 MOV R5,-(SP) ;SAVE R5
91 016550 012701 001644 MOV #BUFTBL+2,R1 ;BEGINNING OF TABLE
92 016554 013702 001642 MOV BUFTBL,R2 ;ENTRY COUNT
93 016560 001424 BEQ 2$ ;BR IF EMPTY TABLE
94 016562 016003 000110 MOV $HLDWC(R0),R3 ;TRIAL ADDRESS
95 016566 006303 ASL R3 ;CHANGE TO BYTE COUNT
96 016570 066003 000006 ADD $BUF(R0),R3 ;ADDRESS OF HIGHER ADJACENT BLOCK
97 016574 021103 1$: CMP (R1),R3 ;UPPER ADJACENT BLOCK
98 016576 001424 BEQ 3$ ;BR IF YES
99 016600 062701 000004 ADD #4,R1 ;INCREMENT POINTER
100 016604 005302 DEC R2 ;DECREMENT ENTRY COUNT
101 016606 001372 BNE 1$ ;CONTINUE SEARCHING
102 016610 016011 000006 MOV $BUF(R0),(R1) ;PUT THE BUFFER BLOCK INTO THE TABLE
103 016614 016061 000110 000002 MOV $HLDWC(R0),2(R1) ;BLOCK WRD CNT
104 016622 005237 001642 INC BUFTBL ;INCREMENT ENTRY COUNT
105 016626 005202 INC R2 ;INCREMENT R2 FOR USE LATER
106 016630 000414 BR 4$ ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
107 016632 016021 000006 2$: MOV $BUF(R0),(R1)+ ;BLOCK ADDRESS TO TABLE
108 016636 016021 000110 MOV $HLDWC(R0),(R1)+ ;WRD CNT TO TABLE
109 016642 005237 001642 INC BUFTBL ;INCREMENT ENTRY COUNT
110 016646 000443 BR 8$ ;EXIT
111 016650 016011 000006 3$: MOV $BUF(R0),(R1) ;RELEASED BUFFER IS LOWER ADJACENT
112 016654 066061 000110 000002 ADD $HLDWC(R0),2(R1) ;INCREMENTED WRD CNT
113 016662 010246 4$: MOV R2,-(SP) ;SAVE R2
114 016664 013702 001642 MOV BUFTBL,R2 ;ENTRY COUNT

```

```

115 016670 012705 001644      MOV      #BUFTBL+2,R5      ;BEGINNING OF TABLE
116 016674 016504 000002      5$:     MOV      2(R5),R4      ;BLOCK SIZE (IN WORDS)
117 016700 006304              ASL      R4                ;CHANGE TO BYTE COUNT
118 016702 061504              ADD      (R5),R4          ;ADD BLOCK BEGINNING ADDRESS
119 016704 020411              CMP      R4,(R1)          ;R1 STILL POINTS TO INSERTED ENTRY
120 016706 001406              BEQ      6$               ;LOWER ADJACENT IN TABLE
121 016710 062705 000004      ADD      #4,R5            ;INCREMENT POINTER
122 016714 005302              DEC      R2                ;DECREMENT ENTRY COUNT
123 016716 001366              BNE      5$               ;CONTINUE LOOKING
124 016720 005726              TST      (SP)+            ;RESTORE STACK POINTER
125 016722 000415              BR       8$               ;END
126 016724 012602              6$:     MOV      (SP)+,R2        ;RESTORE R2
127 016726 066165 000002 000002  ADD      2(R1),2(R5)      ;INCREMENT LOWER BLOCK LENGTH
128 016734 005337 001642      DEC      BUFTBL           ;DECREMENT ENTRY COUNT
129 016740 010105              MOV      R1,R5            ;GET READY TO COMPRESS
130 016742 062705 000004      ADD      #4,R5            ;INCREMENT TO NEXT ENTRY
131 016746 012521              7$:     MOV      (R5)+,(R1)+    ;COMPRESS TABLE
132 016750 012521              MOV      (R5)+,(R1)+    ;MOVE SIZE FIELD DOWN
133 016752 005302              DEC      R2                ;DECREMENT ENTRY COUNT
134 016754 001374              BNE      7$               ;BR IF NOT FINISHED
135 016756 012605              8$:     MOV      (SP)+,R5        ;RESTORE R5
136 016760 012604              MOV      (SP)+,R4        ;RESTORE R4
137 016762 012603              MOV      (SP)+,R3        ;RESTORE R3
138 016764 012602              MOV      (SP)+,R2        ;RESTORE R2
139 016766 012601              MOV      (SP)+,R1        ;RESTORE R1
140 016770 000207              RTS      PC                ;RETURN
141
142      ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK COMMAND)
143      ;CALL:
144      :     MOV      #DPB,R0      ;DPB ADDRESS
145      :     MOV      #BUFADR,$BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
146      :     MOV      #PATTERN,$PATT(R0) ;PATTERN CODE
147      :     JSR      PC,FILBUF
148      :
149
150 016772 104412      FILBUF: SAVREG           ;SAVE THE REGISTERS
151 016774 132760 000004 000024  BITB     #4,$CODE(R0)    ;SEE IF READ COMMAND
152 017002 001020              BNE      4$               ;BR IF READ
153 017004 016001 000006      1$:     MOV      $BUF(R0),R1  ;BUFFER ADDRESS
154 017010 016002 000020      MOV      $WRDL(R0),R2    ;POSITIVE WORD COUNT
155 017014 116004 000030      MOV      $PATT(R0),R4    ;RELATIVE PATTERN ADDRESS
156 017020 016405 002426      2$:     MOV      STNDAT(R4),R5 ;PATTERN ADDRESS
157 017024 012703 000020      MOV      #16,R3         ;PATTERN COUNT
158 017030 012521              3$:     MOV      (R5)+,(R1)+  ;MOVE THE PATTERN INTO THE BUFFER
159 017032 005302              DEC      R2                ;DECREMENT THE WORD COUNT
160 017034 003403              BLE      4$               ;BR IF DONE (WORD COUNT = 0)
161 017036 005303              DEC      R3                ;DECREMENT THE PATTERN COUNT
162 017040 001373              BNE      3$               ;BR IF MORE PATTERN
163 017042 000766              BR       2$               ;CONTINUE DISTRIBUTING THE PATTERN
164 017044 104413      4$:     RESREG           ;RESTORE THE REGISTERS
165 017046 000207              RTS      PC                ;RETURN
166
167      ;START THE COMMAND FOR THE DPB IN R0
168      ;CALL:
169      :     MOV      #DPB,R0      ;DPB ADDRESS
170      :     JSR      PC,GODRIV
171      :     RETURN

```

```
172
173 017050 010046          GODRIV: MOV    RO,-(SP)      ;SAVE RO
174 017052 010037 017062  MOV    RO,2$      ;CURRENT DPB ADDRESS
175 017056 004037 040714  1$:   JSR    RO,RM05  ;CALL THE DRIVE HANDLER
176 017062 000000          2$:   .WORD  0      ;DRIVE BLOCK ADDRESS GOES HERE
177 017064 000000          HALT                    ;DRIVER REJECTED REQUEST
178 017066 012600          MOV    (SP)+,RO    ;RESTORE RO
179 017070 062760 000001 000046  ADD    #1,$OPERC(RO) ;INCREMENT THE OPERATION COUNT
180 017076 005560 000050          ADC    $OPERC+2(RO)
181 017102 026060 000034 000012  CMP    $PREVA+2(RO),$CYL(RO) ;DID COMMAND REQUIRE A CYLINDER CHANGE ?
182 017110 001412          BEQ    3$          ;BR IF NO
183 017112 062760 000001 000042  ADD    #1,$ENDSK(RO) ;INCREMENT END OF PASS SEEK COUNT
184 017120 005560 000044          ADC    $ENDSK+2(RO) ;ADD ANY CARRY
185 017124 062760 000001 000052  ADD    #1,$POSIT(RO) ;INCREMENT SEEK COUNT
186 017132 005560 000054          ADC    $POSIT+2(RO) ;ADD ANY CARRY
187 017136 000207          3$:   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

```

;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
CALL:
      MOV     #DPB,R0           ;DPB ADDRESS
      -----
      MOV     #-2,$PACK(R0)    ;'WRITE PACK' & 'TEST' FLAG
OR
      MOV     #-1,$PACK(R0)    ;'WRITE PACK' FLAG
OR
      MOV     #1,$PACK(R0)     ;'READ PACK' FLAG
      -----
      JSR     PC,WRTPK         ;CALL READ OR WRITE PACK
      RETURN

WRTPK: JSR     PC,GETLMT        ;GET ADDRESS LIMITS
      TST     $OPERC+2(R0)     ;IS THIS THE FIRST OPERATION ?
      BNE     1$              ;BR IF NO
      TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
      BEQ     2$              ;BR IF YES

1$:   MOVB    $RMCS1(R0),$PREVO(R0) ;SAVE CURRENT PARAMETERS
      MOV     $$SEC(R0),$PREVA(R0) ;SAVE PREVIOUS TRACK/SECTOR ADDRESS
      MOV     $CYL(R0),$PREVA+2(R0) ;SAVE PREVIOUS CYLINDER ADDRESS
      MOV     $RMDA(R0),$SEC(R0)   ;CURRENT SECTOR & TRACK ADDRESS
      MOV     $RMDC(R0),$CYL(R0)   ;CURRENT CYLINDER ADDRESS

      MOV     #$$SEC,R4         ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
      JSR     PC,CKLMTS        ;GO CHECK DISK ADDRESS LIMITS
      BR     2$                ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
      MOVB    MINSEC(R0),$SEC(R0) ;RESET SECTOR ADDRESS
      MOVB    MINTRK(R0),$TRK(R0) ;RESET TRACK ADDRESS
      MOV     MINCYL(R0),$CYL(R0) ;RESET CYLINDER ADDRESS
      MOVB    #4,$CODE(R0)      ;SET CODE TO READ DATA
      CMPB    #-2,$PACK(R0)     ;WAS WRITE DATA PACK IN PROGRESS ?
      BEQ     7$                ;BR IF YES (START TESTING)
      JSR     PC,EOP2          ;DROP THE DRIVE (NORMAL TERMINATION)
      BIT     #SW04,@SWR       ;IS SWITCH 4 SET ?
      BEQ     8$                ;BR IF NO

2$:   MOV     #$$SEC,R4         ;GET INDEX TO SECTOR STORAGE
      MOV     WRDCNT,R5        ;WORD COUNT IS MAXIMUM
      JSR     PC,CHKWC        ;CHECK WORD COUNT FOR MAXCYL/MAXTRK
      MOV     R5,$WRDL(R0)     ;GET WORD COUNT
      BIC     #377,$WRDL(R0)   ;IS IT LESS THAN ONE SECTOR WORD COUNT ?
      BNE     3$                ;NO
      INCB    $WRDL+1(R0)      ;SET TO ONE SECTOR
3$:   MOV     $WRDL(R0),$WCNT(R0) ;STORE FOR 2'S COMPLEMENT WORD
      MOV     $WRDL(R0),$HLDWC(R0) ;HOLD WORD FOR 'RELBUF' ROUTINE
      NEG     $WCNT(R0)        ;CHANGE WORD COUNT TO 2'S COMPLEMENT
      MOV     #256,$$SEC(R0)   ;SECTOR SIZE FOR READ

      TSTB   $PACK(R0)        ;READ OR WRITE PACK ?
      BMI     5$                ;BR IF WRITE
4$:   MOVB    #4,$CODE(R0)     ;CODE FOR READ DATA
      MOVB    #RDDAT,$COMND(R0) ;DRIVE CODE FOR OPERATION
      BR     6$                ;SET UP FOR EXIT
5$:   MOVB    #2,$CODE(R0)     ;CODE FOR WRTDAT
      MOVB    #WRTDAT,$COMND(R0) ;OP CODE
  
```

58	017426	004737	020122			JSR	PC,GETPAT	:GET PATTERN CODE
59	017432	110560	000030			MOVB	R5,\$PATT(RO)	:PATTERN CODE
60	017436	012760	177777	000120	6\$:	MOV	#-1,\$NEXT(RO)	:SET PARAMETERS SELECTED INDICATOR
61	017444	000207				RTS	PC	:RETURN
62								
63	017446	005037	001320		7\$:	CLR	PACK	:SET 'TEST' FLAG
64	017452	105060	000026			(LRB	\$PACK(RO)	:SET DPB 'TEST' FLAG
65	017456	005060	000120		8\$:	CLR	\$NEXT(RO)	:CLEAR 'PARAMETER SELECTED' INDICATOR
66	017462	005726				TST	(SP)+	:CLEAR STACK LEVEL
67	017464	000137	006142			JMP	MAIN	:JUMP TO MAIN BACKGROUND LOOP

```

1
2
3
4
5
6
7 017470 004737 027346          ;GENERATE PARAMETERS FOR THE OPERATION
8 017474 004737 036724          ;CALL:
9 017500 032777 000001 161446   ;      MOV      #DPB,R0          ;DPB ADDRESS
10 017506 001012                ;      JSR      PC,GENPAR
11 017510 012705 000010          ;      RETURN
12 017514 004737 031664          GENPAR: JSR      PC,GETLMT      ;GET ADDRESS LIMITS
13 017520 020537 001464          JSR      PC,$RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
14 017524 103003                BIT      #SW0,@SWR         ;SEE IF SW0 SET
15 017526 012705 000002          BNE     1$                ;BR IF SET - READ ONLY
16 017532 000407                MOV      #8.,R5           ;READ/WRITE SELECTION DIVISOR
17                                JSR      PC,GETREM         ;GET SELECTION VALUE
18 017534 013705 037024          CMP      R5,RATIO         ;DETERMINE IF READ OR WRITE
19 017540 000305                BHS     1$                ;BR IF READ
20 017542 042705 177776          MOV      #2,R5           ;SELECT WRITE DATA COMMAND
21 017544 062705 000004          BR      2$                ;SELECT ADDRESS
22 017552 110560 000112          1$:  MOV      $LONUM,R5     ;SELECT READ OPERATION CODE
23 017556 016060 002144 000114   SWAB     R5                ;SWAP BYTES IN R5
24 017564 016060 002172 000116   BIC     #^C1,R5           ;MASK OUT ALL BUT BIT 0
                                ADD      #4,R5             ;TABLE OFFSET FOR READ CODE
                                MOVVB   R5,$NCODE(R0)        ;COMMAND SELECTION CODE TO CONTROL BLOCK
                                MOV      $RMDA(R0),$NSEC(R0)  ;SECTOR AND TRACK
                                MOV      $RMDC(R0),$NCYL(R0)   ;CYLINDER NUMBER
25
26                                .ENABL  LSB
27
28 017572 005737 001474          THEAD: TST     RANDOM      ;ENABLE RANDOM ADDRESS SELECT ?
29 017576 001426                BEQ     RANCYL            ;YES
30 017600 005760 000050          TST     $OPERC+2(R0)     ;IS THIS THE FIRST OPERATION ?
31 017604 001003                BNE     THEAD1           ;BR IF NO
32 017606 005760 000046          TST     $OPERC(R0)      ;IS THIS THE FIRST OPERATION ?
33 017612 001405                BEQ     1$                ;BR IF YES
34
35 017614 012704 000114          THEAD1: MOV     #NSEC,R4   ;GET INDEX TO SECTOR ADDR STORAGE IN DPB
36 017620 004737 020164          JSR      PC,CKLMTS      ;GO CHECK DISK ADDRESS LIMITS
37 017624 000411                BR      2$                ;BR IF NOT AT END OF SEQUENTIAL ADDRESSING
38 017626 116060 000136 000114   1$:  MOVVB   MINSEC(R0),$NSEC(R0) ;RESET SECTOR ADDRESS
39 017634 116060 000132 000115   MOVVB   MINTRK(R0),$NTRK(R0)  ;RESET TRACK ADDRESS
40 017642 016060 000126 000116   MOV      MINCYL(R0),$NCYL(R0) ;RESET CYLINDER ADDRESS
41 017650 000137 020000          2$:  JMP      RANSIZ       ;GO CHECK FOR RANDOM WORD SIZE
42
43                                .DSABL  LSB

```

```

1          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
2
3 017654 016005 000124 RANCYL: MOV    MAXCYL(R0),R5 ;GET MAXIMUM CYLINDER ADDRESS
4 017660 026005 000126      CMP    MINCYL(R0),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
5 017664 001407          BEQ    1$ ;BR IF THEY ARE
6 017666 166005 000126      SUB    MINCYL(R0),R5 ;GET NUMBER OF ALLOWABLE CYLINDERS
7 017672 005205          INC    R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
8 017674 004737 031664      JSR    PC,GETREM ;GET THE RANDOM AUGMENT
9 017700 066005 000126      ADD    MINCYL(R0),R5 ;NEW CYLINDER ADDRESS
10 017704 010560 000116 1$: MOV    R5,$NCYL(R0) ;STORE CYLINDER ADDRESS IN DPB
11
12          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
13
14 017710 016005 000130 RANTRK: MOV    MAXTRK(R0),R5 ;GET MAXIMUM TRACK ADDRESS
15 017714 026005 000132      CMP    MINTRK(R0),R5 ;'MINTRK' AND 'MAXTRK' THE SAME ?
16 017720 001407          BEQ    1$ ;BR IF THEY ARE
17 017722 166005 000132      SUB    MINTRK(R0),R5 ;GET NUMBER OF ALLOWABLE TRACKS
18 017726 005205          INC    R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
19 017730 004737 031664      JSR    PC,GETREM ;GET THE RANDOM AUGMENT
20 017734 066005 000132      ADD    MINTRK(R0),R5 ;NEW TRACK ADDRESS
21 017740 110560 000115 1$: MOVB   R5,$NTRK(R0) ;STORE TRACK ADDRESS IN DPB
22
23          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
24
25 017744 016005 000134 RANSEC: MOV    MAXSEC(R0),R5 ;GET MAXIMUM SECTOR ADDRESS
26 017750 026005 000136      CMP    MINSEC(R0),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
27 017754 001407          BEQ    1$ ;BR IF THEY ARE
28 017756 166005 000136      SUB    MINSEC(R0),R5 ;GET NUMBER OF ALLOWABLE SECTORS
29 017762 005205          INC    R5 ;INCREMENT DIFFERENCE TO USE AS DIVISOR
30 017764 004737 031664      JSR    PC,GETREM ;GET THE RANDOM AUGMENT
31 017770 066005 000136      ADD    MINSEC(R0),R5 ;NEW SECTOR ADDRESS
32 017774 10560 000114 1$: MOVB   R5,$NSEC(R0) ;STORE SECTOR ADDRESS IN DPB
33
34          ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 6 & THE VALUE IN 'WRDCNT'
35
36 020000 013705 001450 RANSIZ: MOV    WRDCNT,R5 ;GET MAX WORD COUNT
37 020004 005737 001462      TST   RANDWC ;SELECT A RANDOM WORD COUNT ?
38 020010 001011          BNE   2$ ;BR IF NOT
39 020012 005205          INC   R5 ;INCREMENT THE MAXIMUM WRD CNT
40 020014 004737 031664      JSR   PC,GETREM ;DIVIDE BY MAX VALUE
41 020020 020527 000006      CMP   R5,#6 ;WORD COUNT LESS THAN 6 ?
42 020024 002003          BGE   2$ ;BR IF NO
43 020026 004737 036724 1$: JSR   PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR
44 020032 000762          BR    RANSIZ
45
46 020034 012704 000114 2$: MOV    #NSEC,R4 ;GET INDEX TO SECTOR STORAGE
47 020040 004737 020316      JSR   PC,CHKWC ;SEE IF WORD COUNT IS TOO LARGE TO FIT
48 ;IN REMAINDER OF TRACK. IF SO, THEN ADJUST
49 ;WORD COUNT TO FIT ON TRACK.
50 020044 122760 000002 000112 3$: CMPB   #2,$NCODE(R0) ;WRITE OPERATION ?
51 020052 001005          BNE   4$ ;BR IF NO
52 020054 042705 000377      BIC   #377,R5 ;WRITING PARTIAL SECTOR ?
53 020060 001002          BNE   4$ ;BR IF NO, ELSE,
54 020062 012705 000400      MOV   #256.,R5 ;WRITE AT LEAST ONE SECTOR
55 020066 010560 000020 4$: MOV    R5,$WRDL(R0) ;WORD COUNT
56
57          ;GET A RANDOM PATTERN NUMBER

```

```
58  
59 020072 122760 00002 000112 RANPAT: CMPB #2,$NCODE(R0) ;WRITE OPERATION ?  
60 020100 001004 BNE RANXIT ;BR IF NO  
61 020102 004737 020122 JSR PC,GETPAT ;GET PATTERN CODE  
62 020106 110560 000113 MOVB R5,$NPATC(R0) ;MOVE PATTERN CODE TO CONTROL BLOCK  
63 020112 012760 177777 000120 RANXIT: MOV #-1,$NEXT(R0) ;SET PARAMETERS SELECTED INDICATOR  
64 020120 000207 RTS PC ;RETURN  
65  
66 ;ROUTINE TO SELECT A PATTERN  
67  
68 020122 012705 000020 GETPAT: MOV #16,,R5 ;SELECT PATTERN  
69 020126 005737 001460 TST PATTERN ;ENABLE RANDOM PATTERN SELECTION ?  
70 020132 001403 BEQ 1$ ;YES  
71 020134 013705 001460 MOV PATTERN,R5 ;USE INDEXED PATTERN  
72 020140 000407 BR 2$ ;NO  
73 020142 004737 031664 1$: JSR PC,GETREM ;GET CODE  
74 020146 005705 TST R5 ;WAS PATTERN ZERO SELECTED ?  
75 020150 001003 BNE 2$ ;BR IF NOT ZERO  
76 020152 004737 036724 JSR PC,$RAND ;CYCLE THE RANDOM NUMBER GENERATOR  
77 020156 000761 BR GETPAT ;TRY AGAIN  
78 020160 006305 2$: ASL R5 ;MAKE CODE INTO TABLE INDEX  
79 020162 000207 RTS PC
```

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

```

;THIS ROUTINE IS USED TO CHECK ADDRESS LIMITS BEFORE THE NEXT COMMAND
;IS PERFORMED. ALSO, IT WILL CHECK FOR MAXIMUM ADDRESS LIMITS TO LOOK
;FOR AN END TO THE SEQUENTIAL ADDRESSING.
;CALL:
      MOV     #DPB,R0           ;DPB ADDRESS
      MOV     #POINTER,R4      ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
      JSR     PC,CKLMTS        ;CALL ADDRESS LIMITS ROUTINE
      BR     ???               ;RETURN HERE IF NOT END OF SEQUENTIAL ADDRESSING
      -----                ;ELSE, RETURN HERE TO RESET DISK ADDRESS
    
```

```

;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
    
```

```

14 020164 060004          CKLMTS: ADD     R0,R4           ;POINT TO SECTOR STORAGE POINT IN DPB
15 020166 026460 000002 000126  CMP     2(R4),MINCYL(R0) ;IS CYLINDER ADDRESS BELOW MIN. ?
16 020174 002003          BGE     1$              ;BR IF NO
17 020176 016064 000126 000002  MOV     MINCYL(R0),2(R4) ;RESET CYLINDER TO MIN.
18 020204 126460 000001 000132 1$:  CMPB   1(R4),MINTRK(R0) ;IS TRACK ADDRESS BELOW MIN. ?
19 020212 002003          BGE     2$              ;BR IF NO
20 020214 116064 000132 000001  MOVB   MINTRK(R0),1(R4) ;RESET TRACK TO MIN.
21 020222 121460 000136          2$:  CMPB   (R4),MINSEC(R0) ;IS SECTOR ADDRESS BELOW MIN. ?
22 020226 002002          BGE     3$              ;BR IF NO
23 020230 116014 000136          MOVB   MINSEC(R0),(R4) ;RESET SECTOR TO MIN.
    
```

```

;LOOK FOR MAXIMUM LIMITS AND END OF SEQUENTIAL ADDRESSING
    
```

```

27 020234 121460 000134          3$:  CMPB   (R4),MAXSEC(R0) ;IS SECTOR ADDRESS AT MAXIMUM ?
28 020240 003404          BLE     4$              ;BR IF NO
29 020242 116014 000136          MOVB   MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
30 020246 105264 000001          INCB   1(R4)           ;INCREMENT TO NEXT TRACK ADDRESS
31 020252 126460 000001 000130 4$:  CMPB   1(R4),MAXTRK(R0) ;IS TRACK ADDRESS OVER MAXIMUM ?
32 020260 003407          BLE     5$              ;BR IF NO
33 020262 116014 000136          MOVB   MINSEC(R0),(R4) ;RESET SECTOR ADDRESS
34 020266 116064 000132 000001  MOVB   MINTRK(R0),1(R4) ;RESET TRACK ADDRESS
35 020274 005264 000002          INC     2(R4)           ;INCREMENT TO NEXT CYLINDER ADDRESS
36 020300 026460 000002 000124 5$:  CMP     2(R4),MAXCYL(R0) ;IS CYLINDER ADDRESS OVER MAXIMUM ?
37 020306 003402          BLE     6$              ;BR IF NO
38 020310 062716 000002          ADD     #2,(SP)        ;ADJUST RETURN TO RESET DISK ADDRESS PARAMETERS
39 020314 000207          6$:  RTS     PC           ;RETURN
    
```

```

1      ; THIS ROUTINE IS USED TO CALCULATE AND CHECK THE WORD COUNT FOR THE
2      ; DRIVE THAT IS TO DO A DATA TRANSFER ON THE MAXIMUM TRACK OF THE MAXIMUM
3      ; CYLINDER. IF THE CALCULATED MAXIMUM WORD COUNT, EXCEEDS THE DESIRED WORD
4      ; COUNT (CONTENTS OF R5), THEN THE DESIRED WORD COUNT IS CHANGED, SO THAT
5      ; THE WORD COUNT WILL NOT CAUSE A TRACK OVERFLOW DURING THE TRANSFER.
6      ; CALL:
7      ;
8      ;       MOV      #DPB,R0          ;DPB ADDRESS
9      ;       MOV      #POINTER,R4     ;POINTER TO SECTOR STORAGE ($SEC OR $NSEC) IN DPB
10     ;       JSR      PC,CHKWC        ;CALL CHECK WORD COUNT ROUTINE
11     ;       RETURN                    ;RETURN WITH R5 CONTAINING THE DESIRED WORD COUNT
12     ;
13     ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
14     ;R4 = POINTER TO SECTOR STORAGE BEFORE CALLING THE ROUTINE
15     ;R5 = DESIRED WORD COUNT BEFORE CALLING THE ROUTINE
16 020316 060004          CHKWC: ADD      R0,R4          ;POINT TO SECTOR STORAGE POINT IN DPB
17 020320 105760 000136  TSTB     MINSEC(R0)      ;ALLOW SPIRAL RD/WRT ?
18 020324 001023          BNE      2$              ;BR IF NO
19 020326 126037 000134 001424  CMPB     MAXSEC(R0),SECLMT      ;ALLOW SPIRAL RD/WRT ?
20 020334 001017          BNE      2$              ;BR IF NO
21 020336 105760 000132  TSTB     MINTRK(R0)      ;ALLOW SPIRAL RD/WRT ?
22 020342 001010          BNE      1$              ;BR IF NO
23 020344 126037 000130 001426  CMPB     MAXTRK(R0),TRKLMT     ;ALLOW SPIRAL RD/WRT ?
24 020352 001004          BNE      1$              ;BR IF NO
25                                ;WHEN SPIRAL RD/WRT IS ALLOWED, THEN CHECK
26                                ;TO MAKE SURE YOU DO NOT SPIRAL OVER MAXIMUM
27                                ;TRACK ON MAXIMUM CYLINDER
28 020354 026064 000124 000002  CMP      MAXCYL(R0),2(R4)      ;ON MAXIMUM CYLINDER ?
29 020362 001021          BNE      4$              ;BR IF NO
30 020364 126064 000130 000001 1$: CMPB     MAXTRK(R0),1(R4)      ;ON MAXIMUM TRACK ?
31 020372 001015          BNE      4$              ;BR IF NO
32 020374 111404          2$: MOVB     (R4),R4          ;GET STARTING SECTOR ADDRESS
33 020376 016046 000134          MOV      MAXSEC(R0),-(SP)      ;GET MAXIMUM SECTOR
34 020402 160416          SUB      R4,(SP)          ;GET NUMBER SECTORS TO BE XFERD
35 020404 005004          CLR      R4              ;CLEAR R4
36 020406 062704 000400 3$: ADD      #256.,R4          ;ADD 1 SECTOR OF WORDS TO R4
37 020412 005316          DEC      (SP)          ;DONE ALL SECTORS YET ?
38 020414 002374          BGE      3$              ;BR IF NO
39 020416 005726          TST      (SP)+          ;RESTORE STACK
40 020420 020504          CMP      R5,R4          ;TOO MANY WORDS FOR TRACK ?
41 020422 003401          BLE      4$              ;BR IF NO
42 020424 010405          MOV      R4,R5          ;YES, CHANGE WORD COUNT
43 020426 000207          4$: RTS      PC          ;RETURN

```

```

1          :ROUTINE TO GET THE PREVIOUSLY SELECTED PAMAMETER VALUES
2          :CALL:
3          :
4          :       MOV      #DPB,RO          ;DPB ADDRESS
5          :       JSR      PC,GENPAR        ;GENERATE THE PARAMETERS
6          :       JSR      PC,LODPAR       ;LOAD THE PARAMETERS JUST GENERATED
7          :       RETURN
8 020430   010546
9 020432   105760   000026
10 020436   001106
11 020440   116060   002136   000027
12 020446   142760   177701   000027
13 020454   132760   000006   000112
14 020462   001007
15 020464   016060   000012   000034
16 020472   016060   000010   000032
17 020500   000410
18 020502   004737   023216
19 020506   012660   000034
20 020512   112660   000033
21 020516   112660   000032
22
23 020522   032777   000100   160424
24 020530   001051
25 020532   116060   000112   000024
26 020540   116005   000112
27 020544   116560   002064   000002
28 020552   122760   000151   000002
29 020560   001012
30 020562   122760   000060   000027
31 020570   001431
32 020572   112760   000171   000002
33 020600   112760   000004   000024
34
35 020606   116060   000113   000030
36 020614   016060   000114   000010
37 020622   016060   000116   000012
38 020630   012760   000400   000022
39 020636   132760   000001   000024
40 020644   001403
41 020646   062760   000002   000022
42 020654   016060   000020   000004
43 020662   016060   000020   000110
44 020670   005460   000004
45 020674   012605
46 020676   000207

LODPAR: MOV      R5,-(SP)          ;SAVE R5
        TSTB     $PACK(RO)        ;'R' OR 'W' COMMAND FOR THE DRIVE ?
        BNE      4$              ;BR IF YES
        MOVB     $RMCS1(RO),$PREVO(RO) ;SAVE CURRENT PARAMETERS
        BICB     #^C76,$PREVO(RO) ;STRIP GO,AND IE BITS
        BITB     #6,$NCODE(RO)    ;SEE IF NEXT OPERATION IS READ OR WRITE
        BNE      1$              ;B. IF EITHER
        MOV      $CYL(RO),$PREVA+2(RO) ;SAVE STARTING CYLINDER
        MOV      $SEC(RO),$PREVA(RO) ;SAVE STARTING SECTOR AND TRACK
        BR       2$
1$:     JSR      PC,READDR        ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
        MOV      (SP)+,$PREVA+2(RO) ;CYLINDER ADDRESS
        MOVB     (SP)+,$PREVA+1(RO) ;TRACK ADDRESS
        MOVB     (SP)+,$PREVA(RO)  ;SECTOR ADDRESS
2$:     BIT      #SW06,@SWR        ;SWITCH6 SET ?
        BNE      4$              ;BR IF SET
        MOVB     $NCODE(RO),$CODE(RO) ;LOGICAL CODE FOR OPERATION
        MOVB     $NCODE(RO),R5    ;LOAD R5 FOR USE AS TABLE INDEX
        MOVB     COMTBL(R5),$COMND(RO) ;COMMAND CODE
        LMPB     #WCKD,$COMND(RO) ;IS NEW COMMAND A WRITE CHECK DATA ?
        BNE      3$              ;BR IF NO
        CMPB     #60,$PREVO(RO)  ;WAS PREVIOUS COMMAND ,, WRITE DATA ?
        BEQ      4$              ;BR IF YES
        MOVB     #RDDAT,$COMND(RO) ;CHANGE WRITE CHECK TO READ DATA COMMAND
        MOVB     #4,$CODE(RO)    ;CODE NUMBER CHANGED TO READ DATA
3$:     MOVB     $NPATC(RO),$PATIC(RO) ;PATTERN CODE
        MOV      $NSEC(RO),$SEC(RO) ;TRACK AND SECTOR ADDRESSES
        MOV      $NCYL(RO),$CYL(RO) ;CYLINDER ADDRESS
        MOV      #256,$SSEC(RO)  ;INITIAL VALUE OF SECTOR SIZE
        BITB     #1,$CODE(RO)    ;HEADER OPERATION ?
        BEQ      4$              ;BR IF NOT
        ADD      #2,$SSEC(RO)    ;ADD HEADER SIZE
4$:     MOV      $WRDL(RO),$WCNT(RO) ;GET WORD COUNT AND
        MOV      $WRDL(RO),$HLDWC(RO) ;HOLD WORD FOR 'RELBUF' ROUTINE
        NEG      $WCNT(RO)       ;MAKE IT 2'S COMPLEMENT
        MOV      (SP)+,R5        ;RESTORE R5
        RTS      PC              ;RETURN

```



```

1      ;ROUTINE TO COMPRESS A LIST
2      :CALL:
3      :
4      :       MOV      #ADDRS,R1      ;COMPRESS LIST STARTING AT THIS ADDRESS
5      :       JSR      PC,CMPRES
6      :       RETURN
7      020700 016111 000002      CMPRES: MOV      2(R1),(R1)      ;COMPRESS THE TABLE IN R1
8      020704 001402              BEQ      1$                    ;BR WHEN ZERO FOUND
9      020706 005721              TST      (R1)+                 ;INCREMENT R1
10     020710 000773              BR       CMPRES                ;CONTINUE COMPRESSING TABLE
11     020712 000207      1$:      RTS      PC          ;RETURN
12
13     ;ROUTINE TO DETERMINE IF THE ERROR IS AT A LOCATION ON THE DISK, DEFINED
14     ;IN THE BAD SECTOR TABLE FOR THE DRIVE.
15     :CALL:
16     :
17     :       JSR      PC,SPOTCK
18     :       RETURN1
19     :       RETURN2              ;ERROR AT AN ADDRESS IN TABLE
20     :                               ;NO TABLE ENTRY FOR ERROR ADDRESS OR
21     :                               ;PARAMETER 'MESSAGE' IS 0
22     020714 010146              SPOTCK: MOV      R1,-(SP)           ;;PUSH R1 ON STACK
23     020716 012701 000144      MOV      #BADSEC,R1          ;INCREMENT FOR BAD SECTOR TABLE
24     020722 060001              ADD      R0,R1              ;ADD THE BLOCK'S STARTING ADDRESS
25     020724 004737 023216      1$:      JSR      PC,READDR      ;DECREMENT THE SECTOR/TRACK ADDRESS
26     020730 021126              CMP      (R1),(SP)+         ;ON THE SAME CYLINDER ?
27     020732 001023              BNE     5$                    ;BRANCH IF NOT
28     020734 122761 177777 000003  CMPB     #-1,3(R1)          ;ALL BAD TRACKS ?
29     020742 001002              BNE     2$                    ;BR IF NO
30     020744 005726              TST     (SP)+                ;ADJUST STACK AND
31     020746 000403              BR      3$                    ;GO CHECK SECTORS
32     020750 122661 000003      2$:      CMPB     (SP)+,3(R1)      ;COMPARE THE TRACK ADDRESS
33     020754 001013              BNE     6$                    ;BR IF IT IS NOT EQUAL
34     020756 122761 177777 000002  3$:      CMPB     #-1,2(R1)          ;ALL BAD SECTORS ?
35     020764 001002              BNE     4$                    ;BR IF NO
36     020766 005726              TST     (SP)+                ;ADJUST STACK AND
37     020770 000413              BR      8$                    ;CHECK 'MESSAGE'
38     020772 122661 000002      4$:      CMPB     (SP)+,2(R1)      ;COMPARE THE SECTOR ADDRESS
39     020776 001003              BNE     7$                    ;BR IF NOT EQUAL
40     021000 000407              BR      8$                    ;CHECK 'MESSAGE'
41     021002 005726              5$:      TST     (SP)+                ;CLEAR OFF THE STACK
42     021004 005726              6$:      TST     (SP)+                ;INCREMENT THE STACK POINTER
43     021006 062701 000004      7$:      ADD      #4,R1              ;GO TO THE NEXT LOCATION IN THE TABLE
44     021012 005711              TST     (R1)                 ;EMPTY ENTRY OR TERMINATOR ?
45     021014 100407              BMI     9$                    ;BR IF YES
46     021016 000742              BR      1$                    ;TRY NEXT SECTOR
47     021020 005737 001472      8$:      TST     MESSAGE            ;PRINT THE ERROR ANYWAY ?
48     021024 001006              BNE     10$                   ;BR IF NOT
49     021026 012737 177777 001342  9$:      MOV      #-1,BADSEC         ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
50     021034 062766 000002 000002  9$:      ADD      #2,2(SP)          ;INCREMENT THE RETURN
51     021042 012601              10$:     MOV      (SP)+,R1        ;;POP STACK INTO R1
52     021044 000207              RTS      PC                  ;RETURN
    
```

:SBTTL ERROR MESSAGE GENERATION ROUTINES

:PRINT LINE 1 OF ERROR MESSAGE:
:'HH:MM:SS'

```
8 021046 032777 002000 160100 LINE1: BIT #SW10,@SWR ;SWITCH 10 SET ?
9 021054 001402 BEQ 1$ ;BR IF NOT
10 021056 104401 001176 TYPE ,S$ELL ;RING THE BELL
11 021062 032777 020000 160064 1$: BIT #SW13,@SWR ;INHIBIT TYPEOUT ?
12 021070 001405 BEQ 2$ ;BR IF NOT
13 021072 104414 001203 DISPLY ,S$RLF ;CR-LF
14 021076 104414 001203 DISPLY ,S$RLF ;CR-LF
15 021102 000410 BR 3$ ;EXIT
16 021104 104414 001203 2$: DISPLY ,S$RLF ;CR-LF
17 021110 104414 001203 DISPLY ,S$RLF ;CR-LF
18 021114 004737 024750 JSR PC,$TIME ;TYPE THE TIME
19 021120 104414 075555 DISPLY ,BLNK$1 ;TYPE 1 BLANK
20 021124 000207 3$: RTS PC ;RETURN & TYPE DESCRIPTION
```

:PRINT LINE 2 OF ERROR MESSAGE
:'PRSNT COMMAND = XXXX PREV COMMAND = XXXX'
:'* ERROR AT BAD TRACK/SECTOR'
:'DRV RMCS1 RMCS2 RMD\$1 RMER1 RMMR2 RMER2 RMEC1 RMEC2'
:'RMWC RMBA RMDA RMAS RMLA RMDB RMMR1 RMDT'
:'RMSN RMOF RMDC RMCC STATUS'
:'RMBAE RMCS3' (RH70 ONLY)
:'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
:'RMBA = XXXXXX RMWC = XXXXXX'
:'BUFFER ADR = XXXXXX WRD CNT = XXXX ACTUAL NMBR WRDS XFRD XXX'

```
35 021126 LINE2: MOV R3,-(SP) ;;PUSH R3 ON STACK
021126 010346 MOV R4,-(SP) ;;PUSH R4 ON STACK
021130 010446 MOV R5,-(SP) ;;PUSH R5 ON STACK
021132 010546 DISPLY ,S$RLF ;CR-LF
36 021134 104414 001203 CLR 4$ ;CLEAR MESSAGE ADDRESS STORAGE
37 021140 005037 021266 CLR R4 ;WORKING REGISTER
38 021144 005004 MOV #LIN2C,4$ ;ADDRESS OF 'PRSNT COMMAND - ' MSC
39 021146 012737 073666 021266 MOV$B $RMCS1(R0),R4 ;GET THE OPCODE
40 021154 116004 002136 BIC #*C76,R4 ;SAVE ONLY SIGNIFICANT BITS
41 021160 042704 177701 JSR PC,1$ ;TYPE THE FIRST MNEMONIC
42 021164 004737 021222 TST 5$ ;SEE IF MNEMONIC ENTRY FOUND
43 021170 005737 021272 BEQ LINE2A ;BR IF NOT
44 021174 001440 MOV #LIN2P,4$ ;ADDRESS OF 'PREVS COMMAND = ' MSG
45 021176 012737 073706 021266 MOV$B $PREVO(R0),R4 ;PREVIOUS OPERATION CODE
46 021204 116004 000027 BIC #*C76,R4 ;SAVE ONLY SIGNIFICANT BITS
47 021210 042704 177701 JSR PC,1$ ;TYPE THE PREVIOUS MNEMONIC
48 021214 004737 021222 BR LINE2A ;CONTINUE
49 021220 000426 1$: CLR R5 ;CLEAR THE TABLE INDEX
50 021222 005005 2$: (MPB OPTBL(R5),R4 ;LOOK FOR THE OPCODE
51 021224 126504 002072 BEQ 3$ ;BR WHEN OPCODE COUNT EQUALS OPCODE
52 021230 001405 3$ TSTB OPTBL(R5) ;LOOK FOR END OF TABLE
53 021232 105765 002072 BMI 3$ ;BR IF END
54 021236 100402 INC R5 ;INCREMENT THE POINTER
55 021240 005205 BR 2$ ;CONTINUE - NOT END OF TAB E
56 021242 000770
```

```

57 021244 006305          3$:   ASL      R5          ;SHIFT INDEX
58 021246 006305          ASL      R5          ;SHIFT THE INDEX
59 021250 006305          ASL      R5          ;SHIFT THE INDEX
60 021252 012737 002112 021272  MOV     #MNTBL,5$    ;ADDRESS OF ASCII TEXT TABLE
61 021260 060537 021272  ADD      R5,5$      ;ADD THE INDEX
62 021264 104414          DISPLY          ;TYPE IT
63 021266 000000          .WORD     0          ;ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
64 021270 104414          DISPLY          ;TYPE THE OPERATION MNEMONIC
65 021272 000000          5$:   .WORD     0          ;ADDRESS OF MESSAGE
66 021274 000207          RTS       PC         ;RETURN TO MAIN ROUTINE
67
68 021276 005737 001342  LINE2A: TST      BADSEC    ;PRINT THE BAD SECTOR LINE ?
69 021302 001404          BEQ      LINE2B     ;BR IF NOT
70 021304 104414 001203  DISPLY   , $CRLF    ;CR-LF
71 021310 104414 073727  DISPLY   , LINE2S   ;ERROR ADDRESS DEFINED AS BAD AREA
72 021314 104414 001203  LINE2B: DISPLY   , $CRLF    ;CR-LF
73 021320 104414 073240  DISPLY   , DH14     ;STANDARD RM REGISTER HEADER
74 021324 104414 075555  DISPLY   , BLNKS1   ;TYPE 1 BLANK
75 021330 013746 001324  MOV      UNIT,-(SP) ;PUT THE DRIVE NUMBER ON THE STACK
76 021334 004737 023176  JSR      PC,LINDEC ;TYPE DRIVE NUMBER
77 021340 104414 075554  DISPLY   , BLNKS2   ;TYPE 2 BLANKS
78 021344 012705 073600  MOV      #DT14,R5   ;REGISTER INDEXES
79 021350 004737 021532  JSR      PC,3$     ;PRINT THE REGISTERS
80 021354 032777 000040 157572 BIT      #SW05,@SWR ;PRINT THE OPTIONAL REGISTERS ?
81 021362 001031          BNE      1$        ;BR IF NOT
82 021364 104414 073342  DISPLY   , DH15     ;
83 021370 012705 073622  MOV      #DT15,R5   ;SECOND DATA LINE
84 021374 004737 021532  JSR      PC,3$     ;PRINT THEM
85 021400 104414 073440  DISPLY   , DH16     ;
86 021404 012705 073644  MOV      #DT16,R5   ;THIRD DATA LINE
87 021410 004737 021532  JSR      PC,3$     ;PRINT THE REGISTERS
88 021414 013746 001234  MOV      $CPUOP,-(SP);CHECK THE CPU (RH) TYPE
89 021420 042716 003777  BIC      #^C174000,(SP);LEAVE THE CPU BITS
90 021424 022726 030000  CMP      #30000,(SP)+;SEE IF RM7?
91 021430 001006          BNE      1$        ;BR IF NO
92 021432 104414 073510  DISPLY   , DH17     ;
93 021436 012705 073660  MOV      #DT17,R5   ;OPTIONAL FOURTH DATA LINE
94 021442 004737 021532  JSR      PC,3$     ;PRINT THE REGISTERS
95 021446 032760 000100 000016 1$:  BIT      #BIT6,$STATUS(RO);DATA ERROR ?
96 021454 001422          BEQ      2$        ;BR IF NOT
97 021456 016046 000020  MOV      $WRDL(RO),-(SP);TRANSFER WRD CNT
98 021462 066016 002140  ADD      $RMWC(RO),(SP);ADD REMAINING WORD COUNT
99 021466 006316          ASL      (SP)       ;CONVERT TO AN BYTE INCREMENT
100 021470 066016 000006  ADD      $BUF(RO),(SP);BUFFER STARTING ADDRESS
101 021474 022660 002142  CMP      (SP)+,$RMB(RO);CORRECT BUFFER ADDRESS ?
102 021500 001410          BEQ      2$        ;BR IF YES
103 021502 104414 072665  DISPLY   , EM46     ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
104 021506 104414 001203  DISPLY   , $CRLF    ;CR-LF
105 021512 004737 021624  JSR      PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
106 021516 004737 022236  JSR      PC,LINE4  ;PRINT LINE 4 OF ERROR MESSAGE
107 021522
108 021522 012605          2$:   MOV      (SP)+,R5   ;;POP STACK INTO R5
109 021524 012604          MOV      (SP)+,R4   ;;POP STACK INTO R4
110 021526 012603          MOV      (SP)+,R3   ;;POP STACK INTO R3
111 021530 000207          RTS       PC       ;RETURN TO ERROR PROCESSING ROUTINE
112 021532 012546          3$:   MOV      (R5)+,-(SP) ;PUT THE REGISTER INDEX ON THE STACK
113 021534 060016          ADD      RO,(SP)   ;ADD DRIVE'S TABLE ADDRESS

```



```

170 021720 116016 000011      MOVB   $TRK(RO), (SP)      ; TRACK TO STACK
171 021724 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
172 021730 104414 075554      DISPLY ,BLNKS2           ; TYPE 2 BLANKS
173 021734 104414 074174      DISPLY ,LINSS3           ; 'START SEC = '
174 021740 005046              CLR    -(SP)              ; CLEAR STACK
175 021742 116016 000010      MOVB   $SEC(RO), (SP)    ; SECTOR ADDR TO STACK
176 021746 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
177 021752 104414 001203      DISPLY , $CRLF
178 021756 000207              RTS    PC
179
180                          ; PRINT LINE 3F OF ERROR MESSAGE
181                          ; 'RMDA = XXXXXX  RMCA - XXXXXX'
182
183 021760 032777 000040 157166 LINE3F: BIT   #SW5,@SWR      ; SWITCH 5 SET ?
184 021766 001420              BFO    1$                ; BR IF NOT
185 021770 104414 074131      DISPLY ,LINDA3           ; 'RMDA = '
186 021774 016046 002144      MOV    $RMDA(RO), -(SP)  ; PUT SECTOR/TRACK ADDRESS ON THE STACK
187 022000 004737 023144      JSR    PC, LINOCT        ; TYPE IT
188 022004 104414 075554      DISPLY ,BLNKS2           ; TYPE 2 BLANKS
189 022010 104414 074120      DISPLY ,LINCA3           ; ' RMDC = '
190 022014 016046 002172      MOV    $RMDC(RO), -(SP)  ; PUT DESIRED CYLINDER ADDRESS ON THE STACK
191 022020 004737 023144      JSR    PC, LINOCT        ; TYPE IT
192 022024 104414 001203      DISPLY , $CRLF
193 022030 000207              1$:   RTS    PC
194
195                          ; 'CCC TT SS  PREV ADR = CCC TT SS'
196
200 022032 004737 023216      LIN3.1: JSR   PC, READDR   ; DECREMENT TRACK AND SECTOR ADDRESS
201 022036 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
202 022042 104414 073776      DISPLY ,T                 ; PRINT ' T '
206 022046 004737 023176      JSR    PC, LINDEC         ; TYPE TRACK IN DECIMAL
207 022052 104414 074017      DISPLY ,S                 ; PRINT ' S '
208 022056 004737 023176      JSR    PC, LINDEC         ; TYPE SECTOR ADDRESS
209 022062 104414 074022      DISPLY ,LIMP3            ; PRINT 'PREV ADDR'
210 022066 016046 000034      MOV    $PREVA+2(RO), -(SP) ; PREVIOUS CYLINDER
211 022072 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
212 022076 104414 073776      DISPLY ,T                 ; PRINT ' T '
213 022102 005046              CLR    -(SP)              ; MAKE ROOM ON THE STACK
214 022104 116016 000033      MOVB   $PREVA+1(RO), (SP) ; PREVIOUS TRACK ADDRESS
215 022110 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
216 022114 104414 074017      DISPLY ,S                 ; PRINT ' S '
217 022120 005046              CLR    -(SP)              ; MAKE ROOM ON THE STACK
218 022122 116016 000032      MOVB   $PREVA(RO), (SP)   ; PREVIOUS SECTOR ADDRESS
219 022126 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
220 022132 104414 001203      DISPLY , $CRLF
221 022136 000207              RTS    ; PC
222
223                          ; 'START CYL = CCC  END CYL = CCC'
224
225 022140 104414 074041      LIN3.3: DISPLY ,LINS3      ; LINE '3B & 3C' ENTRANCE
226 022144 016046 000034      MOV    $PREVA+2(RO), -(SP) ; PREVIOUS CYLINDER
227 022150 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
228 022154 104414 074055      DISPLY ,LINS3            ; PRINT 'END CYL'
229 022160 016046 002172      MOV    $RMDC(RO), -(SP)   ; PRESENT CYLINDER
230 022164 004737 023176      JSR    PC, LINDEC         ; TYPE IT IN DECIMAL
231 022170 000207              RTS    PC
232

```

```

233          ;'ACTUAL CYL = CCC   TRK   TT'
234
235 022172  104414  074071  LIN3.4: DISPLY  ,LINA3      ;PRINT 'ACTUAL'
236 022176  013746  101066      MOV    CYLNDR,-(SP)  ;ACTUAL CYLINDER
237 022202  042716  010000      BIC   #BIT12,(SP)  ;CLEAR THE FORMAT BIT
238 022206  004737  023176      JSR   PC,LINDEC    ;TYPE IT IN DECIMAL
239 022212  104414  074110      DISPLY ,LINT3      ;PRINT TRACK
240 022216  005046      CLR   -(SP)        ;CLEAR STACK WORD
241 022220  116016  002145      MOVB  $RMDA+1(RO),(SP) ;PUT TRACK ON STACK
242 022224  004737  023176      JSR   PC,LINDEC    ;TYPE IT IN DECIMAL
243 022230  104414  001203      DISPLY ,$CRLF      ;
244 022234  000207      RTS    PC
245
246          ;PRINT LINE 4 OF ERROR MESSAGE
247          ;'BUFFER ADR = XXXXXX   WRD CNT = XXXX   ACTUAL NMBR WRDS XFRD   XXX'
248
249 022236  032760  000100  000016  LINE4: BIT    #BIT06,$STATUS(RO) ;DATA ERROR ?
250 022244  001427      BEQ   1$          ;BR IF NOT
251 022246  104414  074210      DISPLY ,LINM4      ;'PRINT BUFFER'
252 022252  016046  0C0006      MOV   $BUF(RO),-(SP) ;BUFFER ADDR ON STACK
253 022256  004737  023144      JSR   PC,LINOC1    ;CONVERT TO OCTAL & PRINT
254 022262  104414  074226      DISPLY ,LINS4      ;PRINT 'WRD CNT'
255 022266  016046  000020      MOV   $WRDL(RO),-(SP) ;WORD LENGTH SIZE(WORD COUNT)
256 022272  004737  023176      JSR   PC,LINDEC    ;TYPE IT IN DECIMAL
257 022276  104414  074242      DISPLY ,LINX4      ;'ACTUAL NMBR WRDS XFRD = '
258 022302  016046  002142      MOV   $RMB4(RO),-(SP) ;VALUE IN BUFFER ADDR REGISTER
259 022306  166016  000006      SUB   $BUF(RO),(SP) ;SUBTRACT STARTING ADDRESS
260 022312  006216      ASR   (SP)        ;CONVERT INTO A WORD COUNT
261 022314  004737  023176      JSR   PC,LINDEC    ;TYPE IT IN DECIMAL
262 022320  104414  001203      DISPLY , $CRLF    ;CR-LF
263 022324  000207      1$: RTS    PC      ;RETURN
264
265          ;PRINT LINE 5 OF ERROR MESSAGE
266          ;'GOOD DATA = XXXXXX   BAD DATA = XXXXXX   SECT POS = XXX'
267
268 022326  104414  074274  LINES: DISPLY  ,LIND5      ;PRINT 'GOOD DATA'
269 022332  162760  000002  002142  SUB    #2,$RMB4(RO) ;BACK THE ADDRESS UP
270 022340  013746  001234      MOV   $CPUOP,-(SP) ;CHECK THE CPU (RH) TYPE
271 022344  042716  003777      BIC   #^C17400,(SP) ;LEAVE THE CPU BITS
272 022350  022726  030000      CMP   #30000,(SP)+ ;SEE IF RH70
273 022354  001012      BNE   1$          ;BR IF NO
274 022356  162760  000004  002142  SUB    #4,$RMB4(RO) ;BACKUP THE BUFFER POINTER
275 022364  032760  004000  002210  BIT   #BIT11,$RMCS3(RO) ;SEE WHICH WORD HALF DIDN'T COMPARE
276 022372  001403      BEQ   1$          ;IF EQ, EVEN HALF DIDN'T COMPARE
277 022374  162760  000002  002142  SUB    #2,$RMB4(RO) ;BACKUP THE BUFFER POINTER AGAIN
278 022402  017046  002142  1$: MOV   @RMB4(RO),-(SP) ;'GOOD' DATA - AT THE BUFFER LOCATION
279 022406  004737  023144      JSR   PC,LINOC1    ;TYPE IT
280 022412  104414  074310      DISPLY ,LINS5      ;PRINT 'BAD DATA'
281 022416  016046  002160      MOV   $RMD5(RO),-(SP) ;BAD DATA FROM BUFFER
282 022422  004737  023144      JSR   PC,LINOC1    ;TYPE IT
283 022426  016046  002140      MOV   $RMWC(RO),-(SP) ;WORD LENGTH ON STACK
284 022432  066016  000020      ADD   $WRDL(RO),(SP) ;MAKE INTO A POSITIVE NUMBER
285 022436  005046      CLR   -(SP)        ;UPPER DIVIDEND TO ZERO
286 022440  016046  000022      MOV   $SSEC(RO),-(SP) ;SECTOR SIZE ON THE STACK
287 022444  004737  031710      JSR   PC,$DIV      ;DIVIDE WORDS XFERED BY SECTOR SIZE
288 022450  012616      MOV   (SP)+(SP)    ;MOVE REMAINDER UP THE STACK
289 022452  104414  074325      DISPLY ,LINP5      ;PRINT 'SECT POS'

```

294 022456 004737 023176
 295 022462 104414 001203
 296 022466 000207

```
JSR PC,LINDEC ;TYPE THE POSITION
DISPLY , $CRLF
RTS PC
```

297
 298
 299
 300

```
;PRINT LINE 5A OF THE ERROR MESSAGE
;'HEADER FROM ERROR SECTOR XXXXXX XXXXXX XXXXXX XXXXXX'
```

301 022470
 302 022470 104414 074342
 307 022474 013746 101066

```
LINE5A:
2$: DISPLY ,LINS5 ;'HEADER CONTENTS OF ERROR SECTOR'
```

022500 004737 023144
 022504 104414 075554
 022510 013746 101070
 022514 004737 023144
 022520 104414 075554

```
MOV CY,LNDR,-(SP) ;HEADER POSITION
JSR PC,LINOC1 ;TYPE IT
DISPLY ,BLNKS2 ;TYPE 2 BLANKS
MOV CY,LNDR+2,-(SP) ;HEADER POSITION +2
JSR PC,LINOC1 ;TYPE IT
DISPLY ,BLNKS2 ;TYPE 2 BLANKS
```

308 022524 104414 075557
 309 022530 104414 001203
 310 022534 000207

```
3$: DISPLY , $CRLF
RTS PC
```

311
 312
 313
 314

```
;PRINT LINE 5B OF ERROR MESSAGE
;'RMEC1 = XXXXXX RMEC2 = XXXXXX'
```

315 022536 104414 074375
 316 022542 016046 002202
 317 022546 004737 023144

```
LINE5B: DISPLY ,LINEP5 ;'RMEC1 = '
MOV $RMEC1(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
```

318 022552 104414 075554
 319 022556 104414 074405
 320 022562 016046 002204
 321 022566 004737 023144
 322 022572 104414 001203

```
JSR PC,LINOC1 ;TYPE IT
DISPLY ,BLNKS2 ;TYPE 2 BLANKS
DISPLY ,LINEO5 ;' RMEC2 = '
MOV $RMEC2(RO),-(SP) ;PUT REGISTER CONTENTS ON THE STACK
JSR PC,LINOC1 ;TYPE IT
DISPLY , $CRLF
RTS PC ;RETURN
```

323 022576 000207
 324
 325
 326
 327

```
;PRINT LINE 6 OF ERROR MESSAGE
;'SECTOR IS ECC CORRECTABLE'
```

328 022600 104414 074417
 329 022604 104414 001203
 330 022610 000207

```
LINE6: DISPLY ,LINB6 ;ECC CORRECTABLE
DISPLY , $CRLF
RTS PC
```

331
 332
 333
 334

```
;PRINT LINE 6A OF THE ERROR MESSAGE
;'SECTOR READ CORRECTLY AT OFFSET N'
```

335 022612 104414 074452
 336 022616 000411

```
LINE6A: DISPLY ,LINC6 ;PRINT 'READ CORRECTLY AT OFFSET N'
BR LIN6.1 ;TYPE THE REST OF THE LINE
```

337
 338
 339
 340

```
;PRINT LINE 6B OF THE ERROR MESSAGE
;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
```

341 022620 104414 074417
 342 022624 000406

```
LINE6B: DISPLY ,LINB6 ;PRINT 'SECTOR IS ECC CORRECTABLE '
BR LIN6.1
```

343
 344
 345
 346

```
;PRINT LINE 6C OF THE ERROR MESSAGE
;'CORRECTED ON NTH RETRY'
```

347 022626 104414 074501
 348 022632 000414
 349

```
LINE6C: DISPLY ,LING6 ;'CORRECTED ON NTH RETRY'
BR LIN6.2 ;TYPE THE REST OF THE LINE
```

```

350 ;PRINT LINE 6D OF THE ERROR MESSAGE
351 ;'UNCORRECTABLE AFTER N RETRIES'
352
353 022634 104414 074531 LINE6D: DISPLY ,LIN06 ;'UNCORRECTABLE AFTER N RETRIES'
354 022640 000411 BR LIN6.2 ;FINISH
355
356 ;TYPE THE OFFSET VALUE IN MICRO-INCHES
357
358 022642 006301 LIN6.1: ASL R1 ;DOUBLE THE OFFSET TABLE INDEX
359 022644 016137 002420 022654 MOV OFMTBL(R1),1$ ;ADDRESS OF OFFSET POSITION MESSAGE
360 022652 104414 DISPLY
361 022654 000000 1$: .WORD 0 ;OFFSET VALUE
362 022656 104414 001203 DISPLY ,$CRLF
363 022662 000207 RTS PC
364
365 ;RETRY COUNT TYPEOUT
366
367 022664 005046 LIN6.2: CLR -(SP) ;CLEAR STACK
368 022666 113716 001331 MOVB RETRY+1,(SP) ;RETRY COUNT
369 022672 004737 023176 JSR PC,LINDEC ;TYPE IT IN DECIMAL
370 022676 104414 074517 DISPLY ,LINR6 ;'RETRY'
371 022702 104414 001203 DISPLY ,$CRLF
372 022706 000207 RTS PC
373
374 ;PRINT LINE 7 OF THE ERROR MESSAGE
375 ;'TOTAL ERRORS:XXX WOFL:X WRDS WRITN:XXXXXXX ROFL:0 WRDS READ:XXXXXXX'
376
377 022710 104414 074643 LINE7: DISPLY ,LIN7T ;TOTAL ERRORS
378 022714 016046 000072 MOV $TOTAL(RO),-(SP) ;TO STACK
379 022720 004737 023176 JSR PC,LINDEC ;TYPE IT IN DECIMAL
380 022724 104414 074661 DISPLY ,LIN7OX ;PRINT 'WTOFL'
381 022730 016046 000056 MOV $WTOFL(RO),-(SP) ;PUSH $WTOFL(RO) ON STACK
382 022734 004737 023176 JSR PC,LINDEC ;TYPE IT IN DECIMAL
383 022740 104414 074671 DISPLY ,LIN7X ;PRINT 'WRDS WRITN'
384 022744 012746 000060 MOV #$WRITN, -(SP) ;ADDRESS OF LOW WORD ON STACK
385 022750 060016 ADD RO,(SP)
386 022752 004737 037122 JSR PC,$DB2D ;CONVERT
387 022756 004737 032234 JSR PC,$SUPRL ;PRINT
388 022762 104414 074706 DISPLY ,LIN7OR ;PRINT 'ROFL'
389 022766 016046 000064 MOV $RDOFL(RO),-(SP) ;PUSH $RDOFL(RO) ON STACK
390 022772 004737 023176 JSR PC,LINDEC ;TYPE IT IN DECIMAL
391 022776 104414 074716 DISPLY ,LIN7R ;'WRDS READ'
392 023002 012746 000066 MOV #$READ, -(SP) ;LOW WORD ADDRESS
393 023006 060016 ADD RO,(SP)
394 023010 004737 037122 JSR PC,$DB2D ;CONVERT
395 023014 004737 032234 JSR PC,$SUPRL ;PRINT IT
396 023020 104414 001203 DISPLY ,$CRLF ;CR-LF
397 023024 032777 100000 156122 BIT #$SW15,@SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15
398 023032 001401 BEQ 1$ ;BR IF NOT
399 023034 000000 HALT ;SWITCH 15 HALT
400 023036 000207 1$: RTS PC
401
402 ;PRINT LINE 7A OF ERROR MESSAGE
403 ;'TOTAL SEEKS=XXXXX TOTAL MISFOS ERR = XXX TOTAL SKI- XXX'
404
405 023040 104414 074603 LINE7A: DISPLY ,LIN7P ;'TOTAL SEEKS = '
406 023044 012746 000052 MOV , $POSIT, -(SP) ;TOTAL SEEKS

```



```

421 023050 060016          ADD    RO,(SP)          ;DEVICE TABLE ADDRESS
422 023052 004737 037122  JSR    PC,$DB2D        ;CONVERT THE SEEK COUNT
423 023056 004737 032234  JSR    PC,$SUPRL       ;PRINT IT
424 023062 104414 074556  DISPLY ,LIN7M          ;' TOTAL MISPOS ERR - '
425 023066 016046 000102  MOV    $MISPO(RO),-(SP) ;TOTAL ERRORS
426 023072 004737 023176  JSR    PC,LINDEC       ;TYPE IT IN DECIMAL
427 023076 104414 074621  DISPLY ,LIN7S          ;' TOTAL SKI ERR = '
428 023102 016046 000100  MOV    $SKI(RO),-(SP)  ;CONVERT & PRINT IT
429 023106 004737 023176  JSR    PC,LINDEC       ;TYPE IT IN DECIMAL
430 023112 104414 001203  DISPLY ,$CRLF          ;CR-LF
431 023116 032777 100000 156030 BIT    #SW15,@SWR      ;SEE IF HALT ON ERROR - SWITCH 15 SET
432 023124 001401          BEQ    1$              ;BR IF NOT
433 023126 000000          HALT                    ;SWITCH 15 HALT
434 023130 000207          1$:    RTS    PC

435
436          ;PRINT LINE 8 OF THE ERROR MESSAGE
437          ;'DIFFERENT ERROR DURING RETRY'
438
439 023132 104414 074732  LINE8: DISPLY ,LIN8M
440 023136 004737 021126  JSR    PC,LIN2        ;PRINT LINE 2 OF ERROR MESSAGE
441 023142 000207          RTS    PC

442
443          ;OCTAL TYPEOUT ROUTINE
444          ;CALL:
445          ;   MOV    NUM,-(SP)          ;PUT THE NUMBER ON THE STACK
446          ;   JSR    PC,LINOC
447          ;   RETURN
448
449 023144 016646 000002  LINOC: MOV    2(SP),-(SP) ;PUT NUMBER IN PROPER LOCATION ON STACK
450 023150 004737 033160  JSR    PC,$SB2D       ;CONVERT THE NUMBER TO OCTAL
451 023154 012637 023170  MOV    (SP)+,1$       ;GET THE ADDRESS OF THE ASCII STRING
452 023160 062737 000005 023170 ADD    #5.,1$         ;ADDRESS THE LAST 6 ASCII DIGITS
453 023166 104414          DISPLY                    ;TYPE IT
454 023170 000000          1$:    .WORD    0          ;ADDRESS
455 023172 012616          MOV    (SP)+,(SP)      ;CORRECT THE STACK
456 023174 000207          RTS    PC          ;RETURN
457
458          ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
459          ;LEADING ZERO SUPPRESSION
460          ;CALL:
461          ;   MOV    NUM,-(SP)          ;PUT THE NUMBER ON THE STACK
462          ;   JSR    PC,LINDEC
463          ;   RETURN
464
465 023176 016646 000002  LINDEC: MOV    2(SP),-(SP) ;SET UP STACK FOR CONVERT
466 023202 004737 033130  JSR    PC,$SB2D       ;CONVERT IT TO DECIMAL
467 023206 004737 032234  JSR    PC,$SUPRL       ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
468 023212 012616          MOV    (SP)+,(SP)      ;RESTORE STACK POINTER
469 023214 000207          RTS    PC

```

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

.SBTTL GENERAL SUPPORT SUBROUTINES

:DECREMENT THE SECTOR-TRACK ADDRESS

```
:CALL:
      MOV    #DPB,R0          ;DPB ADDRESS
      JSR    PC,READDR
      RETURN
```

:ON RETURN THE STACK CONTAINS THE FOLLOWING:

```
4(SP) = SECTOR ADDRESS
2(SP) = TRACK ADDRESS
(SP) = CYLINDER ADDRESS
```

```
READDR: JSR    PC,GETLMT      ;GET ADDRESS LIMITS
        SUB    #6,SP          ;DECREMENT THE STACK POINTER
        MOV    6(SP),(SP)     ;MOVE THE RETURN ADDR DOWN THE STACK
        CLR    6(SP)         ;CLEAR STACK FOR SECTOR
        CLR    4(SP)         ;CLEAR STACK FOR TRACK
        MOVB  $RMDA(R0),6(SP) ;SECTOR ON STACK
        MOVB  $RMDA+1(R0),4(SP) ;TRACK ADDRESS
        MOV    $RMDC(R0),2(SP) ;CYLINDER ADDRESS
1$:     TST    6(SP)          ;SECTOR 0 ?
        BEQ   2$             ;BRANCH IF SO
        DECB  6(SP)         ;DECREMENT ONE SECTOR
        BR    4$             ;BRANCH TO EXIT
2$:     TST    4(SP)          ;ALSO ON TRACK 0 ?
        BEQ   3$             ;BRANCH IF SO
        MOVB  SECLMT,6(SP)   ;LAST SECTOR
        DECB  4(SP)         ;DECREMENT ONE TRACK
        BR    4$             ;EXIT
3$:     MOVB  SECLMT,6(SP)   ;LAST SECTOR
        MOVB  TRKLMT,4(SP)  ;GET LAST TRACK
        DEC   2(SP)         ;DECREMENT ONE CYLINDER COUNT
4$:     RTS    PC            ;RETURN
```

:ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS

```
CKCLK: MOV    #-1,CLKFLG    ;CLEAR CLOCK AVAILABILITY FLAG
        MOV    #-1,PCLOCK   ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
        MOV    ERRVEC,-(SP) ;:PUSH ERRVEC ON STACK
        MOV    #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
        TST   @ $LKCSR      ;CHECK FOR KW11-P
        CLR   CLKFLG       ;SET CLOCK AVAILABILITY FLAG
        CLR   PCLOCK       ;SET KW11-P CLOCK FLAG
        MOV   $LPVEC,R1     ;KW11-P VECTOR ADDRESS
        MOV   #CLOCK,(R1)+  ;SET UP KW11-P VECTOR
        MOV   #300,(R1)    ;PSW - PRI 6
        MOV   #-1667,@ $LKCSB ;LOAD COUNTER BUFFER WITH 16.67
        MOV   #131,@ $LKCSR ;SET CLOCK - CNT UP, 10US, CONT INT
        BR    CKCLK3

CKCLK1: MOV   #1$,(SP)     ;SETUP RETURN ADDRESS
        RTI

1$:     MOV   #CKCLK2,ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
        TST  @ $LKS        ;LOOK FOR KW11-L
        CLR  CLKFLG       ;SET CLOCK FLAG
        MOV  $LLVEC,R1     ;KW11-L VECTOR ADDRESS
```

```
58 023470 012721 025074      MOV      #CLOCK,(R1)+      ;SET UP KW11-L VECTOR
59 023474 012711 000300      MOV      #300,(R1)        ;PSW - PRI 6
60 023500 012777 000100 155576  MOV      #100,@$LKS       ;SET KW11-L INTERRUPT
61 023506 000415              BR        CKCLK3
62
63 023510 012716 023516      CKCLK2: MOV      #1$, (SP)      ;SETUP RETURN ADDRESS
64 023514 000002              RTI
65 023516 104401 076422      1$:     TYPE      ,NEDCLK      ;'P OR L CLOCK MUST BE ON SYSTEM'
66 023522 105737 001150      TSTB     $AUTOB          ;RUNNING IN AUTO MODE ?
67 023526 001402              BEQ      2$              ;BR IF NOT
68 023530 000137 031612      JMP      $GET42          ;ABORT PROGRAM
69 023534 000000      2$:     HALT
70 023536 000137 003634      JMP      START          ;HALT
71 023542 012637 000004      CKCLK3: MOV      (SP)+,ERRVEC ;TRY AGAIN
72 023546 000207              RTS                     ;RESTORE THE ERROR VECTOR
PC
```

```

1
2
3
4
5
6
7
8
9
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
58
59
60
61
62
63
64

```

```

;ROUTINE TO DISPLAY STATISTICS FOR ASSIGNED DRIVES
;CALL:
;      JSR      PC,STATPR
;      RETURN
;
;STATPR:
;      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
;      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
;      TST      ASNLST        ;ANY DRIVES ASSIGNED ?
;      BEQ      5$           ;BR IF NOT
;      TYPE     ,$CRLF        ;CR-LF
;      JSR      PC,SHDTYP     ;TYPE THE HEADING
;      CLR      R4           ;CLEAR THE DRIVE INDEX
;      ASL      R4           ;CHANGE TO INDEX WORDS
;      MOV      BLKADR(R.),R0 ;GET THE DRIVE'S BLOCK ADDRESS
;      ASR      R4           ;RESTORE R4
;      BITB     ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
;      BEQ      4$           ;BR IF NOT
;      JSR      PC,SDETAL     ;TYPE THE PERFORMANCE SUMMARY
;      INC      R4           ;INCREMENT THE INDEX
;      CMP      R4,#8.       ;FINISHED ?
;      BNE     3$           ;BR IF NO
;
;      MOV      (SP)+,R4      ;;POP STACK INTO R4
;      MOV      (SP)+,R0      ;;POP STACK INTO R0
;      RTS      PC           ;RETURN

```

```

;ROUTINE TO TYPE THE PERFORMANCE SUMMARY (STATISTICS) FOR AN INDIVIDUAL
;DRIVE.
;CALL:
;      MOV      #DPB,R0      ;DPB ADDRESS
;      JSR      PC,SUMARY
;      RETURN
;
;SUMARY:
;      MOV      R0,-(SP)      ;SAVE R0
;      MOV      R4,-(SP)      ;SAVE R4
;      JSR      PC,SHDTYP     ;TYPE THE HEADING
;      CLR      R4           ;CLEAR R4 FOR DRIVE NUMBER
;      MOV      (R0),R4      ;DRIVE NUMBER
;      JSR      PC,SDETAL     ;TYPE THE STATISTICS
;      MOV      (SP)+,R4      ;RESTORE R4
;      MOV      (SP)+,R0      ;RESTORE R0
;      RTS      PC           ;RETURN

```

```

;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
;CALL:
;      JSR      PC,SHDTYP
;      RETURN
;
;SHDTYP:
;      JSR      R5,TYPRI4     ;TYPE SUMMARY HEADER
;      SUMHD
;      RTS      PC           ;RETURN

```

```

;TYPE THE PERFORMANCE SUMMARY
;CALL:
;      MOV      #DRIVE,R4    ;DRIVE NUMBER

```

```

65      :      MOV      #DPB,RO      ;DPB ADDRESS
66      :      RETURN
67
68 023674      SDETAL:
023674      TYPE      ,65$          ;;TYPE ASCIZ STRING
023700      BR        64$          ;;GET OVER THE ASCIZ
        ;;65$: .ASCIZ <CRLF>/TIME /
64$:
69 023712      JSR      PC,$TIME      ;TYPE ELAPSED TIME
70
71      ;TYPE LINE 2 OF SUMMARY
72 023716      TYPE      ,SCLF          ;CR-LF
023722      TYPE      ,UNMSG          ;TYPE 'DRIVE'
74 023726      MOV      R4,-(SP)      ;;SAVE R4 FOR TYPEOUT
023730      TYPOS          ;;GO TYPE--OCTAL ASCII
023732      .BYTE      2              ;;TYPE 2 DIGIT(S)
023733      .BYTE      0              ;;SUPPRESS LEADING ZEROS
75 023734      TYPE      ,BLNKS1      ;TYPE 1 BLANK
76 023740      TYPE      ,DASH          ;TYPE '-'
77 023744      TYPE      ,BLNKS1      ;TYPE 1 BLANK
78 023750      MOV      #RMO3,2$      ;ADDRESS OF RM03 MESSAGE
79 023756      CMPB     #4,DRVTYP(R4) ;IS DEVICE AN RM03 ?
80 023764      BEQ      1$              ;BR IF YES
81 023766      MOV      #RMO2,2$      ;ADDRESS OF RM02 MESSAGE
82 023774      CMPB     #5,DRVTYP(R4) ;IS DEVICE AN RM02 ?
83 024002      BEQ      1$              ;BR IF YES
84 024004      MOV      #RMO5,2$      ;ADDRESS OF RM05 MESSAGE
85 024012      TYPE      ,DRIVE        ;TYPE THE DRIVE TYPE MESSAGE
86 024014      .WORD      0              ;MESSAGE ADDRESS HERE
87 024016      TYPE      ,COMMA        ;TYPE ','
88
89 024022      TYPE      ,67$          ;;TYPE ASCIZ STRING
024026      BR        66$          ;;GET OVER THE ASCIZ
        ;;67$: .ASCIZ / PASS /
66$:
90 024040      MOV      $PASSC(RO),-(SP) ;PUT THE PASS COUNT ON THE STACK
91 024044      JSR      PC,$SB2D      ;CONVERT IT
92 024050      JSR      R5,REPLZ      ;TYPE IT
93 024054      .WORD      3              ;TYPE 3 DIGITS
94 024056      TYPE      ,PERIOD        ;TYPE '.'
95 024062      TYPE      ,BLNKS1      ;TYPE 1 BLANK
96 024066      JSR      PC,TY MBA      ;TYPE MBA SERIAL NUMBER
97 024072      TYPE      ,BLNKS1      ;TYPE 1 BLANK
98 024076      JSR      PC,TYPACK      ;TYPE PACK SERIAL NUMBER
99
100      ;TYPE LINE 3 OF SUMMARY
101 024102      TYPE      ,69$          ;;TYPE ASCIZ STRING
024106      BR        68$          ;;GET OVER THE ASCIZ
        ;;69$: .ASCIZ <CRLF><LF>/WT OFLOW /
68$:
102 024126      MOV      $WTOFL(RO),-(SP) ;;SAVE $WTOFL(RO) FOR TYPEOUT
024132      TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
103 024134      TYPE      ,PERIOD        ;TYPE '.'
104 024140      TYPE      ,71$          ;;TYPE ASCIZ STRING
024144      BR        70$          ;;GET OVER THE ASCIZ
        ;;71$: .ASCIZ / WRDS WRITN /
70$:
024164

```

```

105 024164 010046      MOV      RO,-(SP)      ;GET ADDRESS OF DPB
106 024166 062716 000060  ADD      #SWRTN,(SP)  ;POINT TO LOW NUMBER OF WRDS WRITTEN
107 024172 004737 037122  JSR      PC,$DB2D    ;CONVERT DECIMAL NUMBER
108 024176 004737 032150  JSR      PC,SUPRS    ;SUPPRESS LEADING ZEROS AND TYPE
109 024202 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
110
111
112 024206 104401 024214  ;TYPE LINE 4 OF SUMMARY
      TYPE     ,73$    ;:TYPE ASCIZ STRING
      BR      72$     ;:GET OVER THE ASCIZ
      ;:73$: .ASCIZ <CRLF>/RD OFLOW /
      72$:
113 024230 016046 000064  MOV      $RDOFL(RO),-(SP) ;:SAVE $RDOFL(RO) FOR TYPEOUT
      TYPDS   ;:GO TYPE--DECIMAL ASCII WITH SIGN
114 024234 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
115 024242 104401 024250  TYPE     ,75$    ;:TYPE ASCIZ STRING
      BR      74$     ;:GET OVER THE ASCIZ
      ;:75$: .ASCIZ / WRDS READ /
      74$:
116 024266 010046      MOV      RO,-(SP)      ;GET ADDRESS OF DPB
117 024270 062716 000066  ADD      #SREAD,(SP)  ;POINT TO LOW NUMBER OF WRDS READ
118 024274 004737 037122  JSR      PC,$DB2D    ;CONVERT DECIMAL NUMBER
119 024300 004737 032150  JSR      PC,SUPRS    ;SUPPRESS LEADING ZEROS AND TYPE
120 024304 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
121
122
123 024310 104401 024316  ;TYPE LINE 5 OF SUMMARY
      TYPE     ,77$    ;:TYPE ASCIZ STRING
      BR      76$     ;:GET OVER THE ASCIZ
      ;:77$: .ASCIZ <CRLF>/SEEKS /
      76$:
124 024326 010046      MOV      RO,-(SP)      ;PUT $POSIT ON THE STACK
125 024330 062716 000052  ADD      #$POSIT,(SP) ;POINT TO LOW NUMBER OF SEEK COUNT
126 024334 004737 037122  JSR      PC,$DB2D    ;CONVERT DECIMAL NUMBER
127 024340 004737 032150  JSR      PC,SUPRS    ;SUPPRESS LEADING ZEROS AND TYPE
128 024344 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
129
130
131
132
133
134
135
136
137
138
139 024350 104401 024356  ;TYPE LINES 6 AND 7 OF SUMMARY
      TYPE     ,79$    ;:TYPE ASCIZ STRING
      BR      78$     ;:GET OVER THE ASCIZ
      ;:79$: .ASCIZ <CRLF>/ERRORS:/
      78$:
140 024370 104401 024376  TYPE     ,81$    ;:TYPE ASCIZ STRING
      BR      80$     ;:GET OVER THE ASCIZ
      ;:81$: .ASCIZ <CRLF>/SOFT /
      80$:
141 024406 016046 000074  MOV      $SOFT(RO),-(SP) ;:SAVE $SOFT(RO) FOR TYPEOUT
      TYPDS   ;:GO TYPE--DECIMAL ASCII WITH SIGN
142 024414 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
143 024420 104401 024426  TYPE     ,83$    ;:TYPE ASCIZ STRING
      BR      82$     ;:GET OVER THE ASCIZ
      ;:83$: .ASCIZ / HARD /
      82$:
144 024436 016046 000076  MOV      $HARD(RO),-(SP) ;:SAVE $HARD(RO) FOR TYPEOUT
      TYPDS   ;:GO TYPE--DECIMAL ASCII WITH SIGN
145 024444 104401 076477  TYPE     ,PERIOD     ;TYPE '.'
146 024450 104401 024456  TYPE     ,85$    ;:TYPE ASCIZ STRING
      BR      84$     ;:GET OVER THE ASCIZ
      ;:85$: .ASCIZ / SKI /

```

147	024464	016046	000100
	024470	104405	
148	024472	104401	076477
149	024476	104401	024504
	024502	000404	
150	024514	016046	000102
	024520	104405	
151	024522	104401	076477
152	024526	104401	024534
	024532	000404	
153	024544	016046	000072
156	024550	166016	000074
	024554	166016	000076
	024560	166016	000100
	024564	166016	000102
157	024570	104405	
158	024572	104401	076477
159	024576	104401	001203
160	024602	000207	

```

84$: MOV $SKI(R0),-(SP) ;;SAVE $SKI(R0) FOR TYPEOUT
      TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,PERIOD ;;TYPE '.'
      TYPE ,87$ ;;TYPE ASCII STRING
      BR 86$ ;;GET OVER THE ASCII
      .:87$: .ASCII / MISP /
86$: MOV $MISPO(R0),-(SP) ;;SAVE $MISPO(R0) FOR TYPEOUT
      TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,PERIOD ;;TYPE '.'
      TYPE ,89$ ;;TYPE ASCII STRING
      BR 88$ ;;GET OVER THE ASCII
      .:89$: .ASCII / OTHER /
88$: MOV $TOTAL(R0),-(SP) ;;CALCULATE NUMBER OF OTHER ERRORS
      SUB $SOFT(R0),(SP) ;;SUBTRACT $SOFT FROM $TOTAL
      SUB $SHARD(R0),(SP) ;;SUBTRACT $SHARD FROM $TOTAL
      SUB $SKI(R0),(SP) ;;SUBTRACT $SKI FROM $TOTAL
      SUB $MISPO(R0),(SP) ;;SUBTRACT $MISPO FROM $TOTAL
      TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPE ,PERIOD ;;TYPE '.'
      TYPE ,$CRLF ;;CR-LF
      RTS PC
  
```

1
15
16

;ROUTINE TO INCREMENT \$SOFT

;NOTE: \$SOFT WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

024604	005737	001342	INCSOF:	TST	BADSEC		;SEE IF BAD TRK/SEC INDICATOR SET
024610	001006			BNE	3\$;BR IF IT'S SET, DON'T INCREMENT COUNT
024612	026027	000074		CMP	\$SOFT(RO),#77777		;IS \$SOFT ALREADY AT MAXIMUM ?
024620	103002			BHIS	3\$;BR IF IT IS
024622	005260	000074		INC	\$SOFT(RO)		;INCREMENT \$SOFT
024626	000207		3\$:	RTS	PC		;RETURN

17

;ROUTINE TO INCREMENT \$SHARD

;NOTE: \$SHARD WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

024630	005737	001342	INCHRD:	TST	BADSEC		;SEE IF BAD TRK/SEC INDICATOR SET
024634	001006			BNE	3\$;BR IF IT'S SET, DON'T INCREMENT COUNT
024636	026027	000076		CMP	\$SHARD(RO),#77777		;IS \$SHARD ALREADY AT MAXIMUM ?
024644	103002			BHIS	3\$;BR IF IT IS
024646	005260	000076		INC	\$SHARD(RO)		;INCREMENT \$SHARD
024652	000207		3\$:	RTS	PC		;RETURN

18

;ROUTINE TO INCREMENT \$SKI

;NOTE: \$SKI WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

024654	005737	001342	INCSKI:	TST	BADSEC		;SEE IF BAD TRK/SEC INDICATOR SET
024660	001006			BNE	3\$;BR IF IT'S SET, DON'T INCREMENT COUNT
024662	026027	000100		CMP	\$SKI(RO),#77777		;IS \$SKI ALREADY AT MAXIMUM ?
024670	103002			BHIS	3\$;BR IF IT IS
024672	005260	000100		INC	\$SKI(RO)		;INCREMENT \$SKI
024676	000207		3\$:	RTS	PC		;RETURN

19

;ROUTINE TO INCREMENT \$MISPO

;NOTE: \$MISPO WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

024700	005737	001342	INCMIS:	TST	BADSEC		;SEE IF BAD TRK/SEC INDICATOR SET
024704	001006			BNE	3\$;BR IF IT'S SET, DON'T INCREMENT COUNT
024706	026027	000102		CMP	\$MISPO(RO),#77777		;IS \$MISPO ALREADY AT MAXIMUM ?
024714	103002			BHIS	3\$;BR IF IT IS
024716	005260	000102		INC	\$MISPO(RO)		;INCREMENT \$MISPO
024722	000207		3\$:	RTS	PC		;RETURN

20

;ROUTINE TO INCREMENT \$TOTAL

;NOTE: \$TOTAL WILL NOT BE INCREMENTED BEYOND 77777 (32767.)

024724	005737	001342	INCTOT:	TST	BADSEC		;SEE IF BAD TRK/SEC INDICATOR SET
024730	001006			BNE	3\$;BR IF IT'S SET, DON'T INCREMENT COUNT
024732	026027	000072		CMP	\$TOTAL(RO),#77777		;IS \$TOTAL ALREADY AT MAXIMUM ?
024740	103002			BHIS	3\$;BR IF IT IS
024742	005260	000072		INC	\$TOTAL(RO)		;INCREMENT \$TOTAL
024746	000207		3\$:	RTS	PC		;RETURN


```

1
2
3
4 024750 005737 001312 $TIME: TST CLKFLG ;CLOCK ON THE SYSTEM ?
5 024754 001046 BNE 3$ ;BR IF NOT
6 024756 012737 000002 025024 MOV #2,2$ ;ASSUME 2 DIGITS TO TYPE
7 024764 013746 001344 MOV HOUR,-(SP) ;PUT 'HOURS' ON THE STACK
8 024770 021627 000144 CMP (SP),#100. ;100. HOURS OR MORE ?
9 024774 002407 BLT 1$ ;BR IF NO
10 024776 005237 025024 INC 2$ ;TYPE 3 DIGITS
11 025002 021627 001750 CMP (SP),#1000. ;1000. HOURS OR MORE ?
12 025006 002402 BLT 1$ ;BR IF NO
13 025010 005237 025024 INC 2$ ;TYPE 4 DIGITS
14 025014 004737 033130 1$: JSR PC,$SSE2D ;CONVERT TO DECIMAL
15 025020 004537 032334 JSR R5,FILLZ ;TYPE IT
16 025024 000000 2$: .WORD 0 ;NUMBER OF HOUR DIGITS TO TYPE
17 025026 104401 076670 TYPE ,COLON ;'.'
18 025032 013746 001346 MOV MINUTE,-(SP) ;PUT 'MINUTES' ON THE STACK
19 025036 004737 033130 JSR PC,$SB2D ;CONVERT TO DECIMAL
20 025042 004537 032334 JSR R5,FILLZ ;TYPE IT
21 025046 000002 .WORD 2 ;TYPE 2 DIGITS
22 025050 104401 076670 TYPE ,COLON ;'.'
23 025054 013746 001350 MOV SECOND,-(SP) ;PUT SECONDS ON THE STACK
24 025060 004737 033130 JSR PC,$SB2D ;CONVERT TO DECIMAL
25 025064 004537 032334 JSR R5,FILLZ ;TYPE IT
26 025070 000002 .WORD 2 ;TYPE 2 DIGITS
27 025072 000207 3$: RTS PC
28
29 ;CLOCK HANDLER ROUTINE
30
31 025074 005337 001352 CLOCK: DEC ONESEC ;INCREMENT THE ONE SECOND COUNTER
32 025100 001027 BNE 1$ ;BR IF A SECOND NOT COUNTED
33 025102 013737 001314 001352 MOV #2,ONESEC ;RESTORE THE VALUE
34 025110 005237 001350 INC SECOND ;COUNT THE SECOND
35 025114 022737 000074 001350 CMP #60.,SECOND ;AT MAXIMUM ?
36 025122 001016 BNE 1$ ;BR IF NOT
37 025124 005037 001350 CLR SECOND ;CLEAR THE SECOND'S COUNTER
38 025130 005237 001454 INC INTRVL+2 ;COUNT THE PERFORMANCE SUMMARY INTERVAL
39 025134 005237 001346 INC MINUTE ;COUNT THE MINUTE
40 025140 022737 000074 001346 CMP #60.,MINUTE ;AT MAXIMUM ?
41 025146 001004 BNE 1$ ;BR IF NOT
42 025150 005037 001346 CLR MINUTE ;CLEAR THE MINUTE'S COUNTER
43 025154 005237 001344 INC HOUR ;COUNT THE HOURS
44 025160 022737 000062 001314 1$: CMP #50.,HZ ;CPU RUNNING @ 50HZ ?
45 025166 001403 BEQ 2$ ;BR IF YES
46 025170 012746 000020 MOV #16.,-(SP) ;16MS ON THE STACK @ 60HZ
47 025174 000402 BR 3$
48 025176 012746 000024 2$: MOV #20.,-(SP) ;20MS ON THE STACK @ 50HZ
49 025202 004737 044254 3$: JSR PC,RMTMR ;DRIVER TIMER ROUTINE
50 025206 005737 001452 TST INTRVL ;DISPLAY THE PERFORMANCE SUMMARY ?
51 025212 001411 BEQ 4$ ;BR IF NOT
52 025214 023737 001452 001454 CMP INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
53 025222 001005 BNE 4$ ;BR IF NOT
54 025224 012737 177777 001316 MOV #-1,STATIN ;SET PERFORMANCE SUMMARY DISPLAY FLAG
55 025232 005037 001454 CLR INTRVL+2 ;CLEAR THE PERFORMANCE INTERVAL COUNTER
56 025236 000002 4$: RTI

```

```

1
2
3
4
5
6
7
8
9
10 025240 005737 040140      KSR:  TST      DTJW      ;ANY DATA TRANSFERS UNDER WAY ?
11 025244 100375              BPL      KSR           ;BR IF YES
12 025246 104412              KSR1:  SAVREG      ;SAVE THE REGISTERS
13 025250 012737 000200 177776  MOV      #PR4,PS      ;SET PRIORITY TO 4
14 025256 005037 001340      *$:  CLR      CFLAG     ;CLEAR THE 'CONTROL C' FLAG
15 025262 104401 001203      TYPE    ,$CRLF       ;CR-LF
16 025266 004737 024750      JSR     PC,$TIME     ;TYPE THE TIME
17 025272 004737 033226      JSR     PC,$TKINT    ;INITIALIZE TTY KEYBOARD
18 025276 104401 076567      TYPE    ,ENTCOM     ;'ENTER COMMAND'
19
20 025302 104411              RDLIN                    ;READ THE KEYBOARD
21 025304 012605              MOV      (SP)-,R5      ;GET ADDRESS OF INPUT STRING
22 025306 005737 001340      TST     CFLAG        ;CHECK THE CONTROL C FLAG
23 025312 001405              BEQ     2$           ;BR IF NO 'CONTROL C' ENTERED
24 025314 005737 001530      TST     ASNLST       ;ANY DRIVES ASSIGNED ?
25 025320 001130              BNE     13$         ;BR IF YES
26 025322 000137 003634      JMP     START        ;JUMP TO START
27
28 025326 005205              2$:  INC      R5        ;POINT TO SECOND CHARACTER
29 025330 122715 000124      CMPB   #'T,(R5)      ;EQ TO A 'T' ?
30 025334 001462              BEQ     9$           ;YES
31 025336 122715 000101      CMPB   #'A,(R5)      ;EQ TO AN 'A'
32 025342 001410              BEQ     3$           ;BR IF IT IS
33 025344 121527 000067      CMPB   (R5),#7       ;DRIVE NUMBER GREATER THAN AN ASCII 7 ?
34 025350 101111              BHI     12$         ;BR IF IT IS
35 025352 121527 000060      CMPB   (R5),#0       ;DRIVE NUMBER LESS THAN AN ASCII 0 ?
36 025356 103506              BLO     12$         ;BR IF IT IS
37 025360 142715 177770      BICB   #'C7,(R5)     ;LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
38 025364 122765 000124 177777 3$:  CMPB   #'T,-1(R5)    ;EQ TO 'T'
39 025372 001003              BNE     4$           ;BR IF NOT EQ
40 025374 004737 026530      JSR     PC,NEWASN    ;ASSIGN DRIVE FOR TEST
41 025400 000500              BR      13$         ;EXIT
42 025402 122765 000104 177777 4$:  CMPB   #'D,-1(R5)    ;EQ TO 'D' ?
43 025410 001003              BNE     5$           ;BR IF NOT EQ
44 025412 004737 026330      JSR     PC,DEASGN    ;DEASSIGN DRIVE
45 025416 000471              BR      13$         ;EXIT
46 025420 122765 000123 177777 5$:  CMPB   #'S,-1(R5)    ;EQ TO 'S'
47 025426 001003              BNE     6$           ;BR IF NOT EQ
48 025430 004737 026436      JSR     PC,SCMND     ;TYPE STATISTICS
49 025434 000462              BR      13$         ;EXIT
50 025436 122765 000127 177777 6$:  CMPB   #'W,-1(R5)    ;EQ TO 'W'
51 025444 001007              BNE     8$           ;BR IF NOT EQ
52 025446 032777 000001 153500      BIT     #SW0,@SWR    ;IS SWITCH 0 SET ?
53 025454 001044              BNE     11$         ;BR IF SET, CAN'T DO 'W' COMMAND
54 025456 004737 026552      7$:  JSR     PC,DATAPK   ;WRITE A DATA PACK
55 025462 000447              BR      13$         ;EXIT
56 025464 122765 000122 177777 8$:  CMPB   #'R,-1(R5)    ;EQ TO 'R' ?
57 025472 001040              BNE     12$         ;BR IF NOT EQ

```

```

58 025474 004737 026540 JSR PC,REDAPK ;READ A DATA PACK
59 025500 000440 BR 13$ ;EXIT
60 025502 122765 000127 *77777 9$: CMPB #'W,-1(R5) ;WT COMMAND ?
61 025510 001031 BNE 12$ ;NO
62 025512 032777 000001 153434 BIT #SW0,@SWR ;IS SWITCH 0 SET ?
63 025520 001022 BNE 11$ ;BR IF SET, CAN'T DO 'W' COMMAND
64 025522 122765 000101 000001 CMPB #'A,1(R5) ;ALL DRIVES ?
65 025530 001413 BEQ 10$ ;YES
66 025532 126527 000001 000067 CMPB 1(R5),#'7 ;GREAT THAN 7
67 025540 101015 BHI 12$ ;YES
68 025542 126527 000001 000060 CMPB 1(R5),#'0 ;LESS THAN 0
69 025550 103411 BLO 12$ ;YES
70 025552 142765 177770 000001 BICB #'*07,1(R5) ;CHOP OFF THE HIGHER BITS
71 025560 004737 026564 10$: JSR PC,WATPAK ;ASSIGN DRIVES WITH WT COMMAND
72 025564 000406 BR 13$
73 025566 104401 076501 11$: TYPE ,MSWRO ;TYPE 'CAN'T WRITE IN READ ONLY MODE'
74 025572 000631 BR 1$ ;TRY AGAIN
75 025574 104401 076544 12$: TYPE ,INVLD ;TYPE 'INVALID COMMAND' MESSAGE
76 025600 000626 BR 1$ ;TRY AGAIN
77 025602 104413 13$: RESREG ;RESTORE R0 - R5
78 025604 005777 153352 TST @STKB ;CLEAR THE TTY BUFFER
79 025610 052777 000100 153342 BIS #BIT06,@STKB ;SET TTY INTERRUPT ENABLE
80 025616 005037 177776 CLR PS ;SET PRIORITY BACK TO ZERO
81 025622 000207 RTS PC ;RETURN
82
83 ;ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
84
85 025624 111504 ASSIGN: MOVB (R5),R4 ;PUT DRIVE # IN R4
86 025626 005037 001340 1$: CLR CFLAG ;CLEAR CONTROL C FLAG
87 025632 005037 001430 CLR DRVPAR ;ASSUME CHANGING DRIVE PARAMETERS
88 025636 104401 077315 TYPE ,MSPRM ;TYPE 'CHANGE DRIVE PARAMETERS ?'
89 025642 104411 RDLIN ;READ THE ENTRY
90 025644 012600 MOV (SP)+,R0 ;SAVE ADDRESS OF RESPONSE
91 025646 005737 001340 TST CFLAG ;WAS IT CONTROL C ?
92 025652 001365 BNE 1$ ;BR IF YES
93 025654 105710 TSTB (R0) ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
94 025656 001414 BEQ 3$ ;BR IF YES
95 025660 105760 000001 TSTB 1(R0) ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
96 025664 001006 BNE 2$ ;BR IF NO
97 025666 122710 000131 CMPB #'Y,(R0) ;WAS IT A 'Y' RESPONSE ?
98 025672 001410 BEQ 4$ ;BR IF YES
99 025674 122710 000116 CMPB #'N,(R0) ;WAS IT A 'N' RESPONSE ?
100 025700 001403 BEQ 3$ ;BR IF YES
101 025702 104401 076677 2$: TYPE ,BADENT ;TYPE BAD ENTRY MESSAGE
102 025706 000747 BR 1$ ;TRY AGAIN
103 025710 005237 001430 3$: INC DRVPAR ;DO NOT CHANGE DRIVE PARAMETERS
104 025714 122704 000101 4$: CMPB #'A,R4 ;ASSIGN ALL DRIVES ?
105 025720 001431 BEQ ASGN2 ;BR IF YES
106
107 025722 012737 075676 031050 ASGN1: MOV #UNTSN,ASNMSG ;'DRIVE ASSIGNED' MESSAGE ADDRESS
108 025730 005737 001432 TST XXDP ;LOADED FROM THIS DEVICE ?
109 025734 001412 BEQ 1$ ;BR IF NO
110 025736 123704 001432 CMPB XXDP,R4 ;LOADED FROM THIS DRIVE ?
111 025742 001007 BNE 1$ ;BR IF NO
112 025744 146437 040142 001530 BICB ATABIT(R4),ASNLS1 ;DELETE THE DRIVE FROM THE ASSIGNED LIST
113 025752 012737 076011 031050 MOV #LODEV,ASNMSG ;'LOAD DEVICE' MESSAGE ADDRESS
114 025760 000407 BR 2$

```

```

115 025762 136437 040142 001530 1$: BITB ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
116 025770 001003 BNE 2$ ;BR IF IT IS
117 025772 004737 026102 JSR PC,ASGN3 ;SEE IF DRIVE ON THE SYSTEM
118 025776 000207 RTS ;RETURN
119 026000 000137 031024 2$: JMP ASNERR ;EXIT ERROR
120
121 026004 005004 ASGN2: CLR R4 ;START WITH DRIVE 0
122 026006 012737 075676 031050 1$: MOV #UNTASN,ASNMSG ;ERROR MESSAGE
123 026014 005737 001432 TST XXDP ;LOADED FROM THIS DEVICE ?
124 026020 001412 BEQ 2$ ;BR IF NO
125 026022 123704 001432 CMPB XXDP,R4 ;LOADED FROM THIS DRIVE ?
126 026026 001007 BNE 2$ ;BR IF NO
127 026030 146437 040142 001530 BICB ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
128 026036 012737 076011 031050 MOV #LODEV,ASNMSG ;'LOAD DEVICE' MESSAGE ADDRESS
129 026044 000413 BR 4$
130 026046 136437 040142 001530 2$: BITB ATABIT(R4),ASNLST ;ALREADY ASSIGNED ?
131 026054 001007 BNE 4$ ;YES
132 026056 004737 026102 JSR PC,ASGN3 ;ASSIGN THE DRIVE
133 026062 005204 3$: INC R4 ;INCREMENT DRIVE #
134 026064 020427 000007 CMP R4,#7 ;ALL DRIVE CHECKED ?
135 026070 003746 BLE 1$ ;NO
136 026072 000207 RTS PC ;YES
137 026074 004737 031024 4$: JSR PC,ASNERR ;ERROR MESSAGE
138 026100 000770 BR 3$ ;TO LOOP
139
140 026102 136437 040142 001530 ASGN3: BITB ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
141 026110 001060 BNE ASGN4 ;BR IF IT IS
142 026112 110437 070322 MOVB R4,GENDPB ;GET DRIVE NUMBER
143 026116 006304 ASL R4 ;MAKE R4 WORD INDEX
144 026120 016400 002044 MOV BLKADR(R4),R0 ;PUT BLOCK'S ADDR INTO R0
145 026124 004737 015740 JSR PC,RECALO ;RECALIBRATE DRIVE
146 026130 006204 ASR R4 ;MAKE R4 BYTE INDEX
147 026132 105764 040036 TSTB DRVSTA(R4) ;DRIVE AVAILABLE?
148 026136 001453 BEQ ASGN7 ;BR IF DRIVE OFFLINE OR NONEXISTENT
149 026140 100445 BMI ASGN6 ;BR IF DRIVE UNSAFE
150 026142 004737 026602 JSR PC,CLRDPB ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
151 026146 004737 027420 JSR PC,GETID ;GET DRIVE (MBA) SERIAL NUMBER
152 026152 004537 027520 JSR R5,GETADR ;RETRIEVE BAD SECTOR FILE
153 026156 005737 001430 TST DRVPAR ;CHANGE DRIVE PARAMETERS ?
154 026162 001017 BNE 1$ ;BR IF NO
155 026164 104401 076372 TYPE ,DRNUM ;TYPE DRIVE MESSAGE
156 026170 010446 MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
026172 104403 TYPOS ;;GO TYPE--OCTAL ASCII
026174 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
026175 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
157 026176 104401 001203 TYPE ,$CRLF ;CR-LF
158 026202 004737 032472 JSR PC,TYMBA ;TYPE MBA (DRIVE) SERIAL NUMBER
159 026206 104401 075623 TYPE ,TAB ;TYPE TAB CONTROL
160 026212 004737 032522 JSR PC,TYPACK ;TYPE PACK SERIAL NUMBER
161 026216 104401 001203 TYPE ,$CRLF ;CR-LF
162 026222 006304 1$: ASL R4 ;MAKE R4 WORD INDEX
163 026224 004737 027020 JSR PC,DRVPRM ;GET THE DRIVE'S ADDRESS LIMITS
164 026230 004737 030124 JSR PC,MANTER ;MANUALLY ENTER BAD SECTOR INFORMATION
165 026234 016464 002044 001554 MOV BLKADR(R4),NEWUNT(R4) ;DPB ADDRESS
166 026242 113760 001320 000026 MOVB PACK,$PACK(R0) ;SET READ/WRITE DATA PACK INDICATOR
167 026250 006204 ASR R4 ;MAKE R4 BYTE INDEX
168 026252 000207 ASGN4: RTS PC ;RETURN

```

```

169
170 026254 012737 076001 031050 ASGN6: MOV #NOTSAFE,ASNMSG ;'UNSAFE' MESSAGE ADDRESS
171 026262 000137 031024 JMP ASNERR ;TO ERROR ROUTINE
172
173 026266 105764 040046 ASGN7: TSTB DRVTP(R4) ;DRIVE PRESENT?
174 026272 001405 BEQ 1$ ;BR IF NOT
175 026274 100010 BPL 2$ ;BR IF DRIVE OFFLINE
176 026276 012737 075724 031050 MOV #NOTRM,ASNMSG ;ADDRESS OF 'NOT RM05/3/2' MSG
177 026304 000407 BR 3$ ;EXIT
178 026306 012737 075745 031050 1$: MOV #NOTPRS,ASNMSG ;ADDRESS OF 'NOT PRESENT' MSG
179 026314 000403 BR 3$ ;EXIT
180 026316 012737 075633 031050 2$: MOV #UNTOFF,ASNMSG ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
181 026324 000137 031024 3$: JMP ASNERR ;TO ERROR ROUTINE

```

```

1
2           ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
3
4 026330 005004           DEASGN: CLR      R4           ;START WITH DRIVE 0
5 026332 012703 000010   MOV      #8,R3           ;COUNTER
6 026336 122715 000101   CMPB    #'A',(R5)       ;DEASSIGN ALL DRIVES ?
7 026342 001403           BEQ     1$              ;BR IF YES
8 026344 111504           MOVB    (R5),R4         ;GET DRIVE NUMBER
9 026346 012703 000001   MOV      #1,R3         ;SET R3 FOR ONE UNIT
10 026352 136437 040142 001530 1$: BITB    ATABIT(R4),ASNLS ;DRIVE ASSIGNED ?
11 026360 001417           BEQ     3$              ;BR IF NOT
12 026362 146437 040142 001530 BICB    ATABIT(R4),ASNLS ;DELETE THE DRIVE FROM THE ASSIGNED LIST
13 026370 146437 040142 031662 BICB    ATABIT(R4),AUTLS ;DELETE DRIVE FROM AUTO ASSIGN LIST
14 026376 006304           ASL     R4              ;MAKE ADDR INTO A WORD INDEX
15 026400 016464 002044 001532 MOV      BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
16 026406 006204           ASR     R4
17 026410 005303           2$: DEC     R3              ;ANY MORE DRIVES ?
18 026412 0014 0           BEQ     4$              ;BR IF NOT
19 026414 005204           INC     R4
20 026416 000755           BR      1$
21 026420 012737 075654 031050 3$: MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
22 026426 004737 031024   JSR     PC,ASNERR      ;REPORT IT
23 026432 000766           BR      2$
24 026434 000207           4$: RTS     PC
25
26           ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
27
28 026436           SCMDND:
29 026442 122715 000101   MOV      ASNLS,-(SP)    ;:PUSH ASNLS ON STACK
30 026446 001416           CMPB    #'A',(R5)     ;ALL STATISTICS ?
31 026450 111504           BEQ     2$              ;BR IF YES
32 026452 136416 040142   MOVB    (R5),R4         ;GET DRIVE NUMBER
33 026456 001404           BITB    ATABIT(R4),(SP) ;IS THIS DRIVE ASSIGNED ?
34 026460 116437 040142 001530 BEQ     1$              ;BR IF NO
35 026466 000411           MOVB    ATABIT(R4),ASNLS ;GET DRIVE ASSIGN BIT
36
37 026470 012737 075654 031050 1$: MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
38 026476 004737 031024   JSR     PC,ASNERR      ;TYPE ERROR MESSAGE
39 026502 000407           BR      4$              ;EXIT
40 026504 105737 001530   2$: TSTB   ASNLS        ;ANY DRIVE ASSIGNED ?
41 026510 001404           BEQ     4$              ;BR IF NO
42 026512 004737 023550   3$: JSR     PC,STATPR    ;TYPE ALL STATISTICS
43 026516 104401 076155   TYPE    ,STAR5         ;TYPE '****...ETC'
44 026522           4$:
45 026522 012637 001530   MOV      (SP)+,ASNLS   ;:POP STACK INTO ASNLS
46 026526 000207           RTS     PC
47
48           ;'T' COMMAND (ROUTINE TO TEST A DRIVE)
49 026530 005037 001320   NEWASN: CLR     PACK    ;SET 'T' COMMAND INDICATOR
50 026534 000137 025624   JMP     ASSIGN         ;GO TO THE ASSIGN ROUTINE
51
52           ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
53
54 026540 012737 000001 001320 REDAPK: MOV      #1,PACK ;SET THE 'READ' INDICATOR
55 026546 000137 025624   JMP     ASSIGN         ;ASSIGN THE REQUESTED DRIVE

```

```

56
57           ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
58
59 026552 012737 177777 001320 DATAPK: MOV    #-1,PACK    ;SET THE 'W' COMMAND INDICATOR
60 026560 000137 025624           JMP     ASSIGN    ;ASSIGN REQUESTED DRIVE
61
62           ;'WT' COMMAND (TO WRITE A PACK AND TEST A DRIVE)
63
64 026564 116515 000001           WATPAK: MOVB  1(R5),(R5) ;ADJUST DRIVE NUMBER ADDRESS
65 026570 012737 177776 001320     MOV    #-2,PACK    ;PACK WRITE COMMAND
66 026576 000137 025624           JMP     ASSIGN    ;JUMP TO ASSIGN ROUTINE

```

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

```

026602
026602 010146
026604 010346
026606 010446
026610 010546
11 026612 010004
12 026614 062704 000002
13 026620 012703 000012
14 026624 005024
15 026626 162703 000002
16 026632 001374
17 026634 062704 000002
18 026640 012703 000104
19 026644 005024
20 026646 162703 000002
21 026652 001374
22 026654 062704 002004
23
25 026660 012703 C00064
29 026664 012764 177777 177776
30 026672 005024
31 026674 162703 000002
32 026700 001374
33 026702 113760 001502 000024
34 026710 013701 001502
35 026714 116160 002064 000002
36 026722 113760 001500 000030
37 026730 106360 000030
38 026734 013760 001504 000020
39 026742 013760 001504 000004
40 026750 005460 000004
41 026754 012760 000400 000022
42 026762 012760 000001 000104
43 026770 132760 000001 000024
44 026776 001403
45 027000 062760 000002 0 J022
46 027006
027006 012605
027010 012604
027012 012603
027014 012601
47 027016 000207

```

;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE

```

;CALL:
      MOV     #DPB,R0           ;DPB ADDRESS
      JSR    PC,CLRDPB
      RETURN

```

;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE

```

CLRDPB:
      MOV     R1,-(SP)          ;;PUSH R1 ON STACK
      MOV     R3,-(SP)          ;;PUSH R3 ON STACK
      MOV     R4,-(SP)          ;;PUSH R4 ON STACK
      MOV     R5,-(SP)          ;;PUSH R5 ON STACK
      MOV     R0,R4             ;GET THE DPB ADDRESS
      ADD     #2,R4             ;ADDRESS OF FIRST LOCN TO BE CLEARED
      MOV     #<$CYL-$COMND>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
3$:   CLR     (R4)+              ;CLEAR THE LOCATION
      SUB     #2,R3             ;DONE CLEARING YET ?
      BNE    1$                ;BR IF NO
      ADD     #2,R4             ;SKIP OVER THE '$REG' LOCATION
      MOV     #<$NEXT-$STATUS>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
2$:   CLR     (R4)+              ;CLEAR THE LOCATION
      SUB     #2,R3             ;DONE CLEARING YET ?
      BNE    2$                ;BR IF NO
      ADD     #<$MBASN-$FIRST>,R4 ;SKIP OVER FIRST FLAG, MIN/MAX ADRS
                                           ;LIMITS AND BAD SECTOR TABLE
      MOV     #<$RMCS3-$MBASN>+2,R3 ;NUMBER OF LOCNS TO BE CLEARED
      MOV     #-1,-2(R4)        ;INITIALIZE TERMINATOR FOR BAD SECTOR TABLE
3$:   CLR     (R4)+              ;CLEAR A LOCATION
      SUB     #2,R3             ;DONE CLEARING YET ?
      BNE    3$                ;BR IF NO
      MOVB   BEGCOD,$CODE(R0)   ;INITIAL COMMAND CODE
      MOV     BEGCOD,R1         ;GET THE ACTUAL OP CODE
      MOVB   COMTBL(R1),$COMND(R0) ;OPERATION CODE
      MOVB   BEGPAT,$PATT(R0)  ;PATTERN CODE
      ASLB   $PATT(R0)         ;CONVERT CODE TO A TABLE INDEX
      MOV     BEGWC,$WRDL(R0)  ;BEGINNING WORD COUNT
      MOV     BEGWC,$WCNT(R0)  ;VALUE FOR DATA TRANSFER
      NEG    $WCNT(R0)         ;MAKE IT INTO 2'S COMPLEMENT
      MOV     #256,$SSEC(R0)   ;INITIAL VALUE OF SECTOR SIZE
      MOV     #1,$PASSC(R0)    ;PRESET PASS COUNT TO 1
      BITB   #1,$CODE(R0)     ;HEADER COMMAND ?
      BEQ    4$                ;BR IF NOT
      ADD     #2,$SSEC(R0)     ;ADD HEADER SIZE TO SECTOR SIZE
4$:   MOV     (SP)+,R5          ;;POP STACK INTO R5
      MOV     (SP)+,R4          ;;POP STACK INTO R4
      MOV     (SP)+,R3          ;;POP STACK INTO R3
      MOV     (SP)+,R1          ;;POP STACK INTO R1
      RTS    PC                ;RETURN

```

;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR

```

;CALL:
      MOV     #DPB,R0           ;DPB ADDRESS
      JSR    PC,DRVPRM        ;CALL ROUTINE

```


:RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

```

54
55
56 027020 010346          DRVPRM: MOV    R3,-(SP)      ;SAVE R3
57 027022 010446          MOV    R4,-(SP)      ;SAVE R4
58 027024 105737 001150   TSTB  $AUTOB        ;RUNNING IN AUTO MODE ?
59 027030 001010          BNE   1$            ;BR IF YES
60 027032 005737 001336   TST   CHGADR        ;PROGRAM STARTED AT 200 ?
61 027036 003005          BGT   1$            ;BR IF YES
62 027040 005737 001430   TST   DRVPAR        ;CHANGE DRIVE PARAMETERS ?
63 027044 001002          BNE   1$            ;BR IF NO
64 027046 104401 076610   TYPE  ,ENTLMT       ;'ENTER ADDRESS LIMITS'
65
66 027052 004737 027346   1$:   JSR   PC,GETLMT  ;GET ADDRESS LIMITS
67 027056 062760 177777 000122   ADD   #-1,$FIRST(RO) ;SEE IF FIRST TIME STARTED
68 027064 103417          BCS   2$            ;BR IF NOT
69 027066 013760 001422 000124   MOV   CYLIMT,MAXCYL(RO) ;LOAD MAXIMUM CYLINDER
70 027074 013760 001426 000130   MOV   TRKLMT,MAXTRK(RO) ;LOAD MAXIMUM TRACK
71 027102 013760 001424 000134   MOV   SECLMT,MAXSEC(RO) ;LOAD MAXIMUM SECTOR
72 027110 005060 000126   CLR   MINCYL(RO)     ;CLEAR MINIMUM CYLINDER
73 027114 005060 000132   CLR   MINTRK(RO)    ;CLEAR MINIMUM TRACK
74 027120 005060 000136   CLR   MINSEC(RO)    ;CLEAR MINIMUM SECTOR
75
76 027124 105737 001150   2$:   TSTB  $AUTOB        ;RUNNING IN AUTO MODE ?
77 027130 001072          BNE   5$            ;BR IF YES
78 027132 005737 001336   TST   CHGADR        ;PROGRAM STARTED AT 200 ?
79 027136 003067          BGT   5$            ;BR IF YES
80 027140 005737 001430   TST   DRVPAR        ;CHANGE DRIVE PARAMETERS ?
81 027144 001064          BNE   5$            ;BR IF NO
82 027146 016403 077734   MOV   TABLE(R4),R3 ;PARAMETER TABLE ADDRESS
83 027152 013763 001422 000002   MOV   CYLIMT,2(R3)   ;LOAD CYLINDER LIMIT FOR MINCYL
84 027160 013763 001422 000010   MOV   CYLIMT,10(R3) ;LOAD CYLINDER LIMIT FOR MAXCYL
85 027166 013763 001426 000016   MOV   TRKLMT,16(R3) ;LOAD TRACK LIMIT FOR MINTRK
86 027174 013763 001426 000024   MOV   TRKLMT,24(R3) ;LOAD TRACK LIMIT FOR MAXTRK
87 027202 013763 001424 000032   MOV   SECLMT,32(R3) ;LOAD SECTOR LIMIT FOR MINSEC
88 027210 013763 001424 000040   MOV   SECLMT,40(R3) ;LOAD SECTOR LIMIT FOR MAXSEC
89 027216 004737 030700   JSR   PC,PARENT     ;GET THE DRIVE'S PARAMETERS
90
91 027222 016003 000126   MOV   MINCYL(RO),R3 ;STORE MINCYL VALUE
92 027226 016004 000124   MOV   MAXCYL(RO),R4 ;STORE MAXCYL VALUE
93 027232 020304          CMP   R3,R4         ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
94 027234 003404          BLE   3$            ;BR IF YES
95 027236 010360 000124   MOV   R3,MAXCYL(RO) ;SWAP MIN. TO MAX.
96 027242 010460 000126   MOV   R4,MINCYL(RO) ;SWAP MAX. TO MIN.
97 027246 016003 000132   3$:   MOV   MINTRK(RO),R3 ;STORE MINTRK VALUE
98 027252 016004 000130   MOV   MAXTRK(RO),R4 ;STORE MAXTRK VALUE
99 027256 020304          CMP   R3,R4         ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
100 027260 003404          BLE   4$            ;BR IF YES
101 027262 010360 000130   MOV   R3,MAXTRK(RO) ;SWAP MIN. TO MAX.
102 027266 010460 000132   MOV   R4,MINTRK(RO) ;SWAP MAX. TO MIN.
103 027272 016003 000136   4$:   MOV   MINSEC(RO),R3 ;STORE MINSEC VALUE
104 027276 016004 000134   MOV   MAXSEC(RO),R4 ;STORE MAXSEC VALUE
105 027302 020304          CMP   R3,R4         ;IS MIN. LESS THAN OR EQUAL TO MAX. ?
106 027304 003404          BLE   5$            ;BR IF YES
107 027306 010360 000134   MOV   R3,MAXSEC(RO) ;SWAP MIN. TO MAX.
108 027312 010460 000136   MOV   R4,MINSEC(RO) ;SWAP MAX. TO MIN.
109
110 027316 016003 000126 000012 5$:   MOV   MINCYL(RO),$CYL(RO) ;INITIAL CYLINDER VALUE

```

```

111 027324 116060 000132 000011      MOVB   MINTRK(R0), $TRK(R0)      ;INITIAL TRACK VALUE
112 027332 116060 000136 000010      MOVB   MINSEC(R0), $SEC(R0)     ;INITIAL SECTOR VALUE
113 027340 012604                MOV    (SP)+, R4                ;;POP STACK INTO R4
      027342 012603                MOV    (SP)+, R3                ;;POP STACK INTO R3
114 027344 000207                RTS     PC                      ;RETURN
115
116                                     ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
117                                     ;CALL:
118                                     ;      JSR     PC, GETLMT      ;CALL ROUTINE
119                                     ;
120                                     ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
121
122 027346                GETLMT:
      027346 010146                MOV    R1, -(SP)                ;;PUSH R1 ON STACK
123 027350 005001                CLR    R1                      ;START FRESH
124 027352 012737 001465 001422      MOV    #821., CYLMT            ;GET CYLINDER LIMIT
125 027360 012737 000037 001424      MOV    #31., SECLMT           ;GET SECTOR LIMIT
126 027366 111001                MOVB   (R0), R1                ;GET DRIVE NUMBER
127 027370 012737 000022 001426      MOV    #18., TRKLMT           ;ASSUME LAST TRACK FOR AN RM05
128 027376 122761 000007 040046      CMPB   #7, DRVTYP(R1)         ;IS DRIVE AN RM05 ?
129 027404 001403                BEQ    1$                      ;BR IF YES
130 027406 012737 000004 001426      MOV    #4, TRKLMT            ;GET LAST TRACK FOR AN RM03/2
131 027414                1$:
      027414 012601                MOV    (SP)+, R1                ;;POP STACK INTO R1
132 027416 000207                RTS     PC                      ;RETURN

```

1
2
3
4
5
6
7
8
9
10

:ROUTINE TO GET THE DRIVE (MBA) SERIAL NUMBER FROM RMSN REGISTER
:THIS NUMBERS CONTAINED IN THE REGISTER ARE ONLY THE 4 LSD'S OF THE
:SERIAL NUMBER.

:CALL:
:MOV #DPB,R0 :DPB ADDRESS
:JSR PC,GETID :CALL ROUTINE
:RO - DPB ADDRESS BEFORE CALLING THE ROUTINE

10 027420
027420 010046
027422 010146
027424 010246
027426 010546
11 027430 010002
12 027432 004737 045230
13 027436 012702 000004
14 027442 016001 002166
15 027446 005005
16 027450 006101
17 027452 006105
18 027454 006101
19 027456 006105
20 027460 006101
21 027462 006105
22 027464 006101
23 027466 006105
24 027470 062705 000060
25 027474 110560 002126
26 027500 005200
27 027502 005302
28 027504 003360
29 027506 012605
027510 012602
027512 012601
027514 012600
30 027516 000207

GETID:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV R0,R2 :GET INDEX TO DPB
JSR PC,SVRH70 :SAVE ALL REGISTERS
MOV #4,R2 :FOUR DIGITS TO STORE
MOV \$RMSN(R0),R1 :SERIAL NUMBER
1\$: CLR R5 :ZERO
ROL R1 :PUT THE NEXT DIGIT
ROL R5 :INTO R5
ROL R1
ROL R5
ROL R1
ROL R5
ROL R1
ROL R5
ADD #'0,R5 :MAKE IT ASCII
MOVSB R5,\$MBASN(R0) :SAVE DRIVE (MBA) SERIAL NUMBER DIGIT
INC R0 :GET NEXT INDEX FOR DRIVE (MBA) SERIAL NUMBER
DEC R2 :ALL DIGITS TYPED?
BGT 1\$:NO -- BRANCH
MOV (SP)+,R5 ;;POP STACK INTO R5
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTS PC :RETURN

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33

.SBTTL READ DEC144 FILE

:THIS ROUTINE IS USED TO READ THE DEC144 BAD SECTOR FILE FROM CYLINDER
:558. TRACK 13. AND TO STORE THE FILE IN IT'S RESPECTIVE DPB TABLE.
:THE DPB TABLE HAS ENOUGH ROOM TO SAVE THE ENTIRE MFG AND USR PORTIONS
:OF THE DEC144 FILE. (MFG=126. ENTRIES AND USR=126. ENTRIES) EVERY TIME
:THE DRIVE IS ASSIGNED THE DEC144 FILE IS READ TO DETERMINE THE STATUS
:OF THE CURRENT PACK SERIAL NUMBER. BUT, IN ORDER TO INITIALIZE THE
:BAD SECTOR ENTRY TABLE, AT LEAST ONE OF FOLLOWING STATEMENTS MUST BE VALID.

- :1) FIRST TIME PROGRAM WAS STARTED
- : OR
- :2) LOCATION 'BADBLK' IS EQUAL TO 1
- : OR
- :3) LOCATION 'BADBLK' IS EQUAL TO 0 AND THE PACK
SERIAL NUMBER CHANGED SINCE THE LAST TIME IT WAS
READ. (DEFAULT)

:NOTE: IF THE SERIAL NUMBER HAS CHANGED, THIS MOST LIKELY MEANS THAT THE
PACK OR DRIVE HAD BEEN REPLACED WHILE THE DRIVE WAS DEASSIGNED.

:THIS ROUTINE CHECKS THAT THE TWO SERIAL NUMBER WORDS ARE NOT ZERO
:AND ARE POSITIVE NUMBERS. ALSO, WORDS 3 AND 4 ARE CHECKED TO BE ALL
:ZERO WORDS. IF THE DEC144 FILE DOES NOT COMPLY WITH THIS STRUCTURE,
:AN ERROR MESSAGE IS TYPED AND THE ROUTINE IS EXITED.

```

:CALL
:      MOV      #DPB,RO      ;DPB ADDRESS
:      JSR      R5,GETADR    ;READ DEC144 BAD SECTOR FILES
:
:RO - DPB ADDRESS BEFORE CALLING THE ROUTINE

```

```

GETADR:
MOV      R1,-(SP)          ;;PUSH R1 ON STACK
MOV      R2,-(SP)          ;;PUSH R2 ON STACK
MOV      R3,-(SP)          ;;PUSH R3 ON STACK
JSR      PC,GETLMT         ;GET ADDRESS LIMITS
MOV      RO,R1             ;DPB ADDRESS
ADD      #BADSEC,R1        ;ADDRESS OF BAD SECTOR TABLE
MOV      R1,-(SP)          ;;PUSH R1 ON STACK
MOVB    (R0),GENDPB        ;DRIVE NUMBER
MOV      #822,GENDPB+$CYL  ;LAST CYLINDER
MOVB    TRKLMT,GENDPB+$TRK ;GET LAST TRACK
MOVB    #0,GENDPB+$SEC     ;GET STARTING SECTOR OF 16 BIT MFG FILE
MOV      #-256,GENDPB+$WCNT ;ONE SECTOR WORD COUNT
MOVB    #RDDAT,GENDPB+$CMD  ;READ DATA COMMAND
MOV      #8,$CDW2          ;GET LAST SECTOR OF 16 BIT MFG FILE
1$:      MOV      #CYLNR,R3 ;GET READ BUFFER ADDRESS
        JSR      RO,RMOS    ;READ CURRENT SECTOR
        GENDPB
        BR      1$         ;WAIT FOR QUE
2$:      TST      GENDPB+$STATUS ;READ DONE YET ?
        BEQ      2$         ;BR IF NO
        BPL      3$         ;BR IF NO ERROR, ELSE
        ADD      #2,GENDPB+$SEC ;INCREMENT NEXT SECTOR TO READ
        CMPB    $CDW2,GENDPB+$SEC ;WERE ALL SECTORS TRIED ?
        BHIS    1$         ;BR IF NO

```

```

027520
027520 010146
027522 010246
027524 010346
34 027526 004737 027346
35 027532 010001
36 027534 062701 000144
37 027540 010146
38 027542 111037 070322
39 027546 012737 001466 070334
40 027554 113737 001426 070333
41 027562 112737 000000 070332
42 027570 012737 177400 070326
43 027576 112737 000171 070324
44 027604 012737 000010 001270
45 027612 012703 101066
46 027616 004037 040714
47 027622 070322
48 027624 000772
49 027626 005737 070340
50 027632 001775
51 027634 100010
52 027636 062737 000002 070332
53 027644 123737 001270 070332
54 027652 103357

```

55	027654	0004701			BR	10\$:BR IF UNSUCCESSFUL ON RETRIES
56	027656	005723	3\$:		TST	(R3)+		:ARE LSB'S OF SERIAL NUMBER VALID ?
57	027660	005723			TST	(R3)+		:ARE MSB'S OF SERIAL NUMBER VALID ?
58	027662	100462			BMI	9\$:BR IF MINUS
59	027664	001003			BNE	4\$:BR IF PLUS
60	027666	005763	177774		TST	-4(R3)		:ARE SERIAL NUMBERS ZERO ?
61	027672	001456			BEQ	9\$:BR IF YES
62	027674	005723	4\$:		TST	(R3)+		:IS 3RD WORD ALL 0'S ?
63	027676	001054			BNE	9\$:BR IF NO
64	027700	005723			TST	(R3)+		:IS 4TH WORD ALL 0'S ?
65	027702	001052			BNE	9\$:BR IF NO
66	027704	123727	070332	000012	CMPB	GENDPB+\$SEC,#10.		:READING USR BAD FILE ?
67	027712	103021			BHIS	6\$:BR IF YES
68	027714	005737	001476		TST	BADBLK		:INIT. BAD SECTOR TABLE ENTRIES ?
69	027720	001010			BNE	5\$:BR IF YES
70	027722	026360	177770	000140	CMP	-10(R3), \$PSNL(R0)		:ARE LSB'S OF S/N SAME AS BEFORE ?
71	027730	001004			BNE	5\$:BR IF NO
72	027732	026360	177772	000142	CMP	-6(R3), \$PSNM(R0)		:ARE MSB'S OF S/N SAME AS BEFORE ?
73	027740	001464			BEQ	13\$:BR IF YES
74	027742	016360	177770	000140	MOV	-10(R3), \$PSNL(R0)	5\$:	:STORE PACK SERIAL NUMBER
75	027750	016360	177772	000142	MOV	-6(R3), \$PSNM(R0)		
76								
77	027756	012702	000176		MOV	#126.,R2	6\$:	:NUMBER OF ENTRIES PER FILE (MFG/USR)
78	027762	012321			MOV	(R3)+,(R1)+	7\$:	:STORE BAD CYLINDER ADDRESS
79	027764	012321			MOV	(R3)+,(R1)+		:STORE BAD TRK/SEC ADDRESS
80	027766	005302			DEC	R2		:DONE WITH ENTRIES ?
81	027770	001374			BNE	7\$:BR IF NO
82	027772	123727	070332	000012	CMPB	GENDPB+\$SEC,#10.		:USR BAD FILE DONE YET ?
83	030000	103044			BHIS	13\$:BR IF YES
84	030002	112737	000012	070332	MOV	#10.,GENDPB+\$SEC		:GET STARTING SECTOR OF USR FILE
85	030010	012737	000036	001270	MOV	#30., \$CDW2		:GET LAST SECTOR OF USR FILE
86	030016	011601			MOV	(SP),R1		:GET BEGINNING OF \$BDSEC TABLE
87	030020	005721			TST	(R1)+	8\$:	:IS THIS TERMINATOR ?
88	030022	100376			BPL	8\$:BR IF NO
89	030024	005741			TST	-(R1)		:FOUND TERMINATOR, BACKUP 1 WORD
90	030026	000671			BR	1\$		
91								
92	030030	104401	077000		TYPE	,MERR2	9\$:	:REPORT, INVALID DEC144 FILE STRUCTURE
93	030034	000402			BR	11\$		
94	030036	104401	076720		TYPE	,MERR1	10\$:	:REPORT, FAILED TO RETRIEVE DEC144 FILES
95	030042	104401	075625		TYPE	,UNMSG	11\$:	:TYPE 'ON DRIVE'
96	030046	011046			MOV	(R0),-(SP)		:SAVE (R0) FOR TYPEOUT
	030050	104403			TYPOS			:GO TYPE--OCTAL ASCII
	030052	002			.BYTE	2		:TYPE 2 DIGIT(S)
	030053	000			.BYTE	0		:SUPPRESS LEADING ZEROS
97	030054	104401	001203		TYPE	, \$CRLF		:CR-LF
98	030060	012601			MOV	(SP)+,R1		:POP STACK INTO R1
99	030062	012702	000374		MOV	#252.,R2		:TOTAL NUMBER OF ENTRIES ALLOWED
100	030066	012721	177777		MOV	#-1,(R1)+	12\$:	:INITIALIZE CYLINDER LOCATIONS TO -1
101	030072	012721	177777		MOV	#-1,(R1)+		:INITIALIZE TRK/SEC LOCATIONS TO -1
102	030076	005302			DEC	R2		:DONE YET ?
103	030100	001372			BNE	12\$:BR IF NO
104	030102	012760	177777	000142	MOV	#-1,\$PSNM(R0)		:INDICATE SERIAL NUMBER IS UNKNOWN
105	030110	000401			BR	14\$		
106	030112	005726			TST	(SP)+	13\$:	:RESTORE STACK
107	030114						14\$:	
	030114	012603			MOV	(SP)+,R3		:POP STACK INTO R3

ZRMUBG RM05/3/2 PERF EXER
READ DEC144 FILE

MACRO V04.00 4-APR-81 01:42:23 PAGE 31-2

E 11

SEQ 0134

030116 012602
030120 012601
108 030122 000205

MOV (SP)+,R2 ::POP STACK INTO R2
MOV (SP)+,R1 ::POP STACK INTO R1
RTS R5 :EXIT

.SBTTL ENTER BAD SECTOR ROUTINE

:ROUTINE TO ENTER BAD SECTOR INFORMATION MANUALLY

:CALL:

```

:      MOV    #DPB,RO      ;DPB ADDRESS
:      JSR    PC,MANTER    ;CALL ROUTINE

```

:RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

MANTER:

```

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
TSTB  $AUTOB        ;RUNNING IN AUTO MODE ?
BNE   19$           ;BRANCH IF SO
TST   DRVPAR        ;CHANGE DRIVE PARAMETERS ?
BNE   19$           ;BR IF NO
1$:   CLR   CFLAG    ;CLEAR THE CONTROL-C FLAG
      TYPE ,ENTADR  ;MESSAGE TO ENTER...

MOV    #SBDSEC,R4   ;INDEX VALUE OF TABLE ADDRESS
ADD   R0,R4         ;TABLE STARTING ADDRESS
MOV   #<126.*2>,R1 ;256. TOTAL BAD SECTORS ALLOWED
2$:   CMP   #-1,(R4) ;ENTRY IN THE TABLE ?
      BEQ  3$       ;BRANCH IF SO
      ADD  #4,R4    ;ADJUST THE TABLE ENTRY POINTER
      DEC  R1       ;DECREMENT THE BAD SECTOR COUNT
      BNE 2$       ;BR IF TO NEXT ENTRIES POSITION
      TYPE ,MSFULL ;TYPE 'BAD SECTOR TABLE IS FULL'
      BR  19$      ;EXIT..

3$:   MOV   R1,-(SP) ;SAVE THE COUNTER AND FIRST
      MOV   R4,-(SP) ;ENTRY POINTER PAIR
4$:   MOV   #-1,(R4) ;RESET CYLINDER TO -1
      MOV   #-1,2(R4) ;RESET TRACK/SECTOR FIELD TO -1
      TYPE ,MSGCTS  ;TYPE 'CYL,TRK,SEC = '
      RDLIN ;READ THE ADDRESS
      MOV   (SP)+,R1 ;READ IN TEXT ADDRESS
      TST  CFLAG    ;CONTROL-C ENTERED ?
      BNE  5$       ;BRANCH IF YES
      TSTB 1(R1)    ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
      BNE  7$       ;BR IF NO
      CMPB #'L',(R1) ;WAS CHARACTER AN 'L' ?
      BNE  7$       ;BR IF NO
      JSR  PC,TYLIST ;TYPE BAD SECTOR LIST FOR USER
      BR  4$

5$:   MOV   (SP)+,R4 ;RETRIEVE THE ENTRY POINTER
      MOV   (SP)+,R1 ;RETRIEVE THE COUNT
6$:   MOV   #-1,(R4)+ ;RESET THE TABLE
      MOV   #-1,(R4)+ ;TO -1
      DEC  R1       ;ALL DONE ?
      BNE  6$       ;BRANCH IF NOT
      TYPE ,ALOST  ;TYPE ' * ALL CURRENT ENTRIES LOST * '
      BR  1$       ;ENTER AGAIN FROM THE FIRST POINTER

```

```

1
2
3
4
5
6
7
8
9
10 030124 010146
    030124 010246
    030130 010346
    030132 010446
11 030134 105737 001150
12 030140 001162
13 030142 005737 00430
14 030146 001157
15 030150 005037 001340
16 030154 104401 076640
17
18 030160 012704 000144
19 030164 060004
20 030166 012701 000374
21 030172 022714 177777
22 030176 001407
23 030200 062704 000004
24 030204 005301
25 030206 001371
26 030210 104401 077044
27 030214 000534
28
29 030216 010146
30 030220 010446
31 030222 012714 177777
32 030226 012764 177777 000002
33 030234 104401 077102
34 030240 104411
35 030242 012601
36 030244 005737 001340
37 030250 001011
38 030252 105761 000001
39 030256 001021
40 030260 122711 000114
41 030264 001016
42 030266 004737 030520
43 030272 000753
44 030274 012604
45 030276 012601
46 030300 012724 177777
47 030304 012724 177777
48 030310 005301
49 030312 001372
50 030314 104401 077232
51 030320 000713
52
53 030322

```

```

030322 013702 001422      MOV      CYLIMT,R2      ;UPPER LIMIT OF INPUT
030326 004537 032772      JSR      R5,CK.DIG     ;CHECK THE DIGIT(S)
030332 030502              18$          ;CARRIAGE RETURN ONLY ENTERED
030334 030502              18$          ;PERIOD ONLY ENTERED
030336 030474              17$          ;ILLEGAL INPUT
030340 030346              8$           ;TERMINATED WITH A CARRIAGE RETURN
030342 030356              10$          ;TERMINATED WITH A '...'
030344 030352              9$           ;TERMINATED WITH A '...'
54 030346 010214          8$:      MOV      R2,(R4)      ;CYLINDER ADDRESS
55 030350 000444          BR       16$          ;FINISH WITH THE CURRENT ADDRESS
56 030352 010214          9$:      MOV      R2,(R4)      ;CYLINDER ADDRESS
57 030354 000452          BR       18$          ;EXIT,PERIOD ENTERED
58 030356 010214          10$:     MOV      R2,(R4)      ;CYLINDER ADDRESS FOLLOWED BY ', '
59
60 030360 013702 001426      MOV      TRKLMT,R2     ;UPPER LIMIT OF INPUT
030364 004537 032772      JSR      R5,CK.DIG     ;CHECK THE DIGIT(S)
030370 030502              18$          ;CARRIAGE RETURN ONLY ENTERED
030372 030502              18$          ;PERIOD ONLY ENTERED
030374 030474              7$           ;ILLEGAL INPUT
030376 030404              11$          ;TERMINATED WITH A CARRIAGE RETURN
030400 030420              13$          ;TERMINATED WITH A '...'
030402 030412              12$          ;TERMINATED WITH A '...'
61 030404 110264 000003      11$:     MOVVB   R2,3(R4)      ;TRACK ADDRESS
62 030410 000424          BR       16$          ;TRACK NUMBER FOLLOWED BY CR
63 030412 110264 000003      12$:     MOVVB   R2,3(R4)      ;TRACK ADDRESS
64 030416 000431          BR       8$           ;EXIT, TRACK NUMBER FOLLOWED BY ', '
65 030420 110264 000003      13$:     MOVVB   R2,3(R4)      ;TRACK ADDRESS FOLLOWED BY ', '
66
67 030424 013702 001424      MOV      SECLMT,R2    ;UPPER LIMIT OF INPUT
030430 004537 032772      JSR      R5,CK.DIG     ;CHECK THE DIGIT(S)
030434 030502              18$          ;CARRIAGE RETURN ONLY ENTERED
030436 030502              18$          ;PERIOD ONLY ENTERED
030440 030474              17$          ;ILLEGAL INPUT
030442 030456              15$          ;TERMINATED WITH A CARRIAGE RETURN
030444 030474              17$          ;TERMINATED WITH A '...'
030446 030450              14$          ;TERMINATED WITH A '...'
68 030450 110264 000002      14$:     MOVVB   R2,2(R4)      ;SECTOR ADDRESS
69 030454 000412          BR       18$          ;EXIT,SECTOR ADDRESS FOLLOWED BY ', '
70 030456 110264 000002      15$:     MOVVB   R2,2(R4)      ;SECTOR ADDRESS
71
72 030462 005303          16$:     DEC      R3           ;MORE ENTRIES ?
73 030464 001406          BEQ     18$          ;BRANCH IF EXHAUSTED
74 030466 062704 000004      ADD     #4,R4        ;ADJUST FOR THE NEXT TABLE ENTRY
75 030472 000653          BR      4$           ;ENTER NEXT SECTOR ADDRESS
76
77 030474 104401 076677      17$:     TYPE   ,BADENT     ;MESSAGE BAD ENTRY
78 030500 000650          BR      4$           ;ENTER SECTOR ADDRESS AGAIN
79 030502 062706 000004      18$:     ADD     #4,SP      ;CLEAR OFF THE STACK POINT
80 030506          19$:
030506 012604          MOV     (SP)+,R4     ;;POP STACK INTO R4
030510 012603          MOV     (SP)+,R3     ;;POP STACK INTO R3
030512 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
030514 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
81 030516 000207          RTS     PC           ;EXIT
82
83
84

```

.SBTTL TYPE BAD SECTOR LIST


```

85
86
87
88
89
90
91
92
93 030520
   030520 010146
94 030522 104401 077166
95 030526 012701 000144
96 030532 060001
97 030534 010146
98 030536 022711 177777
99 030542 001444
100 030544 011146
101 030546 004737 033130
102 030552 004737 032134
103 030556 005046
104 030560 116116 000003
105 030564 100407
106 030566 104401 075617
107 030572 004737 033130
108 030576 004737 032134
109 030602 000401
110 030604 005726
111 030606 005046
112 030610 116116 000002
113 030614 100407
114 030616 104401 075617
115 030622 004737 033130
116 030626 004737 032134
117 030632 000401
118 030634 005726
119 030636 104401 001203
120 030642 062701 000004
121 030646 005737 001340
122 030652 001731
123 030654 022601
124 030656 001002
125 030660 104401 077146
126 030664 104401 001203
127 030670 005037 001340
128 030674 012601
129 030676 000207

```

```

:ROUTINE TO LIST BAD SECTORS ON THE TERMINAL IN DECIMAL NUMBERS
:FORMAT IS: CYL,TRK,SEC
:CALL:
:      MOV      #DPB,RO      ;DPB ADDRESS
:      JSR      PC,TYLIST    ;CALL ROUTINE
:
:RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

```

```

TYLIST:
MOV      R1,-(SP)          ;;PUSH R1 ON STACK
TYPE     ,LSTHDR          ;TYPE 'DEC144 AND MANUAL BAD SECTOR LIST'
MOV      #SBDSEC,R1       ;INDEX VALUE OF TABLE ADDRESS
ADD      RO,R1            ;TABLE STARTING ADDRESS
MOV      R1,-(SP)         ;SAVE ADDRESS FOR LATER
1$:      CMP      #-1,(R1)  ;TERMINATOR OR NO ENTRY IN THE TABLE ?
BEQ      6$              ;BRANCH IF YES
MOV      (R1),-(SP)       ;GET CYLINDER NUMBER
JSR      PC,$SB2D        ;CONVERT NUMBER
JSR      PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
CLR      -(SP)           ;CLEAR HI BYTE AND PUSH STACK
MOVB     3(R1),(SP)       ;GET TRACK NUMBER
BMI      2$              ;BR IF ALL BAD
TYPE     ,COMMA          ;TYPE ','
JSR      PC,$SB2D        ;CONVERT NUMBER
JSR      PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
BR       3$
2$:      TST      (SP)+    ;RESTORE STACK
CLR      -(SP)           ;CLEAR HI BYTE AND PUSH STACK
3$:      MOVB     2(R1),(SP) ;GET SECTOR NUMBER
BMI      4$              ;BR IF ALL BAD
TYPE     ,COMMA          ;TYPE ','
JSR      PC,$SB2D        ;CONVERT NUMBER
JSR      PC,SUPRSL       ;LEFT JUSTIFY AND TYPE
BR       5$
4$:      TST      (SP)+    ;RESTORE STACK
5$:      TYPE     ,$CRLF   ;CR-LF
ADD      #4,R1           ;INCREMENT POINTER
TST      CFLAG          ;CONTROL-C ENTERED ?
BEQ      1$             ;BRANCH IF NO
6$:      CMP      (SP)+,R1 ;ANY ENTRIES ?
BNE      7$             ;BR IF YES
TYPE     ,NOENTY        ;TYPE 'NO ENTRIES'
7$:      TYPE     ,$CRLF   ;CR-LF
CLR      CFLAG          ;CLEAR CONTROL FLAG
MOV      (SP)+,R1       ;POP STACK INTO R1
PTS      PC             ;RETURN

```

.SBTTL PARAMETER ENTRY ROUTINE

			:PARAMETER ENTRY ROUTINE	
			:CALL:	
			:MOV #ADR,R3	:PARAMETER TABLE ADDRESS
			:JSR PC,PARENT	:GET THE PARAMETERS
8	030700	010346	PARENT: MOV R3,-(SP)	:SAVE THE PARAMETER TABLE ADDRESS
9	030702	005037 001340	CLR CFLAG	:CLEAR THE 'CONTROL C' FLAG
10	030706	012337 030716	1\$: MOV (R3)+,2\$:ADDRESS OF PARAMETER NAME
11	030712	0C1442	BEQ 7\$:BR IF AT END OF TABLE
12	030714	104401	TYPE	:TYPE THE PARAMETER NAME
13	030716	000000	2\$: .WORD 0	:ADDRESS OF PARAMETER NAME TEXT
14	030720	012302	MOV (R3)+,R2	:MAXIMUM PARAMETER VALUE
15	030722	012305	MOV (R3)+,R5	:ADDRESS OF PARAMETER
16	030724	011546	MOV (R5),-(SP)	:CURRENT VALUE OF PARAMETER
17	030726	104405	TYPDS	:TYPE THE CURRENT VALUE OF THE PARAMETER
18	030730	104401 077464	TYPE .SLASH	: '/'
19	030734	104411	RDLIN	:READ THE KEYBOARD
20	030736	012601	MOV (SP)+,R1	:INPUT ASCII STRING ADDRESS
21	030740	005737 001340	TST CFLAG	: 'CONTROL C' ENTERED ?
22	030744	001021	BNE 6\$:BR IF IT WAS
23	030746	004537 032772	JSR R5,CK.DIG	:CHECK THE DIGIT(S)
	030752	030706	1\$:CARRIAGE RETURN ONLY ENTERED
	030754	031020	7\$:PERIOD ONLY ENTERED
	030756	030772	4\$:ILLEGAL INPUT
	030760	030766	3\$:TERMINATED WITH A CARRIAGE RETURN
	030762	030772	4\$:TERMINATED WITH A '...'
	030764	031004	5\$:TERMINATED WITH A '...'
24	030766	010215	3\$: MOV R2,(R5)	:MOVE NEW VALUE TO PARAMETER LOCATION
25	030770	000746	BR 1\$:GET MORE PARAMETERS
26	030772	104401 076677	4\$: TYPE .BADENT	: 'BAD ENTRY'
27	030776	162703 000006	SUB #6,R3	:DECREMENT THE TABLE POINTER
28	031002	000741	BR 1\$:TRY AGAIN
29	031004	010215	5\$: MOV R2,(R5)	:NEW VALUE
30	031006	000404	BR 7\$:EXIT
31	031010	005037 001340	6\$: CLR CFLAG	:CLEAR THE 'CONTROL C' FLAG
32	031014	011603	MOV (SP),R3	:RELOAD THE PARAMETER TABLE ADDRESS
33	031016	000733	BR 1\$:TRY AGAIN
34	031020	005726	7\$: TST (SP)+	:CORRECT THE STACK POINTER
35	031022	000207	RTS PC	:RETURN

```

1      :TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
2      :CALL:
3      :      MOV      #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
4      :      JSR      PC,ASNERR
5      :      RETURN
6
7 031024 104401 001203  ASNERR: TYPE      ,SCLRF      ;CR-LF
8 031030 104401 075613  TYPE      ,QUES      ;'?'
9 031034 104401 075625  TYPE      ,UNTMSG     ;TYPE 'DRIVE'
10 031040 010446  MOV      R4,-(SP)    ;;SAVE R4 FOR TYPEOUT
                        ;;TYPE DRIVE NUMBER
                        ;;GO TYPE--OCTAL ASCII
                        ;;TYPE 2 DIGIT(S)
                        ;;SUPPRESS LEADING ZEROS
                        ;;TYPE SPECIFIC MESSAGE
                        ;;MESSAGE ADDRESS
        031042 104403  TYPOS
        031044      002  .BYTE      2
        031045      000  .BYTE      0
11 031046 104401  TYPE
12 031050 000000  ASNMSG: .WORD      0
13 031052 000207  RTS      PC
14
15      :DEASSIGN DRIVE IF A FATAL ERROR OCCURS
16      :CALL:
17      :      JSR      PC,DROP
18      :      RETURN
19
20 031054 005004  DROP:   CLR      R4      ;CLEAR R4 FOR DRIVE NUMBER
21 031056 111004  MOVB     (R0),R4      ;MOVE DRIVE NUMBER TO R4
22 031060 146437 040142 001530  BICB     ATABIT(R4),ASNLIST ;REMOVE DRIVE FROM ASSIGNED LIST
23 031066 146437 040142 031662  BICB     ATABIT(R4),AUTLIST ;DELETE DRIVE FROM AUTO ASSIGN LIST
24 031074 006304  ASL      R4      ;MAKE DRIVE NUMBER INTO A TABLE INDEX
25 031076 010064 001532  MOV      R0,DUNIT(R4) ;PUT DRIVE I' DROP LIST
26 031102 104401 001203  TYPE      ,SCLRF
27 031106 104401 076236  TYPE      ,DROPNG     ;TYPE 'FATAL OR EXCESSIVE ERRORS'
28 031112 104401 076311  TYPE      ,MSGON      ;TYPE 'ON'
29 031116 104401 075625  TYPE      ,UNTMSG     ;TYPE 'DRIVE'
30 031122 006204  ASR      R4      ;DRIVE NUMBER
31 031124 010446  MOV      R4,-(SP)    ;;SAVE R4 FOR TYPEOUT
                        ;;TYPE DRIVE NUMBER
                        ;;GO TYPE--OCTAL ASCII
                        ;;TYPE 2 DIGIT(S)
                        ;;SUPPRESS LEADING ZEROS
        031126 104403  TYPOS
        031130      002  .BYTE      2
        031131      000  .BYTE      0
32 031132 000207  1$:    RTS      PC
33
34      :ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
35
36 031134 032777 000020 150012  ABNRML: BIT      #SW04,@SWR ;SEE IF SWITCH 4 SET
37 031142 001006 1$      BNE      1$      ;BR IF IT'S SET
38 031144 023760 001444 000072  CMP      MAXER,$TOTAL(R0) ;CHECK TOTAL ERROR VALUE
39 031152 101002 1$      BHI      1$      ;BR IF ERRORS DO NOT EXCEED MAX
40 031154 000137 031054  JMP      DROP        ;DEASSING THE DRIVE
41 031160 000207  1$:    RTS      PC      ;RETURN
42
43      :ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
44

```

.SBTTL END OF PASS ROUTINE

;*INCREMENT THE PASS NUMBER (\$PASS)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO RETURN

```

031162          $EOP:
031162 005737 001466      TST      ENDING          ;END OF PASS DETERMINED BY SEEKS OR DATA WORDS ?
031166 001412          BEQ      EOP1            ;BR IF SEEKS
031170 026037 000040 001436  CMP      $ENDAT+2(RO),ENDCON+2      ;CHECK MSW OF WORDS DATA COUNT
031176 101020          BHI      EOP2            ;BR IF MSW GREATER THAN LIMIT
031200 103404          BLO      1$              ;BR IF MSW LESS THAN LIMIT
031202 026037 000036 001434  CMP      $ENDAT(RO),FNDCON          ;CHECK LSW AGAINST LIMIT
031210 103013          BHIS     EOP2            ;BR IF EQUAL OR GREATER
031212 000207          1$:      RTS      PC

031214 026037 000044 001442  EOP1:   CMP      $ENDSK+2(RO),ENDSEK+2      ;CHECK MSW OF SEEK COUNT
031222 101006          BHI      EOP2            ;BR IF MSW GREATER THAN LIMIT
031224 103404          BLO      1$              ;EXIT IF MSW LESS THAN LIMIT
031226 026037 000042 001440  CMP      $ENDSK(RO),ENDSEK          ;CHECK LSW OF SEEK COUNT
031234 103001          BHIS     EOP2            ;BR IF EQUAL OR GREATER
031236 000207          1$:      RTS      PC

031240 010446          EOP2:   MOV      R4,-(SP)          ;SAVE R4
031242 032777 000400 147704  BIT      #SW0B,@SWR                ;INHIBIT END OF PASS TYPEOUT ?
031250 001023          BNE      1$              ;BR IF YES
031252 104401 001203          TYPE     ,$CRLF                ;CR-LF
031256 104401 076272          TYPE     ,ENDPAS                ;END OF PASS FOR THE DRIVE
031262 016046 000104          MOV      $PASSC(RO),-(SP)          ;;SAVE $PASSC(RO) FOR TYPEOUT
031266 104405          TYPDS     ;;GO TYPE--DECIMAL ASCII WITH SIGN
031270 111037 001324          MOVB     (RO),UNIT                ;STORE THE DRIVE NUMBER
031274 104401 076311          TYPE     ,MSGON                ;TYPE 'ON'
031300 104401 075625          TYPE     ,UNTMMSG                ;'DRIVE '
031304 013746 001324          MOV      UNIT,-(SP)              ;;SAVE UNIT FOR TYPEOUT
031310 104403          TYPOS     ;;GO TYPE--OCTAL ASCII
031312 002          .BYTE     2                ;;TYPE 2 DIGIT(S)
031313 000          .BYTE     0                ;;SUPPRESS LEADING ZEROS
031314 104401 001203          TYPE     ,$CRLF                ;CR-LF
031320 111004          1$:      MOVB     (RO),R4            ;MOVE DRIVE NUMBER

031322 105737 001150          TSTB     $AUTOB                  ;RUNNING IN AUTO MODE ?
031326 001410          BEQ      2$                    ;BR IF NO
031330 136437 040142 031662  BITB     ATABIT(R4),AULST          ;IS DRIVE ALRFADY ASSIGNED TO AUTO LIST ?
031336 001071          BNE      6$                    ;BR IF YES
031340 156437 040142 031662  BISB     ATABIT(R4),AULST          ;ADD DRIVE TO AUTO ASSIGN LIST
031346 000443          BR      3$

031350 026037 000104 001456  2$:      CMP      $PASSC(RO),PASSES          ;SEE IF AT END OF TEST
031356 103437          BLO      3$                    ;BR IF NOT
031360 032777 000020 147566  BIT      #SW04,@SWR                ;TYPE END OF TEST MESSAGE ?
031366 001033          BNE      3$                    ;BR IF NO
031370 104401 076316          TYPE     ,ENDTST                ;TYPE 'END OF TEST'
031374 104401 076334          TYPE     ,MSGFOR                ;TYPE 'FOR'
031400 104401 075625          TYPE     ,UNTMMSG                ;'DRIVE '
031404 013746 001324          MOV      UNIT,-(SP)              ;;SAVE UNIT FOR TYPEOUT
031410 104403          TYPOS     ;;GO TYPE--OCTAL ASCII

```

```

031412      002      .BYTE      2      ;;TYPE 2 DIGIT(S)
031413      000      .BYTE      0      ;;SUPPRESS LEADING ZEROS
031414      146437   040142  001530   BICB      ATABIT(R4),ASNLST ;;DELETE DRIVE FROM ASSIGNED LIST
031422      006304   ASL      R4      ;MAKE DRIVE NUMBER INTO TABLE INDEX
031424      010064   001532   MOV      R0,DUNIT(R4) ;PUT BLOCK ADDRESS INTO DROP LIST
031430      105737   001530   TSTB     ASNLST    ;ALL DRIVES ARE DEASSIGNED ?
031434      001041   BNE      7$      ;BR IF NO
031436      005237   001216   INC      $DEVCT    ;INCREMENT DEVICE COUNT
031442      005237   001214   INC      $PASS     ;INCREMENT THE PASS COUNT
031446      042737   100000  001214   BIC      #100000,$PASS ;AVOID NEGATIVE NUMBER
031454      000431   BR       7$

031456      032777   000400  147470  3$:      BIT      #SW08,@SWR    ;INHIBIT END OF PASS TYPEOUT ?
031464      001002   BNE      4$      ;BR IF YES
031466      004737   023636   JSR      PC,SUMARY ;TYPE THE DRIVE'S STATISTICS SUMMARY
031472      010346   4$:      MOV      R3,-(SP) ;SAVE R3
031474      010004   MOV      R0,R4    ;DRIVE'S BLOCK ADDRESS
031476      062704   000036   ADD      #$ENDAT,R4 ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
031502      012703   000006   MOV      #6,R3    ;NUMBER OF LOCNS TO BE CLEARED
                                ;(CLEAR $ENDAT, $ENDSK AND $OPERC COUNTERS)
031506      005024   5$:      CLR      (R4)+    ;CLEAR THE LOCN
031510      005303   DEC      R3      ;DECREMENT THE LOCATION COUNTER
031512      001375   BNE      5$      ;BR IF MORE TO GO
031514      012603   MOV      (SP)+,R3 ;RESTORE R3
031516      005260   000104   INC      $PASSC(R0) ;INCREMENT THE PASS COUNT

031522      105737   001150   6$:      TSTB     $AUTOB    ;RUNNING IN AUTO MODE ?
031526      001404   BEQ      7$      ;BR IF NO
031530      023737   001530  031662   (MP      ASNLST,AUTLST ;HAVE ALL DRIVES COMPLETED PASS IN AUTO MODE ?
031536      001402   BEQ      8$      ;BR IF YES
031540      012604   7$:      MOV      (SP)+,R4 ;RESTORE R4
031542      000207   RTS      PC      ;RETURN

031544      005237   031662   8$:      INC      AUTLST    ;CLEAR AUTO ASSIGN LIST FOR NEXT PASS AND
031550      001375   BNE      8$      ;WAIT FOR TTY
031552      012737   000340  177776   MOV      #PR7,PS   ;DON'T ALLOW ANY INTERRUPTS
031560      005237   001216   INC      $DEVCT    ;INCREMENT DEVICE COUNT
031564      005237   001214   INC      $PASS     ;;INCREMENT THE PASS NUMBER
031570      042737   100000  001214   BIC      #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
031576      005327   DEC      (PC)+    ;;LOOP?
031600      000001   $EOPCT: .WORD     1
031602      003013   BGT      $DOAGN    ;;YES
031604      012737   MOV      (PC)+,@(PC)+ ;;RESTORE COUNTER
031606      000001   $ENDCT: .WORD     1
031610      031600   $EOPCT
031612      013700   000042   $GET42: MOV      @#42,R0 ;;GET MONITOR ADDRESS
031616      001405   BEQ      $DOAGN    ;;BRANCH IF NO MONITOR
031620      000005   RESET    ;;CLEAR THE WORLD
031622      004710   $ENDAD: JSR      PC,(R0) ;;GO TO MONITOR
031624      000240   NOP      ;;SAVE ROOM
031626      000240   NOP      ;;FOR
031630      000240   NOP      ;;ACT11
031632      $DOAGN:
031632      000137   JMP      @(PC)+    ;;RETURN
031634      031636   $RTNAD: .WORD     RTURN
2
3 031636      012706   001100   RTURN:  MOV      #STACK,SP ;RESTORE STACK

```

CRMU80 RM05/3/2 PERF EXER
END OF PASS ROUTINE

MACRO V04.00 4-APR-81 01:42:23 PAGE 35-2 M 11

SEQ 0142

4 031642 005237 001212
5 031646 004737 033226
6 031652 004737 023344
7 031656 000137 006142
8
9 031662 000000

INC \$TESTN
JSR PC,\$TKINT
JSR PC,CKCLK
JMP MAIN

;INCREMENT THE TEST NUMBER IN THE MAIL BOX
;MAKE SURE KEYBOARD INTERRUPT AND
;SYSTEM CLOCK ARE STILL ON.
;RETURN TO LOOP

AUTLST: .WORD 0

;AUTO ASSIGN LIST (USED IN AUTO RUN MODE)

```

1
2
3
4
5
6
7
8 031664 013746 037024
9 031670 013746 037022
10 031674 010546
11 031676 004737 031710
12 031702 012605
13 031704 005726
14 031706 000207
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 031710 104412
40 031712 016605 000026
41 031716 005004
42 031720 016602 000030
43 031724 016603 000032
44 031730 005000
45 031732 005001
46 031734 004737 031756
47 031740 010166 000030
48 031744 010366 000032
49 031750 104413
50 031752 012616
51 031754 000207
52
53
54
55
56
57

:ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
:CALL:
:      MOV     NUMBER,R5      ;DIVISOR INTO R5
:      JSR     PC,GETREM
:      RETURN                    ;REMAINDER IS IN R5

GETREM: MOV     $LONUM,-(SP)   ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
        MOV     $HINUM,-(SP)   ;UPPER PART
        MOV     R5,-(SP)       ;PUT THE DIVISOR ONTO THE STACK
        JSR     PC,$DIV        ;DIVIDE THE RANDOM NUMBERS
        MOV     (SP)+,R5       ;PUT THE REMAINDER INTO R5
        TST     (SP)+         ;ADJUST THE STACK POINTER
        RTS     PC

.SBTTL  INTEGER DIVIDE ROUTINE

:*****
:*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER
:*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING
:*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.
:*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE
:*SAME SIGN AS THE DIVIDEND.
:*CALL:
:*      MOV     LOW DIVIDEND,-(SP)   ;;THE HIGH DIVIDEND MUST BE < 1/2
:*      MOV     HIGH DIVIDEND,-(SP)  ;;AS LARGE AS THE DIVISOR
:*      MOV     DIVISOR,-(SP)
:*      JSR     PC,$DIV
:*      RETURN                        ;;QUOTIENT & REMAINDER ARE ON THE STACK

:      STACK  NO ERROR      OVERFLOW      DIVIDE BY ZERO
:      -----
:*      TOP    REMAINDER    ALL ZEROS      ALL ONES
:*      +2     QUOTIENT     ALL ZEROS      ALL ONES
:
:*NOTE: THIS ROUTINE WILL LINK TO THE DIVISION SUBROUTINE ('M.DPID').

$DIV:  SAVREG                    ;STORE R0 - R5
        MOV     26(SP),R5        ;DIVISOR
        CLR     R4               ;OTHER DIVISOR WORD
        MOV     30(SP),R2        ;UPPER DIVIDEND WORD
        MOV     32(SP),R3        ;LOWER DIVIDEND WORD
        CLR     R0               ;CLEAR OTHER DIVIDEND REGISTERS
        CLR     R1
        JSR     PC,M.DPID        ;GO TO THE DIVIDE ROUTINE
        MOV     R1,30(SP)        ;REMAINDER ON THE STACK
        MOV     R3,32(SP)        ;QUOTIENT ON THE STACK
        RESREG                    ;RESTORE R0 - R5
        MOV     (SP)+,(SP)       ;MOVE RETURN UP THE STACK
        RTS     PC

.SBTTL  DOUBLE PRECISION DIVISION SUBROUTINE

:CALL:
:      JSR     PC,M.DPID
:

```

```

58      :      DIVIDEND = R0-R1-R2-R3 (R0=MSD)
59      :      DIVISOR = R4-R5 (R4=MSD)
60      :
61      : RETURN
62      :
63      :      REMAINDER AFTER DIVISION = R0-R1 (R0=MSD)
64      :      QUOTIENT AFTER DIVISION = R2-R3 (R2=MSD)
65      :
66 031756 012746 000040 M.DPID: MOV #40,(SP) ;COUNTER FOR DIVISION CYCLES
67 031762 010446 MOV R4,-(SP) ;HIGH ORDER
68 031764 010546 MOV R5,-(SP) ;LOW ORDER DIVISOR TO THE STACK
69 031766 005466 000002 NEG 2(SP) ;FORM NEGATIVE
70 031772 005416 NEG @SP ;VERSION OF THE DIVISOR
71 031774 005666 000002 SBC 2(SP)
72 032000 061601 ADD @SP,R1
73 032002 005500 ADC R0 ;PERFORM THE INITIAL SUBTRACTION
74 032004 066600 000002 ADD 2(SP),R0
75 032010 103445 BCS M.DP50 ;IF CARRY THEN OVERFLOW HAS OCCURRED
76 032012 005046 CLR -(SP) ;THIS IS A LONGER LASTING CARRY BIT
77 032014 006103 M.DP40: ROL R3
78 032016 006102 ROL R2
79 032020 006101 ROL R1
80 032022 006100 ROL R0
81 032024 005716 TST @SP ;TEST "CARRY" INDICATOR
82 032026 011410 BEQ M.DP41 ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
83 032030 005016 CLR @SP ;CLEAR UP FOR NEXT TIME
84 032032 066601 000002 ADD 2(SP),R1
85 032036 005500 ADC R0 ;ADD -(DIVISOR)
86 032040 005516 ADC @SP ;SET "CARRY"
87 032042 066600 000004 ADD 4(SP),R0;<-
88 032046 000404 BR M.DP42
89 032050 060501 M.DP41: ADD R5,R1
90 032052 005500 ADC R0 ;ADD +(DIVISOR)
91 032054 005516 ADC @SP ;SET "CARRY"
92 032056 060400 ADD R4,R0;<-
93 032060 005516 M.DP42: ADC @SP ;SET "CARRY"
94 032062 005716 TST @SP ;TEST THE UPDATE INDICATOR
95 032064 001401 BEQ .+4 ;-> ;IF ZERO FORGET IT
96 032066 005203 INC R3 ;I ;NO CARRY POSSIBLE HERE
97 032070 005366 000006 DEC 6(SP) ;<- ;DECREMENT COUNTER
98 032074 003347 BGT M.DP40 ;BRANCH IF MORE TO DO
99 032076 006003 ROR R3
100 032100 103404 BCS M.DP44
101 032102 060501 ADD R5,R1
102 032104 005500 ADC R0
103 032106 060400 ADD R4,R0
104 032110 000241 CLC
105 032112 006103 M.DP44: ROL R3
106 032114 062706 000010 ADD #10,SP ;ADJUST STACK BY 4 WORDS
107 032120 000242 CLV
108 032122 000207 RTS PC
109 032124 062706 000006 M.DP50: ADD #6,SP
110 032130 000262 SEV
111 032132 000207 RTS PC
    
```


.SBTTL SUPRS - TYPE ASCIZ, REPLACE LEADING 0'S WITH BLANKS
 .SBTTL SUPRSL -TYPE ASCIZ, LEFT JUSTIFY

.....

```

CALL:
      MOV   #NUMADR, -(SP) ;FIRST ADDRESS OF ASCIZ STRING
      JSR   PC, SUPRS
      OR
      MOV   #NUMADR, -(SP) ;FIRST ADDRESS OF ASCI7 STRING
      JSR   PC, SUPRSL

SUPRSL: MOV   RO, -(SP)      ;SAVE RO
        MOV   4(SP), RO     ;GET POINTER TO MESSAGE
        CLR   SUPR2
        BR    SUPR1

SUPRS:  MOV   RO, -(SP)      ;SAVE RO
        MOV   4(SP), RO     ;GET POINTER TO MESSAGE
        MOV   RO, SUPR2     ;GET POINTER FOR TYPING

1$:    TSTB  (RO)           ;TEST FOR TERMINATOR
        BEQ  2$             ;YES
        CMPB #'0', (RO)    ;IS THIS A '0' ?
        BNE  3$             ;NO
        MOVB #40, (RO)+    ;REPLACE IT WITH A 'BLANK'
        BR   1$            ;NEXT CHAR.

2$:    DEC   RO             ;BACKUP 1
        MOVB #'0', (RO)    ;MAKE IT '0'

3$:    TST   SUPR2          ;LEFT JUSTIFY ?
        BNE  4$             ;NO
        MOV  RO, SUPR2     ;YES

4$:    TYPE  .WORD         0
SUPR2: MOV   (SP)+, RO      ;RESTORE RO
        MOV  (SP)+, (SP)   ;RESTORE STACK
        RTS  PC
  
```

4
5
6
7
8
9
10
11
12 032134 0 0046
13 032136 015600 000004
14 032142 005037 032224
15 032146 000405
16
17 032150 010046
18 032152 016600 000004
19 032156 010037 032224
20 032162
21 032162 105710
22 032164 001406
23 032166 122710 000060
24 032172 001006
25 032174 112720 000040
26 032200 000770
27 032202 005300
28 032204 112710 000060
29 032210 005737 032224
30 032214 001002
31 032216 010037 032224
32 032222 104401
33 032224 000000
34 032226 012600
35 032230 012616
36 032232 000207

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

.SBTTL \$SUPRS - TYPE ASCII, REPLACE LEADING 0'S WITH BLANKS
.SBTTL \$SUPRL - TYPE ASCII, LEFT JUSTIFY

: THIS ROUTINE IS SAME AS 'SUPRSL' AND 'SUPRS', EXCEPT THAT IT
: WILL SUPPRESS THE ERROR TYPEOUT IF SW13=1. THIS ACCOMPLISHED BY
: USED TRAP CALL 'DISPLY', INSTEAD OF 'TYPE'.
: CALL:

: MOV #NUMADR, -(SP) ; FIRST ADDRESS OF ASCII STRING
: JSR PC, \$SUPRS

: OR
: MOV #NUMADR, -(SP) ; FIRST ADDRESS OF ASCII STRING
: JSR PC, \$SUPRL

\$SUPRL: MOV R0, -(SP) ; SAVE R0
: MOV 4(SP), R0 ; GET POINTER TO MESSAGE
: CLR \$SUPR2
: BR \$SUPR1

\$SUPRS: MOV R0, -(SP) ; SAVE R0
: MOV 4(SP), R0 ; GET POINTER TO MESSAGE
: MOV R0, \$SUPR2 ; GET POINTER FOR TYPING

\$SUPR1:
1\$: TSTB (R0) ; TEST FOR TERMINATOR
: BEQ 2\$; YES
: CMPB #'0', (R0) ; IS THIS A '0' ?
: BNE 3\$; NO
: MOVB #40, (R0)+ ; REPLACE IT WITH A 'BLANK'
: BR 1\$; NEXT CHAR.

2\$: DEC R0 ; BACKUP 1
: MOVB #'0', (R0) ; MAKE IT '0'
3\$: TST \$SUPR2 ; LEFT JUSTIFY ?

: BNE 4\$; NO
: MOV R0, \$SUPR2 ; YES
4\$: DISPLY .WORD 0 ; TYPE, UNLESS SW13-1

\$SUPR2: MOV (SP)+, R0 ; RESTORE R0
: MOV (SP)+, (SP) ; RESTORE STACK
: RTS PC

032234 010046
032236 016600 000004
032242 005037 032324
032246 000405
032250 010046
032252 016600 000004
032256 010037 032324
032262 105710
032264 001406
032266 122710 000060
032272 001006
032274 112720 000040
032300 000770
032302 005300
032304 112710 000060
032310 005737 032324
032314 001002
032316 010037 032324
032322 104414
032324 000000
032326 012600
032330 012616
032332 000207

```

1
2
3
4
5
6
7
8
9
10
11 032334 005237 032442
12
13 032340 010046
14 032342 016600 000004
15 032346 005737 032442
16 032352 001014
17 032354 122710 000060
18 032360 001004
19 032362 112710 000040
20 032366 005200
21 032370 000771
22 032372 105710
23 032374 001003
24 032376 005300
25 032400 112710 000060
26 032404 016600 000004
27 032410 105720
28 032412 001376
29 032414 005300
30 032416 162500
31 032420 010037 032426
32 032424 104401
33 032426 000000
34 032430 012600
35 032432 012616
36 032434 005037 032442
37 032440 000205
38
39 032442 000000
40
41
42
43 032444 013746 177776
44 032450 012737 000200 177776
45 032456 012537 032466
46 032462 004737 035450
47 032466 000000
48 032470 000205
49
50
51
52
53
54
55
56
57

```

```

:ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
:CALL:
:
:   MOV   #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
:   JSR   R5,REPLZ      ;REPLACE PRECEDING ZEROS WITH BLANKS
:   .WORD N              ;'N' IS NUMBER OF DIGITS TO BE TYPED
:
:   OR
:
:   MOV   #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
:   JSR   R5,FILLZ      ;TYPE PRECEDING ZEROS
:   .WORD N              ;'N' IS NUMBER OF DIGITS TO BE TYPED
:
FILLZ: INC   FILL0      ;LEAVE ZERO'S
:
REPLZ: MOV   R0,-(SP)      ;SAVE R0
:   MOV   4(SP),R0        ;ADDRESS OF NUMBER TO R0
:   TST   FILL0          ;LEAVE PRECEDING ZEROS ?
:   BNE   3$             ;BR IF YES
1$:   CMPB #'0,(R0)        ;BYTE EQUAL TO ASCII '0' ?
:   BNE   2$             ;BR IF NOT
:   MOVB  #40,(R0)       ;REPLACE THE ZERO WITH A SPACE
:   INC   R0             ;INCREMENT THE BYTE ADDRESS
:   BR    1$            ;GO BACK AND LOOK FOR MORE LEADING ZEROS
2$:   TSTB (R0)          ;SEE IF ZERO BYTE TERMINATOR
:   BNE   3$            ;BR IF NOT
:   DEC   R0             ;BACKUP STRING POINTER
:   MOVB  #'0,(R0)       ;PUT A ZERO BACK IN
3$:   MOV   4(SP),R0      ;PUT ADDRESS OF FIRST CHARACTER ON STACK
4$:   TSTB (R0)+         ;SEE IF ZERO BYTE TERMINATOR
:   BNE   4$            ;BR IF NOT
:   DEC   R0             ;BACKUP STRING POINTER
:   SUB   (R5)+,R0       ;ADJUST ADDRESS
:   MOV   R0,5$         ;GET ADDRESS FOR TYPEOUT
:   TYPE
5$:   .WORD 0            ;TYPE THE NUMBER
:   MOV   (SP)+,R0       ;ADDRESS OF NUMBER
:   MOV   (SP)+,(SP)    ;POP STACK INTO R0
:   CLR   FILL0         ;RESTORE STACK
:   RTS   R5            ;RESET FILL FLAG
:                       ;RETURN
:
FILL0: .WORD 0          ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE
:
:ROUTINE TO TYPE AT PRIORITY 4
TYPRI4: MOV   @#PS,-(SP)  ;SAVE THE PRESENT STATUS
:   MOV   #200,@#PS     ;CHANGE THE PRIORITY TO 4
:   MOV   (R5)+,1$      ;MESSAGE ADDRESS
:   JSR   PC,$TYPE      ;TYPE THE MESSAGE
1$:   .WORD 0            ;MESSAGE ADDRESS GOES HERE
:   RTS   R5            ;RETURN
:
:ROUTINE TO TYPE THE MBA (DRIVE) SERIAL NUMBER IN DECIMAL
:CALL:
:
:   MOV   #DPB,R0       ;ADDRESS OF DRIVE PARAMETER BLOCK
:   JSR   PC,TYMBA     ;CALL ROUTINE
:
:   OR
:
:   MOV   #DPB,R0       ;ADDRESS OF DRIVE PARAMETER BLOCK
:   JSR   PC,TYPMBA    ;CALL ROUTINE (WITH NO HEADER MESSAGE)
:

```

```

58
59
60 032472 104401 077303
61 032476 010037 032512
62 032502 062737 002126 032512
63 032510 104401
64 032512 000000
65 032514 104401 076477
66 032520 000207

```

```

;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
TYMBA: TYPE ,MBAASN ;TYPE 'MBA S/N:'
TYPMBA: MOV RO,1$ ;ADDRESS OF DPB
ADD #SMBASN,1$ ;INDEX TO DRIVE (MBA) SERIAL NUMBER
TYPE ;TYPE THE DRIVE (MBA) SERIAL NUMBER
1$: .WORD 0 ;ADDRESS OF DRIVE (MBA) SERIAL NUMBER FIELD
TYPE ,PERIOD ;TYPE '.'
RTS PC ;RETURN

```

```

67
68
69
70
71
72
73
74
75
76
77

```

```

;ROUTINE TO TYPE THE PACK SERIAL NUMBER IN OCTAL
:CALL:
: MOV #DPB,RO ;ADDRESS OF DRIVE PARAMETER BLOCK
: JSR PC,TYPACK ;CALL ROUTINE
: OR
: MOV #DPB,RO ;ADDRESS OF DRIVE PARAMETER BLOCK
: JSR PC,TYPCK ;CALL ROUTINE (WITH NO HEADER MESSAGE)
;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE

```

```

78 032522 104401 077270
79 032526 005760 000142
80 032532 002003
81 032534 104401 076672
82 032540 000425
83 032542 016046 000142
84 032546 001414
85 032550 004737 033160
86 032554 004737 032134
87 032560 016046 000140
88 032564 004737 033160
89 032570 004537 032334
90 032574 000005
91 032576 000406
92 032600 016016 000140
93 032604 004737 033160
94 032610 004737 032134
95 032614 000207

```

```

TYPACK: TYPE ,PACKSN ;TYPE 'PACK S/N:'
TYPCK: TST $PSNM(RO) ;IS SERIAL NUMBER VALID ?
BGE 1$ ;BR IF VALID
TYPE ,NONE ;TYPE 'NONE'
BR 3$
1$: MOV $PSNM(RO),-(SP) ;GET HI NUMBER (MSB)
BEQ 2$ ;BR IF ZERO
JSR PC,$SB20 ;CONVERT TO OCTAL NUMBER
JSR PC,SUPRSL ;AND TYPE IT LEFT JUSTIFIED
MOV $PSNL(RO),-(SP) ;GET LOW NUMBER (LSB)
JSR PC,$SB20 ;CONVERT TO OCTAL NUMBER
JSR R5,FILLZ ;TYPE WITH PRECEDING ZEROS
.WORD 5 ;5 DIGITS
BR 3$
2$: MOV $PSNL(RO),(SP) ;GET LOW NUMBER (LSB)
JSR PC,$SB20 ;CONVERT TO OCTAL NUMBER
JSR PC,SUPRSL ;AND TYPE IT LEFT JUSTIFIED
3$: RTS PC ;RETURN

```

```

96
97
98
99
100
101
102

```

```

;ROUTINE TO TYPE ERRORS
:CALL:
: DISPLY ;MUST DEFINED IN 'TRAP' TABLE
: MESADR ;ADDRESS OF MESSAGE
: RETURN

```

```

103 032616 032777 020000 146330
104 032624 001004
105 032626 005037 177776
106 032632 000137 035450
107 032636 062716 000002
108 032642 000002

```

```

$DSPLY: BIT #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
BNE 1$ ;BR IF YES
CLR @#PS ;SET PRIORITY TO ZERO
JMP $TYPE ;TYPE THE MESSAGE
1$: ADD #2,(SP) ;INCREMENT THE RETURN
RTI ;RETURN

```

1
 2
 3
 4
 5
 6
 7
 8
 9
 10 032644 121127 000060
 11 032650 103407
 12 032652 121127 000067
 13 032656 101004
 14 032660 111102
 15 032662 042702 177770
 16 032666 005725
 17 032670 000205
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28 032672 121127 000060
 29 032676 103407
 30 032700 121127 000071
 31 032704 101004
 32 032706 111102
 33 032710 042702 000060
 34 032714 005725
 35 032716 000205
 36
 37
 38
 39
 40
 41
 42
 43
 44
 45
 46
 47
 48
 49
 50 032720 105711
 51 032722 001417
 52 032724 121127 000054
 53 032730 001413
 54 032732 121127 000056
 55 032736 001407
 56 032740 004537 032672
 57 032744 000410

```

;THIS ROUTINE IS USED TO CHECK IF AN
;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
;CALL:
      MOV     #ADR,R1           ;ADDRESS OF ASCII CHARACTER
      JSR     R5,CK.OCT        ;CHECK THE CHARACTER
      RETURN1                    ;CHARACTER IS NOT BETWEEN 0-7
      RETURN2                    ;CHARACTER IS IN R2 AS A
                                   ;OCTAL DIGIT
CK.OCT:  CMPB   (R1),#'0        ;LESS THAN ZERO?
          BLO   1$             ;YES -- BRANCH
          CMPB   (R1),#'7        ;GREATER THAN SEVEN?
          BHI   1$             ;YES -- BRANCH
          MOVB   (R1),R2         ;GET THE CHARACTER
          BIC   #'(7,R2         ;STRIP AWAY THE ASCII
          TST   (R5)+           ;ADJUST FOR RETURN
          RTS    R5             ;RETURN

;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
;CALL:
      MOV     #ADR,R1           ;ADDRESS OF ASCII CHARACTER
      JSR     R5,CK.DEC        ;CHECK THE CHARACTER
      RETURN1                    ;NOT BETWEEN 0 AND 9
      RETURN2                    ;BETWEEN 0 AND 9
                                   ;R2 = DIGIT
CK.DEC:  CMPB   (R1),#'0        ;LESS THAN ZERO?
          BLO   1$             ;YES -- BRANCH
          CMPB   (R1),#'9        ;GREATER THAN NINE?
          BHI   1$             ;YES -- BRANCH
          MOVB   (R1),R2         ;GET THE CHARACTER
          BIC   #'0,R2         ;STRIP AWAY THE ASCII
          TST   (R5)+           ;ADJUST FOR RETURN
          RTS    R5             ;RETURN

;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
;DETERMINE WHAT IT IS.
;CALL:
      MOV     #ADR,R1           ;ADDRESS OF ASCII CHARACTER
      JSR     R5,CK.CHR        ;CHECK CHARACTER
      RETURN ADR1                ;UNKNOWN CHARACTER
      RETURN ADR2                ;CARRIAGE RETURN * (R1)=ADR+1
      RETURN ADR3                ;COMMA * (R1)=ADR+1
      RETURN ADR4                ;PERIOD * (R1)=ADR+1
      RETURN ADR5                ;DIGIT BETWEEN 0 AND 7.
      RETURN ADR6                ;DIGIT BETWEEN 8 AND 9.
                                   ;R2 = DIGIT * (R1)=ADR+1
CK.CHR:  TSTB   (R1)            ;"CARRIAGE RETURN"?
          BEQ   3$             ;YES -- BRANCH
          CMPB   (R1),#',        ;"COMMA"?
          BEQ   2$             ;YES -- BRANCH
          CMPB   (R1),#'.        ;"PERIOD"?
          BEQ   1$             ;YES -- BRANCH
          JSR   R5,CK.DEC        ;"DIGIT"?
          BR    4$             ;NO -- BRANCH
    
```

```

58 032746 004537 032644 JSR R5,CK.OCT ;OCTAL ?
59 032752 005725 TST (R5)+ ;DIGIT BETWEEN 8-9
60 032754 005725 TST (R5)+ ;DIGIT BETWEEN 0-7
61 032756 005725 1$: TST (R5)+ ;PERIOD
62 032760 005725 2$: TST (R5)+ ;COMMA
63 032762 005725 3$: TST (R5)+ ;CARRIAGE RETURN
64 032764 005201 INC R1 ;MOVE POINTER TO NEXT CHARACTER
65 032766 011505 4$: MOV (R5),R5 ;UNKNOWN CHARACTER
66 032770 000205 RTS R5 ;RETURN
67
68 ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
69 ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
70 ;CALL:
71 : MOV #ADR,R1 ;ADDRESS OF ASCIZ STRING
72 : MOV #NUM,R2 ;MAX. MAGNITUDE OF INPUT NUMBER
73 : JSR R5,CK.DIG ;CHECK DIGITS
74 : RETURN ADR1 ;'CR' ONLY ENTERED -- R2=0
75 : RETURN ADR2 ;'PERIOD' ONLY ENTERED -- R2=0
76 : RETURN ADR3 ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
77 : RETURN ADR4 ;'CR' -- R2 = NUMBER
78 : RETURN ADR5 ;'COMMA' -- R2 = NUMBER
79 : RETURN ADR6 ;'PERIOD' -- R2 = NUMBER
80
81 032772 010446 CK.DIG: MOV R4,-(SP) ;SAVE R4
82 032774 010346 MOV R3,-(SP) ;SAVE R3
83 032776 010246 MOV R2,-(SP) ;SAVE THE MAX. SIZE ON THE STACK
84 033000 005002 CLR R2 ;START WITH 0
85 033002 005003 CLR R3
86 033004 005004 CLR R4
87 033006 004537 032720 JSR R5,CK.CHR ;CHECK ONE CHARACTER
    033012 033106 6$ ;ILLEGAL CHARACTER
    033014 033114 9$ ;CARRIAGE RETURN
    033016 033106 6$ ;
    033020 033110 7$ ;
    033022 033026 1$ ;DIGIT 0-7
    033024 033026 1$ ;DIGIT 8-9
88 033026 062705 000004 1$: ADD #4,R5 ;STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
89 033032 006303 2$: ASL R3 ;INPUT NUMBER *2
90 033034 010346 MOV R3,-(SP) ;SAVE *2
91 033036 006303 ASL R3 ;*4
92 033040 006303 ASL R3 ;*8
93 033042 062603 ADD (SP)+,R3 ;>(*2)+(*8) *10
94 033044 060203 ADD R2,R3 ;UPDATE THE INPUT NUMBER
95 033046 004537 032720 JSR R5,CK.CHR ;CHECK ONE CHARACTER
    033052 033112 8$ ;ILLEGAL CHARACTER
    033054 033076 5$ ;CARRIAGE RETURN
    033056 033074 4$ ;
    033060 033066 3$ ;
    033062 033032 2$ ;DIGIT 0-7
    033064 033032 2$ ;DIGIT 8-9
96 033066 105711 3$: TSTB (R1) ;DOES A 'CR' FOLLOW THE 'PERIOD'?
97 033070 001010 BNE 8$ ;BR IF NOT
98 033072 005724 TST (R4)+ ;INCREMENT THE RETURN
99 033074 005724 4$: TST (R4)+ ;INCREMENT THE RETURN
100 033076 005724 5$: TST (R4)+ ;INCREMENT THE RETURN
101 033100 020316 CMP R3,(SP) ;CHECK THE MAGNITUDE OF THE NUMBER
102 033102 101004 BHI 9$ ;BR IF ENTERED NUMBER TOO LARGE

```

```

103 033104 000402          BR      8$          ;BYPASS INCREMENT
104 033106 005725          6$:   TST      (R5)+       ;INCREMENT RETURN PAST INVALID RETURN
105 033110 005725          7$:   TST      (R5)+       ;INCREMENT RETURN
106 033112 060405          8$:   ADD      R4,R5       ;SETUP RETURN POINTER
107 033114 010302          9$:   MOV      R3,R2       ;ENTERED VALUE
108 033116 005726          TST      (SP)+       ;CLEAN MAX. SIZE OFF OF STACK
109 033120 012603          MOV      (SP)+,R3     ;RESTORE R3
110 033122 012604          MOV      (SP)+,R4     ;RESTORE R4
111 033124 011505          MOV      (R5),R5      ;GET RETURN ADDRESS
112 033126 000205          RTS      R5           ;RETURN
113
114
115          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
116          ;UNSIGNED DECIMAL ASCIZ NUMBER.
117          ;CALL:
118          ;       MOV      NUMBER,-(SP)   ;PUT THE NUMBER ON THE STACK
119          ;       JSR      PC,$SB2D      ;CALL:
120          ;       RETURN                    ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
121          ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
122          ;       THE SYSMAC LIBRARY, REV C AND LATER
123
124 033130 016637 000002 033154 $SB2D: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
125 033136 012746 033154          MOV      #1$,-(SP)    ;SET THE POINTER
126 033142 004737 037122          JSR      PC,$DB2D     ;CALL THE DOUBLE LENGTH CONVERT
127 033146 012666 000002          MOV      (SP)+,2(SP)  ;PICKUP THE POINTER
128 033152 000207          RTS      PC           ;RETURN
129 033154 000000 000000          1$:   .WORD 0,0
130
131          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
132          ;UNSIGNED OCTAL ASCIZ NUMBER.
133          ;CALL:
134          ;       MOV      NUMBER,-(SP)   ;PUT THE NUMBER ON THE STACK
135          ;       JSR      PC,$SB20      ;CALL:
136          ;       RETURN                    ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
137          ;NOTE:  THE PROGRAM REQUIRES THIS FORM OF '$SB20', NOT THE VERSION ON
138          ;       THE SYSMAC LIBRARY, REV C AND LATER
139
140
141 033160 016637 000002 033204 $SB20: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
142 033166 012746 033204          MOV      #1$,-(SP)    ;SET THE POINTER
143 033172 004737 037316          JSR      PC,$DB20     ;CALL THE DOUBLE LENGTH CONVERT
144 033176 012666 000002          MOV      (SP)+,2(SP)  ;PICKUP THE POINTER
145 033202 000207          RTS      PC           ;RETURN
146 033204 000000 000000          1$:   .WORD 0,0
  
```

.SBTTL TTY INPUT ROUTINE

033210 000000
033212 000000
033214 000000
033216 033225

```
.....  
:ENABL LSB  
$TKCNT: .WORD 0 ;;NUMBER OF ITEMS IN QUEUE  
$TKQIN: .WORD 0 ;;INPUT POINTER  
$TKQOUT: .WORD 0 ;;OUTPUT POINTER  
$TKQSR: .BLKB 7 ;;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
```

033226 005037 033210
033232 012737 033216 033212
033240 013737 033212 033214
033246 012737 033276 000060
033254 012737 000200 000062
033262 005777 145674
033266 012777 000100 145664
033274 000207

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

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

033276 117746 145660
033302 042716 177600
033306 021627 000021
033312 001002
033314 005726
033316 000002
033320
033320 021627 000003
033324 001007
033326 104401 034433
033332 004737 033226
033336 005726
033340 000137 034474
033344 021627 000007
033350 001004
033352 022737 000176 001154
033360 001500

033362
033362 022737 000007 033210
033370 001004
033372 104401 001176

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



```

033376 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
033400 000451          BR       5$        ;;EXIT
033402 021627 000023  3$:    CMP      (SP),#23    ;;IS IT A CONTROL-S?
033406 001021          BNE      32$        ;;BRANCH IF NO
033410 005077 145544          CLR      @STKS     ;;DISABLE TTY KEYBOARD INTERRUPTS
033414 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
033416 105777 145536  31$:    TSTB     @STKS     ;;WAIT FOR A CHAR
033422 100375          BPL      31$        ;;LOOP UNTIL ITS THERE
033424 117746 145532          MOVB     @STKB,-(SP) ;;GET THE CHARACTER
033430 042716 177600          BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
033434 022627 000021          CMP      (SP)+,#21  ;;IS IT A CONTROL-Q?
033440 001366          BNE      31$        ;;BRANCH IF NO
033442 012777 000100 145510  MOV      #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
033450 000002          RTI              ;;RETURN
033452 005237 033210  32$:    INC      $TKCNT    ;;COUNT THIS CHARACTER
033456 021627 000140          CMP      (SP),#140 ;;IS IT UPPER CASE?
033462 002405          BLT      4$        ;;BRANCH IF YES
033464 021627 000175          CMP      (SP),#175 ;;IS IT A SPECIAL CHAR?
033470 003002          BGT      4$        ;;BRANCH IF YES
033472 042716 0C0040          BIC      #40,(SP)  ;;MAKE IT UPPER CASE
033476 112677 177510  4$:    MOVB     (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
033502 005237 033212          INC      $TKQIN    ;;UPDATE THE POINTER
033506 023727 033212 033225  CMP      $TKQIN,$STKQEND ;;GO OFF THE END?
033514 001003          BNE      5$        ;;BRANCH IF NO
033516 012737 033216 033212  MOV      #$TKQSR,$STKQIN ;;RESET THE POINTER
033524 000002          RTI              ;;RETURN

```

```

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

```

```

033526 022737 000176 001154  $CKSWR: CMP      #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED
033534 001124          BNE      15$        ;;EXIT IF NOT
033536 105777 145416          TSTB     @STKS     ;;IS A CHAR WAITING?
033542 100121          BPL      15$        ;;IF NOT, EXIT
033544 117746 145412          MOVB     @STKB,-(SP) ;;YES
033550 042716 177600          BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
033554 021627 000007          CMP      (SP),#7   ;;IS IT A CONTROL-G?
033560 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.

```

```

033562 123727 001150 000001  6$:    CMPSB   $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
033570 001674          BEQ      2$        ;;BRANCH IF YES
033572 005726          TST      (SP)+      ;;CLEAR CONTROL-G OFF STACK
033574 004737 033226          JSR      PC,$TKINT  ;;FLUSH THE TTY INPUT QUEUE
033600 005077 145354          CLR      @STKS     ;;DISABLE TTY KEYBOARD INTERRUPTS
033604 112737 000001 001151  MOVB     #1,$INTAG  ;;SET INTERRUPT MODE INDICATOR

033612 104401 034445          TYPE     .$CNTLG   ;;ECHO THE CONTROL-G (^G)
033616 104401 034452  $GTSWR: TYPE     .$MSWR   ;;TYPE CURRENT CONTENTS
033622 013746 000176          MOV      SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
033626 104402          TYOC     ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```

033630	104401	034463		TYPE	,SMNEW	::PROMPT FOR NEW SWR
033634	005046		19\$:	CLR	-(SP)	::CLEAR COUNTER
033636	005046			CLR	-(SP)	::THE NEW SWR
033640	105777	145314	7\$:	TSTB	@\$TKS	::CHAR iHERE?
033644	100375			BPL	7\$::IF NOT TRY AGAIN
033646	117746	145310		MOVB	@\$TKB, -(SP)	::PICK UP CHAR
033652	042716	177600		BIC	#^C177, (SP)	::MAKE IT 7-BIT ASCII
033656	021627	000003		CMF	(S?), #3	::IS IT A CONTROL-C?
033662	001015			BNE	9\$::BRANCH IF NOT
033664	104401	034433		TYPE	, \$CNTLC	::YES, ECHO CONTROL-C (^C)
033670	062706	000006		ADD	#6, SP	::CLEAN UP STACK
033674	123727	001151	000001	CMFB	\$INTAG, #1	::REENABLE TTY KEYBOARD INTERRUPTS?
033702	001003			BNE	8\$::BRANCH IF NO
033704	012777	000100	145246	MOV	#100, @\$TKS	::ALLOW TTY KEYBOARD INTERRUPTS
033712	000137	034474	8\$:	JMP	CTRAP	::CONTROL-C RESTART
033716	021627	000025	9\$:	CMF	(SP), #25	::IS IT A CONTROL-U?
033722	001005			BNE	10\$::BRANCH IF NOT
033724	104401	034440		TYPE	, \$CNTLU	::YES, ECHO CONTROL-U (^U)
033730	062706	000006	20\$:	ADD	#6, SP	::IGNORE PREVIOUS INPUT
033734	000737			BR	19\$::LET'S TRY IT AGAIN
033736	021627	000015	10\$:	CMF	(SP), #15	::IS IT A <CR>?
033742	001022			BNE	16\$::BRANCH IF NO
033744	005766	000004		TST	4(SP)	::YES, IS IT THE FIRST CHAR?
033750	001403			BEQ	11\$::BRANCH IF YES
033752	016677	000007	145174	MOV	2(SP), @SWR	::SAVE NEW SWR
033760	062706	000006	11\$:	ADD	#6, SP	::CLEAR UP STACK
033764	104401	001203	14\$:	TYPE	, \$CRLF	::ECHO <CR> AND <LF>
033770	123727	001151	000001	CMFB	\$INTAG, #1	::RE-ENABLE TTY KBD INTERRUPTS?
033776	001003			BNE	15\$::BRANCH IF NOT
034000	012777	000100	145152	MOV	#100, @\$TKS	::RE-ENABLE TTY KBD INTERRUPTS
034006	000002		15\$:	RTI		::RETURN
034010	004737	035662	16\$:	JSR	PC, \$TYPEC	::ECHO CHAR
034014	021627	000060		CMF	(SP), #60	::CHAR < 0?
034020	002420			BLT	18\$::BRANCH IF YES
034022	021627	000067		CMF	(SP), #67	::CHAR > 7?
034026	003015			BGT	18\$::BRANCH IF YES
034030	042726	000060		BIC	#60, (SP)+	::STRIP-OFF ASCII
034034	005766	000002		TST	2(SP)	::IS THIS THE FIRST CHAR
034040	001403			BEQ	17\$::BRANCH IF YES
034042	006316			ASL	(SP)	::NO, SHIFT PRESENT
034044	006316			ASL	(SP)	:: CHAR OVER TO MAKE
034046	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
034050	005266	000002	17\$:	INC	2(SP)	::KEEP COUNT OF CHAR
034054	056616	177776		BIS	-2(SP), (SP)	::SET IN NEW CHAR
034060	000667			BR	7\$::GET THE NEXT ONE
034062	104401	001202	18\$:	TYPE	, \$QUES	::TYPE ?<CR><LF>
034066	000720			BR	20\$::SIMULATE CONTROL-U
				.DSABL	LSB	

::*****

```

: *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: *CALL:
: *      RDCHR          ;; GET A CHARACTER FROM THE QUEUE
: *      RETURN HERE   ;; CHARACTER IS ON THE STACK
: *                   ;; WITH PARITY BIT STRIPPED OFF
:
034070 011646          SRDCHR: MOV      (SP),-(SP)      ;; PUSH DOWN THE PC AND
034072 016666 000004 000002 MOV      4(SP),2(SP)      ;; THE PS
034100 005066 000004          CLR      4(SP)          ;; GET READY FOR A CHARACTER
034104 005046          CLR      -(SP)          ;; PUT NEW PS ON STACK
034106 012746 034114          MOV      #64$,-(SP)      ;; PUT NEW PC ON STACK
034112 000002          RTI          ;; POP NEW PC AND PS
034114
034114 005737 033210          64$:  TST      $TKCNT          ;; WAIT ON A CHARACTER
034120 001775          1$:  BEQ      1$
034122 005337 033210          DEC      $TKCNT          ;; DECREMENT THE COUNTER
034126 117766 177062 000004 MOVVB   @$TKQOUT,4(SP)    ;; GET ONE CHARACTER
034134 005237 033214          INC      $TKQOUT          ;; UPDATE THE POINTER
034140 023727 033214 033225 CMP      $TKQOUT,#$TKQEND ;; DID IT GO OFF OF THE END?
034146 001003          BNE      2$          ;; BRANCH IF NO
034150 012737 033216 033214 MOV      #$TKQSPT,$TKQOUT ;; RESET THE POINTER
034156 000002          2$:  RTI          ;; RETURN
: *****
: *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
: *CALL:
: *      RDLIN         ;; INPUT A STRING FROM THE TTY
: *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
: *                   ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
:
034160 010346          SRDLIN: MOV      R3,-(SP)      ;; SAVE R3
034162 005046          CLR      -(SP)          ;; CLEAR THE RUBOUT KEY
034164 012703 034414          1$:  MOV      #$TTYIN,R3      ;; GET ADDRESS
034170 022703 034433          2$:  CMP      #$TTYIN+15.,R3    ;; BUFFER FULL?
034174 101456          BLOS     4$          ;; BR IF YES
034176 104410          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
034200 112613          MOVVB   (SP)+,(R3)      ;; GET CHARACTER
034202 122713 000177          10$: CMPB    #177,(R3)      ;; IS IT A RUBOUT
034206 001022          BNE      5$          ;; BR IF NO
034210 005716          TST      (SP)          ;; IS THIS THE FIRST RUBOUT?
034212 001007          BNE      6$          ;; BR IF NO
034214 112737 000134 034412 MOVVB   #' \ ,9$      ;; TYPE A BACK SLASH
034222 104401 034412          TYPE   ,9$
034226 012716 177777          MOV      #-1,(SP)      ;; SET THE RUBOUT KEY
034232 005303          6$:  DEC      R3          ;; BACKUP BY ONE
034234 020327 034414          CMP      R3,$$TTYIN    ;; STACK EMPTY?
034240 103434          BLO     4$          ;; BR IF YES
034242 111337 034412          MOVVB   (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
034246 104401 034412          TYPE   ,9$          ;; GO TYPE
034252 000746          BR      2$          ;; GO READ ANOTHER CHAR.
034254 005716          5$:  TST      (SP)          ;; RUBOUT KEY SET?
034256 001406          BEQ     7$          ;; BR IF NO
034260 112737 000134 034412 MOVVB   #' \ ,9$      ;; TYPE A BACK SLASH
034266 104401 034412          TYPE   ,9$
034272 005016          CLR      (SP)          ;; CLEAR THE RUBOUT KEY
034274 122713 000025          7$:  CMPB    #25,(R3)      ;; IS CHARACTER A CTRL U?
034300 001003          BNE     8$          ;; BR IF NO

```

```

034302 104401 034440          TYPE      ,SCNTLU      ;;TYPE A CONTROL 'U'
034306 000726          BR          1$          ;;GO START OVER
034310 122713 000022      8$:      CMPB      #22,(R3)      ;;IS CHARACTER A '^R'?
034314 001011          BNE          3$          ;;BRANCH IF NO
034316 105013          CLRB      (R3)          ;;CLEAR THE CHARACTER
034320 104401 001203      TYPE      ,SCRLF      ;;TYPE A 'CR' & 'LF'
034324 104401 034414      TYPE      ,STTYIN     ;;TYPE THE INPUT STRING
034330 000717          BR          2$          ;;GO PICKUP ANOTHER CHACTER
034332 104401 001202      4$:      TYPE      ,SQUES      ;;TYPE A '?'
034336 000712          BR          1$          ;;CLEAR THE BUFFER AND LOOP
034340 111337 034412      3$:      MOVB      (R3),9$          ;;ECHO THE CHARACTER
034344 104401 034412      TYPE      ,9$
034350 122723 000015      CMPB      #15,(R3)+      ;;CHECK FOR RETURN
034354 001305          BNE          2$          ;;LOOP IF NOT RETURN
034356 105063 177777      CLRB      -1(R3)          ;;CLEAR RETURN (THE 15)
034362 104401 001204      TYPE      ,SLF          ;;TYPE A LINE FEED
034366 005726          TST      (SP)+          ;;CLEAN RUBOUT KEY FROM THE STACK
034370 012603          MOV      (SP)+,R3      ;;RESTORE R3
034372 011646          MOV      (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034374 016666 000004 000002 MOV      4(SP),2(SP)      ;; FIRST ASCII CHARACTER ON IT
034402 012766 034414 000004 MOV      #STTYIN,4(SP)
034410 000002          RTI
034412          000          9$:      .BYTE      0          ;;RETURN
034413          000          .BYTE      0          ;;STORAGE FOR ASCII CHAR. TO TYPE
034414          .BLKB     15.          ;;TERMINATOR
034433          136          103          015 $TTYIN: .BLKB     15.          ;;RESERVE 15. BYTES FOR TTY INPUT
034440          136          125          015 $CNTLC: .ASCIZ  /^C/<15><12>      ;;CONTROL 'C'
034445          136          107          015 $CNTLU: .ASCIZ  /^U/<15><12>      ;;CONTROL 'U'
034452          015          012          123 $CNTLG: .ASCIZ  /^G/<15><12>      ;;CONTROL 'G'
034463          040          040          116 $MSWR: .ASCIZ  <15><12>/SWR = /
                                $MNEW: .ASCIZ  / NEW = /

2
3
4
5 034474 012737 000001 001340 CTRAP: MOV      #1,CFLAG      ;;SET THE 'CONTROL C' FLAG
6 034502 005237 033210          INC      $TKCNT      ;;COUNT THIS CHARACTER
7 034506 112777 000015 176476      MOVB     #15,@$TKQIN    ;;PUT 'RETURN' CHARACTER IN QUEUE
8 034514 005237 033212          INC      $TKQIN      ;;UPDATE THE POINTER
9 034520 023727 033212 033225      CMP      $TKQIN,#$TKQEND ;;GO OFF THE END ?
10 034526 001003          BNE      1$          ;;BR IF YES
11 034530 012737 033216 033212      MOV      #$TKQSRST,$TKQIN ;;RESET THE POINTER
12 034536 000002          1$:      RTI          ;;RETURN

```

;THIS ROUTINE WILL PROCESS THE (^C) CHARACTER

.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 $ERRTYP 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
*CALL
*          ERROR  N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  
```

```

034540 105037 035126 $ERROR: CLR B IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
034544 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
034546 010337 001322 MOV R3,ATIN          ;;SAVE THE ATTENTION REGISTER CONTENTS
034552 010137 001220 MOV R1,DRIVE        ;;DRIVE NUMBER
034556 032777 020000 144370 BIT #SW13,@SWR      ;;INHIBIT PRINTOUTS ?
034564 001004          BNE .+12          ;;BR IF YES
034566 104401 001203 TYPE .$CRLF         ;;CR-LF
034572 104401 001203 TYPE .$CRLF         ;;CR-LF
034576 004737 024750 JSR PC,$TIME        ;;TYPE THE TIME
034602 105237 001117 7$: INCB $ERFLG          ;;SET THE ERROR FLAG
034606 001775          BEQ 7$           ;;DON'T LET THE FLAG GO TO ZERO
034610 013777 001116 144340 MOV $STMM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
034616 032777 002000 144330 BIT #BIT10,@SWR    ;;BELL ON ERROR?
034624 001402          BEQ 1$          ;;NO - SKIP
034626 104401 001176 TYPE .$BELL         ;;RING BELL
034632 005237 001126 1$: INC $ERTTL          ;;COUNT THE NUMBER OF ERRORS
034636 011637 001132 MOV (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
034642 162737 000002 001132 SUB #2,$ERRPC
034650 117737 144256 001130 MOV B,@$ERRPC,$ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
034656 032777 001000 144270 BIT #BIT09,@SWR    ;;SEE IF LOOP ON ERROR IS SET
034664 001060          BNE 1004$       ;;BRANCH AROUND ROUTINE IF SO
034666 122737 000177 001130 CMPB #177,$ITEMB   ;;SEE IF THIS IS THE POWER FAIL CALL
034674 001454          BEQ 1004$       ;;BRANCH AROUND ROUTINE IF IT IS
034676 105737 035126 TSTB IBSAVE        ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
034702 001047          BNE 1003$       ;;BRANCH IF SO
034704 022737 177777 035124 CMP #-1,CPSAVE     ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
034712 001445          BEQ 1004$       ;;BRANCH IF SO
034714 013746 000004 MOV ERRVEC,-(SP)   ;;SAVE CONTENTS OF ERROR VECTOR
034720 012737 034736 000004 MOV #1000$,ERRVEC  ;;SETUP 'TRAP' RETURN ADDRESS
034726 013737 177766 035124 MOV 177766,CPSAVE ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
034734 000406          BR 1001$
034736 012737 177777 035124 1000$: MOV #-1,CPSAVE    ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
034744 012716 034752 MOV #1001$, (SP)   ;;SETUP RETURN ADDRESS
034750 000002          RTI
034752 012637 000004 1001$: MOV (SP)+,ERRVEC    ;;RESTORE CONTENTS OF ERROR VECTOR

034756 022737 177777 035124 1002$: CMP #-1,CPSAVE    ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
034764 001420          BEQ 1004$       ;;BRANCH IF SO
034766 032737 000001 035124 BIT #BIT00,CPSAVE  ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
034774 001414          BEQ 1004$       ;;BRANCH IF OK
034776 042737 000001 177766 BIC #BIT00,177766 ;;CLEAR THE BIT FOUND SET
035004 113737 001130 035126 MOV B,$ITEMB,IPSAVE ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
035012 112737 000177 001130 MOV B,#177,$ITEMB ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
035020 000402          BR 1004$       ;;BRANCH OVER IBSAVE CLEARING
  
```

```

035022 105037 035126      1003$: CLRB   IBSAVE      ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
035026      1004$:
035026 032777 020000 144120      BIT   #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
035034 001004      BNE   20$              ;;SKIP TYPEOUTS
035036 004737 035130      JSR   PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
035042 104401 001203      TYPE  ,$CRLF
035046      20$:
035046 122737 000001 001226      CMPB  #APTENV,$ENV     ;;RUNNING IN APT MODE
035054 001007      BNE   2$              ;;NO SKIP APT ERROR REPORT
035056 113737 001130 035070      MOVB  $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
035064 004737 036474      JSR   PC,$ATY4        ;;REPORT FATAL ERROR TO APT
035070      21$:
035071      .BYTE 0
035071      .BYTE 0
035072 000777      22$: BR   22$          ;;APT ERROR LOOP
035074 105737 035126      2$:  TSTB  IBSAVE      ;;SEE IF IBSAVE IS LOADED
035100 001005      BNE   3$              ;;BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
035102 005777 144046      TST   @SWR            ;;HALT ON ERROR
035106 100002      BPL   3$              ;;SKIP IF CONTINUE
035110 000000      HALT                          ;;HALT ON ERROR!
035112 104407      CKSWR                      ;;TEST FOR CHANGE IN SOFT-SWR
035114      3$:
035114 105737 035126      TSTB  IBSAVE      ;;SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
035120 001230      BNE   7$              ;;BRANCH BACK TO CALL ORIGINAL ERROR
035122 000002      RTI                          ;;RETURN
035124 000000      CPSAVE: .WORD 0        ;;LOCATION TO SAVE CPU ERROR REG CONTENTS
035126 000000      IBSAVE: .WORD 0        ;;LOCATION TO SAVE ITEM BYTE

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

 : *THIS ROUTINE USES THE "ITEM CONTROL BYTE" (\$ITEMB) TO DETERMINE WHICH
 : *ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (\$ERRTB),
 : *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

035130          035130 104401 001203          $ERRTYP:
035134          035134 010046          TYPE      , $CRLF      ;; 'CARRIAGE RETURN' & 'LINE FEED'
035136          035136 005000          MOV       RO, -(SP)   ;; SAVE RO
035140          035140 153700 001130          CLR       RO         ;; PICKUP THE ITEM INDEX
035144          035144 001004          BISB     @($ITEMB,RO
                                BNE      1$      ;; IF ITEM NUMBER IS ZERO, JUST
                                ;; TYPE THE PC OF THE ERROR
035146          035146 013746 001132          MOV       $ERRPC, -(SP) ;; SAVE $ERRPC FOR TYPEOUT
                                ;; ERROR ADDRESS
                                ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
035152          035152 104402          TYP0C    BR          6$      ;; GET OUT
035154          035154 000437          1$:      CMPB     #177,RO     ;; SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
035156          035156 122700 000177          BNE     1000$       ;; BRANCH IF NOT
035162          035162 001006          MOV     $TESTN,PFTSTN ;; GET TEST NUMBER
035164          035164 013737 001212 035446  MOV     #PFECB,RO    ;; MOVE POWER FAIL ERROR CALL TABLE TO RO
035172          035172 012700 035306          BR      1001$       ;; BRANCH TO CALL ERROR
035176          035176 000406          1000$:  DEC     RO         ;; ADJUST THE INDEX SO THAT IT WILL
035200          035200 005300          ASL     RO         ;; WORK FOR THE ERROR TABLE
035202          035202 006300          ASL     RO
035204          035204 006300          ASL     RO
035206          035206 006300          ADD     #$ERRTB,RO   ;; FORM TABLE POINTER
035210          035210 062700 003472          1001$: MOV     (RO)+,2$     ;; PICKUP 'ERROR MESSAGE' POINTER
035214          035214 012037 035224          BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
035220          035220 001404          TYPE   'ERROR MESSAGE' ;; TYPE THE 'ERROR MESSAGE'
035222          035222 104401          .WORD  0           ;; 'ERROR MESSAGE' POINTER GOES HERE
035224          035224 000000          2$:      TYPE   , $CRLF ;; 'CARRIAGE RETURN' & 'LINE FEED'
035226          035226 104401 001203          MOV     (RO)+,4$     ;; PICKUP 'DATA HEADER' POINTER
035232          035232 012037 035242          BEQ     5$          ;; SKIP TYPEOUT IF 0
035236          035236 001404          TYPE   'DATA HEADER'  ;; TYPE THE 'DATA HEADER'
035240          035240 104401          .WORD  0           ;; 'DATA HEADER' POINTER GOES HERE
035242          035242 000000          4$:      TYPE   , $CRLF ;; 'CARRIAGE RETURN' & 'LINE FEED'
035244          035244 104401 001203          MOV     (RO),RO      ;; PICKUP 'DATA TABLE' POINTER
035250          035250 011000          BNE     7$          ;; GO TYPE THE DATA
035252          035252 001004          5$:      MOV     (SP)+,RO     ;; RESTORE RO
035254          035254 012600          6$:      TYPE   , $CRLF ;; 'CARRIAGE RETURN' & 'LINE FEED'
035256          035256 104401 001203          RTS     PC          ;; RETURN
035262          035262 000207          7$:
035264          035264 013046          MOV     @(RO)+, -(SP) ;; SAVE @(RO)+ FOR TYPEOUT
035266          035266 104402          TYP0C    GO TYPE--OCTAL ASCII(ALL DIGITS)
035270          035270 005710          TST     (RO)        ;; IS THERE ANOTHER NUMBER?
035272          035272 001770          BEQ     6$          ;; BR IF NO
035274          035274 104401 035302          TYPE   , 8$        ;; TYPE TWO(2) SPACES
035300          035300 000771          BR      7$          ;; LOOP
035302          035302 040 040 000 8$:      .ASCIZ  / /        ;; TWO(2) SPACES
                                .EVEN
035306          035306 035316 035400 035432  PFECB:  PFECB1,PFECB2,PFECB3,PFECB4 ;; WORDS DEFINING TABLES BELOW
035316          035316 120 117 127  PFECB1:  .ASCIZ  ?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SFT?
035400          035400 124 105 123  PFECB2:  .ASCIZ  ?TESTNO ERR PC CPUERREG?
                                .EVEN
035432          035432 035446 001132 035124  PFECB3:  .WORD   PFTSTN,$ERRPC,CPSAVE,0
  
```

ZRMUBG RM05/3/2 PERF EXER
ERROR MESSAGE TYPEOUT ROUTINE

MACRO V04.00 4-APR-81 01:42:23 PAGE 43-

SEQ 0160

035442 000 000
035446 000000

000 PFECH4: .BYTE 0.0.0.0
PFTSTN: .WORD 0

::CONTAINS TEST NUMBER FOR P1 BIT ERROR

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

*OR
* TYPE
* MESADR

```

```

035450 105737 001173 $TYPE: TSTB $TPFLG ;:IS THERE A TERMINAL?
035454 100002 BPL 1$ ;:BR IF YES
035456 000000 HALT ;:HALT HERE IF NO TERMINAL
035460 000430 BR 3$ ;:LEAVE
035462 010046 1$: MOV RO,-(SP) ;:SAVE RO
035464 017600 000002 MOV @2(SP),RO ;:GET ADDRESS OF ASCIZ STRING
035470 122737 000001 001226 CMPB #APTENV,$ENV ;:RUNNING IN APT MODE
035476 001011 BNE 62$ ;:NO,GO CHECK FOR APT CONSOLE
035500 132737 000100 001227 BITB #APTSPOOL,$ENVM ;:SPOOL MESSAGE TO APT
035506 001405 BEQ 62$ ;:NO,GO CHECK FOR CONSOLE
035510 010037 035520 MOV RO,61$ ;:SETUP MESSAGE ADDRESS FOR APT
035514 004737 036464 JSR PC,$ATY3 ;:SPOOL MESSAGE TO APT
035520 000000 61$: .WORD 0 ;:MESSAGE ADDRESS
035522 132737 000040 001227 62$: BITB #APTCSUP,$ENVM ;:APT CONSOLE SUPPRESSED
035530 001003 BNE 60$ ;:YES,SKIP TYPE OUT
035532 112046 2$: MOVB (RO)+,-(SP) ;:PUSH CHARACTER TO BE TYPED ONTO STACK
035534 001005 BNE 4$ ;:BR IF IT ISN'T THE TERMINATOR
035536 005726 TST (SP)+ ;:IF TERMINATOR POP IT OFF THE STACK
035540 012600 60$: MOV (SP)+,RO ;:RESTORE RO
035542 062716 000002 3$: ADD #2,(SP) ;:ADJUST RETURN PC
035546 000002 RTI ;:RETURN
035550 122716 000011 4$: CMPB #HT,(SP) ;:BRANCH IF <HT>
035554 001430 BEQ 8$ ;:BRANCH IF NOT <CR>
035556 122716 000200 CMPB #CRLF,(SP) ;:BRANCH IF NOT <CRLF>
035562 001006 BNE 5$ ;:POP <CR><LF> EQUIV
035564 005726 TST (SP)+ ;:TYPE A CR AND LF
035566 104401 TYPE ;:TYPE A CR AND LF
035570 001203 $CRLF ;:TYPE A CR AND LF
035572 105037 036000 CLRB $CHARCNT ;:CLEAR CHARACTER COUNT
035576 000755 BR 2$ ;:GET NEXT CHARACTER
035600 004737 035662 5$: JSR PC,$TYPEC ;:GO TYPE THIS CHARACTER
035604 123726 001172 6$: CMPB $FILLC,(SP)+ ;:IS IT TIME FOR FILLER CHARS.?
035610 001350 BNE 2$ ;:IF NO GO GET NEXT CHAR.
035612 013746 001170 MOV $NULL,-(SP) ;:GET # OF FILLER CHARS. NEEDED
;:AND THE NULL CHAR.
035616 105366 000001 7$: DECB 1(SP) ;:DOES A NULL NEED TO BE TYPED?
035622 002770 BLT 6$ ;:BR IF NO--GO POP THE NULL OFF OF STACK
035624 004737 035662 JSR PC,$TYPEC ;:GO TYPE A NULL
035630 105337 036000 DECB $CHARCNT ;:DO NOT COUNT AS A COUNT
035634 000770 BR 7$ ;:LOOP

```

;HORIZONTAL TAB PROCESSOR

```

035636 112716 000040      8$:  MOVB  #' , (SP)      ;; REPLACE TAB WITH SPACE
035642 004737 035662      9$:  JSR   PC, $TYPEC      ;; TYPE A SPACE
035646 132737 000007 036000  BITB  #7, $CHARCNT      ;; BRANCH IF NOT AT
035654 001372          BNE   9$                ;; TAB STOP
035656 005726          TST   (SP)+          ;; POP SPACE OFF STACK
035660 000724          BR    2$                ;; GET NEXT CHARACTER
035662          $TYPEC:
035662 105777 143272      TSTB  @ $TKS          ;; CHAR IN KYBD BUFFER?
035666 100022          BPL   10$          ;; BR IF NOT
035670 017746 143266      MOV   @ $TKB, -(SP)      ;; GET CHAR
035674 042716 177600      BIC   #177600, (SP)     ;; STRIP EXTRANEIOUS BITS
035700 122716 000023      CMPB  # $XOFF, (SP)     ;; WAS CHAR XOFF
035704 001012          BNE   102$         ;; BR IF NOT
035706          101$:
035706 105777 143246      TSTB  @ $TKS          ;; WAIT FOR CHAR
035712 100375          BPL   101$          ;; BR IF NOT
035714 117716 143242      MOVB  @ $TKB, (SP)      ;; GET CHAR
035720 042716 177600      BIC   #177600, (SP)     ;; STRIP IT
035724 122716 000021      CMPB  # $XON, (SP)     ;; WAS IT XON?
035730 001366          BNE   101$          ;; BR IF NOT
035732          102$:
035732 005726          TST   (SP)+          ;; FIX STACK
035734          10$:
035734 105777 143224      TSTB  @ $TPS          ;; WAIT UNTIL PRINTER IS READY
035740 100375          BPL   10$                ;; BR IF NOT
035742 116677 000002 143216  MOVB  2(SP), @ $TPB     ;; LOAD CHAR TO BE TYPED INTO DATA REG.
035750 122766 000015 000002  CMPB  #CR, 2(SP)        ;; IS CHARACTER A CARRIAGE RETURN?
035756 001003          BNE   1$                ;; BRANCH IF NO
035760 105037 036000      CLRB  $CHARCNT        ;; YES--CLEAR CHARACTER COUNT
035764 000406          BR    $TYPEX          ;; EXIT
035766 122766 000012 000002 1$:  CMPB  #LF, 2(SP)        ;; IS CHARACTER A LINE FEED?
035774 001402          BEQ   $TYPEX          ;; BRANCH IF YES
035776 105227          INCB  (PL)+          ;; COUNT THE CHARACTER
036000 000000      $CHARCNT: .WORD 0    ;; CHARACTER COUNT STORAGE
036002 000207      $TYPEX. RTS  PL

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---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
*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
TYPON   ;;CALL FOR TYPEOUT
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
TYPOC   ;;CALL FOR TYPEOUT

```

```

036004 017646 000000          STYPOS: MOV     @ (SP),-(SP)      ;;PICKUP THE MODE
036010 116637 000001 036227  MOVVB  1(SP),%OFILL    ;;LOAD ZERO FILL SWITCH
036016 112637 036231          MOVVB  (SP)+,%SOMODE+1  ;;NUMBER OF DIGITS TO TYPE
036022 062716 000002          ADD     #2,(SP)        ;;ADJUST RETURN ADDRESS
036026 000406                    BR     $TYPON
036030 112737 000001 036227  STYPOC: MOVVB  #1,%OFILL    ;;SET THE ZERO FILL SWITCH
036036 112737 000006 036231  MOVVB  #6,%SOMODE+1    ;;SET FOR SIX(6) DIGITS
036044 112737 000005 036226  STYPON: MOVVB  #5,%SOCNT  ;;SET THE ITERATION COUNT
036052 010346                    MOV     R3,-(SP)        ;;SAVE R3
036054 010446                    MOV     R4,-(SP)        ;;SAVE R4
036056 010546                    MOV     R5,-(SP)        ;;SAVE R5
036060 113704 036231          MOVVB  %SOMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
036064 005404                    NEG     R4
036066 062704 000006          ADD     #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
036072 110437 036230          MOVVB  R4,%SOMODE    ;;SAVE IT FOR USE
036076 113704 036227          MOVVB  %OFILL,R4      ;;GET THE ZERO FILL SWITCH
036102 016605 000012          MOV     12(SP),R5     ;;PICKUP THE INPUT NUMBER
036106 005003                    CLR     R3             ;;CLEAR THE OUTPUT WORD
036110 006105                    1$:   ROL     R5          ;;ROTATE MSB INTO 'C'
036112 000404                    BR     3$            ;;GO DO MSB
036114 006105                    2$:   ROL     R5          ;;FORM THIS DIGIT
036116 006105                    ROL     R5
036120 006105                    ROL     R5
036122 010503                    MOV     R5,R3
036124 006103                    3$:   ROL     R3          ;;GET LSB OF THIS DIGIT
036126 105337 036230          DECB  %SOMODE        ;;TYPE THIS DIGIT?
036132 100016                    BPL     7$            ;;BR IF NO
036134 042703 177770          BIC  #177770,R3     ;;GET RID OF JUNK
036140 001002                    BNE     4$            ;;TEST FOR 0
036142 005704                    TST     R4            ;;SUPPRESS THIS 0?
036144 001403                    BEQ     5$            ;;BR IF YES
036146 005204                    4$:   INC     R4          ;;DON'T SUPPRESS ANYMORE 0'S

```

```

036150 052703 000060          BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
036154 052703 000040          BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
036160 110337 036224          MOVB     R3,R5      ;;SAVE FOR TYPING
036164 104401 036224          TYPE    .R5      ;;GO TYPE THIS DIGIT
036170 105337 036226          DECB   $OCNT      ;;COUNT BY 1
036174 003347          BGT     2$        ;;BR IF MORE TO DO
036176 002402          BLT    6$        ;;BR IF DONE
036200 005204          INC    R4        ;;INSURE LAST DIGIT ISN'T A BLANK
036202 000744          BR     2$        ;;GO DO THE LAST DIGIT
036204 012605          6$:     MOV     (SP)+,R5  ;;RESTORE R5
036206 012604          MOV     (SP)+,R4  ;;RESTORE R4
036210 012603          MOV     (SP)+,R3  ;;RESTORE R3
036212 016666 000002 000004          MOV     2(SP),4(SP) ;;SET THE STACK FOR RETURNING
036220 012616          MOV     (SP)+,(SP)
036222 000002          RTI          ;;RETURN
036224      000          8$:     .BYTE 0      ;;STORAGE FOR ASCII DIGIT
036225      000          .BYTE 0      ;;TERMINATOR FOR TYPE ROUTINE
036226      000          $OCNT: .BYTE 0  ;;OCTAL DIGIT COUNTER
036227      000          $ZFILL: .BYTE 0 ;;ZERO FILL SWITCH
036230 000000          $OMODE: .WORD 0  ;;NUMBER OF DIGITS TO TYPE
    
```

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

036232			\$TYPDS:	MOV	R0,-(SP)	;;PUSH R0 ON STACK
036232	010046			MOV	R1,-(SP)	;;PUSH R1 ON STACK
036234	010146			MOV	R2,-(SP)	;;PUSH R2 ON STACK
036236	010246			MOV	R3,-(SP)	;;PUSH R3 ON STACK
036240	010346			MOV	R5,-(SP)	;;PUSH R5 ON STACK
036242	010546			MOV	#20200,-(SP)	;;SET BLANK SWITCH AND SIGN
036244	012746	020200		MOV	20(SP),R5	;;GET THE INPUT NUMBER
036250	016605	000020		BPL	1\$;;BR IF INPUT IS POS.
036254	100004			NEG	R5	;;MAKE THE BINARY NUMBER POS.
036256	005405			MOV	#'-,1(SP)	;;MAKE THE ASCII NUMBER NEG.
036260	112766	000055	000001	1\$: CLR	R0	;;ZERO THE CONSTANTS INDEX
036266	005000			MOV	#SDBLK,R3	;;SETUP THE OUTPUT POINTER
036270	012703	036446		MOV	#',(R3)+	;;SET THE FIRST CHARACTER TO A BLANK
036274	112723	000040		2\$: CLR	R2	;;CLEAR THE BCD NUMBER
036300	005002			MOV	\$DTBL(R0),R1	;;GET THE CONSTANT
036302	016001	036436		3\$: SUB	R1,R5	;;FORM THIS BCD DIGIT
036306	160105			BLT	4\$;;BR IF DONE
036310	002402			INC	R2	;;INCREASE THE BCD DIGIT BY 1
036312	005202			BR	3\$	
036314	000774			ADD	R1,R5	;;ADD BACK THE CONSTANT
036316	060105			4\$: TST	R2	;;CHECK IF BCD DIGIT=0
036320	005702			BNE	5\$;;FALL THROUGH IF 0
036322	001002			TSTB	(SP)	;;STILL DOING LEADING 0'S?
036324	105716			BMI	7\$;;BR IF YES
036326	100407			5\$: ASLB	(SP)	;;MSD?
036330	105316			BCC	6\$;;BR IF NO
036332	103003			MOV	1(SP),-1(R3)	;;YES--SET THE SIGN
036334	116663	000001	177777	6\$: BIS	#'0,R2	;;MAKE THE BCD DIGIT ASCII
036342	052702	000060		7\$: BIS	#',R2	;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
036346	052702	000040		MOV	R2,(R3)+	;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
036352	110223			TST	(R0)+	;;JUST INCREMENTING
036354	005720			CMR	R0,#10	;;CHECK THE TABLE INDEX
036356	020027	000010		BLT	2\$;;GO DO THE NEXT DIGIT
036362	002746			BGT	8\$;;GO TO EXIT
036364	003002			MOV	R5,R2	;;GET THE LSD
036366	010502			BR	6\$;;GO CHANGE TO ASCII
036370	000764			8\$: TSTB	(SP)+	;;WAS THE LSD THE FIRST NON-ZERO?
036372	105726			BPL	9\$;;BR IF NO
036374	100003			MOV	-1(SP),-2(R3)	;;YES--SET THE SIGN FOR TYPING
036376	116663	177777	177776	9\$: CLRB	(R3)	;;SET THE TERMINATOR
036404	105013			MOV	(SP)+,R5	;;POP STACK INTO R5
036406	012605			MOV	(SP)+,R3	;;POP STACK INTO R3
036410	012603			MOV	(SP)+,R2	;;POP STACK INTO R2
036412	012602			MOV	(SP)+,R1	;;POP STACK INTO R1
036414	012601					

```

036416 012600          MOV      (SP)+,R0      ;;POP STACK INTO R0
036420 104401  C36446  TYPE      $DBLK      ;;NOW TYPE THE NUMBER
036424 016666  G00002 000004  MOV      2(SP),4(SP)   ;;ADJUST THE STACK
036432 012616          MOV      (SP)+,(SP)
036434 000002          RTI          ;;RETURN TO USER
036436 023420          $DTBL: 10000.
036440 001750          1000.
036442 000144          100.
036444 000012          10.
036446          $DBLK: .BLKW 4

```

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
036456 112737 000001 036722 $ATY1:  MOV  #1,$FFLG      ;;TO REPORT FATAL ERROP
036464 112737 000001 036720 $ATY3:  MOV  #1,$MFLG      ;;TO TYPE A MESSAGE
036472 000403                BR      $ATYC
036474 112737 000001 036722 $ATY4:  MOV  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
036502                $ATYC:
036502 010046                MOV  R0,-(SP)      ;;PUSH R0 ON STACK
036504 010146                MOV  R1,-(SP)      ;;PUSH R1 ON STACK
036506 105737 036720                TSTB $MFLG      ;;SHOULD TYPE A MESSAGE?
036512 001450                BEQ  5$          ;;IF NOT: BR
036514 122737 000001 001226        CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
036522 001031                BNE  3$          ;;IF NOT: BR
036524 132737 000100 001227        BITB #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
036532 001425                BEQ  3$          ;;IF NOT: BR
036534 017600 000004                MOV  @4(SP),R0    ;;GET MESSAGE ADDR.
036540 062766 000002 000004        ADD  #2,4(SP)    ;;BUMP RETURN ADDR.
036546 005737 001206        1$:  TST  $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
036552 001375                BNE  1$          ;;IF NOT: WAIT
036554 010037 001222                MOV  R0,$MSGAD   ;;PUT ADDR IN MAILBOX
036560 105720                2$:  TSTB (R0)+     ;;FIND END OF MESSAGE
036562 001376                BNE  2$
036564 163700 001222                SUB  $MSGAD,R0   ;;SUB START OF MESSAGE
036570 006200                ASR  R0          ;;GET MESSAGE LNTH IN WORDS
036572 010037 001224                MOV  R0,$MSGGLT  ;;PUT LENGTH IN MAILBOX
036576 012737 000004 001206        MOV  #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
036604 000413                BR   5$
036606 017637 000004 036632        3$:  MOV  @4(SP),4$   ;;PUT MSG ADDR IN JSR LINKAGE
036614 062766 000002 000004        ADD  #2,4(SP)    ;;BUMP RETURN ADDRESS
036622 013746 177776                MOV  177776,-(SP) ;;PUSH 177776 ON STACK
036626 004737 035450                JSR  PC,$TYPE   ;;CALL TYPE MACRO
036632 000000                4$:  .WORD 0
036634                5$:
036634 105737 036722                10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROP?
036640 001416                BEQ  12$        ;;IF NOT: BR
036642 005737 001226                TST  $ENV       ;;RUNNING UNDER APT?
036646 001413                BEQ  12$        ;;IF NOT: BR
036650 005737 001206                11$: TST  $MSGTYPE   ;;FINISHED LAST MESSAGE?
036654 001375                BNE  11$       ;;IF NOT: WAIT
036656 017637 000004 001210        MOV  @4(SP),$FATAL ;;GET ERROR #
036664 062766 000002 000004        ADD  #2,4(SP)    ;;BUMP RETURN ADDR.
036672 005237 001206                INC  $MSGTYPE   ;;TELL APT TO TAKE ERROR
036676 105037 036722                12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
036702 105037 036721                CLRB $LFLG      ;;CLEAR LOG FLAG
036706 105037 036720                CLRB $MFLG      ;;CLEAR MESSAGE FLAG
036712 012601                MOV  (SP)+,R1   ;;POP STACK INTO R1
036714 012600                MOV  (SP)+,R0   ;;POP STACK INTO R0
036716 000207                RTS  PC         ;;RETURN
036720 000                $MFLG: .BYTE 0   ;;MESSG. FLAG
036721 000                $LFLG: .BYTE 0   ;;LOG FLAG
036722 000                $FFLG: .BYTE 0   ;;FATAL FLAG
                .EVEN
000200                APTSIZE = 200
000001                APTENV  = 001
000100                APTSPOOL= 100
000040                APTCSUP = 040
  
```

.SBTTL RANDOM NUMBER GENERATOR ROUTINE

*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
*WITH A RANGE OF 0 TO 2⁽⁺³³⁾-1.

*CALL:
* JSR PC,\$RAND ;;CALL THE ROUTINE
* RETURN ;;RETURN HERE THE RANDOM
* ;;NUMBER WILL BE IN
* ;;\$HINUM,\$LONUM

036724
036724 010046
036726 010146
036730 010246
036732 013700 037024
036736 013701 037022
036742 012702 177771
036746 006300
036750 006101
036752 005202
036754 001374
036756 063700 037024
036762 005501
036764 063701 037022
036770 062700 001057
036774 005501
036776 062701 047401
037002 010037 037024
037006 010137 037022
037012 012602
037014 012601
037016 012600
037020 000207
037022 176543
037024 123456

\$RAND:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV \$LONUM,R0 ;;SET R0 WITH LOW
MOV \$HINUM,R1 ;;SET R1 WITH HIGH
MOV #-7,R2 ;;SET SHIFT COUNT
1\$: ASL R0 ;;SHIFT R0 LEFT AND
ROL R1 ;;ROTATE CARRY INTO R1 AND
INC R2 ;;CHECK FOR DONE
BNE 1\$;;CONTINUE SHIFT LOOP
ADD \$LONUM,R0 ;;ADD NUMBER MAKE X 129
ADC R1 ;;PROPAGATE CARRY
ADD \$HINUM,R1 ;;ADD NUMBER MAKE X 129
ADD #1057,R0 ;;ADD LOW CONSTANT
ADC R1 ;;PROPAGATE CARRY
ADD #47401,R1 ;;ADD HIGH CONSTANT
MOV R0,\$LONUM ;;SAVE R0
MOV R1,\$HINUM ;;SAVE R1
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTS PC ;;RETURN
\$HINUM: .WORD 176543
\$LONUM: .WORD 123456

.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
    
```

```

037026
037026 010046
037030 010146
037032 010246
037034 010346
037036 010446
037040 010546
037042 016646 000022
037046 016646 000022
037052 016646 000022
037056 016646 000022
037062 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) ;;PU H R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI
    
```

```

*RESTORE R0-R5
*CALL:
*   RESREG
$RESREG:
    
```

```

037064
037064 012666 000022
037070 012666 000022
037074 012666 000022
037100 012666 000022
037104 012605
037106 012604
037110 012603
037112 012602
037114 012601
037116 012600
037120 000002
    
```

```

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

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

*****
*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
*POSITIVE.

```

```

*CALL
*   MOV   #PNTR,-(SP)   ;; POINTER TO LOW WORD OF BINARY NUMBER
*   JSR   PC,@#$DB2D   ;; THE FIRST ADDRESS OF ASCII
*   RETURN              ;; IS ON THE STACK

```

```

037122 104412          $DB2D: SAVREG          ;; SAVE REGISTERS
037124 016602 000002 MOV          2(SP),R2          ;; PICKUP THE DATA POINTER
037130 012700 037302 MOV          #$DECVL,R0        ;; GET ADDRESS OF '$DECVL' STRING
037134 010066 000002 MOV          R0,2(SP)          ;; PUT ADDRESS OF ASCII STRING ON STACK
037140 012201          MOV          (R2)+,R1        ;; PICKUP THE BINARY NUMBER
037142 012202          MOV          (R2)+,R2
037144 012737 0C0012 037220 MOV          #10,4$          ;; SET UP TO DO 10 CONVERSIONS
037152 012704 037232 MOV          #$TNPWR,R4        ;; ADDRESS OF TEN POWER
037156 012705 037234 MOV          #$TNPWR+2,R5
037162 005003          1$: CLR          R3          ;; CLEAR PARTIAL
037164 161401          2$: SUB          (R4),R1        ;; SUBTRACT TEN POWER
037166 005602          SBC          R2
037170 161502          SUB          (R5),R2
037172 002402          BLT          3$          ;; BR IF TEN POWER TOO LARGE
037174 005203          INC          R3          ;; ADD 1 TO PARTIAL
037176 000772          BR          2$          ;; LOOP
037200 062401          3$: ADD          (R4)+,R1        ;; RESTORE SUBTRACTED VALUE
037202 005502          ADC          R2
037204 062402          ADD          (R4)+,R2
037206 022525          CMP          (R5)+,(R5)+      ;; MOVE TO NEXT TEN POWER
037210 052703 000060 BIS          #0,R3          ;; CHANGE PARTIAL TO ASCII
037214 110320          MOVB         R3,(R0)+        ;; SAVE IT
037216 005327          DEC          (PC)+          ;; DONE?
037220 000000          4$: .WORD          0
037222 001357          BNE          1$          ;; BR IF NO
037224 105020          CLR          (R0)+          ;; TERMINATOR
037226 104413          RESREG          ;; RESTORE REGISTERS
037230 000207          RTS          PC          ;; RETURN
037232 145000          $TNPWR: 145000          ;; 1.0E09
037234 035632          35632
037236 160400          160400          ;; 1.0E08
037240 002765          2765
037242 113200          113200          ;; 1.0E07
037244 000230          230
037246 041100          041100          ;; 1.0E06
037250 000017          17
037252 103240          103240          ;; 1.0E05
037254 000001          1
037256 023420          23420          ;; 1.0E04
037260 000000          0
037262 001750          1750          ;; 1.0E03
037264 000000          0
037266 000144          144          ;; 1.0E02
037270 000000          0

```

037272 000012
037274 000000
037276 000001
037300 000000
037302

12
0
1
0
\$DECVL: .BLKB 12.

:::1.0E01
:::1.0E00
:::RESERVE STORAGE FOR ASCII STRING

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

*THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED OCTAL ASCII NUMBER.
*CALL

* MOV #PNTR,-(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
* JSR PC,@#SDB20 ;; CALL THE ROUTINE
* RETURN ;; THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

037316	104412		\$DB20:	SAVREG		;; SAVE ALL REGISTERS
037320	016601	000002		MOV	2(SP),R1	;; PICKUP THE POINTER TO LOW WORD
037324	012705	037455		MOV	#SOCTVL+13.,R5	;; POINTER TO DATA TABLE
037330	012704	000014		MOV	#12.,R4	;; DO ELEVEN CHARACTERS
037334	012703	177770		MOV	#^C7,R3	;; MASK
037340	012100			MOV	(R1)+,R0	;; LOWER WORD
037342	012101			MOV	(R1)+,R1	;; HIGH WORD
037344	005002			CLR	R2	;; TERMINATOR
037346	1 0245		1\$:	MOV	R2,-(R5)	;; PUT CHARACTER IN DATA TABLE
037350	010002			MOV	R0,R2	;; GET THIS DIGIT
037352	005304			DEC	R4	;; COUNT THIS CHARACTER
037354	003007			BGT	3\$;; BR IF NOT THE LAST DIGIT
037356	001405			BEG	2\$;; BR IF IT IS THE LAST DIGIT
037360	005205			INC	R5	;; ALL DIGITS DONE-ADJUST POINTER FOR FIRST
037362	010566	000002		MOV	R5,2(SP)	;; ASCII CHAR. & PUT IT ON THE STACK
037366	104413			RESREG		;; RESTORE ALL REGISTERS
037370	000207			RTS	PC	;; RETURN TO USER
037372	006203		2\$:	ASR	R3	;; POSITION THE MASK FOR THE LAST DIGIT
037374	006001		3\$:	ROR	R1	;; POSITION THE BINARY NUMBER FOR
037376	006000			ROR	R0	;; THE NEXT OCTAL DIGIT
037400	006001			ROR	R1	
037402	006000			ROR	R0	
037404	006001			ROR	R1	
037406	006000			ROR	R0	
037410	040302			BIC	R3,R2	;; MASK OUT ALL JUNK
037412	062702	000060		ADD	#'0,R2	;; MAKE THIS CHAR. ASCII
037416	000753			BR	1\$;; GO PUT IT IN THE DATA TABLE
037420			\$OCTVL:	.BLKB	14.	;; RESERVE DATA TABLE

.SBTIL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
037436 012737 037610 000024 $PWRDN: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST UP
037444 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
037452 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
037454 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
037456 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
037460 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
037462 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
037464 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
037466 017746 141462 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
037472 010637 037614 MOV SP,$SAVR6 ;;SAVE SP
037476 012737 037510 000024 MOV # $PWRUP,@#PWRVEC ;;SET UP VECTOR
037504 000000 HALT
037506 000776 BR -2 ;;HANG UP
*****
:POWER UP ROUTINE
037510 012737 037610 000024 $PWRUP: MOV # $ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
037516 013706 037614 MOV $SAVR6,SP ;;GET SP
037522 005037 037614 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
037526 005237 037614 1$: INC $SAVR6 ;;WAIT FOR THE INC
037532 001375 BNE 1$ ;;OF WORD
037534 005337 037616 2$: DEC PWRFLG ;;WAIT AND SET POWER FAIL FLAG
037540 003775 BLE 2$ ;;WAIT FOR FLAG= 1
037542 012677 141406 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
037546 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
037550 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
037552 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
037554 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
037556 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
037560 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
037562 012737 037436 000024 MOV # $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
037570 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
037576 104401 TYPE ;;REPORT THE POWER FAILURE
037600 037620 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
037602 012716 MOV (PC)+,(SP) ;;RESTART AT SATPOW
037604 037636 $PWRAD: .WORD SATPOW ;;RESTART ADDRESS
037606 000002 RTI
037610 000000 $ILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
037612 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
037614 000000 $SAVR6: 0 ;;PUT THE SP HERE
2 037616 000000 PWRFLG: .WORD 0 ;;INDICATES POWER FAIL OCCURRED WHEN SET
3 037620 200 042 120 $POWER: .ASCII <CRLF>/'POWER UP'/<CRLF>
4
5
6
7
8
9 037636 005227 000000 SATPOW: INC #0 ;;TTY LOOP, WAIT FOR INCREMENT
10 037642 001375 BNE -4 ;;OF WORD
11 037644 000005 RESET ;;CLEAR THE WORLD
12 037646 005037 001346 CLR MINUTE ;;RESET MINUTE COUNT
13 037652 005037 001350 CLR SECOND ;;RESET SECOND COUNT
14 037656 005037 001454 CLR INTRVL+2 ;;RESET THE INTERVAL COUNT
:POWER UP ROUTINE ,WAIT TWO MINUTES,
:THEN AUTO STARTS AT SIZMEM

```

ZRMJBU RM05/3/2 PERFORM EXERCISE
POWER DOWN AND UP ROUTINES

MACRO V04.00 4-APR-81 01:42:23 PAGE 52-1

15	037662	005037	177776		CLR	@#PS	:CLEAR PSW
16	037666	004737	023344		JSR	PC,CKCLK	:START THE CLOCK
17	037672	022737	000002	001454	CMP	#2,INTRVL+2	:TWO MINUTES YET ?
18	037700	101374			BHI	1\$:WAIT IF NOT
19	037702	012737	000400	001336	MOV	#400,CHGADR	:FUDGE THE AUTO START
20	037710	012705	001506		MOV	#ORDERQ,R5	:CLEAR UP THE QUE AND BUFFER
21	037714	005025		2\$:	CLR	(R5)+	
22	037716	022705	002044		CMP	#BLKADR,R5	:ALL DONE ?
23	037722	001374			BNE	2\$:BRANCH IF NOT
24	037724	000137	005112		JMP	SIZMEM	:LOOP BACK

.SBTTL TRAP DECODER

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

037730	010046		\$TRAP:	MOV	RO,-(SP)	::SAVE R0
037732	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
037736	005740			TST	-(R0)	::BACKUP BY 2
037740	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
037742	006300			ASL	R0	::POSITION FOR INDEXING
037744	016000	037764		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
037750	000200			RTS	R0	::GO TO ROUTINE

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

037752	011646		\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
037754	016666	000004		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
037762	000002	000002		RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			:	ROUTINE		
			:	-----		
037764	037752		\$TRPAD:	.WORD	\$TRAP2	
037766	035450			\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
037770	036030			\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
037772	036004			\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
037774	036044			\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
037776	036232			\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
040000	033616			\$GTSWR	::CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
040002	033526			\$CKSWR	::CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
040004	034070			\$RDCHR	::CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
040006	034160			\$RDLIN	::CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE
040010	037026			\$SAVREG	::CALL=SAVREG	TRAP+12(104412) SAVE R0-R5 ROUTINE
040012	037064			\$RESREG	::CALL=RESREG	TRAP+13(104413) RESTORE R0-R5 ROUTINE
2 040014	032616			\$DSPLY	::CALL=DISPLY	TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGE
3 000032			\$TERM:	-\$TRPAD		

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
 33
 34
 35
 36
 37
 38
 39
 40
 43
 44
 45
 46
 47
 48
 49

.SBTTL SINGLE/DUAL PORT RH/RM DRIVER (REV 6.5) 1981

```

:NEW DRIVE TYPE ID FOR RM02 *****
:10-AUG-77 *****
:10-MAR-78 THE SC, SC5 CHANGES
:NEW DRIVE TYPE ID FOR RM05 *****
:1980 *****
    
```

```

:COPYRIGHT (C) 1977,1981
:DIGITAL EQUIPMENT CORP.
:MAYNARD, MA 01754
:AUTHOR(S): JIM LACEY/CHUCK HESS
:REVISED BY: MIKE LEAVITT 11-APR-80, 27-MAR-81
    
```

:STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'

```

:RMERRS = RMD5
:RMERRS+2 = RMER1
:RMERRS+4 = RMER2
:RMERRS+6 = RMMR2
    
```

040016 000000 000000 000000 RMERRS: .WORD 0,0,0,0

:TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)

```

:DRVACT=0 IF DRIVE IS IDLE
:DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
:DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
    
```

040026	000	DRVACT: .BYTE 0	:DRIVE 0
040027	000	.BYTE 0	:DRIVE 1
040030	000	.BYTE 0	:DRIVE 2
040031	000	.BYTE 0	:DRIVE 3
040032	000	.BYTE 0	:DRIVE 4
040033	000	.BYTE 0	:DRIVE 5
040034	000	.BYTE 0	:DRIVE 6
040035	000	.BYTE 0	:DRIVE 7

:TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)

```

:DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
:DRVSTA>0 IF DRIVE IS ONLINE
:DRVSTA<0 IF DRIVE IS UNSAFE
    
```

040036	000	DRVSTA: .BYTE 0	:DRIVE 0
040037	000	.BYTE 0	:DRIVE 1
040040	000	.BYTE 0	:DRIVE 2
040041	000	.BYTE 0	:DRIVE 3
040042	000	.BYTE 0	:DRIVE 4
040043	000	.BYTE 0	:DRIVE 5
040044	000	.BYTE 0	:DRIVE 6
040045	000	.BYTE 0	:DRIVE 7

:TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)

```

:DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
:DRV TYP=7 IF DRIVE IS RM05 *****
:DRV TYP=5 IF DRIVE IS RM02 *****
:DRV TYP=4 IF DRIVE IS RM03
    
```



```

50                                     ;DRVTYP -1 IF NOT RM05/3/2
51
52 040046      000      DRVTYP: .BYTE 0          ;DRIVE 0
53 040047      000      .BYTE 0          ;DRIVE 1
54 040050      000      .BYTE 0          ;DRIVE 2
55 040051      000      .BYTE 0          ;DRIVE 3
56 040052      000      .BYTE 0          ;DRIVE 4
57 040053      000      .BYTE C          ;DRIVE 5
58 040054      000      .BYTE 0          ;DRIVE 6
59 040055      000      .BYTE 0          ;DRIVE 7
60
61 040056      000      ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
62 040057      000      ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
63 040060      000      ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
64 040061      000      DPINT: .BYTE 0          ;DRIVE 0
65 040062      000      .BYTE 0          ;DRIVE 1
66 040063      000      .BYTE 0          ;DRIVE 2
67 040064      000      .BYTE 0          ;DRIVE 3
68 040065      000      .BYTE 0          ;DRIVE 4
69 040066      000      .BYTE 0          ;DRIVE 5
70 040067      000      .BYTE 0          ;DRIVE 6
71 040070      000      .BYTE 0          ;DRIVE 7
72 040071      000
73 040072      000
74 040073      000
75 040074      000
76 040075      000
77
78                                     ;TABLE OF PENDING DUAL PORT REQUESTS
79 040076      000000    ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
80                                     ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
81 DPRQS: .BYTE 0          ;DRIVE 0
82 .BYTE 0          ;DRIVE 1
83 .BYTE 0          ;DRIVE 2
84 .BYTE 0          ;DRIVE 3
85 .BYTE 0          ;DRIVE 4
86 .BYTE 0          ;DRIVE 5
87 .BYTE 0          ;DRIVE 6
88 .BYTE 0          ;DRIVE 7
89
90                                     ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
91                                     ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
92                                     ;"DPB" OF THE I/O OPERATION.
93 040100      000000    TRNSWT: .WORD 0
94
95                                     ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
96                                     ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
97                                     ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
98                                     ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
99                                     ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
100 SRCHWT: .WORD 0
101
102                                     ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
103                                     ;ACTDRV=0 IF DRIVER IS INACTIVE
104                                     ;ACTDRV>0 IF DRIVER IS ACTIVE
105 040102      000      ACTDRV: .BYTE 0
  
```

```

95          ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
96          ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
97          ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
98
99 040103    000          ACTSTR: .BYTE 0
100
101          ;UNLOAD FLAG (ULDFLG=8 BYTES)
102          ;ULDFLG=0 IF NO UNLOAD COMMAND
103          ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
104          ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
105
106 040104    000          ULDFLG: .BYTE 0          ;DRIVE 0
107 040105    000          .BYTE 0          ;DRIVE 1
108 040106    000          .BYTE 0          ;DRIVE 2
109 040107    000          .BYTE 0          ;DRIVE 3
110 040110    000          .BYTE 0          ;DRIVE 4
111 040111    000          .BYTE 0          ;DRIVE 5
112 040112    000          .BYTE 0          ;DRIVE 6
113 040113    000          .BYTE 0          ;DRIVE 7
114
115          ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
116          ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
117          ;OPERATION IS COMPLETED AS PER (DPB+14).
118          ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
119          ;(DPB+14), AFTER AN ERROR.
120
121 040114    000000      SAVEFG: .WORD 0
122
123          ;SEEK FLAG (SEEKFG=1 WORD)
124          ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
125          ;FOR A DATA TRANSFER START A SEARCH COMMAND
126          ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
127          ;DISREGARD THE WINDOW
128
129 040116    177777      SEEKFG: .WORD -1
130
131          ;TIMEOUT TABLE (TIMER=8 WORDS)
132          ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
133
134 040120    177777      TIMER: .WORD -1          ;DRIVE 0
135 040122    177777      .WORD -1          ;DRIVE 1
136 040124    177777      .WORD -1          ;DRIVE 2
137 040126    177777      .WORD -1          ;DRIVE 3
138 040130    177777      .WORD -1          ;DRIVE 4
139 040132    177777      .WORD -1          ;DRIVE 5
140 040134    177777      .WORD -1          ;DRIVE 6
141 040136    177777      .WORD -1          ;DRIVE 7
142
143          ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
144          ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
145          ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
146
147 040140    177777      DTUW: .WORD -1
148
149          ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
150          ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
151          ;ATTENTION BIT

```

```

144
145 040142 001          ABIT: .BYTE 1          ;DRIVE 0
146 040143 002          .BYTE 2          ;DRIVE 1
147 040144 004          .BYTE 4          ;DRIVE 2
148 040145 010          .BYTE 10         ;DRIVE 3
149 040146 020          .BYTE 20         ;DRIVE 4
150 040147 040          .BYTE 40         ;DRIVE 5
151 040150 100          .BYTE 100        ;DRIVE 6
152 040151 200          .BYTE 200        ;DRIVE 7
153
154          ;NUMBER OF 'MASSBUS CONTROL PARITY ERRORS' (MCPE) ALLOWED BEFORE
155          ;CALLING IT FATAL (MCPEMX=1 WORD)
156
157 040152 000003      MCPEMX: .WORD 3
158
159          ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
160          ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
161
162 040154 176700      RMADR: .WORD 176700
163 040156 000254      RMVEC: .WORD 254,5*32.
165 040162 000050      RHEXT: .WORD 50          ;OFFSET TO RMBAE
167
168          ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWINDW=1 WORD)
169 040164 000005      MXWINDW: .WORD 5
170
171          ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
172
173          000000      RMCS1 = 0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
174          000002      RMWC = 2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
175          000004      RMBA = 4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
176          000006      RMDA = 6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 5)
177          000010      RMCS2 = 10         ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
178          000012      RMD5 = 12         ;DRIVE STATUS REGISTER (DRIVE REG 1)
179          000014      RMER1 = 14        ;ERROR REGISTER #1 (DRIVE REG. 2)
180          000016      RMAS = 16        ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
181          000020      RMLA = 20        ;LOOK AHEAD REGISTER (DRIVE REG. 7)
182          000022      RMDB = 22        ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
183          000024      RMMR1 = 24       ;MAINTAINABILITY REGISTER (DRIVE REG. 3)
184          000026      RMDT = 26        ;DRIVE TYPE REGISTER (DRIVE REG. 6)
185          000030      RMSN = 30        ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
186          000032      RMOF = 32        ;OFFSET REGISTER (DRIVE REG. 11)
187          000034      RMDC = 34        ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
188          000036      RMHR = 36        ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
189          000040      RMMR2 = 40       ;MAINTENANCE REGISTER #2
190          000042      RMER2 = 42       ;ERROR REGISTER #2 (DRIVE REG. 15)
191          000044      RMEC1 = 44       ;ECC POSITION REGISTER (DRIVE REG. 16)
192          000046      RMEC2 = 46       ;ECC PATTERN REGISTER (DRIVE REG. 17)
194          000050      RMBAE = 50       ;BUS ADDRESS EXTENTION REGISTER
195          000052      RMCS3 = 52       ;CONTROL AND STATUS REGISTER #3
197
198          .SBTTL RH/RM DRIVER INITIALIZATION CODE
199
200          ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
201          ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
202          ;TO THE PROPER STATE FOR EACH DRIVE.
203          ;NOTE: THIS ROUTINE CALLS DRVINT
204          ;
  
```

```

205          :CALL
206          :
207          JSR    PC,RMINIT
208          RETURN
209          :
210          :NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
211          :
212          RMINIT: SAVREG          ;SAVE R0 - R5
213          MOV    PS,-(SP)        ;SAVE THE PRESENT PROCFSSOR STATUS
214          MOV    #<5*32.>,PS    ;CHANGE THE PRIORITY TO 5
215          JSR    PC,CLRQUE      ;CLEAR ALL REQUEST QUEUES
216          MOV    #RMERRS,R1     ;FIRST ADDRESS TO BE CLEARED
217          MOV    #SEEKFG,R2    ;LAST ADDRESS TO BE CLEARED
218          1$: CLR    (R1)+       ;CLEAR
219          CMP    R1,R2         ;ARE WE DONE?
220          BLO   1$            ;BR IF NO
221          MOV    #DTUW,R2       ;LAST ADDRESS
222          2$: MOV    #-1,(R1)+  ;INITIALIZE
223          CMP    R1,R2         ;DONE?
224          BLOS  2$            ;LOOP IF NO
225          CLR    DRVSTA        ;SET ALL DRIVES TO OFFLINE
226          CLR    DRVSTA+2
227          CLR    DRVSTA+4
228          CLR    DRVSTA+6
229          MOV    RMVEC,R3      ;SETUP THE RH/RM VECTOR
230          MOV    #ISR,(R3)+
231          MOV    RMVEC+2,(R3)
232          MOV    RMADR,R4
233          MOV    #CLR,RMC52(R4) ;FIRST ADDRESS OF RH/RM
234          CLR    R1            ;MASSBUS INIT
235          JSR    R0,DRVINT      ;START WITH DRIVE 0
236          BR    4$            ;INIT THE DRIVE
237          BR    5$            ;'DVA' NOT SET OR PARITY ERROR
238          4$: CLRB  DRVSTA(R1)  ;NORMAL RETURN
239          5$: INC    R1          ;SET DRIVE STATUS TO OFFLINE
240          BIC    #^C7,R1       ;GO TO NEXT DRIVE
241          BNE   3$            ;MASK OUT UNUSED BITS
242          MOV    #7,R1         ;BR IF MORE DRIVES TO GO
243          CLR    PS           ;START WITH DRIVE 7
244          6$: TSTB  DPINT(R1)   ;CLEAR THE PROCESSOR STATUS
245          BEQ   8$            ;WAITING FOR DRIVE TO SWITCH PORTS ?
246          JSR    PC,SET.IE     ;BR NOT WAITING
247          7$: TSTB  DPINT(R1)  ;SET INTERRUPT
248          BNE   7$           ;DRIVE SWITCHED PORTS ?
249          8$: DEC    R1        ;BR IF NOT
250          BPL   6$           ;GO TO THE NEXT DRIVE
251          MOV    (SP)+,PS      ;CHECK NEXT DRIVE
252          RESREG          ;RESTORE THE PROCESSOR STATUS
253          RTS    PC          ;RESTORE R0 - R5
254          :BYE-BYF
255          :
256          :DRIVE INITILIZATION ROUTINE
257          :THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
258          :AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
259          :IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
260          :INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
261          :DRVSTA IS SET TO THE PROPER CONDITION.
          :CALL
  
```

```

262      :      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
263      :      MOV      RMADR,R4      ;UNIBUS ADDRESS OF RH/RM (RMCS1)
264      :      JSR      RO,DRVINT      ;CALLED BY A JSR
265      :      RETURN1     ;ERROR OCCURRED (PARITY)
266      :      RETURN2     ;NORMAL RETURN
267      :
268
269 040400 010546      DRVINT: MOV      R5,-(SP)      ;SAVE R5
270 040402 105061 040036 CLRB   DRVSTA(R1)      ;START DRIVE STATUS AS OFFLINE
271 040406 105061 040046 CLRB   DRVYP(R1)      ;CLEAR THE DRIVE TYPE INDICATOR
272 040412 105061 040104 CLRB   ULDFLG(R1)     ;CLEAR THE UNLOAD FLAG
273 040416 010164 000010 MOV     R1,RMCS2(R4)   ;SELECT A DRIVE
274 040422 112764 000111 000000 MOVVB  #111,RMCS1(R4)   ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
275 040430 032764 010000 000010 BIT    #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
276 040436 001403      BEQ     1$          ;NO
277 040440 004737 045374      JSR    PC,SET.IE      ;GO SET 'IE' WITHOUT A 'TRE''
278 040444 000520      BR     4$          ;LEAVE THIS ROUTINE
279
280 040446 105061 040036 1$: CLRB   DRVSTA(R1)      ;SET DRIVE STATUS TO OFFLINE
281 040452 032764 004000 000000 BIT    #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
282 040460 001512      BEQ     4$          ;BR IF DRIVE NOT AVAILABLE
283 040462 004037 044666      JSR    RO,RD.RM      ;READ THE DRIVE TYPE REG.
284 040466 000026      RMDT   5$          ;ERROR RETURN ADDRESS
285 040470 040710      MOV     (SP)+,R5      ;PUT DRIVE TYPE IN R5
286 040472 012605      MOVVB  #4,DRVYP(R1)   ;SET RM03 INDICATOR
287 040474 112761 000004 040046 CMP    #20024,R5      ;SINGLE PORT RM03 ?
288 040502 022705 020024      BEQ     2$          ;BR IF YES
289 040506 001431      CMP    #24024,R5     ;DUAL PORT RM03 ?
290 040510 022705 024024      BEQ     2$          ;BR IF YES
291 040514 001426      MOVVB  #5,DRVYP(R1)   ;SET RM02 INDICATOR
292 040516 112761 000005 040046 CMP    #20025,R5      ;SINGLE PORT RM02 ?
293 040524 022705 020025      BEQ     2$          ;BR IF SO
294 040530 001420      CMP    #24025,R5     ;DUAL PORT RM02 ?
295 040532 022705 024025      BEQ     2$          ;BR IF SO
296 040536 001415      MOVVB  #7,DRVYP(R1)   ;SET RM05 INDICATOR
297 040540 112761 000007 040046 CMP    #20027,R5      ;SINGLE PORT RM05 ?
298 040546 022705 020027      BEQ     2$          ;BR IF YES
299 040552 001407      CMP    #24027,R5     ;DUAL PORT RM05 ?
300 040554 022705 024027      BEQ     2$          ;BR IF YES
301 040560 001404      MOVVB  #-1,DRVYP(R1) ;SET INDICATOR TO 'OTHER'
302 040562 112761 177777 040046 BR     4$          ;EXIT
303 040570 000446
304
305 040572 012746 000121 2$: MOV    #121,-(SP)      ;DO A 'READ-IN PRESET''
306 040576 004037 045042      JSR    RO,WRT.RM
307 040602 000000      RMDT   5$
308 040604 040710      MOV    #BIT12,-(SP)   ;SET FMT16=1
309 040606 012746 010060      JSR    RO,WRT.RM
310 040612 004037 045042      RMOF   5$
311 040616 000032      JSR    RO,RD.RM      ;READ RMD5
312 040620 040710      RMD5   5$
313 040622 004037 044666      MOV    (SP)+,R5      ;AND SAVE IT IN R5
314 040626 000012      BPL   3$          ;BR IF ATA=0
315 040630 040710      MOVVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
316 040632 012605
317 040634 100015
318 040636 116164 040142 000016

```

```

319 040644 004037 044666 JSR RO, RD.RM ;FIND OUT WHY ATA 1
320 040650 000014 RMER1
321 040652 040710 5$
322 040654 006126 ROL (SP)+ ;IS IT UNSAFE?
323 040656 100004 BPL 3$ ;BR IF NOT
324 040660 112761 177777 040036 MOV# #1, DRVSTA(R1) ;SET UNSAFE INDICATOR
325 040666 000407 BR 4$ ;EXIT
326
327 040670 005105 3$: COM R5 ;CHECK MOL, DPR, DRY, AND VV
328 040672 042705 167077 BIC #^C<BIT12:BIT08:BIT07:BIT06>, R5
329 040676 001003 BNE 4$ ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
330 040700 112761 000001 040036 MOV# #1, DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
331 040706 005720 4$: TST (R0)+ ;STEP OVER THE ERROR RETURN
332 040710 012605 5$: MOV (SP)+, R5 ;RESTORE R5
333 040712 000200 RTS RO ;EXIT
334
335 ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
336
337 ;CALL
338
339 ;
340 JSR RO, RM05 ;CALL THE RM05 DRIVER
341 PNTADR ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
342 RETURN1 ;RETURN HERE IF QUEUE IS FULL
343 RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
344 ;IS AN ERROR CONDITION
345 040714 013746 177776 RM05: MOV PS, -(SP) ;SAVE THE CALLING STATUS
346 040720 013737 040160 177776 MOV RMVEC+2, PS ;DON'T ALLOW ANY RM INTERRUPTS
347 040726 112737 000001 040102 MOV# #1, ACTDRV ;SET 'ACTIVE DRIVER' FLAG
348 040734 104412 SAVREG ;SAVE R0 - R5
349 040736 011002 MOV (R0), R2 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
350 040740 005062 000016 CLR 16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
351 040744 111201 MOV# (R2), R1 ;PICKUP THE DRIVE NUMBER
352 040746 013704 040154 MOV RMADR, R4 ;UNIBUS ADDRESS OF RMCS1
353 040752 105761 040036 TSTB DRVSTA(R1) ;CHECK DRIVES STATUS
354 040756 003014 PGT 1$ ;BR IF ONLINE
355 040760 105761 040104 TSTB ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
356 040764 001036 BNE 3$ ;BR IF YES
357 040766 105761 040056 TSTB DPINT(R1) ;TRYING TO INIT THE DRIVE
358 040772 001042 BNE 5$ ;BR IF YES
359 040774 004037 040400 JSR RO, DRVINT ;GO INIT. THE DRIVE
360 041000 000434 BR 4$ ;ERROR RETURN
361 041002 105761 040036 TSTB DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
362 041006 003445 BLE 6$ ;BR IF NOT
363 041010 105761 040066 1$: TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
364 041014 001031 BNE 5$ ;BR IF YES
365 041016 010164 000010 MOV R1, RMCS2(R4) ;SELECT THE DRIVE
366 041022 004037 046036 JSR RO, DRVQUE ;PUT THIS REQUEST IN QUEUE
367 041026 000460 BR 9$ ;QUEUE IS FULL
368 041030 122762 000103 000002 CMPB #103, 2(R2) ;IS THIS REQ. FOR AN UNLOAD?
369 041036 001003 BNE 2$ ;BR IF NO
370 041040 112761 177777 040104 MOV# #-1, ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
371 041046 105761 040026 2$: TSTB DRVACT(R1) ;IS THIS DRIVE ACTIVE?
372 041052 001043 BNE 8$ ;BR IF YES
373 041054 004737 041206 JSR PC, OPT ;CALL THE OPTIMIZER
374 041060 000440 BR 8$
375 041062 012762 120000 000016 3$: MCV #BIT15.BIT13, 16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
  
```

```

376 041070 000434          BR      8$          ;EXIT
377 041072 004737 042264 4$: JSR    PC,C17      ;GO HANDLE THE PARITY ERROR
378 041076 000431          BR      8$
379 041100 004037 046036 5$: JSR    R0,DRVQUE   ;PUT REQUEST IN QUEUE
380 041104 000431          BR      9$          ;QUEUE IS FULL
381 041106 032714 000100  BIT    #BIT06,(R4)   ;IE BIT SET ?
382 041112 001023          BNE    8$          ;YES
383 041114 004737 045374  JSR    PC,SET.IE    ;SET THE INTERRUPT
384 041120 000420          BR      8$          ;RETURN
385 041122 105761 040036 6$: TSTB   DRVSTA(R1)  ;SEE IF DRIVE OFFLINE OR UNSAFE
386 041126 002412          BLT    7$          ;BR IF UNSAFE
387 041130 012762 140000 000016 MOV    #BIT15.BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
388 041136 105761 040046  TSTB   DRVSTYP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
389 041142 001007          BNE    8$          ;BR IF OFFLINE
390 041144 012762 100002 000016 MOV    #BIT15.BIT01,15(R2) ;REPORT DRIVE NONEXISTENT
391 041152 000403          BR      8$          ;GO TO EXIT
392 041154 012762 110000 000016 7$: MOV    #BIT15.BIT12,16(R2) ;DRIVE IS UNSAFE
393 041162 104413          8$: RESREG          ;RESTORE R0 - R5
394 041164 005720          TST    (R0)+        ;SETUP FOR NORMAL RETURN
395 041166 000401          BR      10$        ;FINISH UP, THEN EXIT
396 041170 104413          9$: RESREG          ;RESTORE R0 - R5
397 041172 005720          10$: TST    (R0)+       ;CORRECT THE RETURN ADDRESS
398 041174 105037 040102  CLRB   ACTDRV      ;CLEAR 'ACTIVE DRIVER' FLAG
399 041200 012637 177776  MOV    (SP)+,PS    ;RETURN 'PS' TO USER LEVEL
400 041204 000200          RTS    R0          ;RETURN TO CALLER
401
402 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
403 ;
404 ;CALL
405 ;
406 ;
407 ;
408 041206 104412          OPT: SAVREG          ;SAVE R0 - R5
409 041210 013746 177776  MOV    PS,-(SP)    ;SAVE PROC. STATUS
410 041214 146137 040142 040100 BICB   ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
411 041222 105061 040066  CLRB   DPRQS(R1)  ;RESET THE PORT REQ FLAG ****
412 041226 004737 046112  JSR    PC,GETREQ   ;GET 'DPB' POINTER OF REQUEST
413 041232 005702          TST    R2          ;IS THERE A REQUEST IN QUEUE?
414 041234 001466          BEQ    7$          ;NO--BR TO EXIT
415 041236 010164 000010  MOV    R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
416 041242 012764 000111 000000 MOV    #111,RMCS1(R4) ;CLEAR THE DRIVE
417 041250 032764 004000 000000 BIT    #BIT11,RMCS1(R4) ;DVA SET ?
418 041256 001442          BEQ    5$          ;TO PORT REQUEST,IF NOT
419 041260 105761 040036 1$: TSTB   DRVSTA(R1)  ;IS DRIVE ONLINE?
420 041264 003014          BGT    2$          ;YES
421 041266 004737 046134  JSR    PC,POPQUE   ;NO--REMOVE REQUEST FROM QUEUE
422 041272 012762 140000 000016 MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
423 041300 105761 040036  TSTB   DRVSTA(R1)  ;IS DRIVE UNSAFE ?
424 041304 100047          BPL    8$          ;BR TO EXIT IF NOT
425 041306 012762 110000 000016 MOV    #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
426 041314 000443          BR      8$          ;BR TO EXIT
427 041316
428 041316 122762 000150 000002 2$: CMPB   #150,2(R2)  ;IS THE REQUEST FOR I/O?
429 041324 002403          BLT    3$          ;YES
430 041326 004737 041650  JSR    PC,C14      ;CALL THE COMMAND INITIATOR
431 041332 000434          BR      8$          ;BR TO EXIT
432 041334 005737 040140 3$: TST    DTUW       ;DATA TRANSFER UNDERWAY?

```

```

441 041340 002006          BGE      4$          ;YES--GO START A SEARCH
442 041342 005737 040116   TST      SEFKFG      ;DO IMPLIED SEEKS?
443 041346 100003          BPL      4$          ;NO, DO A SEARCH
444 041350 004737 041434   JSR      PC,C11     ;START A DATA TRANSFER
445 041354 000423          BR       8$
446 041356 004737 041542   4$:    JSR      PC,C13     ;START A SEARCH
447 041362 000420          BR       8$          ;GO TO THE EXIT
448 041364 112761 177777 040066 5$:    MOVB     #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
449 041372 010103          MOV      R1,R3      ;SET UP TO ADDRESS WORDS
450 041374 006303          ASL      R3          ;CONVERT TO WORD INDEX
451 041376 012763 035230 040120   MOV      #15000.,TIMER(R3) ;START 15. SECOND TIMER
452 041404 000402          BR       7$          ;EXIT
453 041406 004737 042264   6$:    JSR      PC,C17     ;PROCESS THE PARITY ERROR
454 041412 032714 000100   7$:    BIT      #B!T06,(R4) ;SEE IF 'IE' ALREADY SET
455 041416 001002          BNE      8$          ;BR IF SET
456 041420 004737 045374   JSR      PC,SET.IE  ;SET 'IE' WITHOUT A 'TRE'
457 041424 012637 177776   8$:    MOV      (SP)+,PS  ;RESTORE PROC. STATUS
458 041430 104413          RESREG          ;RESTORE R0 - R5
459 041432 000207          RTS      PC

460
461          ;COMMAND INITIATOR
462
463          ;CALL
464          MOV      #DRVNUM,R1 ;DRIVE NUMBER
465          MOV      #DPB,R2     ;ADDRESS OF DPB
466          JSR      PC,C1?     ;C1?= C11,C13, OR C14
467          ;WHERE:
468          ;C11=DATA TRANSFER
469          ;C13=SEARCH REQUESTED BY DATA XFER
470          ;C14=NOT DATA TRANSFER
471
472 041434 004737 046134   C11:   JSR      PC,POPQUE  ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
473 041440 010237 040076   MOV      R2,TRNSWT  ;PUT REQ. IN TRANSFER WAIT QUEUE
474 041444 010203          MOV      R2,R3      ;DPB ADDRESS TO R3
475 041446 013704 040154   MOV      RMADR,R4    ;RMCS1 ADDRESS
476 041452 010164 000010   MOV      R1,RMCS2(R4) ;SELECT DRIVE
477 041456 062703 000004   ADD      #4,R3       ;DESIRED WORD COUNT
478 041462 062704 000002   ADD      #2,R4       ;RMWC ADDRESS
479 041466 012324          MOV      (R3)+,(R4)+ ;LOAD WORD COUNT
480 041470 012324          MOV      (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
481 041472 012346          MOV      (R3)+,-(SP) ;LOAD SECTOR AND TRACK
482 041474 004037 045042   JSR      R0,WRT.RM  ;CALL THE LOAD(WRITE) ROUTINE
483 041500 000006          RMDA     ;INDEX OF REGISTER TO LOAD
484 041502 042264          C17     ;ERROR RETURN ADDRESS
485 041504 012346          MOV      (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
486 041506 004037 045042   JSR      R0,WRT.RM
487 041512 000034          RMDC     ;
488 041514 042264          C17     ;
489 041516 016246 000002   MOV      2(R2),-(SP) ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSFL'
490 041522 004037 045042   JSR      R0,WRT.RM
491 041526 000000          RMCS1   ;
492 041530 042264          C17     ;
493 041532 010137 040140   MOV      R1,DTUW    ;SET 'DATA TRANSFER UNDERWAY'
494 041536 000137 042226   JMP      C15
495
496 041542 013704 040154   C13:   MOV      RMADR,R4    ;RMCS1 ADDRESS
497 041546 010164 000010   MOV      R1,RMCS2(R4) ;SELECT DRIVE

```

;


```

498 041552 016246 000012      MOV      12(R2),-(SP)      ;DESIRED CYLINDER ADDRESS
499 041556 004037 045042      JSR      R0,WRT.RM
500 041562 000034      RMDC
501 041564 042264      CI7
502 041566 116203 000010      MOV      10(R2),R3      ;PICKUP SECTOR ADDRESS
503 041572 163703 040164      SUB      MXWINDW,R3      ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
504 041576 002002      BGE
505 041600 062703 000040      ADD      #32.,R3
506 041604 010346      MOV      R3,-(SP)      ;COMBINE THE ADJUSTED SECTOR WITH
507 041606 116266 000011 000001 1$:      MOV      11(R2),1(SP)      ;THE DESIRED TRACK
508 041614 094037 045042      JSR      R0,WRT.RM      ;LOAD DESIRED TRACK & SECTOR
509 041620 0C0006      RMDC
510 041622 042264      CI7
511 041624 012746 000131      MOV      #131,-(SP)      ;START A SEARCH
512 041630 004037 045042      JSR      R0,WRT.RM
513 041634 000000      RMCS1
514 041636 042264      CI7
515 041640 156137 040142 040100      BISB    ATABIT(R1),SRCHWT      ;SET 'SEARCH WAIT' KEY
516 041646 000567      BR      C15
517
518 041650 013704 040154      C14:    MOV      RMADR,R4      ;RMCS1 ADDRESS
519 041654 010164 000010      MOV      R1,RMCS2(R4)      ;SELECT DRIVE
520 041660 116203 000002      MOV      2(R2),R3      ;PICKUP THE REQUESTED COMMAND
521 041664 122703 000131      CMPB    #131,R3      ;IS IT A SEARCH COMMAND?
522 041670 001007      BNE      1$      ;BR IF NO
523 041672 016246 000010      MOV      10(R2),-(SP)      ;LOAD DESIRED TRACK & SECTOR
524 041676 004037 045042      JSR      R0,WRT.RM
525 041702 000006      RMDC
526 041704 042264      CI7
527 041706 000403      BR      2$      ;GO LOAD CYLINDER
528 041710 122703 000105      1$:    CMPB    #105,R3      ;IS IT A SEEK COMMAND
529 041714 001007      BNE      3$      ;BR IF NO
530 041716 016246 000012      2$:    MOV      12(R2),-(SP)      ;LOAD DESIRED CYLINDER
531 041722 004037 045042      JSR      R0,WRT.RM
532 041726 000034      RMDC
533 041730 042264      CI7
534 041732 000546      BR      C16
535 041734 122703 000115      3$:    CMPB    #115,R3      ;IS IT AN 'OFFSET' COMMAND?
536 041740 001013      BNE      4$      ;BR IF NO
537 041742 004037 044666      JSR      R0,RD.RM      ;MERGE THE OFFSET VALUE INTO RMOF
538 041746 000032      RMOF
539 041750 042264      CI7
540 041752 116216 000001      MOV      1(R2),(SP)      ;BYTE WHEN LOADING THE
541 041756 004037 045042      JSR      R0,WRT.RM      ;REGISTER (RMOF)
542 041762 000032      RMOF
543 041764 042264      CI7
544 041766 000530      BR      C16
545 041770 122703 000107      4$:    CMPB    #107,R3      ;IS IT A 'RECALIBRATE' COMMAND?
546 041774 001525      BEQ     C16      ;BR IF YES
547 041776 122703 000117      CMPB    #117,R3      ;IS IT A RETURN TO CENTER?
548 042002 001522      BEQ     C16      ;BR IF YES
549 042004 122703 000103      CMPB    #103,R3      ;IS IT AN 'UNLOAD' COMMAND?
550 042010 001016      BNE      5$      ;BR IF NO
551 042012 112761 000001 040026      MOV      #1,DRVACT(R1)      ;SET THE DRIVE ACTIVE INDICATOR
552 042020 105061 040036      CLRB    DRVSTA(R1)      ;PUT DRIVE STATUS TO OFFLINE
553 042024 112761 000001 040104      MOV      #1,ULDFLG(R1)      ;SET 'UNLOAD IN PROGRESS' FLAG
554 042032 010346      MOV      R3,-(SP)      ;START THE 'UNLOAD' COMMAND
  
```

```

555 042034 004037 045042 JSR RO,WRT.RM
556 042040 000000 RMCS1
557 042042 042264 C17
558 042044 000207 RTS PC ;RETURN TO USER
559 042046 122703 000143 5$: CMPB #143,R3 ;IS IT A 'SET FORMAT' COMMAND?
560 042052 001014 6$: BNE ;BR IF NO
561 042054 004037 044666 JSR RO,RD.RM ;READ THE OFFSET REGISTER
562 042060 000032 RMOF
563 042062 042264 C17
564 042064 116266 000001 000001 MOVB 1(R2),1(SP) ;COMBINE 'FM16','ECI', AND 'HCI'
565 042072 004037 045042 JSR RO,WRT.RM ;LOAD 'FMT16','ECI', AND/OR 'HCI'.
566 042076 000032 RMOF
567 042100 042264 C17
568 042102 000436 BR 2$
569 042104 122703 000141 6$: CMPB #141,R3 ;IS IT A 'GET REGISTER' COMMAND?
570 042110 001023 10$: BNE ;BR IF NO
571 042112 016203 000006 7$: MOV 6(R2),R3 ;POINTS TO 1ST ADDRESS OF WHERE
572 ;TO PUT THE REGISTER(S)
573 042116 116237 000010 042134 MOVB 10(R2),9$ ;INIT. THE INDEX FOR THE FIRST REG.
574 042124 116205 000011 MOVB 11(R2),R5 ;INDEX OF LAST REG. TO MOVE
575 042130 004037 044666 8$: JSR RO,RD.RM ;READ RH/RM REGISTER
576 042134 000000 9$: RMCS1 ;INDEX OF REG. TO READ
577 042136 042264 C17
578 042140 012623 MOV (SP)+,(R3)+ ;GET THE CONTENTS OF RH/RM REG.
579 042142 023705 042134 CMP 9$,R5 ;LAST REG. BEEN READ?
580 042146 001414 BEQ 12$ ;GET OUT IF YES
581 042150 062737 000002 042134 ADD #2,9$ ;INCREASE THE INDEX BY 2
582 042156 000764 BR 8$ ;LOOP--MORE TO READ
583 042160 122703 000145 10$: CMPB #145,R3 ;IS IT A 'SELECT DRIVE' COMMAND?
584 042164 001405 BEQ 12$ ;BR IF YES
585 042166 010346 11$: MOV R3,-(SP) ;LOAD THE COMMAND
586 042170 004037 045042 JSR RO,WRT.RM
587 042174 000000 RMCS1
588 042176 042264 C17
589 042200 004737 046134 12$: JSR PC,POPQUE ;REMOVE REQ. FROM QUEUE
590 042204 052762 000200 000016 BIS #BIT07,16(R2) ;SET THE 'DONE' BIT
591 042212 005737 040114 TST SAVEFG ;SAVE THE RH/RM REGISTERS?
592 042216 100002 BPL 13$ ;BR IF NO
593 042220 004737 045230 JSR PC,SVRH70 ;YES--GO SAVE THE REGISTERS
594 042224 000207 13$: RTS ;RETURN TO USER
595
596 042226 006301 15$: ASL R1
597 042230 012761 023420 040120 MOV #10000.,TIMER(R1) ;START 10. SECOND TIMER
598 042236 006201 ASR R1
599 042240 112761 000001 040026 MOVB #1,DRVACT(R1) ;SET THE DRIVE ACTIVE
600 042246 000207 RTS ;RETURN TO THE USER
601
602 042250 010346 16$: MOV R3,-(SP) ;LOAD THE COMMAND
603 042252 004037 045042 JSR RO,WRT.RM
604 042256 000000 RMCS1
605 042260 042264 C17
606 042262 000761 BR C15
607
608 042264 032764 010000 000010 C17: BIT #BIT12,RMCS2(R4) ;DRIVE NON-EXISTENT ?
612 042272 005702 1$: TST R2 ;ANYTHING IN QUEUE ?
616 042274 001001 BNE 2$ ;BR IF QUEUE IS THERE
617 042276 000207 RTS PC ;OTHERWISE EXIT

```

```

618 042300 012762 104000 000016 2$: MOV #BIT15.BIT11,16(R2) ;SET 'PARITY' ERROR INDICATOR
622
623 042306 012766 000111 C17B: MOV #111,-(SP) ;DO A 'DRIVE CLEAR'
624 042312 004037 045042 JSR R0,WRT.RM
625 042316 000000 RMCS1
626 042320 042364 C18
627 042322 004737 046016 1$: JSR PC,EMPTYQ ;EMPTY THE QUEUE
628 042326 105061 040066 CLR B DPRQS(R1) ;CLEAR THE PORT REQUEST FLAG
629 042332 105061 040104 CLR B ULDFLG(R1) ;CLEAR THE UNLOAD IN QUEUE FLAG
630 042336 105061 040026 CLR B DRVACT(R1) ;DRIVE IS IDLE
631 042342 020237 040076 CMP R2,TRNSWT ;IF THIS DRIVE HAD AN I/O REQUEST
635 042346 001005 BNE 2$ ;IN PROGRESS CLEAR ALL OF THE FLAGS
636 042350 005037 040076 CLR TRNSWT
637 042354 012737 177777 040140 MOV #-1,DTUW
638 042362 000207 2$: RTS PC
639
640 042364 104412 C18: SAVREG ;SAVE R0 - R5
641 042366 005001 CLR R1
642 042370 005003 CLR R3
643 042372 105761 040026 1$: TST B DRVACT(R1) ;DRIVE ACTIVE?
644 042376 001003 BNE 2$ ;BR IF IN ACTIVE
645 042400 105761 040066 TST B DPRQS(R1) ;PORT REQUEST
646 042404 001443 BEQ 6$ ;BR IF NOT
647 042406 013702 040076 2$: MOV TRNSWT,R2 ;GET THE 'TRANSFER WAIT' QUEUE
648 042412 020137 040140 CMP R1,DTUW ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
649 042416 001402 BEQ 3$ ;BR IF YES
650 042420 004737 046112 JSR PC,GETREQ ;GET THE DPB POINTER
651 042424 005702 3$: TST R2 ;QUEUE ENTRY FOR DRIVE ?
652 042426 001413 BEQ 5$ ;BR IF NOT
653 042430 032764 010000 000010 BIT #BIT12,RMCS2(R4) ;'NED' SET ?
654 042436 001404 BEQ 4$ ;BR IF NOT
655 042440 012762 100002 000016 MOV #BIT15:BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
656 042446 000403 BR 5$ ;CONTINUE
657 042450 012762 102000 000016 4$: MOV #BIT15:BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
661 042456 012763 177777 040120 5$: MOV #-1,TIMER(R3) ;STOP THE TIMER
662 042464 105061 040026 CLR B DRVACT(R1) ;SET 'DRIVE ACTIVE' TO IDLE
663 042470 105061 040066 CLR B DPRQS(R1) ;CLEAR PORT REQUEST FLAG
664 042474 020137 040140 CMP R1,DTUW ;IS THIS DRIVE SETUP FOR A TRANSFER
665 042500 001005 BNE 6$ ;BR IF NOT
666 042502 012737 177777 040140 MOV #-1,DTUW ;RESET THE INDICATOR
667 042510 005037 040076 CLR TRNSWT ;CLEAR THE TRANSFER QUEUE
668 042514 105061 040104 6$: CLR B ULDFLG(R1) ;CLEAR UNLOAD FLAG
669 042520 032764 010000 000010 BIT #BIT12,RMCS2(R4) ;'NED' SET ?
673 042526 005201 INC R1 ;MOVE TO THE NEXT DRIVE
674 042530 062703 000002 ADD #2,R3
675 042534 042701 177770 BIC #^C7,R1
676 042540 001314 BNE 1$ ;BR IF MORE DRIVES
677 042542 012737 177777 040140 MOV #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
678 042550 005037 040076 CLR TRNSWT ;CLEAR THE 'TRANSFER WAIT' QUEUE
679 042554 004737 045740 JSR PC,CLRQUE ;CLEAR ALL OF THE REQUEST QUEUES
680 042560 012764 000040 000010 MOV #CLR,RMCS2(R4) ;DO A MASSBUS INIT.
681 042566 000406 BR 8$ ;CONTINUE
682 042570 004737 046016 7$: JSR PC,EMPTYQ ;CLEAR THE DRIVE'S QUEUE
683 042574 105061 040036 CLR B DRVSTA(R1) ;SET DRIVE TO OFFLINE
684 042600 105061 040046 CLR B DRVTYPE(R1) ;CLEAR THE DRIVE TYPE INDICATOR
685 042604 004737 045374 8$: JSR PC,SET.IE ;SET 'IE' WITHOUT 'TRE'
686 042610 104413 RESREG ;RESTORE R0 - R5

```

```

687 042612 000207          RTS      PC          ;RETURN
688
689                          ;INTERRUPT SERVICE ROUTINE
690
691 042614 112737 000001 040102 ISR:   MOVB   #1,ACTDRV   ;SET 'ACTIVE DRIVER' FLAG
692 042622 104412          SAVREG          ;SAVE R0 - R5
693 042624 013704 040154          MOV    RMADR,R4   ;ADDRESS OF RMCS1
694 042630 013701 040140          MOV    DTUW,R1   ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
695 042634 002402          BLT    1$       ;BR IF NO DATA TRANSFER UNDERWAY
696 042636 004737 042656          JSR   PC,TD      ;CALL TRANSFER DONE
697 042642 004737 043120          1$:   JSR   PC,SC   ;CALL SPECIAL CONDITIONS
698 042646 104413          2$:   RESREG          ;RESTORE R0 - R5
699 042650 105037 040102          CLRB  ACTDRV    ;CLEAR 'ACTIVE DRIVER' FLAG
700 042654 000002          RTI           ;RETURN
701
702                          ;TRANSFER DONE ROUTINE
703
704 042656 105061 040026          TD:   CLRB  DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
705 042662 012737 177777 040140          MOV   #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
706 042670 006301          ASL   R1
707 042672 012761 177777 040120          MOV   #-1,TIMER(R1) ;CANCEL TIMEOUT
708 042700 006201          ASR   R1
709 042702 013702 040076          MOV   TRNSWT,R2    ;GET 'DPB' ADDRESS FROM THE
710 042706 005037 040076          CLR   TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
711 042712 052762 000200 000016          BIS  #BIT07,16(R2) ;SET DONE
712 042720 010164 000010          MOV  R1,RMCS2(R4) ;SELECT THE DRIVE
713 042724 004037 044666          JSR  R0,RD.RM     ;TRANSFER ERROR(TRE=1)?
714 042730 000000          RMCS1
715 042732 042264          CI7
716 042734 006126          ROL  (SP)+
717 042736 100421          BMI  3$
718 042740 005737 040114          TST  SAVEFG
719 042744 100002          BPL  1$
720 042746 004737 045230          JSR  PC,SVRH70    ;YES--SAVE THE REGISTERS
721 042752 004737 043032          1$:  JSR  PC,WC.HK    ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
722 042756 004737 046112          JSR  PC,GETREQ    ;GET DPB POINTER
723 042762 005702          TST  R2
724 042764 001403          BEQ  2$
725 042766 004737 041206          JSR  PC,OPT       ;CALL OPTIMIZER
726 042772 000207          RTS
727 042774 012714 000113          2$:  MOV  #113,(R4)   ;RELEASE THE DRIVE
728 042776 043000          RTS
729
730 043002 052762 100100 000016 3$:  BIS  #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
731 043010 004737 046016          JSR  PC,EMPTYQ    ;EMPTY THE 'DRIVE'S WAIT' QUEUE
732 043014 004737 045230          JSR  PC,SVRH70    ;SAVE THE RH/RM REGISTERS
733 043020 012714 040111          MOV  #40111,(R4)  ;ISSUE A 'DRIVE CLEAR'
734 043024 012714 000113          MOV  #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
735 043030 000207          RTS
736
737                          ;FORCED WRITE CHECK ROUTINE
738
739 043032 005737 001470          WC.HK: TST  WRTCHK    ;DO WRITE CHECK ?
740 043036 001427          BEQ  2$
741 043040 122762 000161 000002          CMPB #WRTDAT,$COMND(R2) ;LAST OPERATION A WRITE ORDER ?
742 043046 001023          BNE  2$
743 043050 004037 046036          1$:  JSR  R0,DRVQUE   ;PUT THE OPERATION IN THE QUEUE

```

```

753 043054 000420          BR      2$          ;QUEUE IS FULL
754 043056 005062 000016  CLR     $STATUS(R2) ;CLEAR 'DONE' BIT IN DPB
755 043062 116262 002136 000027  MOVFB  $RMCS1(R2),$PREVO(R2) ;SAVE WRITE OPERATION CODE
756 043070 016262 000012 000034  MOV    $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
757 043076 016262 000010 000032  MOV    $SEC(R2),$PREVA(R2) ;SAVE SECTOR AND TRACK ADDRESSES
758 043104 105062 000024          CLRFB  $CODE(R2) ;CHANGE WRITE DATA TO WRITE CHECK
759 043110 112762 000151 000002  MOVFB  #WCKD,$COMND(R2) ;CHANGE FUNCTION CODE TO WRITE CHECK
760 043116 000207          2$:   RTS     PC          ;EXIT
761
763          ;SPECIAL CONDITION ROUTINE
764
765 043120 116403 000016  SC:    MOVFB  RMAS(R4),R3 ;READ 'RMAS'
766 043124 001012          BNE     2$          ;BR IF ANY 'ATA' BITS SET
767 043126 004037 044666  JSR    RO,RD,RM ;READ CONTROL AND STATUS REGISTER
768 043132 000000          RMCS1
769 043134 042364          CIB
770 043136 106126          ROLB   (SP)+ ;IS 'IE'=1?
771 043140 100403          BMI   1$          ;YES, NO DRIVES TO CHECK
772 043142 104001          EMT   1 ;REPORT AN ILLEGAL INTERRUPT
773 043144 004737 045374  JSR    PC,SET.IE ;SET INTERRUPT ENABLE
774 043150 000207          1$:   RTS     PC          ;RETURN
775 043152 005046          2$:   CLR   -(SP) ;PROCESS ALL DRIVES THAT HAVE
776 043154 110316          MOVFB  R3,(SP) ;AN 'ATA'=1
777 043156 012703 000001  MOV    #1,R3
778 043162 005001          CLR   R1
779
780 043164 030316          SC3:  BIT    R3,(SP) ;ATA=1?
781 043166 001005          BNE   SC5 ;YES
782
783 043170 005201          SC4:  INC   R1 ;MOVE TO THE NEXT DRIVE
784 043172 106303          ASLB  R3
785 043174 001373          BNE   SC3 ;BR IF MORE TO CHECK?
786 043176 005726          TST   (SP)+ ;CLEAN OFF THE STACK
787 043200 000207          RTS   PC ;RETURN TO USER
788
789 043202 105761 040056  SC5:  TSTB  DPINT(R1) ;INITIALIZING THE DRIVE ?
790 043206 001402          BEQ   1$          ;BR IF NOT
791 043210 000137 044116  JMP    SC13 ;PROCESS THE DRIVE
792 043214 105761 040066  1$:   TSTB  DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
793 043220 001402          BEQ   2$          ;BR IF NOT
794 043222 000137 044116  JMP    SC13 ;START THE OUTSTANDING COMMAND
795 043226 105761 040036  2$:   TSTB  DRVSTA(R1) ;CHECK THE DRIVE STATUS
796 043232 003023          BGT   4$          ;BR IF ONLINE
797 043234 105761 040104  TSTB  ULDFLG(R1) ;UNLOAD IN PROGRESS?
798 043240 003420          BLE   4$          ;BR IF NOT
799 043242 004737 046112  JSR    PC,GETREQ ;GET DPB POINTER
800 043246 004737 045230  JSR    PC,SVRH70 ;SAVE THE RH/RM REGISTERS
801 043252 004737 044046  JSR    PC,SC12 ;SAVE RMD5, RMER1, RMER2, AND RMMR2
802          ;ALSO DO A DRIVE INIT (DRVINT)
803 043256 105761 040036  TSTB  DRVSTA(R1) ;DID DRIVE COME ONLINE?
804 043262 003414          BLE   5$          ;NO
805 043264 032737 040000 040016  BIT    #BIT14,RMERRS ;WAS THERE AN ERROR?
806 043272 001000          BNE   3$          ;BR IF ERROR
807 043274 013705 040020  3$:   MOV    RMERRS+2,R5 ;YES -- PICKUP RMER1 AND
808 043300 000500          BR    SC6A ;GO PROCESS THE ERROR
809 043302 105761 040026  4$:   TSTB  DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
810 043306 001027          BNE   SC6 ;BR IF EITHER

```

```

814 043310 004737 044046      JSR      PC,SC12      ;SAVE RMD5, RMER1, RMER2, AND RMMR2
815                               ;ALSO DO A DRVINT
816 043314 105761 040056      5$:     TSTB      DPINT(R1) ;TRYING TO INIT THE DRIVE ?
817 043320 001323                BNE      SC4          ;BR IF YES, CHECK ON MORE DRIVES
818 043322 105761 040036      TSTB      DRVSTA(R1) ;CHECK ON DRIVE'S STATUS
819 043326 100412                BMI      6$          ;BR IF UNSAFE
820 043330 032737 020000 040022 BIT      #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
821 043336 001011                BNE      7$          ;BR IF YES
825 043340 012746 000111      MOV      #111,-(SP)   ;DRIVE CLEAR
826 043344 004037 045042      JSR      RO,WRT.RM   ;WRITE THE COMMAND INTO RMCS1
827 043350 000000                RMCS1          ;REGISTER INDEX
828 043352 043706                SC8           ;PARITY EXIT ADDRESS
829 043354 011605                6$:     MOV      (SP),R5 ;PICKUP (RMAS) BEFORE THE ERROR CALL
830 043356 104002                EMT         2        ;REPORT THE UNEXPECTED ATTENTION
831 043360 000703                BR         SC4      ;GO CHECK FOR MORE ATA'S
832 043362                               7$:
833 043364 104005                EMT         5        ;REPORT THE ADDRESS PLUG CHANGE
834 043364 000701                BR         SC4      ;CHECK FOR MORE DRIVES
835 043366 006301                SC6:     ASL      R1          ;SETUP TO ADDRESS WORDS
836 043370 012761 177777 040120 MOV      #-1,TIMER(R1) ;STOP THE TIMER
837 043376 006201                ASR      R1          ;RESTORE THE DRIVE ADDRESS
838 043400 004737 046112      JSR      PC,GETREQ   ;GET THE DPB POINTER FROM THE QUEUE
839 043404 010164 00001C      MOV      R1,RMCS2(R4) ;SELECT DPIPE
840 043410 000137 043736      JMP      SC11        ;PROCESS THE SEARCH
841 043414 004037 044666      JSR      RO,RD.RM   ;READ THE RM'S STATUS REG.
842 043420 000012                RMD5
843 043422 043706                SC8
844 043424 011605                MOV      (SP),R5    ;AND PUT IT IN R5
845 043426 006126                ROL      (SP)+      ;WAS THERE AN ERROR?
846 043430 100407                BMI      1$        ;BR IF ERROR
847 043432 105761 040026      TSTB      DRVACT(R1) ;CHECK DRIVE'S STATE
848 043436 003137                BGT      SC11      ;BR IF DRIVE ACTIVE WITH ORDER
849 043440 052762 100210 000016 BIS      #BIT15:BIT07:BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
850 043446 000470                BR         SC7
851 043450 004037 044666      1$:     JSR      RO,RD.RM   ;READ ERROR REGISTER #1
852 043454 000014                RMER1
853 043456 043706                SC8
854 043460 012605                MOV      (SP)+,R5  ;AND SAVE IT IN R5
855 043462 004737 045230      JSR      PC,SVRH70  ;SAVE RH/RM REGISTERS
856 043466 012746 000111      MOV      #111,-(SP) ;ISSUE A DRIVE CLEAR
857 043472 004037 045042      JSR      RO,WRT.RM
858 043476 000000                RMCS1
859 043500 043706                SC8
860
861 043502 006105                SC6A:    ROL      R5          ;WAS 'UNSAFE' CONDITION =1?
862 043504 100406                BMI      1$        ;BR IF YES
863 043506 005702                TST      R2          ;ANYTHING IN QUEUE ?
864 043510 001447                BEQ      SC7        ;BR IF NOT
865 043512 052762 100240 000016 BIS      #BIT15.BIT07.BIT05,16(R2) ;INFORM USER OF ERROR
866 043520 000443                BR         SC7
867 043522 004037 044666      1$:     JSR      RO,RD.RM   ;READ DRIVE STATUS REG. #1
868 043526 000012                RMD5
869 043530 043706                SC8
870 043532 011605                MOV      (SP),R5  ;SAVE RMD5 IN R5
871 043534 006126                ROL      (SP)+    ;'ERR'=1?
872 043536 100011                BPL      2$        ;BR IF NO--UNSAFE CLEARED
  
```

```

873 043540 112761 177777 040036      MOVB  #1,DRVSTA(R1) ;DRIVE IS UNSAFE
874 043546 004737 045230      JSR   PC,SVRH70    ;SAVE RH/RM REGISTERS
875 043552 052762 110000 000016      BIS   #BIT15.BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
876 043560 000423                BR    SC7
877 043562 032705 010000      2$:  BIT   #BIT12,R5    ;'MOL' = 1 ?
878 043566 001015                BNE   3$           ;BR IF YES
879 043570 112761 177777 040026      MOVB  #1,DRVACT(R1) ;ACTIVE ERROR RECOVER
880 043576 112761 000001 040036      MOVB  #1,DRVSTA(R1) ;ONLINE
881 043604 006301                ASL   R1
882 043606 012761 035230 040120      MOV   #15000.,TIMER(R1) ;START 15. SECOND TIMER
883 043614 006201                ASR   R1
884 043616 000137 043170      JMP   SC4
885 043622 052762 100220 000016 3$:  BIS   #BIT15.BIT07.BIT04,16(R2) ;INFORM USER OF ERROR
886
887 043630 105061 040026      SC7:  CLRB  DRVACT(R1) ;DRIVE IS IDLE
888 043634 004737 046134      JSR   PC,POPQUE   ;REMOVE THE QUEUE
889 043640 105761 040104      TSTB  ULDFLG(R1)  ;UNLOAD IN PROGRESS OR QUEUE?
890 043644 003002                BGT   1$           ;BR IF NOT
891 043646 105061 040104      CLRB  ULDFLG(R1)  ;CLEAR UNLOAD FLAG
892 043652 116164 040142 000016 1$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
893 043656 105761 040036      TSTB  DRVSTA(R1)  ;IS THE DRIVE UNSAFE ?
894 043654 100406                BMI   2$           ;BR IF IT IS
895 043666 012746 000111      MOV   #111,-(SP)  ;DRIVE CLEAR COMMAND
896 043672 004037 045042      JSR   RO,WRT,RM   ;WRITE THE COMMAND INTO RPCS1
897 043676 000000                RMCS1 ;REGISTER INDEX
898 043700 043706                SC8  ;PARITY EXIT ADDRESS
899 043702 000137 043170      2$:  JMP   SC4         ;CHECK FOR MORE DRIVES
900
901 043706 105761 040026      SC8:  TSTB  DRVACT(R1) ;IS DRIVE IDLE?
902 043712 001405                BEQ   1$           ;YES
903 043714 004737 046112      JSR   PC,GETREQ   ;GET DPB POINTER
904 043720 004737 042264      JSR   PC,C17      ;PROCESS THE PARITY ERROR
905 043724 000402                BR    2$           ;CONTINUE
906
907 043726                1$:  JSR   PC,C17B     ;PROCESS THE UNCORRECTABLE PARITY ERROR
908 043726 004737 042306      2$:  JMP   SC4         ;CHECK MORE DRIVES
909
910 043732 000137 043170      SC11: TSTB  ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
911 043736 105761 040104      BLE   1$           ;BR IF NO
912 043742 003402                CLRB  ULDFLG(R1)  ;CLEAR UNLOAD FLAG
913 043744 105061 040104      CLRB  DRVACT(R1)  ;SET DRIVE IDLE
914 043750 105061 040026 040100 1$:  BITB  ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
915 043754 136137 040142                ;AN I/O COMMAND?
916
917 043762 001012                BNE   2$           ;BR IF YES
918 043764 004737 046134      JSR   PC,POPQUE   ;REMOVE REQUEST FROM QUEUE
919 043770 052762 000200 000016      BIS   #BIT07,16(R2) ;SET 'DONE' BIT
920 043776 005737 040114      TST   SAVEFG      ;SAVE THE REGISTERS?
921 044002 100002                BPL   2$           ;BR IF NO
922 044004 004737 045230      JSR   PC,SVRH70   ;YES--SAVE ALL OF THE RH/RM REG'S
923 044010 116164 040142 000016 2$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
924 044016 146137 040142 040100      BICB  ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
925
926 044024 006301                ASL   R1           ;WORD INDEX
927 044026 012761 177777 040120      MOV   #-1,TIMER(R1) ;STOP CLOCK
928 044034 006201                ASR   R1           ;RESTORE R1
929 044036 004737 041206      JSR   PC,OPT      ;START A REQUEST
930 044042 000137 043170      JMP   SC4         ;CHECK FOR MORE DRIVES
931
932
933
934
935
936
937
938

```

```

939 044046 010164 000010          SC12:  MOV      R1,RMCS2(R4)      ;SELECT DRIVE
940 044052 016437 000012 040016    MOV      RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
941 044060 016437 000014 040020    MOV      RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
942 044066 016437 000014 040022    MOV      RMER2(R4),RMERRS+4
943 044074 016437 000040 040024    MOV      RMMR2(R4),RMERRS+6
944 044102 004037 040400          JSR      RO,DRVINT        ;INIT. THE STATE OF THE DRIVE
945 044106 000401          BR       1$              ;TAKE ERROR EXIT
946 044110 000207          RTS      PC              ;RETURN
947 044112 005726          1$:     TST      (SP)+      ;POP PC OFF OF THE STACK
948 044114 000674          BR       SC3            ;PROCESS THE PARITY ERROR
949
950 044116 006301          SC13:  ASL      R1              ;SETUP TO ADDRESS WORDS
951 044120 012761 177777 040120    MOV      #-1,TIMER(R1)   ;STOP THE TIMER.
952 044126 006201          ASR      R1              ;
953 044130 010164 000010          MOV      R1,RMCS2(R4)   ;SELECT THE DRIVE
954 044134 116164 040142 000016    MOVVB   ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
955 044142 105761 040056          1$:     TSTB   DPINT(R1)    ;INITIALIZING THE DRIVE ?
956 044146 001424          BEQ     2$              ;BR IF NOT
957 044150 105061 040056          CLRB   DPINT(R1)       ;CLEAR THE INIT INDICATOR
958 044154 004037 040400          JSR      RO,DRVINT        ;GO INIT THE DRIVE
959 044160 000240          NOP                    ;DUMMY PARITY ERROR RETURN
960 044162 105761 040036          TSTB   DRVSTA(R1)      ;DRIVE ONLINE ?
961 044166 003014          BGT     2$              ;BR IF YES -- START ORDER
962 044170 005702          TST     R2              ;QUEUE ENTRY FOR THE DRIVE
963 044172 001426          BEQ     3$              ;BR IF NOT
964 044174 004737 046112          JSR      PC,GETREQ      ;GET DPB ADDRESS
965 044200 052762 140000 000016    BIS    #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
966 044206 004737 045230          JSR      PC,SVRH70      ;SAVE THE REGISTERS
967 044212 004737 046134          JSR      PC,POPQUE      ;REMOVE THE QUEUE
968 044216 000414          BR       3$
969 044220 032764 004000 000000    2$:     BIT    #BIT11,RMCS1(R4) ;DVA SET ?
970 044226 001006          BNE     4$              ;SET THEN CALL OPT
971 044230 006301          ASL     R1
972 044232 012761 035230 040120    MOV     #15000.,TIMER(R1) ;START 15. SECOND TIMER
973 044240 006201          ASR     R1
974 044242 000402          BR     3$
975 044244 004737 041206          4$:     JSR     PC,OPT      ;START THE PENDING REQUEST
976 044250 000137 043170          3$:     JMP     SC4        ;PROCESS OTHER DRIVES
977
978          ;RM TIMER ROUTINE
979          ;CALL
980          ;
981          ;     MOV     #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
982          ;     JSR     PC,RMTMR ;CALL RM05 TIME ROUTINE
983
984
985
986 044254 005737 040102          RMTMR: TST     ACTDRV        ;CHECK 'ACTDRV & ACTSTR'
987 044260 001027          BNE     4$              ;IF NON ZERO EXIT
988 044262 112737 000001 040103    MOVVB   #1,ACTSTR      ;SET 'ACTSTR'
989 044270 104412          SAVREG ;SAVE R0 - R5
990 044272 005001          CLR     R1              ;START WITH DRIVE 0
991 044274 005003          CLR     R3
992 044276 005763 040120          1$:     TST     TIMER(R3)   ;IS THE TIMER RUNNING?
993 044302 002406          BLT     2$              ;BR IF NO
994 044304 166663 000002 040120    SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
995 044312 003002          BGT     2$              ;BR IF NO SOFTWARE TIMEOUT
996 044314 004737 044344          JSR     PC,STO          ;CALL SOFTWARE TIMEOUT ROUTINE
997 044320 005201          2$:     INC     R1          ;MOVE TO NEXT DRIVE
998 044322 005723          TST     (R3)+

```



```

999 044324 022701 000010      CMP      #8.,R1      :OUT OF DRIVES?
1000 044330 003362          BGT      1$         :BR IF NO
1001 044332 104413          3$: RESREG          :RESTORE R0 - R5
1002 044334 105037 040103      CLRB     ACTSTR     :ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1003 044340 012616          4$: MOV      (SP)+,(SP) :ADJUST THE STACK
1004 044342 000207          RTS      PC        :RETURN
1005
1006      :SOFTWARE TIMEOUT ROUTINE
1007
1008      :NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1009      :OR GREATER
1010
1011      :CALL:
1012      :   MOV      #DRVNUM,R1      :DRIVE NUMBER
1013      :   JSR      PC,STO          :CALL
1014      :   RETURN
1015
1016 044344 010146          STO:  MOV      R1,-(SP)      :SAVE R1
1017 044346 010246          MOV      R2,-(SP)      :SAVE R2
1018 044350 010346          MOV      R3,-(SP)      :SAVE R3
1019 044352 010446          MOV      R4,-(SP)      :SAVE R4
1020 044354 013704 040154      MOV      RMADR,R4      :GET ADDRESS OF 'RMCS1'
1021 044360 010164 000010      MOV      R1,RMCS2(R4)   :SELECT THE DRIVE
1022 044364 004037 044666          JSR      R0,RD.RM      :READ 'DRIVE STATUS REG'
1023 044370 000012          RMDS
1027 044372 044654          STO9
1028 044374 105726          TSTB     (SP)+        :IS 'DRY'=1?
1029 044376 100436          BMI     ST02         :BR IF YES
1030 044400 105761 040056          ST01: TSTB     DPINT(R1)   :TRYING TO INITIALIZE THE DRIVE ?
1031 044404 001033          BNE     ST02         :BR IF YES
1032 044406 105761 040066          TSTB     DPRQS(R1)     :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1033 044412 001030          BNE     ST02         :BR IF YES
1034 044414 013702 040076          MOV      TRNSWT,R2    :PICKUP TRANSFER WAIT QUEUE
1035 044420 020137 040140          CMP      R1,DTUW     :TRANSFER UNDERWAY ON THIS DRIVE?
1036 044424 001404          BEQ     1$          :BR IF YES
1037 044426 000137 044654          JMP     ST09         :IF NOT DON'T BOTHER DRIVES
1038 044432 004737 046112          JSR     PC,GETREQ    :GET DPB ADDRESS
1039 044436 052762 101000 000016 1$: BIS      #BIT15!BIT09,16(R2) :SET THE ERROR FLAGS
1040 044444 004737 045230          JSR     PC,SVRH70    :SAVE RH/RM REGISTERS
1044 044450 105061 040026          CLRB     DRVACT(R1)   :DRIVE IS IDLE
1045 044454 105061 040104          CLRB     ULDFLG(R1)   :CLEAR THE UNLOAD FLAG
1046 044460 005037 040076          CLR     TRNSWT      :CLEAR DPB ADDRESS
1047 044464 012737 177777 040140          MOV     #-1,DTUW     :CLEAR THE TRANSFER DRIVE #
1048 044472 000470          BR      ST09        :DON'T BOTHER OTHER DRIVES
1049
1050 044474 116405 000016          ST02: MOV     RMAS(R4),R5 :READ ATTENTION REG
1051 044500 136105 040142          BITB     ATABIT(R1),R5 :IS ATTENTION FOR THIS DRIVE UP ?
1052 044504 001007          BNE     ST03         :YES
1053 044506 105761 040056          TSTB     DPINT(R1)   :TRYING TO INITIALIZE THE DRIVE ?
1054 044512 001021          BNE     ST06         :BR IF YES - DRIVE NOT ONLINE
1055 044514 105761 040066          TSTB     DPRQS(R1)   :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1056 044520 001035          BNE     ST07         :BR IF YES - NO RESPONSE TO REQUEST
1057 044522 000454          BR      ST09        :OTHER WISE EXIT
1058
1059 044524 105761 040056          ST03: TSTB     DPINT(R1)   :INITIALIZING THE DRIVE ?
1060 044530 001003          BNE     1$          :BR IF INIT PENDING
1061 044532 105761 040066          TSTB     DPRQS(R1)   :PORT REQUEST PENDING ?
```

```

1062 044536 001446          BEQ      ST09          ;BR IF NOT
1063 044540 012763 177777 040120 1$:  MOV      #-1,TIMER(R3) ;STOP THE TIMER
1064 044546 000442          BR       ST09          ;EXIT
1065
1066 044550 004737 042364          ST05:  JSR      PC,C18          ;GO HANDLE THE PARITY ERROR
1067 044554 000437          BR       ST09
1068
1069 044556 105061 040056          ST06:  CLRB     DPINT(R1)        ;CLEAR THE INITIALIZE INDICATOR
1070 044562 105061 040036          CLRB     DRVSTA(R1)        ;SET UNIT OFFLINE
1071 044566 012763 177777 040120  MOV      #-1,TIMER(R3)    ;STOP THE TIMER
1072 044574 004737 046112          JSR      PC,GETREQ        ;GET THE DPB ADDRESS
1073 044600 005702          TST      R2              ;REQUEST IN QUEUE ?
1074 044602 001424          BEQ      ST09          ;BR IF NOT
1075 044604 052762 140000 000016  BIS      #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1076 044612 000414          BR       ST08          ;FINISH
1077
1078 044614 012763 177777 040120  ST07:  MOV      #-1,TIMER(R3)    ;STOP THE TIMER
1079 044622 105061 040066          CLRB     DPRQS(R1)        ;CLEAR PORT REQUEST INDICATOR
1080 044626 004737 046112          JSR      PC,GETREQ        ;GET DPB ADDRESS
1081 044632 005702          TST      R2              ;QUEUE ENTRY FOR DRIVE ?
1082 044634 001407          BEQ      ST09          ;BR IF NONE
1083 044636 012762 100004 000016  MOV      #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1084 044644 004737 046016          ST08:  JSR      PC,EMPTYQ        ;CLEAR THE QUEUE FOR THE DRIVE
1085 044650 004737 045230          JSR      PC,SVRH70        ;SAVE THE REGISTERS
1086 044654 012604          ST09:  MOV      (SP)+,R4        ;RESTORE R4
1087 044656 012603          MOV      (SP)+,R3        ;RESTORE R3
1088 044660 012602          MOV      (SP)+,R2        ;RESTORE R2
1089 044662 012601          MOV      (SP)+,R1        ;RESTORE R1
1090 044664 000207          RTS      PC              ;RETURN
1091
1092          ;ROUTINE TO READ A RH/RM REGISTER
1093          ;CALL
1094          ;CALL
1095          ;CALL      JSR      R0,RD.RM          ;GO READ A REGISTER
1096          ;CALL      INDEX     ;REG. INDEX FROM BASE
1097          ;CALL      ERRADR    ;ERROR ADDRESS--PROCESS ERROR STARTING
1098          ;CALL      ;AT THIS ADDRESS
1099          ;CALL      RETURN    ;CONTENTS OF REG. IS ON THE STACK
1100
1101 044666 013737 040152 045030  RD.RM:  MOV      MCPEMX,RD.RM2    ;MAX. RETRYS ALLOWED
1102 044674 011646          MOV      (SP),-(SP)        ;SAVE R0 FOR RETURN
1103 044676 013737 040154 044712  MOV      RMADR,RD.ADR      ;FORM THE DESIRED ADDRESS
1104 044704 062037 044712          ADD      (R0)+,RD.ADR      ;USING THE BASE AND THE INDEX
1105 044710 013727          RD.RM1: MOV      @ (PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM DRIVE
1106 044712 000000          RD.ADR: .WORD     0        ;ADDRESS IS FORMED HERE
1107 044714 000000          RD.WRD: .WORD     0        ;REG. CONTENTS PUT HERE
1108 044716 013766 044714 000002  MOV      RD.WRD,2(SP)      ;RETURN IT TO THE USER
1109 044724 013746 040154          MOV      RMADR, -(SP)      ;PUT THE ADDRESS ON THE STACK
1110 044730 062716 000010          ADD      #RMCS2,(SP)      ;FORM THE ADDRESS OF RMCS2
1111 044734 032736 010000          BIT      #BIT12,@(SP)+    ;CHECK THE 'NED' BIT
1112 044740 001035          BNE      RD.RM3          ;BR IF DRIVE NON-EXISTENT
1113 044742 017746 173206          MOV      @RMADR, -(SP)    ;READ RMCS1
1114 044746 032716 020000          BIT      #BIT13,(SP)      ;DID MCPE SET?
1115 044752 001002          BNE      1$              ;BR IF YES
1116 044754 022620          CMP      (SP)+,(R0)+      ;ADJUST FOR RETURN
1117 044756 000430          BR       RD.RM4          ;EXIT
1118 044760          1$:

```

```

044760 104003          EMT      3          ;REPORT 'MCPE' ERROR
1119 044762 005737 040140 TST     DTUW        ;DATA TRANSFER UNDERWAY?
1120 044766 100405          BMI     2$          ;NO
1121 044770 032716 040000 BIT     #BIT14,(SP) ;'TRE' = 1 ?
1122 044774 001402          BEQ     2$          ;NO
1123 044776 005726          TST     (SP)+      ;YES--CLEAN OFF THE STACK AND
1124 045000 000415          BR      RD.RM3     ;TAKE THE FATAL ERROR EXIT
1125 045002 052716 040000 2$:    BIS     #BIT14,(SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1126 045006 000316          SWAB    (SP)       ;POSITION BEFORE WRITING
1127 045010 013737 040154 045024 MOV     RMADR,3$   ;FORM ADDRESS OF HIGH BYTE
1128 045016 005237 045024          INC     3$          ;
1129 045022 112637          MOVB   (SP)+,@(PC)+ ;WRITE THE HIGH BYTE OF RMCS1
1130 045024 000000          3$:    .WORD 0       ;ADDRESS STORAGE
1131 045026 005327          DEC     (PC)+     ;EXCEEDED MAX. RETRYS
1132 045030 000003          RD.RM2: .WORD 3     ;
1133 045032 002326          BGE     RD.RM1    ;BR IF NO
1134 045034 011000          RD.RM3: MOV     (RO),RO ;FATAL ERROR EXIT
1135 045036 012616          MOV     (SP)+,(SP)
1136 045040 000200          RD.RM4: RTS     RO
1137
1138          ;ROUTINE TO WRITE A REGISTER
1139          ;
1140          ;CALL
1141          ;      MOV     DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
1142          ;      JSR     RO,WRT.RM  ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1143          ;      INDEX  ;INDEX OF THE REGISTER TO BE LOADED
1144          ;      ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
1145          ;      RETURN ;ERROR FREE RETURN
1146
1147 045042 013737 040152 045214 WRT.RM: MOV     MCPMX,WRT.R2 ;MAX RETRYS ALLOWED
1148 045050 016637 000002 045130 MOV     2(SP),WRT.WD ;SAVE THE WORD TO WRITE
1149 045056 012616          MOV     (SP)+,(SP) ;ADJUST THE STACK
1150 045060 012037 045132          MOV     (RO)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
1151 045064 001015          BNE     1$          ;BR IF NOT RMCS1
1152 045066 122737 000150 045130 CMPB   #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
1153 045074 002411          BLT     1$          ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1154 045076 004037 044666          JSR     RO,RD.RM  ;NO---COMBINE A16&A17, & PSEL WITH
1155 045102 000000          RMCS1 ;THE COMMAND BEFORE SENDING IT TO
1156 045104 045220          WRT.R3 ;THE RH/RM
1157 045106 000316          SWAB   (SP)
1158 045110 042716 177770          BIC     #^C7,(SP)
1159 045114 112637 045131          MOVB   (SP)+,WRT.WD+1
1160 045120 063737 040154 045132 1$:    ADD     RMADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
1161 045126 012737          WRT.R1: MOV     (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
1162 045130 000000          WRT.WD: .WORD 0   ;WORD TO WRITE GOES HERE
1163 045132 000000          WRT.AD: .WORD 0   ;ADDRESS IS FORMED HERE
1164 045134 013746 040154          MOV     RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
1165 045140 062716 000010          ADD     #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
1166 045144 032736 010000          BIT     #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
1167 045150 001023          BNE     WRT.R3     ;BR IF DRIVE NON-EXISTENT
1168 045152 004037 044666          JSR     RO,RD.RM  ;CHECK FOR PARITY ERROR ON WRITE
1169 045156 000014          RMER1
1170 045160 045220          WRT.R3
1171 045162 032726 000010          BIT     #BIT03,(SP)+
1172 045166 001416          BEQ     WRT.R4     ;BR IF 'PAR=0'
1173 045170 016037 177776 045202 MOV     -2(RO),1$  ;PICKUP THE INDEX
1174 045176 004037 044666          JSR     RO,RD.RM  ;READ THE REG.

```

```

1175 045202 000000          1$:      .WORD      0          ;REG. INDEX
1176 045204 045220          WRT.R3      ;RETURN TO THIS ADDRESS ON ERROR
1177 045206 104004          EMT         4          ;REPORT THE PARITY ON WRITE ERROR
1178 045210 005726          TST        (SP)+      ;CLEAR OFF THE STACK
1179 045212 005327          DEC        (PC)+      ;DECREMENT THE ERROR COUNT
1180 045214 000003          WRT.R2:    .WORD      3          ;RETRY COUNTER
1181 045216 002343          BGE        WRT.R1      ;TRY AGAIN IF NOT FINISHED
1182 045220 011000          WRT.R3:    MOV        (R0),R0    ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1183 045222 000401          BR         WRT.R5      ;EXIT
1184 045224 005720          WRT.R4:    TST        (R0)+      ;ADJUST FOR ERROR FREE EXIT
1185 045226 000200          WRT.R5:    RTS         R0
1186
1187          ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1188          ;
1189          ;CALL
1190          ;      MOV      #DPBNUM,R2      ;DPB POINTER TO R2
1191          ;      JSR      PC,SVRH70      ;SAVE THE DRIVES REG'S
1192
1193 045230 104412          SVRH70:    SAVREG          ;SAVE R0 - R5
1194 045232 005702          TST        R2          ;QUEUE ENTRY FOR THE DRIVE ?
1195 045234 001442          BEQ        6$          ;BR IF NONE
1196 045236 013704 040154          MOV        RMADR,R4
1197 045242 111264 000010          MOVB      (R2),RMCS2(R4) ;SELECT DRIVE
1198 045246 016203 000014          MOV        14(R2),R3   ;GET THE ERROR TABLE POINTER
1199 045252 001433          BEQ        6$          ;EXIT IF NO ADDRESS
1200 045254 005037 045310          CLR        3$          ;COUNTER & POINTER
1201 045260 023727 045310 000022 1$:      CMP        3$,#RMDB     ;REACHED THE BUFFER REGISTER ?
1202 045266 001006          BNE        2$          ;BR IF NOT
1203 045270 032764 000200 000010          BIT        #BIT07,RMCS2(R4) ;'OR' SET ?
1204 045276 001002          BNE        2$          ;BR IF SET
1205 045300 005023          CLR        (R3)+      ;STORE RMDB AS ZEROES
1206 045302 000405          BR         4$          ;CONTINUE
1207 045304 004037 044666          2$:      JSR        R0,RD.RM    ;READ THE SELECTED REGISTER
1208 045310 000000          3$:      .WORD      0          ;REGISTER INDEX
1209 045312 045336          5$:      5$          ;ERROR RETURN ADDRESS
1210 045314 012623          MOV        (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
1211 045316 023727 045310 000046 4$:      CMP        3$,#RMEC2   ;REACHED THE END ?
1212 045324 001406          BEQ        6$          ;BR IF YES
1213 045326 062737 000002 045310          ADD        #2,3$       ;INCREMENT THE REGISTER INDEX
1214 045334 000751          BR         1$          ;CONTINUE READING THE REGISTERS
1215 045336 004737 042264          5$:      JSR        PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
1217 045342 013746 001234          6$:      MOV        $CPUOP,-(SP) ;CHECK THE CPU (RH) TYPE
1218 045346 042716 003777          BIC        #^C174000,(SP) ;LEAVE THE CPU TYPE BITS
1219 045352 022726 030000          CMP        #30000,(SP)+ ;SEE IF RH70
1220 045356 001004          BNE        7$          ;IF NE, NO
1221 045360 063704 040162          ADD        RHEXT,R4    ;POINT TO RMBAE
1222 045364 012423          MOV        (R4)+,(R3)+ ;STORE THE CONTENTS
1223 045366 011413          MOV        (R4),(R3)   ;GET RMCS3
1224 045370 104413          7$:      RESREG          ;RESTORE R0 - R5
1228 045372 000207          RTS        PC          ;RETURN
1229
1230          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
1231          ;CALL
1232          ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
1233          ;      JSR      PC,SET.IE      ;SET 'IE'
1234          ;
1235          ;      RETURN

```

```

1236 045374 010446          SET.IE: MOV      R4,-(SP)          ;SAVE R4
1237 045376 013704 040154    MOV      RMADR,R4             ;PICKUP ADDRESS OF RMCS1
1238 045402 010164 000010    MOV      R1,RMCS2(R4)        ;SELECT DRIVE
1239 045406 011446           MOV      (R4),-(SP)          ;READ RMCS1
1240 045410 052716 040000    BIS      #BIT14,(SP)         ;SET THE 'TRE' BIT OF THE WORD READ
1241 045414 000316           SWAB    (SP)                 ;ADJUST FOR DATO
1242 045416 112714 000100    MOVB    #BIT06,(R4)         ;SET 'IE'
1243 045422 032764 010000 000010 BIT      #BIT12,RMCS2(R4)    ;IS 'NED'=1?
1244 045430 001002           BNE     1$                  ;YES--CLEAR 'TRE'
1245 045432 005726           TST     (SP)+               ;CLEAN OFF THE STACK
1246 045434 000402           BR      2$
1247 045436 112664 000001    1$:     MOVB    (SP)+,1(R4)    ;CLEAR 'TRE'
1248 045442 012604           2$:     MOV      (SP)+,R4      ;RESTORE R4
1249 045444 000207           RTS     PC                  ;RETURN TO CALLFR
1250
1251          ;QUEUE COUNT
1252
1253 045446      000          QCNT:   .BYTE    0            ;DRIVE 0
1256 045447      000          .BYTE    0            ;DRIVE 1
      045450      000          .BYTE    0            ;DRIVE 2
      045451      000          .BYTE    0            ;DRIVE 3
      045452      000          .BYTE    0            ;DRIVE 4
      045453      000          .BYTE    0            ;DRIVE 5
      045454      000          .BYTE    0            ;DRIVE 6
      045455      000          .BYTE    0            ;DRIVE 7
1257
1258          ;QUEUE INPUT POINTERS
1259
1260 045456 045540          QINPT: .WORD    QDRV0        ;DRIVE 0
1263 045460 045560          .WORD    QDRV1        ;DRIVE 1
      045462 045600          .WORD    QDRV2        ;DRIVE 2
      045464 045620          .WORD    QDRV3        ;DRIVE 3
      045466 045640          .WORD    QDRV4        ;DRIVE 4
      045470 045660          .WORD    QDRV5        ;DRIVE 5
      045472 045700          .WORD    QDRV6        ;DRIVE 6
      045474 045720          .WORD    QDRV7        ;DRIVE 7
1264
1265          ;QUEUE OUTPUT POINTERS
1266
1267 045476 045540          QOUTPT: .WORD    QDRV0        ;DRIVE 0
1270 045500 045560          .WORD    QDRV1        ;DRIVE 1
      045502 045600          .WORD    QDRV2        ;DRIVE 2
      045504 045620          .WORD    QDRV3        ;DRIVE 3
      045506 045640          .WORD    QDRV4        ;DRIVE 4
      045510 045660          .WORD    QDRV5        ;DRIVE 5
      045512 045700          .WORD    QDRV6        ;DRIVE 6
      045514 045720          .WORD    QDRV7        ;DRIVE 7
1271
1272 045516 045540          QSTART: .WORD    QDRV0        ;DRIVE 0 START ADDRESS
1273 045520 045560          QSTOP:  .WORD    QDRV1        ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1274 045522 045600          .WORD    QDRV2        ;STOP DRIVE 1--START DRIVE 2
1275 045524 045620          .WORD    QDRV3        ;STOP DRIVE 2--START DRIVE 3
1276 045526 045640          .WORD    QDRV4        ;STOP DRIVE 3--START DRIVE 4
1277 045530 045660          .WORD    QDRV5        ;STOP DRIVE 4--START DRIVE 5
1278 045532 045700          .WORD    QDRV6        ;STOP DRIVE 5--START DRIVE 6
1279 045534 045720          .WORD    QDRV7        ;STOP DRIVE 6--START DRIVE 7
1280 045536 045720          .WORD    QTERM        ;STOP DRIVE 7

```

```

1281
1282
1283
1286 045540 QDRV0: .BLKW 10
      045560 QDRV1: .BLKW 10
      045600 QDRV2: .BLKW 10
      045620 QDRV3: .BLKW 10
      045640 QDRV4: .BLKW 10
      045660 QDRV5: .BLKW 10
      045700 QDRV6: .BLKW 10
      045720 QDRV7: .BLKW 10
1287      QTERM=.
      045740
1288
1289 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1290 ;
1291 ;CALL
1292 ; JSR PC,CLRQUE
1293
1294 045740 104412 CLRQUE: SAVREG ;SAVE R0 - R5
1295 045742 012702 045446 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
1296 045746 005022 CLR (R2)+ ;DRIVES 0 & 1
1297 045750 005022 CLR (R2)+ ;DRIVES 2 & 3
1298 045752 005022 CLR (R2)+ ;DRIVES 4 & 5
1299 045754 005022 CLR (R2)+ ;DRIVES 6 & 7
1300 045756 012703 000010 MOV #8,R3 ;MOVE THE STARTING
1301 045762 012701 045516 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
1302 045766 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
1303 045770 005303 DEC R3
1304 045772 001375 BNE 1$
1305 045774 012703 000010 MOV #8,R3 ;MOVE THE STARTING ADDRESS
1306 046000 012701 045516 MOV #QSTART,R1 ;OF THE QUEUE INTO THE
1307 046004 012122 2$: MOV (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
1308 046006 005303 DEC R3
1309 046010 001375 BNE 2$
1310 046012 104413 RESREG ;RESTORE R0 - R5
1311 046014 000207 RTS PC
1312
1313 ;EMPTY THE QUEUE SPECIFIED BY R1
1314 ;
1315 ;CALL
1316 ; MOV DRVNUM,R1 ;DRIVE NUMBER TO R1
1317 ; JSR PC,EMPTYQ
1318
1319 046016 105061 045446 EMPTYQ: CLRB QCNT(R1) ;CLEAR NUMBER OF ITEMS IN QUEUE
1320 046022 006301 ASL R1
1321 046024 016161 045456 045476 MOV QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER INPUT POINTER
1322 046032 006201 ASR R1
1323 046034 000207 RTS PC
1324
1325 ;ROUTINE TO PUT A REQUEST IN QUEUE
1326 ;
1327 ;CALL
1328 ; MOV #DRVNUM,R1 ;DRIVE NUMBER
1329 ; MOV #DPB,R2 ;ADDRESS OF PARAMETER BLOCK
1330 ; JSR RO,DRVQUE ;GO PUT REQUEST IN QUEUE
1331 ; RETURN1 ;RETURN HERE IF QUEUE IS FULL
1332 ; RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE

```

```

1333
1334 046036 122761 C00010 045446 DRVQUE: CMPB #10,QCNT(R1) ;IS QUEUE FULL?
1335 046044 001421 BEQ 2$ ;BR IF YES-TAKE RETURN1
1336 046046 105261 045446 INCB QCNT(R1) ;INCREMENT QUEUE COUNT
1337 046052 006301 ASL R1
1338 046054 010271 045456 MOV R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1339 046060 062761 000002 045456 ADD #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
1340 046066 026161 045456 045520 CMP QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1341 046074 001003 BNE 1$ ;BR IF NO
1342 046076 016161 045516 045456 MOV QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1343 046104 006201 1$: ASR R1
1344 046106 005720 TST (R0)+ ;TAKE RETURN 2
1345 046110 000200 2$: RTS R0 ;RETURN TO USER
1346
1347 ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1348 ;
1349 ;CALL
1350 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
1351 ; JSR PC,GETREQ ;GO GET THE REQUEST
1352 ; RETURN ;R2='DPB' ADDRESS OF THE REQUEST
1353 ; ;R2=0 IF NO REQUEST IN QUEUE
1354
1355 046112 005002 GETREQ: CLR R2
1356 046114 105761 045446 TSTB QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
1357 046120 001404 BEQ 2$ ;NO
1358 046122 006301 1$: ASL R1
1359 046124 017102 045476 MOV @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1360 046130 006201 ASR R1
1361 046132 000207 2$: RTS PC ;RETURN TO USER
1362
1363 ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1364 ;
1365 ;CALL
1366 ; MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
1367 ; JSR PC,POPQUE ;CALL TO REMOVE REQUEST
1368 ; RETURN ;R2=ADDRESS OF DPB REMOVED
1369
1370 046134 105361 045446 POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
1371 046140 006301 ASL R1
1372 046142 017102 045476 MOV @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1373 046146 005071 045476 CLR @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
1374 046152 062761 000002 045476 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
1375 046160 026161 045476 045520 CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1376 046166 001003 BNE 1$ ;NO--BR TO EXIT
1377 046170 016161 045516 045476 MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1378 046176 006201 1$: ASR R1
1379 046200 000207 2$: RTS PC ;RETURN TO USER
1380

```

.SBTTL DATA, CONTROL, & STATUS BLOCKS

;BLOCK LOCATION EQUATE STATEMENTS

1			
2			
3			
4			
5	000001	\$FMT = 1	;FMT,HCI,ECI OR OFFSET CODE
6	000002	\$COMND = \$FMT+1	;OPERATION CODE
7	000003	\$PSEL = \$FMT+2	;PORT SELECT & BITS A16, A17
8	000004	\$WCNT = \$FMT+3	;WORD COUNT (2'S COMP)
9	000006	\$BUF = \$FMT+5	;BUFFER ADDR OR REGISTER TABLE POINTER
10	000010	\$SEC = \$FMT+7	;SECTOR ADDRESS OR 1ST REG ADDR
11	000011	\$TRK = \$FMT+10	;TRACK ADDRESS OF LAST REG ADDR
12	000012	\$CYL = \$FMT+11	;CYLINDER ADDR
13	000014	\$REG = \$FMT+13	;REGISTER STORAGE (IF ERROR)
14	000016	\$STATUS = \$FMT+15	;STATUS WORD (SET BY DRIVER)

;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES

15			
16			
17			
18	000020	\$WRDL = \$FMT+17	;WORD COUNT LENGTH (POSITIVE)
19	000022	\$SSEC = \$WRDL+2	;SECTOR SIZE FOR CURRENT OPERATION (256. OR 258.)
20	000024	\$CODE = \$WRDL+4	;PRESENT COMMAND SELECTION CODE
21	000026	\$PACK = \$WRDL+6	;WRITE DATA PACK INDICATOR
22	000027	\$PREVO = \$WRDL+7	;PREVIOUS COMMAND SELECTION CODE
23	000030	\$PATTC = \$WRDL+10	;PATTERN CODE
24	000032	\$PREVA = \$WRDL+12	;PREVIOUS ADDRESS- TRK, SEC, CYL (DOUBLE WORD)
25	000036	\$ENDAT = \$WRDL+16	;END OF PASS DATA COUNT (DOUBLE WORD)
26	000042	\$ENDSK = \$WRDL+22	;END OF PASS SEEK COUNT (DOUBLE WORD)
27	000046	\$OPERC = \$WRDL+26	;OPERATION COUNT (DOUBLE WORD) PER PASS
28	000052	\$POSIT = \$WRDL+32	;SEEK COUNT (DOUBLE WORD)
29	000056	\$WTOFL = \$WRDL+36	;TOTAL WORDS WRITTEN OVERFLOW COUNT
30	000060	\$WRITN = \$WRDL+40	;TOTAL WORDS WRITTEN COUNT (DOUBLE WORD)
31	000064	\$RDOFL = \$WRDL+44	;TOTAL WORDS READ OVERFLOW COUNT
32	000066	\$READ = \$WRDL+46	;TOTAL WORDS READ COUNT (DOUBLE WORD)
33	000072	\$TOTAL = \$WRDL+52	;TOTAL ERRORS (ALL TYPES) COUNT
34	000074	\$SOFT = \$WRDL+54	; 'SOFT' ERROR COUNT
35	000076	\$HARD = \$WRDL+56	; 'HARD' ERROR COUNT
36	000100	\$SKI = \$WRDL+60	; 'SKI' ERROR COUNT
37	000102	\$MISPO = \$WRDL+62	;PROG DETECTED MIS-POSITIONING ERROR COUNT
38	000104	\$PASSC = \$WRDL+64	;PASS COUNTER
39	000106	\$FAIR = \$WRDL+66	;OPERATION QUEUE 'FAIRNESS' COUNT
40	000110	\$HLDWC = \$WRDL+70	;HOLD WORD FOR 'RELBUF' ROUTINE

;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS

41			
42			
43			
44	000112	\$NCODE = \$WRDL+72	;NEXT OPERATION CODE
45	000113	\$NPATC = \$NCODE+1	;NEXT PATTERN
46	000114	\$NSEC = \$NCODE+2	;NEXT SECTOR
47	000115	\$NTRK = \$NCODE+3	;NEXT TRACK
48	000116	\$NCYL = \$NCODE+4	;NEXT CYLINDER
49	000120	\$NEXT = \$NCODE+6	;PARAMETER SELECTION INDICATOR
50	000122	\$FIRST = \$NCODE+10	;FIRST OPERATION INDICATOR

;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES

51			
52			
53			
54	000124	MAXCYL = \$NCODE+12	;MAXIMUM CYLINDER ADDRESS
55	000126	MINCYL = MAXCYL+2	;MINIMUM CYLINDER ADDRESS
56	000130	MAXTRK = MAXCYL+4	;MAXIMUM TRACK ADDRESS
57	000132	MINTRK = MAXCYL+6	;MINIMUM TRACK ADDRESS


```

58          000134          MAXSEC = MAXCYL+10          ;MAXIMUM SECTOR ADDRESS
59          000136          MINSEC = MAXCYL+12          ;MINIMUM SECTOR ADDRESS
60
61          ;PACK SERIAL NUMBER CONTAINED IN DEC144 FILE
62
63          000140          $PSNL = MAXCYL+14          ;LSB'S OF SERIAL NUMBER (5 OCTAL DIGITS)
64          000142          $PSNM = $PSNL+2          ;MSB'S OF SERIAL NUMBER (5 OCTAL DIGITS)
65
66          ;DEC144 BAD SECTOR ADDRESS STORAGE AREA INDEX EQUATE
67
68          000144          $BDSEC = $PSNL+4          ;BAD SECTOR STORAGE TABLE PLUS TERMINATOR
69
70          ;DRIVE (MBA) SERIAL NUMBER AREA INDEX EQUATE
71
72          002126          $MBASN = $BDSEC+<126.*8.>+2 ;DRIVE (MBA) SERIAL NUMBER
73
74          ;RH/RM REGISTER EQUATES
75
76          002136          $RMCS1 = $MBASN+10          ;RM REGISTER STORAGE
77          002140          $RMWC = $RMCS1+2
78          002142          $RMBA = $RMCS1+4
79          002144          $RMDB = $RMCS1+6
80          002146          $RMCS2 = $RMCS1+10
81          002150          $RMDS = $RMCS1+12
82          002152          $RMER1 = $RMCS1+14
83          002154          $RMAS = $RMCS1+16
84          002156          $RMLA = $RMCS1+20
85          002160          $RMDB = $RMCS1+22
86          002162          $RMVR1 = $RMCS1+24
87          002164          $RMDT = $RMCS1+26
88          002166          $RMSN = $RMCS1+30
89          002170          $RMOF = $RMCS1+32
90          002172          $RMDC = $RMCS1+34
91          002174          $RMHR = $RMCS1+36
92          002176          $RMVR2 = $RMCS1+40
93          002200          $RMER2 = $RMCS1+42
94          002202          $RMEC1 = $RMCS1+44
95          002204          $RMEC2 = $RMCS1+46
96          002206          $RMBAE = $RMCS1+50
97          002210          $RMCS3 = $RMCS1+52
98
99
100
101
102          ;BLOCK FOR DRIVE 0
103          046202          000          000          DRIVE0: .BYTE 0,0          ;DRIVE NUMBER 0
104          046204          .BLKW 5
105          046216          050340          .WORD .+$RMCS1-$REG
106          046220          .BLKB $RMCS3-$REG
107
108          ;BLOCK FOR DRIVE 1
109          050414          001          000          DRIVE1: .BYTE 1,0          ;DRIVE NUMBER 1
110          050416          .BLKW 5
111          050430          052552          .WORD .+$RMCS1-$REG
112          050432          .BLKB $RMCS3-$REG
113
114          ;BLOCK FOR DRIVE 2
115          052626          002          000          DRIVE2: .BYTE 2,0          ;DRIVE NUMBER 2
116          052630          .BLKW 5
117          052642          054764          .WORD .+$RMCS1-$REG

```

```

052644          .BLKB  $RMCS3-$REG

055040      003      000      ;BLOCK FOR DRIVE 3
055042      057176      DRIVE3: .BYTE  3,0          ;DRIVE NUMBER 3
055054      .BLKW  5
055056      .WORD  .+$RMCS1-$REG
          .BLKB  $RMCS3-$REG

057252      004      000      ;BLOCK FOR DRIVE 4
057254      061410      DRIVE4: .BYTE  4,0          ;DRIVE NUMBER 4
057266      .BLKW  5
057270      .WORD  .+$RMCS1-$REG
          .BLKB  $RMCS3-$REG

061464      005      000      ;BLOCK FOR DRIVE 5
061466      063622      DRIVE5: .BYTE  5,0          ;DRIVE NUMBER 5
061500      .BLKW  5
061502      .WORD  .+$RMCS1-$REG
          .BLKB  $RMCS3-$REG

063676      006      000      ;BLOCK FOR DRIVE 6
063700      066034      DRIVE6: .BYTE  6,0          ;DRIVE NUMBER 6
063712      .BLKW  5
063714      .WORD  .+$RMCS1-$REG
          .BLKB  $RMCS3-$REG

066110      007      000      ;BLOCK FOR DRIVE 7
066112      070246      DRIVE7: .BYTE  7,0          ;DRIVE NUMBER 7
066124      .BLKW  5
066126      .WORD  .+$RMCS1-$REG
          .BLKB  $RMCS3-$REG

113
114      ;GENERAL PURPOSE PARAMETER BLOCK
115
116 070322      000      GENDPB: .BYTE  0          ;DRIVER PARAMETER BLOCK, DRIVE #
117 070323      000      .BYTE  0          ;OFFSET VALUE OR FMI16, HCI OR ECI
118 070324      000      .BYTE  0          ;COMMAND CODE
119 070325      000      .BYTE  0          ;PSEL, A16 AND A17
120 070326      177776      .WORD  -2          ;WORD COUNT (NEG)
121 070330      101066      .WORD  CYLNDR      ;BUFFER ADDRESS
122 070332      000      .BYTE  0          ;SECTOR ADDRESS
123 070333      000      .BYTE  0          ;TRACK ADDRESS
124 070334      000000      .WORD  0          ;CYLINDER ADDRESS
125 070336      070342      .WORD  GENREG      ;ADDRESS TO SAVE ALL RH/RM REG'S
126 070340      000000      .WORD  0          ;STATUS WORD
127
128 070342      GENREG: .BLKW  24          ;REGISTER STORAGE

```

				.SBTTL	ERROR MESSAGES
1					
4	070412	122	110	040	EM1: .ASCIZ /RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/
	070464	125	116	105	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
5	070522	115	101	123	EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
6	070560	115	101	123	EM4: .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
7	070615	101	104	104	EM5: .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
8	070651	122	110	040	EM6: .ASCIZ /RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/
9	070724	125	116	103	EM10: .ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
10	070767	106	101	124	EM11: .ASCIZ /FATAL MASSBUS PARITY ERROR/
11	071022	120	105	122	EM12: .ASCIZ /PERSISTENT DEVICE UNSAFE/
12	071053	117	120	105	EM13: .ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13	071125	104	122	111	EM14: .ASCIZ /DRIVE WENT OFFLINE/
14	071150	116	117	040	EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
15	071204	110	105	101	EM20: .ASCIZ /HEADER CRC ERROR/
16	071225	104	101	124	EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
17	071256	127	122	111	EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18	071331	127	122	111	EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19	071410	110	105	101	EM24: .ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
20	071456	110	105	101	EM25: .ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21	071537	106	117	122	EM26: .ASCIZ /FORMAT ERROR ('FER')/
22	071564	110	105	101	EM27: .ASCIZ /HEADER COMPARE ('HCE') ERROR/
23	071621	115	111	123	EM30: .ASCIZ /MISCELLANEOUS DRIVE ERROR/
24	071653	117	120	105	EM31: .ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
25	071716	104	122	111	EM32: .ASCIZ /DRIVE TIMING ('DTE') ERROR/
26	071751	120	101	122	EM33: .ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27	072026	127	122	111	EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
28	072070	111	116	126	EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
29	072126	127	122	111	EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
30	072157	104	101	124	EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
31	072241	122	110	040	EM40: .ASCIZ /RH CONTROLLER OR UNIBUS TRANSFER ERROR/
32	072310	102	125	123	EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
33	072354	104	101	124	EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34	072435	103	101	116	EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
35	072502	105	122	122	EM44: .ASCIZ /ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH CONTROLLER/
36	072577	105	103	103	EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37	072665	102	125	123	EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38	072737	123	105	105	EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') ERROR/
39	072775	120	122	117	EM51: .ASCIZ /PROGRAM DETECTED POSITIONING ERROR/
40	073040	104	122	111	EM60: .ASCIZ /DRIVE UNSAFE ERROR/

1	073530	001322	000000		DT1:	.WORD	ATTN,0
2	073534	001220	040016	040020	DT2:	.WORD	DRIVE, RMERRS, RMERRS+2, RMERRS+4, RMERRS+6, ATTN, 0
3	073552	001220	044712	044714	DT3:	.WORD	DRIVE, RD. ADR, RD. WRD, 0
4	073562	001220	045132	045130	DT4:	.WORD	DRIVE, WRT. AD, WRT. WD, RD. WRD, 0
5	073574	001272	000000		DT6:	.WORD	\$RMADR, 0
6	073600	002136	002146	002150	DT14:	.WORD	\$RMCS1, \$RMCS2, \$RMDS, \$RMER1, \$RMMR2, \$RMER2, \$RMEC1, \$RMEC2, 0
7	073622	002140	002142	002144	DT15:	.WORD	\$RMWC, \$RMBA, \$RMDA, \$RMAS, \$RMLA, \$RMDB, \$RMMR1, \$RMDT, 0
8	073644	002166	002170	002172	DT16:	.WORD	\$RMSN, \$RMOF, \$RMDC, \$RMHR, \$STATUS, 0
9	073660	002206	002210	000000	DT17:	.WORD	\$RMBAE, \$RMCS3, 0

1	073666	120	122	123	LIN2C:	.ASCIZ	/PRSNT COMMAND= /
2	073706	040	040	120	LIN2P:	.ASCIZ	/ PREV COMMAND= /
3	073727	052	040	105	LIN2S:	.ASCIZ	@* ERROR AT BAD TRACK/SECTOR@
4	073763	105	122	122	LINM3:	.ASCIZ	/ERROR AT C/
5	073776	040	124	000	T:	.ASCIZ	/ T/
6	074001	120	122	123	LINW3:	.ASCIZ	/PRSNT ADDR= C/
7	074017	040	123	000	S:	.ASCIZ	/ S/
8	074022	040	040	120	LINP3:	.ASCIZ	/ PREV ADDR= C/
9	074041	123	124	101	LINS3:	.ASCIZ	/START CYL= /
10	074055	040	040	105	LINEN3:	.ASCIZ	/ END CYL= /
11	074071	040	040	101	LINA3:	.ASCIZ	/ ACTUAL CYL= /
12	074110	040	040	124	LINT3:	.ASCIZ	/ TRK= /
13	074120	040	040	122	LINCA3:	.ASCIZ	/ RMDC= /
14	074131	122	115	104	LINDA3:	.ASCIZ	/RMDA= /
15	074140	122	115	102	LINB3:	.ASCIZ	/RMDA= /
16	074147	040	040	122	LINW3:	.ASCIZ	/ RMWC= /
17	074160	123	124	101	LINST3:	.ASCIZ	/START TRK= /
18	074174	123	124	101	LINSS3:	.ASCIZ	/START SEC= /
19	074210	102	125	106	LINM4:	.ASCIZ	/BUFFER ADDR= /
20	074226	040	040	127	LINS4:	.ASCIZ	/ WRD CNT= /
21	074242	040	040	101	LINX4:	.ASCIZ	/ ACTUAL NMBR WRDS XFRD= /
22	074274	107	117	117	LIND5:	.ASCIZ	/GOOD DATA= /
23	074310	040	040	102	LINB5:	.ASCIZ	/ BAD DATA= /
24	074325	040	040	127	LINP5:	.ASCIZ	/ WORD POS= /
25	074342	110	105	101	LINS5:	.ASCIZ	/HEADER FROM ERROR SECTOR= /
26	074375	122	115	105	LINEP5:	.ASCIZ	/RMEC1= /
27	074405	040	040	122	LINEO5:	.ASCIZ	/ RMEC2= /
28	074417	123	105	103	LINB6:	.ASCIZ	/SECTOR IS ECC CORRECTABLE /
29	074452	123	105	103	LINC6:	.ASCIZ	/SECTOR READ CORRECTLY /
30	074501	103	117	122	LING6:	.ASCIZ	/CORRECTED ON /
31	074517	040	040	122	LINR6:	.ASCIZ	/ RETRIES/
32	074531	125	116	103	LINUO6:	.ASCIZ	/UNCORRECTABLE AFTER /
33	074556	040	040	124	LIN7M:	.ASCIZ	/ TOTAL MISPOS ERR= /
37	074603	124	117	124	LIN7P:	.ASCIZ	/TOTAL SEEKS= /
38	074621	040	040	124	LIN7S:	.ASCIZ	/ TOTAL SKI ERR= /
39	074643	124	117	124	LIN7T:	.ASCIZ	/TOTAL ERRORS:/
40	074661	040	040	127	LIN7OX:	.ASCIZ	/ WOFL:/
41	074671	040	127	122	LIN7X:	.ASCIZ	/ WRDS WRITN:/
42	074706	040	040	122	LIN7OR:	.ASCIZ	/ ROFL:/
43	074716	040	127	122	LIN7R:	.ASCIZ	/ WRDS READ:/
44	074732	105	122	122	LIN8M:	.ASCIZ	/ERROR DURING RETRY/
45	074755	104	101	124	LIN9B:	.ASCIZ	/DATA COMPARISON ERRORS/
46	075004	040	040	040	LIN9H:	.ASCIZ	/ GOOD DATA/ <CRLF>
47	075031	114	117	103	LIN9I:	.ASCIZ	/LOC DATA DATA/ <CRLF>
48	075060	114	117	103	LIN9J:	.ASCIZ	/LOC DATA/ <CRLF>
49	075077	124	117	124	LIN9E:	.ASCIZ	/TOTAL COMPARE ERRORS= /
50	075126	124	110	105	LIN9G:	.ASCIZ	/THE DATA COMPARED OK/ <CRLF>
51	075154	105	122	122	LIN10A:	.ASCIZ	/ERROR BURST BEGINS AT WORD /
52	075210	040	111	116	LIN10B:	.ASCIZ	/ IN DATA FIELD OF ERROR SECTOR/ <CRLF>
53	075250	105	122	122	LIN10C:	.ASCIZ	/ERROR WAS NOT IN THE DATA READ - / <CRLF>
54	075312	105	103	103	LIN10D:	.ASCIZ	/ECC CORRECTION CAN'T BE PERFORMED/
55	075354	105	103	103	LIN10H:	.ASCIZ	/ECC CORRECTION RESULTS/ <CRLF>
56	075403	101	104	104	LIN10I:	.ASCIZ	/ADDR BAD CORRECTED / <CRLF>
57	075437	103	117	116	LIN11H:	.ASCIZ	/CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/ <CRLF>
58	075512	101	104	104	LIN11:	.ASCIZ	/ADDR HEADER/ <CRLF>
59	075533	101	104	104	LIN11A:	.ASCIZ	/ADDR DATA/ <CRLF>
60	075552	040			BLNKS4:	.ASCIZ	/ /

ZRMUBO RM05/3/2 PERF EXER
ERROR MESSAGES

MACRO V04.00 4-APR-81 01:42:23 PAGE 59-1

SEQ 0207

61	075553	040	
62	075554	040	
63	075555	040	000
64	075557	122	105

	BLNKS3:	.ASCII	//
	BLNKS2:	.ASCII	//
	BLNKS1:	.ASCII	//
124	LINK5:	.ASCII	/RETRIEVED BY A RDHD COMMAND/

```

1
2
3 075613 077 000 .SBTTL TELETYPE MESSAGES
4 075615 075 000 QUES: .ASCIZ /?/
5 075617 054 000 EQUAL: .ASCIZ /=/
6 075621 055 000 COMMA: .ASCIZ /,/
7 075623 011 000 DASH: .ASCIZ /-/
8 075625 104 122 111 TAB: .ASCIZ <11>
9 075633 040 117 106 UNTMSG: .ASCIZ /DRIVE/
10 075644 040 117 116 UNTOFF: .ASCIZ / OFFLINE/
11 075654 040 116 116 UNTON: .ASCIZ / ONLINE/
12 075676 040 101 117 UNTNOT: .ASCIZ / NOT BEING TESTED/
13 075724 040 116 114 UNTASN: .ASCIZ / ALREADY BEING TESTED/
14 075745 040 116 117 NOTRM: .ASCIZ @ NOT AN RM05/3/2@
15 075762 040 116 117 NOTPRS: .ASCIZ / NOT PRESENT/
16 076001 040 125 117 NOTAVL: .ASCIZ / NOT AVAILABLE/
17 076011 040 114 116 NOISAF: .ASCIZ / UNSAFE/
18 076026 200 125 117 LODEV: .ASCIZ / LOAD DEVICE/
19 076044 122 115 116 SYSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
20 076051 122 115 060 $RM02: .ASCIZ /RM02/
21 076056 122 115 060 $RM03: .ASCIZ /RM03/
22 076063 200 012 060 $RM05: .ASCIZ /RM05/
23 076124 052 052 052 REPHD: .ASCIZ <CRLF><LF>/***** PERFORMANCE REPORT *****/
24 076155 052 052 052 STAR30: .ASCIZ /*****/
25 076164 200 104 122 STAR5: .ASCIZ /*****/<CRLF>
26 076236 007 077 106 SUMHD: .ASCIZ <CRLF>/DRIVE SUMMARY, (OFLOW= 2,147,483,647.)/<CRLF>
27 076272 200 105 106 DROPNG: .ASCIZ <07>/? FATAL OR EXCESSIVE ERRORS/
28 076311 040 117 116 ENDPAS: .ASCIZ <CRLF>/END OF PASS #/
29 076316 200 105 116 MSGON: .ASCIZ / ON /
30 076334 106 117 116 ENDTST: .ASCIZ <CRLF>/END OF TEST /
31 076341 200 052 052 MSGFOR: .ASCIZ /FOR /
32 076372 200 052 052 DEASSG: .ASCIZ <CRLF>/***** DRIVE DEASSIGNED/<CRLF>
33 076411 040 123 052 DRNUM: .ASCIZ <CRLF>/***** DRIVE #/
34 076422 200 007 124 ASGND: .ASCIZ / STARTED/
35 076473 116 000 077 NEDCLK: .ASCIZ <CRLF><07>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
36 076475 131 000 N: .ASCIZ /N/
37 076477 056 000 Y: .ASCIZ /Y/
38 076501 040 077 103 PERIOD: .ASCIZ /./
39 076544 040 077 111 MSWRO: .ASCIZ / ?CAN'T WRITE IN 'READ ONLY' MODE/<CRLF>
40 076567 200 105 116 INVLD: .ASCIZ / ?INVALID COMMAND/<CRLF>
41 076610 200 105 116 ENTCOM: .ASCIZ <CRLF>/ENTER COMMAND: /
42 076640 200 105 116 ENTLMT: .ASCIZ <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
43 076670 072 000 116 ENTADR: .ASCIZ <CRLF>/ENTER BAD SECTR ADRS:/<CRLF>
44 076672 116 117 116 COLON: .ASCIZ /:/
45 076677 040 077 116 NONE: .ASCIZ /NONE/
46 076720 200 106 111 BADENT: .ASCIZ / ?INVALID ENTRY/<CRLF>
47 077000 200 111 101 MERR1: .ASCIZ <CRLF>/FAILED TO RETRIEVE BAD SECTOR FILE(DEC144) ON /
48 077044 052 040 116 MERR2: .ASCIZ <CRLF>/INVALID FILE(DEC144) STRUCTURE ON /
49 077102 103 131 102 MSFULL: .ASCIZ /* BAD SECTOR TABLE IS FULL */<CRLF>
50 077120 200 012 114 MSGCTS: .ASCIZ /CYL,TRK,SEC= /
51 077146 052 040 116 NODRVS: .ASCIZ <CRLF><LF>/NO DRIVES ASSIGNED/<CRLF>
52 077166 200 104 116 NOENTY: .ASCIZ /* NO ENTRIES */<CRLF>
53 077232 052 040 105 LSTHDR: .ASCIZ <CRLF>/DEC144 AND MANUAL BAD SECTOR LIST/<CRLF>
54 077270 120 101 101 ALOST: .ASCIZ /* ALL CURRENT ENTRIES LOST */<CRLF>
55 077303 115 102 103 PACKSN: .ASCIZ @PACK S/N: @
56 077315 200 103 101 MBASN: .ASCIZ @MBA S/N: @
57 .EVEN MSPRM: .ASCIZ <CRLF>/CHANGE DRIVE PARAMETERS (L) N ? /

```


:PARAMETER ENTRY TABLE

```

1
2
3 077360 077524 020000 001450 PARLST: .WORD PAR1,8192,WRDCNT
4 077366 077534 077777 001452 .WORD PAR2,32767,INTRVL
5 077374 077704 077777 001456 .WORD PAR19,32767,PASSES
6 077402 077544 000017 001460 .WORD PAR3,15,PATTERN
7 077410 077644 000001 001462 .WORD PAR11,1,RANDWC
8 077416 077654 000007 001464 .WORD PAR14,7,RATIO
9 077424 077674 000001 001466 .WORD PAR16,1,ENDING
10 077432 077664 000001 001470 .WORD PAR15,1,WRCHK
11 077440 077714 000001 001472 .WORD PAR20,1,MESSAGE
12 077446 077724 000001 001474 .WORD PAR21,1,RANDOM
13 077454 077634 000001 001476 .WORD PAR10,1,BADBLK
14 077462 000000 .WORD 0 ;TABLE TERMINATOR

```

```

15
16 077464 040 057 040 SLASH: .ASCIZ @ / @
17 077470 200 103 110 ASKPAR: .ASCIZ <CRLF>/CHANGE PARAMETERS (L) N ? /
18 077524 127 122 104 PAR1: .ASCIZ /WRDCNT /
19 077534 111 116 124 PAR2: .ASCIZ /INTRVL /
20 077544 120 101 124 PAR3: .ASCIZ /PATTERN/
21 077554 115 101 130 PAR4: .ASCIZ /MAXCYL /
22 077564 115 111 116 PAR5: .ASCIZ /MINCYL /
23 077574 115 101 130 PAR6: .ASCIZ /MAXTRK /
24 077604 115 111 116 PAR7: .ASCIZ /MINTRK /
25 077614 115 101 130 PAR8: .ASCIZ /MAXSEC /
26 077624 115 111 116 PAR9: .ASCIZ /MINSEC /
27 077634 102 101 104 PAR10: .ASCIZ /BADBLK /
28 077644 122 101 116 PAR11: .ASCIZ /RANDWC /
29 077654 122 101 124 PAR14: .ASCIZ /RATIO /
30 077664 127 122 124 PAR15: .ASCIZ /WRCHK /
31 077674 105 116 104 PAR16: .ASCIZ /ENDING /
32 077704 120 101 123 PAR19: .ASCIZ /PASSES /
33 077714 115 105 123 PAR20: .ASCIZ /MESSAGE/
34 077724 122 101 116 PAR21: .ASCIZ /RANDOM /

```

.EVEN

:PARAMETER TABLE POINTERS FOR ADDRESS LIMITS

```

35
36
37
38
39 077734 077754 TABLE: .WORD TABLE0 ;PARAMETER TABLE FOR DRIVE 0
42 077736 100022 .WORD TABLE1 ;PARAMETER TABLE FOR DRIVE 1
077740 100070 .WORD TABLE2 ;PARAMETER TABLE FOR DRIVE 2
077742 100136 .WORD TABLE3 ;PARAMETER TABLE FOR DRIVE 3
077744 100204 .WORD TABLE4 ;PARAMETER TABLE FOR DRIVE 4
077746 100252 .WORD TABLE5 ;PARAMETER TABLE FOR DRIVE 5
077750 100320 .WORD TABLE6 ;PARAMETER TABLE FOR DRIVE 6
077752 100366 .WORD TABLE7 ;PARAMETER TABLE FOR DRIVE 7

```

:PARAMETER TABLE FOR ADDRESS LIMITS

```

43
44
54 077754 077564 000000 046330 TABLE0: .WORD PAR5,0,MINCYL+DRIVE0
077762 077554 000000 046326 .WORD PAR4,0,MAXCYL+DRIVE0
077770 077604 000000 046334 .WORD PAR7,0,MINTRK+DRIVE0
077776 077574 000000 046332 .WORD PAR6,0,MAXTRK+DRIVE0
100004 077624 000036 046340 .WORD PAR9,30,MINSEC+DRIVE0
100012 077614 000036 046336 .WORD PAR8,30,MAXSEC+DRIVE0
100020 000000 .WORD 0 ;TERMINATOR

100022 077564 000000 050542 TABLE1: .WORD PAR5,0,MINCYL+DRIVE1

```

100030	077554	000000	050540	.WORD	PAR4,0,MAXCYL+DRIVE1
100036	077604	000000	050546	.WORD	PAR7,0,MINTRK+DRIVE1
100044	077574	000000	050544	.WORD	PAR6,0,MAXTRK+DRIVE1
100052	077624	000036	050552	.WORD	PAR9,30.,MINSEC+DRIVE1
100060	077614	000036	050550	.WORD	PAR8,30.,MAXSEC+DRIVE1
100066	000000			.WORD	0 ;TERMINATOR
100070	077564	000000	052754	TABLE2: .WORD	PAR5,0,MINCYL+DRIVE2
100076	077554	000000	052752	.WORD	PAR4,0,MAXCYL+DRIVE2
100104	077604	000000	052760	.WORD	PAR7,0,MINTRK+DRIVE2
100112	077574	000000	052756	.WORD	PAR6,0,MAXTRK+DRIVE2
100120	077624	000036	052764	.WORD	PAR9,30.,MINSEC+DRIVE2
100126	077614	000036	052762	.WORD	PAR8,30.,MAXSEC+DRIVE2
100134	000000			.WORD	0 ;TERMINATOR
100136	077564	000000	055166	TABLE3: .WORD	PAR5,0,MINCYL+DRIVE3
100144	077554	000000	055164	.WORD	PAR4,0,MAXCYL+DRIVE3
100152	077604	000000	055172	.WORD	PAR7,0,MINTRK+DRIVE3
100160	077574	000000	055170	.WORD	PAR6,0,MAXTRK+DRIVE3
100166	077624	000036	055176	.WORD	PAR9,30.,MINSEC+DRIVE3
100174	077614	000036	055174	.WORD	PAR8,30.,MAXSEC+DRIVE3
100202	000000			.WORD	0 ;TERMINATOR
100204	077564	000000	057400	TABLE4: .WORD	PAR5,0,MINCYL+DRIVE4
100212	077554	000000	057376	.WORD	PAR4,0,MAXCYL+DRIVE4
100220	077604	000000	057404	.WORD	PAR7,0,MINTRK+DRIVE4
100226	077574	000000	057402	.WORD	PAR6,0,MAXTRK+DRIVE4
100234	077624	000036	057410	.WORD	PAR9,30.,MINSEC+DRIVE4
100242	077614	000036	057406	.WORD	PAR8,30.,MAXSEC+DRIVE4
100250	000000			.WORD	0 ;TERMINATOR
100252	077564	000000	061612	TABLE5: .WORD	PAR5,0,MINCYL+DRIVE5
100260	077554	000000	061610	.WORD	PAR4,0,MAXCYL+DRIVE5
100266	077604	000000	061616	.WORD	PAR7,0,MINTRK+DRIVE5
100274	077574	000000	061614	.WORD	PAR6,0,MAXTRK+DRIVE5
100302	077624	000036	061622	.WORD	PAR9,30.,MINSEC+DRIVE5
100310	077614	000036	061620	.WORD	PAR8,30.,MAXSEC+DRIVE5
100316	000000			.WORD	0 ;TERMINATOR
100320	077564	000000	064024	TABLE6: .WORD	PAR5,0,MINCYL+DRIVE6
100326	077554	000000	064022	.WORD	PAR4,0,MAXCYL+DRIVE6
100334	077604	000000	064030	.WORD	PAR7,0,MINTRK+DRIVE6
100342	077574	000000	064026	.WORD	PAR6,0,MAXTRK+DRIVE6
100350	077624	000036	064034	.WORD	PAR9,30.,MINSEC+DRIVE6
100356	077614	000036	064032	.WORD	PAR8,30.,MAXSEC+DRIVE6
100364	000000			.WORD	0 ;TERMINATOR
100366	077564	000000	066236	TABLE7: .WORD	PAR5,0,MINCYL+DRIVE7
100374	077554	000000	066234	.WORD	PAR4,0,MAXCYL+DRIVE7
100402	077604	000000	066242	.WORD	PAR7,0,MINTRK+DRIVE7
100410	077574	000000	066240	.WORD	PAR6,0,MAXTRK+DRIVE7
100416	077624	000036	066246	.WORD	PAR9,30.,MINSEC+DRIVE7
100424	077614	000036	066244	.WORD	PAR8,30.,MAXSEC+DRIVE7
100432	000000			.WORD	0 ;TERMINATOR

.SBTTL ROUTINE TO SIZE MEMORY

:CALL:
:JSR PC,\$SIZE
:RETURN
:*\$LSTAD WILL CONTAIN THE LAST AVAILABLE MEMORY LOCATION

100434	010046			\$SIZE: MOV	R0,-(SP)	::SAVE R0 ON THE STACK
100436	010146			MOV	R1,-(SP)	::SAVE R1 ON THE STACK
100440	013746	000114		MOV	@#114,-(SP)	::SAVE MEMORY ERROR VECTOR PS & PC
100444	013746	000116		MOV	@#116,-(SP)	
100450	012737	000116	000114	MOV	#116,@#114	::IGNORE PARITY ERRORS WHILE SIZING
100456	012737	000002	000116	MOV	#RTI,@#116	
100464	013746	000004		MOV	@ERRVEC,-(SP)	::SAVE PRESENT ERROR VECTOR PS & PC
100470	013746	000006		MOV	@ERRVEC+2,-(SP)	
100474	010600			MOV	SP,R0	::SAVE THE STACK POINTER
				::SET THE ERRVEC PS TO THE PRESENT PS		
100476	104400			TRAP		::PUSH OLD PSW AND PC ON STACK
100500	012637	000006		MOV	(SP)+,@ERRVEC+2	::SAVE THE PSW IN @ERRVEC+2
100504	012737	100524	000004	MOV	#2,@ERRVEC	::SET FOR TIMEOUT
100512	012701	020000		MOV	#20000,R1	::FIRST ADDRESS
100516	005711			1\$: TST	(R1)	::TEST THIS ADDRESS
100520	005721			TST	(R1)+	::STEP TO NEXT ADDRESS
100522	000775			BR	1\$::TRY ANOTHER
100524	162701	000002		2\$: SLB	#2,R1	::DROP BACK
100530	010006			MOV	R0,SP	::RESTORE THE STACK
100532	012637	000006		MOV	(SP)+,@ERRVEC+2	::RESTORE ERROR VECTOR
100536	012637	000004		MOV	(SP)+,@ERRVEC	
100542	012637	000116		MOV	(SP)+,@#116	::RESTORE MEMORY ERROR VECTOR
100546	012637	000114		MOV	(SP)+,@#114	
100552	010137	100564		MOV	R1,\$LSTAD	::LAST ADDRESS
100556	012601			MOV	(SP)+,R1	::RESTORE R1
100560	012600			MOV	(SP)+,R0	::RESTORE R0
100562	000207			RTS	PC	
100564	000000			\$LSTAD: .WORD	0	::CONTAINS THE LAST ADDRESS

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

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS

: THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS OF THE RH/RM
 : IS SETUP FOR THE PROPER ADDRESS. IT WILL ALSO READ THE ADDRESS
 : FROM THE TTY IF REQUIRED.

: NOTE: THIS ROUTINE DESTROYS R0-R4
 : CALL:

: JSR PC,BUSADR
 : RETURN

```

BUSADR: TST     CHGADR      ; INPUT FROM TTY REQUESTED?
        BGE     3$        ; NO--BRANCH
        CLR     CHGADR      ; YES--CLEAR THE REQUEST FLAG
        TYPE   , $CRLF     ; TYPE A CR-LF
        INC     #-1        ; FIRST TIME THRU ?
        BEQ     1$        ; BR IF YES
        TYPE   , $CRLF     ; CR-LF
1$:     CLR     CFLAG      ; CLEAR CONTROL C FLAG
        MOV     # $RMADR,R0 ; FIRST ADDRESS
        TYPE   , MRMCS1    ; 'RMCS1='
        MOV     (R0),-(SP) ; PRESENT RMCS1 ADDRESS
        TYPE   , BLNKS2    ; TYPE IT
        TYPE   , BLNKS2    ; TYPE 2 BLANKS
        RDLIN                      ; GET THE ENTRY
        MOV     (SP)+,R1    ; ADDRESS OF ASCII TEXT
        TST     CFLAG      ; WAS IT CONTROL C ?
        BNE     1$        ; BR IF YES
        JSR     R5,CK.NUM   ; ENTER AND STORE THE NEW ADDRESS
        BR     1$        ; ERROR EXIT
2$:     MOV     # $RMVEC,R0 ; VECTOR ADDRESS
        TYPE   , MRMVEC    ; 'RMVEC='
        MOV     (R0),-(SP) ; PRESENT RH/RM VECTOR ADDRESS ON THE STACK
        TYPE   , BLNKS2    ; TYPE IT
        TYPE   , BLNKS2    ; TYPE 2 BLANKS
        RDLIN                      ; READ THE ENTRY
        MOV     (SP)+,R1    ; ASCII TEXT ADDRESS
        TST     CFLAG      ; WAS IT CONTROL C ?
        BNE     1$        ; BR IF YES
        JSR     R5,CK.NUM   ; ENTER AND STORE NEW ADDRESS
        BR     2$        ; ERROR EXIT
3$:     MOV     # $RMADR,R0 ; FIRST ADDRESS OF NEW PARAMETERS
        MOV     # $RMADR,R1 ; FIRST ADDRESS OF WHERE TO PUT THEM
        MOV     (R0)+,(R1)+ ; BUS ADDRESS
        MOV     (R0)+,(R1)+ ; VECTOR ADDRESS
        RTS     PC         ; RETURN
    
```

103 MRMCS1: .ASCIZ @RMCS1=@
 126 MRMVEC: .ASCIZ @RMVEC=@

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

100756 010246
100760 010346
100762 010446
100764 012703 000006
100770 005002
100772 112104
100774 001424
100776 120427 000060
101002 103425
101004 120427 000067
101010 101022
101012 006302
101014 103420
101016 006302
101020 103416
101022 006302
101024 103414
101026 042704 177770
101032 060402
101034 005303
101036 001401
101040 000754
101042 112104
101044 001004
101046 005702
101050 001401
101052 010210
101054 005725
101056 012604
101060 012603
101062 012602
101064 000205
101066
102072
000200

```
.SBTTL CK.NUM - CHECK NUMBER (OCTAL)
;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
;AND FORMS AN OCTAL NUMBER IN R2
;CALL:
;      MOV      #ADR,R0      ;ADDRESS TO PLACE NEW NUMBER
;      MOV      #ADR,R1      ;ADDRESS OF ASCII STRING
;      JSR      R5,CK.NUM     ;R5 CHANGED
;      RET      ;ERROR EXIT
;      RET      ;NORMAL EXIT

CK.NUM: MOV      R2,-(SP)     ;SAVE R2
        MOV      R3,-(SP)     ;SAVE R3
        MOV      R4,-(SP)     ;SAVE R4
        MOV      #6,R3        ;MAX OCTAL DIGITS IN THE NUMBER
        CLR      R2           ;FINAL OCTAL VALUE
1$:     MOVB     (R1)+,R4      ;GET CURRENT POINTED BYTE
        BEQ      3$           ;BRANCH,IF TERMINATOR DETECTED
        CMPB     R4,#'0       ;SMALLER THAN ASCII-0 ?
        BLO      5$           ;YES,ERROR EXIT
        CMPB     R4,#'7       ;LARGER THAN ASCII-7 ?
        BHI      5$           ;YES,ERROR EXIT
        ASL      R2           ;SHIFT LEFT
        BCS      5$           ;
        ASL      R2           ;ONE
        BCS      5$           ;
        ASL      R2           ;OCTAL DIGIT
        BCS      5$           ;ERROR IF CARRY BIT SET
        BIC      #177770,R4   ;CHOP OFF HIGHER BITS
        ADD      R4,R2        ;APPENDING CURRENT DIGIT TO NUMBER
        DEC      R3           ;DECREMENT BYTE COUNT
        BEQ      2$           ;BRANCH,IF LAST BYTE
        BR       1$          ;LOOPING BACK
2$:     MOVB     (R1)+,R4      ;CHECK TERMINATOR
        BNE      5$           ;ERROR EXIT
3$:     TST      R2           ;FINAL VALUE= 0
        BEQ      4$           ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
        MOV      R2,(R0)      ;REPLACE THE ORIGINAL VALUE
4$:     TST      (R5)+         ;ADJUST FOR NORMAL RETURN
5$:     MOV      (SP)+,R4      ;RESTORE R4
        MOV      (SP)+,R3      ;RESTORE R3
        MOV      (SP)+,R2      ;RESTORE R2
        RTS      R5           ;EXIT

CYLNR:  .BLKW   258.         ;ONE SECTOR WORD CTR MAX SIZE

ENDPGM=.

.END      200
```

ABASE = 176700	ASGN6 026254	BIT5 = 000040	CMSTR2 013602	DRIVE3 055040
ABNRML 031134	ASGN7 026266	BIT6 = 000100	CMTRK 001375	DRIVE4 057252
ACDW1 = 000000	ASKPAR 077470	BIT7 = 000200	COLON 076670	DRIVE5 061464
ACDW2 = 000000	ASNERR 031024	BIT8 = 000400	COMMA 075617	DRIVE6 063676
ACK = 000123	ASNLST 001530	BIT9 = 001000	COMTBL 002064	DRIVE7 066110
ACPUOP= 000000	ASNMSG 031050	BLKADR 002044	CPSAVE 035124	DRNUM 076372
ACTDRV 040102	ASSIGN 025624	BLNKS1 075555	CR = 000015	DROP 031054
ACTSTR 040103	ASWREG= 000000	BLNKS2 075554	CRLF = 000200	DROPNG 076236
ADDW0 = 000000	ATA = 100000	BLNKS3 075553	CTRAP 034474	DRO = 004000
ADDW1 = 000000	ATABIT 040142	BLNKS4 075552	CYLIMT 001422	DRVACT 040026
ADDW10= 000000	AATESTN= 000000	BPTVEC= 000014	CYLNDR 101066	DRVCLR= 000111
ADDW11= 000000	ATTN 001322	BSE = 100000	DASH 075621	DRVER 011256
ADDW12= 000000	ATO = 000001	BUFTBL 001642	DATAPK 026552	DRVINT 040400
ADDW13= 000000	AT1 = 000002	BUSADR 100566	DATA0 002472	DRVPAR 001430
ADDW14= 000000	AT2 = 000004	CFLAG 001340	DATA1 002532	DRVPRM 027020
ADDW15= 000000	AT3 = 000010	CHGADR 001336	DATA10 003172	DRVQUE 046036
ADDW2 = 000000	AT4 = 000020	CHKWC 020316	DATA11 003232	DRVSTA 040036
ADDW3 = 000000	AT5 = 000040	CI1 041434	DATA12 003272	DRV TYP 040046
ADDW4 = 000000	AT6 = 000100	CI3 041542	DATA13 003332	DRY = 000200
ADDW5 = 000000	AT7 = 000200	CI4 041650	DATA14 003372	DSWR = 177570
ADDW6 = 000000	AUNIT = 000000	CI5 042226	DATA15 003432	DTE = 010000
ADDW7 = 000000	AUSWR = 000000	CI6 042250	DATA2 002572	DTEER 012050
ADDW8 = 000000	AUTLST 031662	CI7 042264	DATA3 002632	DTUW 040140
ADDW9 000000	AVAIL 001576	CI7B 042306	DATA4 002672	DT00 = 000001
ADEVCT= 000000	AVECT1= 000254	CI8 042364	DATA5 002732	DT01 = 000002
ADEVPM = 000000	AVECT2= 000000	CKBUS 013270	DATA6 002772	DT02 = 000004
AENV = 000000	A16 = 000400	CKCLK 023344	DATA7 003032	DT03 = 000010
AENVM = 000000	A17 = 001000	CKCLK1 023440	DATA8 003072	DT04 = 000020
AFATAL= 000000	BADBLK 001476	CKCLK2 023510	DATA9 003132	DT05 = 000040
ALOST 077232	BADENT 076677	CKCLK3 023542	DCK = 100000	DT06 = 000100
AMADR1= 000000	BADSEC 001342	CKERR 013176	DCKER 010156	DT07 = 000200
AMADR2= 000000	BADTMO 003552	CKFMT 011310	DCKER1 010274	DT08 = 000400
AMADR3= 000000	BAI = 000010	CKHCE 011464	DDISP = 177570	DT1 073530
AMADR4= 000000	BEGCOD 001502	CKLMTS 020164	DEASGN 026330	DT14 073600
AMAMS1= 000000	BEGPAT 001500	CKSWR = 104407	DEASSG 076341	DT15 073622
AMAMS2= 000000	BEGWC 001504	CK.CHR 032720	DH1 073063	DT16 073644
AMAMS3= 000000	BIT0 = 000001	CK.DEC 032672	DH14 073240	DT17 073660
AMAMS4= 000000	BIT00 = 000001	CK.DIG 032772	DH15 073342	DT2 073534
AMSGAD= 000000	BIT01 = 000002	CK.NUM 100756	DH16 073440	DT3 073552
AMSGLG= 000000	BIT02 = 000004	CK.OCT 032644	DH17 073510	DT4 073562
AMSGTY= 000000	BIT03 = 000010	CLKFLG 001312	DH2 073070	DT6 073574
AMTYP1= 000000	BIT04 = 000020	CLOCK 025074	DH3 073144	DUNIT 001532
AMTYP2= 000000	BIT05 = 000040	CLR = 000040	DH4 073172	DVA = 004000
AMTYP3= 000000	BIT06 = 000100	CLRDPB 026602	DH6 073231	DVC = 000200
AMTYP4= 000000	BIT07 = 000200	CLRQUE 045740	DISPLA 001156	ECBADO 001412
AOE = 001000	BIT08 = 000400	CMCNT 001370	DISPLY= 104414	ECBAD1 001420
APASS = 000000	BIT09 = 001000	CMCYL 001372	DISPRE 000174	ECBIT 001376
APRIOR= 000000	BIT1 = 000002	CMDAT 013614	DLT = 100000	ECC 014606
APTCSU= 000040	BIT10 = 002000	CMHED 013540	DONE 007634	ECCX 015344
APTENV= 000001	BIT11 = 004000	CMPAR 013354	DPE = 000010	ECC1 015210
APTSIZ= 000200	BIT12 = 010000	CMPARD 013372	DPINT 040056	ECC2 015340
APTSPO= 000100	BIT13 = 020000	CMPMLT 001446	DPR = 000400	ECGD 001410
ASGND 076411	BIT14 = 040000	CMPRES 020700	DPRQS 040066	ECGD1 001416
ASGN1 025722	BIT15 = 100000	CMPT 014174	DRIVE = 001220	ECH = 000100
ASGN2 026004	BIT2 = 000004	CMPRX 014166	DRIVE0 046202	ECI = 004000
ASGN3 026102	BIT3 = 000010	CMSEC 001374	DRIVE1 050414	ECMSK0 001402
ASGN4 026252	BIT4 = 000020	CMSTR 013456	DRIVE2 052626	ECMSK1 001404

ECSEC	001400	ERCTR	001364	INVLD	076544	LINS4	074226	MERR1	076720
ECWRD	001406	ERPRC1	007254	IOTVEC=	000020	LINS5	074342	MERR2	077000
ECWRD1	001414	ERPROC	007240	IR	= 000100	LINT3	074110	MESSAG	001472
EMPTYQ	046016	ERR	= 040000	ISR	= 042614	LINU06	074531	MINCYL=	000126
EMTVEC=	000030	ERROR	= 104000	IVC	= 010000	LINW3	074147	MINSEC=	000136
EM1	070412	ERRVEC=	000004	KSR	025240	LINX4	074242	MINTRK=	000132
EM10	070724	FACTOR	016400	KSR1	025246	LINX5	075557	MINUTE	001346
EM11	070767	FAIRNS	001332	LBC	= 002000	LIN10A	075154	MNTBL	002112
EM12	071022	FALPAR	007420	LBT	= 002000	LIN10B	075210	MOH	= 020000
EM13	071053	FALPR1	007434	LF	= 003012	LIN10C	075250	MOL	= 010000
EM14	071125	FER	= 000020	LIMIT	001366	LIN10H	075354	MRMCS1	100740
EM15	071150	FILBUF	016772	LINA3	074071	LIN11	075512	MRMVEC	100747
EM2	070464	FILLZ	032334	LINB3	074140	LIN11A	075533	MSFULL	077044
EM20	071204	FILLO	032442	LINB5	074310	LIN11H	075437	MSGCTS	077102
EM21	071225	FMTER	012244	LINB6	074417	LIN2C	073666	MSGFOR	076334
EM22	071256	FMT16 =	010000	LINCA3	074120	LIN2P	073706	MSGON	076311
EM23	071331	FRSTER	001356	LINC6	074452	LIN2S	073727	MSPRM	077315
EM24	071410	F0	= 000002	LINDA3	074131	LIN3.1	022032	MSWRO	076501
EM25	071456	F1	= 000004	LINDEC	023176	LIN3.3	022140	MXF	= 001000
EM26	071537	F2	= 000010	LIND5	074274	LIN3.4	022172	MXWINDW	040164
EM27	071564	F3	= 000020	LINEN3	074055	LIN6.1	022642	M.DPID	031756
EM3	070522	F4	= 000040	LINE05	074405	LIN6.2	022664	M.DP40	032014
EM30	071621	GENDPB	070322	LINEP5	074375	LIN7M	074556	M.DP41	032050
EM31	071653	GENPAR	017470	LINE1	021046	LIN7OR	074706	M.DP42	032060
EM32	071716	GENREG	070342	LINE2	021126	LIN7OX	074661	M.DP44	032112
EM33	071751	GETADR	027520	LINE2A	021276	LIN7P	074603	M.DP50	032124
EM34	072026	GETBUF	016402	LINE2B	021314	LIN7R	074716	N	076473
EM35	072070	GETID	027420	LINE3	021566	LIN7S	074621	NED	= 010000
EM36	072126	GETLMT	027346	LINE3A	021574	LIN7T	074643	NEDCLK	076422
EM37	072157	GETPAT	020122	LINE3B	021602	LIN7X	074671	NEM	= 004000
EM4	070560	GETREG=	000141	LINE3C	021614	LIN8M	074732	NEWASN	026530
EM40	072241	GETREM	031664	LINE3D	021624	LIN9B	074755	NEWUNT	001554
EM41	072310	GETREQ	046112	LINE3E	021672	LIN9E	075077	NODRVS	077120
EM42	072354	GO	= 000001	LINE3F	021760	LIN9G	075126	NOENTY	077146
EM43	072435	GODRIV	017050	LINE4	022236	LIN9H	075004	NOPTCH	012724
EM44	072502	GTSWR =	104406	LINE5	022236	LIN9I	075060	NONE	076672
EM45	072577	HCE	= 000200	LINE5A	022470	LKPAR	005320	NOTAVL	075762
EM46	072665	HCEER	012306	LINE5B	022536	LODEV	076011	NOTPRS	075745
EM5	070615	HCI	= 002000	LINE6	022600	LODPAR	020430	NOTRM	075724
EM50	072737	HCRC	= 000400	LINE6A	022612	LSC	= 004000	NOTSAF	076001
EM51	072775	HRCRCR	011132	LINE6B	022620	LSTAD	001334	NSA	= 100000
EM6	070651	HOUR	001344	LINE6C	022626	LSTHDR	077166	OFFCOD	002414
EM60	073040	HT	= 000011	LINE6D	022634	MAIN	006142	OFFDIR=	000001
ENDCMP	014450	HZ	001314	LINE7	022710	MAINDA	006170	OFFON =	000001
ENDCON	001434	JAE	= 002000	LINE7A	023040	MAIN1	006306	OFFSET=	000115
ENDING	001466	JAEER	012164	LINE8	023132	MAIN2	006424	OFFST	015766
ENDPAS	076272	IBSAVE	035126	LING6	074501	MANTR	030124	OFLIN	007532
ENDPGM=	102072	IDLE	007006	LINM3	073763	MASK	001326	OFMSG0	002312
ENDSEK	001440	IE	= 000100	LINM4	074210	MATCH	014516	OFMSG1	002345
ENDTST	076316	ILF	= 000001	LINM3	074001	MAXCYL=	000124	OFMSG2	002370
ENTADR	076640	ILR	= 000002	LINOCT	023144	MAXER	001444	OFMTBL	002420
ENTCOM	076567	INCHRD	024630	LINP3	074022	MAXSEC=	000134	ONES	003374
ENTLMT	076610	INCMIS	024700	LINP5	074325	MAXTRK=	000130	ONESEC	001352
ENTPR	005412	INCSKI	024654	LINR6	074517	MBASN	077303	OPE	= 020000
EOP1	031214	INCSOF	024604	LINSS3	074174	MCPE	= 020000	OPI	= 020000
EOP2	031240	INCTOT	024724	LINST3	074160	MCPMX	040152	OPIER	011744
EQUAL	075615	INTRVL	001452	LINS3	074041	MDPE	= 000400	OPIER1	012010

OPT	041206	PSEL =	002000	RMAS =	000016	SC8	043706	SW06	=	000100
OPTBL	002072	PSW =	177776	RMBA =	000004	SDETAL	023674	SW07	=	000200
OR	=	PUNSAF	007336	RMBAE =	000050	SEARCH=	000131	SW08	=	000400
ORDERQ	001506	PWRFLG	037616	RMCS1 =	000000	SECLMT	001424	SW09	=	001000
PACK	001320	PWRVEC=	000024	RMCS2 =	000010	SECOND	001350	SW1	=	000002
PACKSN	077270	QCNT	045446	RMCS3 =	000052	SFEK =	000105	SW10	=	002000
PAR	=	QDRVO	045540	RMDA =	000006	SEEKFG	040116	SW11	=	004000
PARENT	030700	QDRV1	045560	RMDB =	000022	SELDRV=	000145	SW12	=	010000
PARER	012072	QDRV2	045600	RMDC =	000034	SETFMT=	000143	SW13	=	020000
PARLST	077360	QDRV3	045620	RMD5 =	000012	SETVEC	005440	SW14	=	040000
PAR1	077524	QDRV4	045640	RMDT =	000026	SET.IE	045374	SW15	=	100000
PAR10	077634	QDRV5	045660	RMEC1 =	000044	SHDTYP	023664	SW2	=	000004
PAR11	077644	QDRV6	045700	RMEC2 =	000046	SIZE70	004744	SW3	=	000010
PAR14	077654	QDRV7	045720	RMERRS	040016	SIZMEM	005112	SW4	=	000020
PAR15	077664	QINPT	045456	RMER1 =	000014	SKI =	040000	SW5	=	000040
PAR16	077674	QOUTPT	045476	RMER2 =	000042	SKIER	012450	SW6	=	000100
PAR19	077704	QSTART	045516	RMHR =	000036	SLASH	077464	SW7	=	000200
PAR2	077534	QSTOP	045520	RMINIT	040166	SPOTCK	020714	SW8	=	000400
PAR20	077714	QTERM =	045740	RMLA =	000020	SRCHWT	040100	SW9	=	001000
PAR21	077724	QUES	075613	RMMR1 =	000024	STA	005730	SYSTAT		076026
PAR3	077544	RANCYL	017654	RMMR2 =	000040	STACK =	001100	T		073776
PAR4	077554	RANDOM	001474	RMOF =	000032	START	003634	TAB		075623
PAR5	077564	RANDWC	001462	RMR =	000004	START1	003644	TABLE		077734
PAR6	077574	RANPAT	020072	RMSN =	000030	START2	003662	TABLE0		077754
PAR7	077604	RANSEC	017744	RMTMR	044254	STAR30	076124	TABLE1		100022
PAR8	077614	RANSIZ	020000	RMVEC	040156	STAR5	076155	TABLE2		100070
PAR9	077624	RANTRK	017710	RMWC =	000002	STATIN	001316	TABLE3		100136
PASSES	001456	RANXIT	020112	RM05	040714	STATIS	016232	TABLE4		100204
PAT	000020	RATIO	001464	RNOP =	000101	STATPR	023550	TABLE5		100252
PATTER	001460	RDCHR =	104410	RTC =	000117	STKLMT=	177774	TABLE6		100320
PCLOCK	001310	RDDAT =	000171	RTNCTR	015624	STNDAT	002426	TABLE7		100366
PERIOD	076477	RDHD =	000173	RTURN	031636	STO	044344	TAB.XY=		001114
PFECH	035306	RDLIN =	104411	R6 =	X000006	STO1	044400	TAP	=	040000
PFECH1	035316	RDY =	000200	R7 =	X000007	STO2	044474	TBITVE=		000014
PFECH2	035400	RD.ADR	044712	S	074017	STO3	044524	TD		042656
PFECH3	035432	RD.RM	044666	SATPOW	037636	STO5	044550	THEAD		017572
PFECH4	035442	RD.RM1	044710	SAVEFG	040114	STO6	044556	THEAD1		017614
PFTSTN	035446	RD.RM2	045030	SAVER1	001360	STO7	044614	TIMER		040120
PGE =	002000	RD.RM3	045034	SAVER5	001362	STO8	044644	TKVEC =		000060
PGM	001000	RD.RM4	045040	SAVREG=	104412	STO9	044654	TPVEC =		000064
PIP	020000	RD.WRD	044714	SC	043120	SUMARY	023636	TRAPVE=		000034
PIRQ	=	READDR	023216	SCMND	026436	SUMHD	076164	TRE =		040000
PIRQVE=	000240	READHD	016012	SCOPE =	000004	SUPRS	032150	TRFER		012350
POPQUE	046134	READIN=	000121	SC04 =	000400	SUPRSL	032134	TRKLMT		001426
POSER	011672	RECAL =	000107	SC1 =	000100	SUPR1	032162	TRNSWT		040076
PROCES	007150	RECALT	015650	SC10 =	001000	SUPR2	032224	TRTVEC=		000014
PRTBAD	015352	RECALO	015740	SC11	043736	SVRH70	045230	TST1		003652
PRTIM	007576	REDAPK	026540	SC12	044046	SWR	001154	TYLIST		030520
PRO	=	RELBUF	016536	SC13	044116	SWREG	000176	TYMBA		032472
PR1 =	000040	RELSE =	000113	SC2 =	000200	SWTIM	007474	TYPACK		032522
PR2 =	000100	REPHD	076063	SC20 =	002000	SWO =	000001	TYPCK		032526
PR3 =	000140	REPLZ	032340	SC3	043164	SW00 =	000001	TYPDS =		104405
PR4 =	000200	RESREG=	104413	SC4	043170	SW01 =	000002	TYPE =		104401
PR5 =	000240	RESVEC=	000010	SC5	043202	SW02 =	000004	TYPMBA		032476
PR6 =	000300	RETRY	001330	SC6	043366	SW03 =	000010	TYPJC =		104402
PR7 =	000340	RHEXT	040162	SC6A	043502	SW04 =	000020	TYPON =		104404
PS =	177776	RMADR	040154	SC7	043630	SW05 =	000040	TYPOS =		104403

TYPRI4	032444	\$BUF =	000006	\$HINUM	037022	\$POWER	037620	\$SOFT =	000074
UCPAR	007402	\$CDW1	001266	\$HLDWC=	000110	\$PREVA=	000032	\$SSEC =	000022
ULDFLG	040104	\$CDW2	001270	\$ICNT	001120	\$PREVO=	000027	\$STUP =	177777
UNIT	001324	\$CHARC	036000	\$ILLUP	037610	\$PSEL =	000003	\$SUPRL	032234
UNS =	040000	\$CKSWR	033526	\$INTAG	001151	\$PSNL =	000140	\$SUPRS	032250
JNSAF	012610	\$CMTAG	001114	\$ITEMB	001130	\$PSNM =	000142	\$SUPR1	032262
UNTASN	075676	\$CM3 =	000000	\$LF	001204	\$PWRAD	037604	\$SUPR2	032324
UNMSG	075625	\$CM4 =	000001	\$LFLG	036721	\$PWRDN	037436	\$SVPC =	000210
UNNOT	075654	\$CNTLC	034433	\$LKCSB	001300	\$PWRMG	037600	\$SWR =	122000
UNTOFF	075633	\$CNTLG	034445	\$LKCSR	001276	\$PWRUP	037510	\$SWREG	001230
UNTON	075644	\$CNTLU	034440	\$LKS	001304	\$QUES	001202	\$STATUS=	000016
UPE =	020000	\$CODE =	000024	\$LLVEC	001306	\$RAND	036724	\$TERM =	000032
US1 =	000001	\$COMND=	000002	\$LONUM	037024	\$RDCHR	034070	\$TESTN	001212
US2 =	000002	\$CPUOP	001234	\$LPADR	001122	\$RDLIN	034160	\$TIME	024750
US4 =	000004	\$CRLF	001203	\$LPERR	001124	\$RDOFL=	000064	\$TKB	001162
VV =	000100	\$CYL =	000012	\$LPVEC	001302	\$RDSZ =	000017	\$TKCNT	033210
WAIT	001620	\$DBLK	036446	\$LSTAD	100564	\$READ =	000066	\$TKINT	033226
WATPAK	026564	\$DB2D	037122	\$MADR1	001240	\$REG =	000014	\$TKQEN=	033225
WCE =	040000	\$DB2O	037316	\$MADR2	001244	\$RESRE	037064	\$TKQIN	033212
WCF =	000040	\$DECVL	037302	\$MADR3	001250	\$RETRY	016104	\$TKQOU	033214
WCFER	012512	\$DEVCT	001216	\$MADR4	001254	\$RHEXT=	000001	\$TKQSR	033216
WCHKX -	000000	\$DEVM	001264	\$MAIL	001206	\$RMADR	001272	\$TKS	001160
WCKD -	000151	\$DIV	031710	\$MAMS1	001236	\$RMAS =	002154	\$TKSRV	033276
WCKER	010624	\$DOAGN	031632	\$MAMS2	001242	\$RMBA =	002142	\$TMP0	001174
WCKHD =	000153	\$DSPLY	032616	\$MAMS3	001246	\$RMBAE=	002206	\$TN =	000002
WC.HK	043032	\$DTBL	036436	\$MAMS4	001252	\$RMCS1=	002136	\$TNPWR	037232
WLE =	004000	\$ENDAD	031622	\$MBADR	001102	\$RMCS2=	002146	\$TOTAL=	000072
WLEER	012216	\$ENDAT=	000036	\$MBASN=	002126	\$RMCS3=	002210	\$TPB	001166
WRDCNT	001450	\$ENDCT	031606	\$MFLG	036720	\$RMDA =	002144	\$TPFLG	001173
WRL	004000	\$ENDSK=	000042	\$MISPO=	000102	\$RMDB =	002160	\$TPS	001164
WRTCHK	001470	\$ENV	001226	\$MNEW	034463	\$RMDC =	002172	\$TRAP	037730
WRTDAT=	000161	\$ENVM	001227	\$MSGAD	001222	\$RMD5 =	002150	\$TRAP2	037752
WRTHD	000163	\$EOP	031162	\$MSGLG	001224	\$RMDT =	002164	\$TRK =	000011
WRTPK	017140	\$EOPCT	031600	\$MSGTY	001206	\$RMEC1=	002202	\$TRP =	000015
WRT.AD	045132	\$ERFLG	001117	\$MSWR	034452	\$RMEC2=	002204	\$TRPAD	037764
WRT.RM	045042	\$ERMAX	001131	\$SMTYP1	001237	\$RMR1=	002152	\$STIM	001104
WRT.R1	045126	\$ERROR	034540	\$SMTYP2	001243	\$RMR2=	002200	\$STNM	001116
WRT.R2	045214	\$ERRPC	001132	\$SMTYP3	001247	\$RMHR =	002174	\$TTYIN	034414
WRT.R3	045220	\$ERRTB	003472	\$SMTYP4	001253	\$RMLA =	002156	\$TYPDS	036232
WRT.R4	045224	\$ERRTY	035130	\$NCODE=	000112	\$RMR1=	002162	\$TYPE	035450
WRT.R5	045226	\$ERTTL	001126	\$NICYL =	000116	\$RMR2=	002176	\$TYPEC	035662
WRT.WD	045130	\$ETABL	001226	\$NEXT =	000120	\$RMOF =	002170	\$TYPEX	036002
XXDP	001432	\$ETEND	001272	\$NPATC=	000113	\$RM5N =	002166	\$TYPOC	036030
Y	076475	\$FAIR =	000106	\$NSEC =	000114	\$RMVEC	001274	\$TYPON	036044
ZEROS	002472	\$FATAL	001210	\$NTRK =	000115	\$RMWC =	002140	\$TYPOS	036004
ZROIND	001354	\$FFLG	036722	\$NULL	001170	\$RMO2	076044	\$UNIT	001220
\$APTHD	001100	\$FILLC	001172	\$NWTST=	000000	\$RMO3	076051	\$UNITM	001110
\$ATYC	036502	\$FILLS	001171	\$OCNT	036226	\$RMO5	076056	\$USWR	001232
\$ATY1	036456	\$FIRST=	000122	\$OCTVL	037420	\$RTNAD	031634	\$VECT1	001256
\$ATY3	036464	\$FMT =	000001	\$OMODE	036230	\$SAVRE	037026	\$VECT2	001260
\$ATY4	036474	\$GDADR	001134	\$OPERC=	000046	\$SAVR6	037614	\$WCNT =	000004
\$AUTOB	001150	\$GDDAT	001140	\$PACK =	000026	\$SB2D	033130	\$WRDL =	000020
\$BASE	001262	\$GET42	031612	\$PASSC=	000104	\$SB2O	033160	\$WRITN=	000060
\$BDADR	001136	\$GT5WR	033616	\$PASTM	001106	\$SEC =	000010	\$WTOFL=	000056
\$BDAT	001142	\$HARD =	000076	\$PATTC=	000030	\$SETUP=	000156	\$XOFF =	000023
\$BDSEC=	000144	\$HD =	000000	\$POSIT=	000052	\$SIZE	100434	\$XON =	000021
\$BELL	001176	\$HIBTS	001100			\$SKI =	000100	\$GET4=	000000

\$OFILL 036227 .SX = 001100

. ABS. 102072 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 62464 WORDS (244 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 70 PAGES
CZRMUB.BIC,CZRMUB/C-CZRMUB.DOC,CZRMUB,SYSMAC/M

SMSGAD	6-0#	47-1	47-1*												
SMSGLG	6-0#	47-1*													
SMSGTY	6-0#	47-1	47-1	47-1*	47-1*										
SMSWR	41-1	41-1#													
SMTYP1	6-0#														
SMTYP2	6-0#														
SMTYP3	6-0#														
SMTYP4	6-0#														
SNCODE	16-22*	17-50	17-59	20-13	20-25	20-26	55-44#	55-45	55-46	55-47	55-48	55-49	55-50	55-54	
SNCYL	16-24*	16-40*	17-10*	20-37	55-48#										
SNEXT	11-100*	11-123	11-137*	15-60*	15-65*	17-63*	29-18	55-49#							
SNPATC	17-62*	20-35	55-45#												
SNSEC	16-23*	16-35	16-38*	17-32*	17-46	20-36	55-46#								
SNTRK	16-39*	17-21*	55-47#												
SNULL	6-0#	44-1	44-1	44-1											
SNWTST	9-19#														
SOCNT	45-1#	45-1*	45-1*												
SOCTVL	51-1	51-1#													
SOMODE	45-1	45-1#	45-1*	45-1*	45-1*	45-1*									
SOPERC	14-179*	14-180*	15-15	15-17	16-30	16-32	55-27#								
SPACK	10-33*	11-125	15-33	15-51	15-64*	20-9	27-166*	55-21#							
SPASS	6-0#	9-25*	35-1	35-1	35-1*	35-1*	35-1*	35-1*							
SPASSC	24-90	29-42*	35-1	35-1	35-1*	55-38#									
SPASTM	5-8#														
SPATTC	14-155	15-59*	20-35*	29-36*	29-37*	55-23#									
SPOSIT	14-185*	14-186*	22-420	24-125	55-28#										
SPOWER	52-1	52-3#													
SPREVA	13-415*	13-416*	13-417*	14-181	15-21*	15-22*	20-15*	20-16*	20-19*	20-20*	20-21*	22-210	22-214	22-218	
	22-226	54-756*	54-757*	55-24#											
SPREVO	13-407*	15-20*	20-11*	20-12*	20-30	22-46	54-755*	55-22#							
SPSEL	55-7#														
SPSNL	31-70	31-74*	39-87	39-92	55-63#	55-64	55-68								
SPSNM	9-150*	9-150*	9-150*	9-150*	9-150*	9-150*	9-150*	9-150*	9-150*	31-72	31-75*	31-104*	39-79	39-83	55-64#
SPWRAD	52-1#														
SPWRDN	9-25	52-1	52-1#												
SPWRMG	52-1#														
SPWRUP	52-1	52-1#													
SQUES	6-0#	41-1	41-1	41-1	41-1	42-1	42-1	44-1	44-1						
SR2A	53-1														
SRAND	16-8	17-43	17-76	48-1#											
SRDCHR	41-1#	53-1	53-1												
SRDDEC	53-1														
SRDLIN	41-1#	53-1	53-1												
SRDOCT	53-1														
SRDOFL	14-32*	22-396	24-113	55-31#											
SRDSZ	41-1	41-1#													
SREAD	14-28*	14-29*	14-31*	22-399	24-117	55-32#									
SREG	55-13#	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112	55-112
	55-112	55-112	55-112												
SREGRE	49-1#	53-1													
SRETRY	12-205	12-291	12-355	12-403	12-447	12-482	12-509	12-584	12-615	12-634	13-483#	13-501			
SRNEXT	4-8#	9-105	22-28	22-88	22-270	29-24	54-164	54-193	54-<16	55-96	55-112	55-112	55-112	55-112	55-112
	55-112	55-112	55-112	55-112	57-11	58-9									
SRM02	9-252	24-81	60-19#												
SRM03	9-249	24-78	60-20#												
SRM05	9-255	24-84	60-21#												
SRMADR	7-0#	9-73*	9-85	9-113	9-127	58-5	63-21	63-43							

ACDW1	6-0	6-0				
ACDW2	6-0	6-0				
ACK	4-246#					
ACPUOP	6-0	6-0				
ACTDRV	54-93#	54-347*	54-398*	54-691*	54-699*	54-986
ACTSTR	54-99#	54-98P*	54--:02*			
ADDW0	6-0					
ADDW1	6-0					
ADDW10	6-0					
ADDW11	6-0					
ADDW12	6-0					
ADDW13	6-0					
ADDW14	6-0					
ADDW15	6-0					
ADDW2	6-0					
ADDW3	6-0					
ADDW4	6-0					
ADDW5	6-0					
ADDW6	6-0					
ADDW7	6-0					
ADDW8	6-0					
ADDW9	6-0					
ADEVCT	6-0	6-0				
ADEVN	6-0	6-0				
AENV	6-0	6-0				
AENVN	6-0	6-0				
AFATAL	6-0	6-0				
AHOST	32-50	60-53#				
AMADR1	6-0	6-0				
AMADR2	6-0	6-0				
AMADR3	6-0	6-0				
AMADR4	6-0	6-0				
AMAMS1	6-0	6-0				
AMAMS2	6-0	6-0				
AMAMS3	6-0	6-0				
AMAMS4	6-0	6-0				
AMSGAD	6-0	6-0				
AMSGLG	6-0	6-0				
AMSGTY	6-0	6-0				
AMTYP1	6-0	6-0				
AMTYP2	6-0	6-0				
AMTYP3	6-0	6-0				
AMTYP4	6-0	6-0				
AOE	4-154#					
APASS	6-0	6-0				
APRIOR	6-0					
APTCSU	44-1	47-1#				
APTENV	42-1	44-1	47-1		47-1#	
APTSIZ	9-25	47-1#				
APTSP0	44-1	47-1	47-1#			
ASGN1	27-107#					
ASGN2	27-105	27-121#				
ASGN3	27-117	27-132	27-140#			
ASGN4	27-141	27-168#				
ASGN6	27-149	27-170#				
ASGN7	27-148	27-173#				
ASGND	11-64	60-33#				

LIN3.1	22-126	22-132	22-200#											
LIN3.3	22-137	22-144	22-225#											
LIN3.4	22-145	22-235#												
LIN6.1	22-336	22-342	22-358#											
LIN6.2	22-348	22-354	22-367#											
LIN7M	22-424	59-33#												
LIN7OR	22-395	59-42#												
LIN7OX	22-387	59-40#												
LIN7P	22-419	59-37#												
LIN7R	22-398	59-43#												
LIN7S	22-427	59-38#												
LIN7T	22-384	59-39#												
LIN7X	22-390	59-41#												
LIN8M	12-237	13-507	22-439	59-44#										
LIN9B	13-158	59-45#												
LIN9E	13-190	59-49#												
LIN9G	12-275	59-50#												
LIN9H	13-160	59-46#												
LIN9I	12-658	59-48#												
LINA3	22-235	59-11#												
LINB3	22-152	59-15#												
LINB5	22-284	59-23#												
LINB6	22-328	22-341	59-28#											
LINC6	22-335	59-29#												
LINCA3	22-189	59-13#												
LIND5	22-268	59-22#												
LINDA3	22-185	59-14#												
LINDEC	13-192	22-76	22-166	22-171	22-176	22-201	22-206	22-208	22-211	22-215	22-219	22-227	22-230	22-238
LINE1	22-242	22-256	22-261	22-294	22-369	22-386	22-389	22-397	22-426	22-429	22-465#			
	12-48	12-81	12-93	12-104	12-200	12-215	12-283	12-299	12-340	12-366	12-388	12-432	12-458	12-474
	12-495	12-501	12-519	12-529	12-538	12-556	12-574	12-594	12-606	12-625	12-650	12-689	12-707	13-152
	22-8#													
LINE2	12-50	12-83	12-95	12-108	12-202	12-218	12-288	12-309	12-342	12-368	12-390	12-434	12-460	12-476
	12-503	12-521	12-531	12-540	12-558	12-576	12-596	12-608	12-627	12-652	12-691	12-709	13-154	22-35#
	22-440													
LINE2A	22-44	22-49	22-68#											
LINE2B	22-69	22-72#												
LINE3	12-51	12-84	12-96	12-108	12-219	12-288	12-309	12-343	12-369	12-391	12-435	12-477	12-541	12-559
	12-577	12-628	12-692	22-125#										
LINE3A	12-609	12-653	13-155	22-131#										
LINE3B	12-597	22-137#												
LINE3C	12-461	22-144#												
LINE3D	12-710	22-107	22-150#											
LINE3E	12-504	22-164#												
LINE3F	12-522	22-183#												
LINE4	12-52	12-85	12-97	12-108	12-220	12-288	12-309	12-344	12-392	12-436	12-478	12-505	12-542	12-560
	12-578	12-610	12-654	12-693	12-711	13-156	22-108	22-249#						
LINE5	12-288	12-309	22-268#											
LINE5A	12-345	12-398	12-442	12-463	12-549	12-567	22-301#							
LINE5B	12-261	22-315#												
LINE6	12-225	22-328#												
LINE6A	12-268	22-335#												
LINE6B	12-260	22-341#												
LINE6C	12-207	12-293	12-318	12-327	12-357	12-405	12-449	12-484	12-511	12-586	12-617	12-636	22-347#	
LINE6D	12-209	12-266	12-295	12-360	12-408	12-452	12-487	12-514	12-588	12-620	12-638	22-353#		
LINE7	12-54	12-87	12-99	12-110	12-210	12-257	12-276	12-331	12-358	12-361	12-371	12-406	12-409	12-450
	12-453	12-485	12-488	12-512	12-524	12-533	12-551	12-569	12-589	12-618	12-639	12-695	12-713	13-195

MERR2	31-92	60-47#												
MESSAG	7-0#	21-46	61-11											
MINCYL	15-31	16-40	17-4	17-6	17-9	18-15	18-17	29-72*	29-91	29-96*	29-110	55-55#	61-54	61-54
	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54
MINSEC	15-29	16-38	17-26	17-28	17-31	18-21	18-23	18-29	18-33	19-17	29-74*	29-103	29-108*	29-112
	55-59#	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54
MINTRK	15-30	16-39	17-15	17-17	17-20	18-18	18-20	18-34	19-21	29-73*	29-97	29-102*	29-111	55-57#
	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54	61-54
MINUTE	7-0#	9-96*	11-87*	26-18	26-39*	26-40	26-42*	52-12*						
MINIBL	7-0#	22-60												
MOH	4-191#													
MOL	4-138#													
MRMCS1	63-22	63-49#												
MRMVEC	63-33	63-50#												
MSFULL	32-26	60-48#												
MSGCTS	32-33	60-49#												
MSGFOR	35-1	60-30#												
MSGON	12-70	34-28	35-1	60-28#										
MSPRM	27-88	60-56#												
MSWRO	27-73	60-38#												
MXF	4-106#													
MXWINDW	54-169#	54-503												
N	60-35#													
NED	4-109#													
NEDCLK	23-65	60-34#												
NEM	4-108#													
NEWASN	27-40	28-49#												
NEWUNT	7-0#	10-30*	11-53	11-69	11-70*	27-165*								
NODRVS	11-90	60-50#												
NOENTY	32-125	60-51#												
NOMTCH	12-648#	13-56												
NONE	39-81	60-44#												
NOTAVL	60-15#													
NOTPRS	9-232	27-178	60-14#											
NOTRM	9-229	27-176	60-13#											
NOTSAF	9-238	2-170	60-16#											
NSA	4-193#													
OFFCOD	7-0#	12-247												
OFFDIR	4-208#													
OFFON	4-131#													
OFFSET	4-243#	13-444												
OFFST	12-250	13-443#												
OFLIN	12-40	12-92#												
OFMSG0	7-0	7-0#												
OFMSG1	7-0	7-0#												
OFMSG2	7-0	7-0#												
OFMTBL	7-0#	22-359												
ONES	7-0	7-0#												
ONESEC	7-0#	9-94*	11-85*	26-31*	26-33*									
OPE	4-226#													
OPI	4-158#													
OPIER	12-149	12-472#												
OPIER1	12-481#													
OPT	54-373	54-408#	54-729	54-936	54-978									
OPTBL	7-0#	22-51	22-53											
OR	4-104#													
ORDER0	7-0#	9-88	11-5	11-104	11-115	11-140	11-164	52-20						

