

RM05/3/2

RM05/3/2 PERF EXER
CZRMUA0

AH-S071A-MC
FICHE 1 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



RM05/3/2

RM05/3/2 PERF EXER
CZRMUAO

AH-S071A-MC
FICHE 2 OF 2

JUN 1980
COPYRIGHT © 1980
MADE IN USA



.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
45
46
47
48
49
50
51

IDENTIFICATION

PRODUCT CODE: AC-S069A-MC
PRODUCT NAME: CZRMUJAO RM05/3/2 PERFORMANCE EXERCISER
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC GROUP
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 1. ABSTRACT
 - 1.1 GENERAL NOTES
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
- 3. OPERATING THE PROGRAM
 - 3.1 LOAD'NG THE PROGRAM
 - 3.2 STAR 'NG THE PROGRAM
 - 3.3 RESTARTING THE PROGRAM
 - 3.4 PROGRAM CONTROL
 - 3.5 PASS/TEST TERMINATION
 - 3.5.1 PASS TERMINATION
 - 3.5.2 TEST TERMINATION
 - 3.6 RUN TIME
 - 3.6.1 DATA TRANSFER MODE
 - 3.6.2 SEEK VERIFICATION MODE
 - 3.7 UNIBUS & VECTOR ADDRESSES
 - 3.8 DUAL PORT OPERATION
 - 3.9 APT
 - 3.10 APT ENVIRONMENTAL TABLE DEFINITIONS
- 4. CONTROLLING THE PROGRAM
 - 4.1 DATE & OPERATOR IDENTIFICATION
 - 4.2 PARAMETERS
 - 4.2.1 PROGRAM CONTROL PARAMETERS
 - 4.2.2 PERIPHERIAL DEVICE ADDRESSES
 - 4.2.3 PARAMETERS FOR THE FIRST OPERATION
 - 4.3 SWITCH REGISTER SETTINGS
 - 4.4 KEYBOARD COMMANDS
 - 4.4.1 'T' COMMAND
 - 4.4.2 'D' COMMAND
 - 4.4.3 'S' COMMAND
 - 4.4.4 'W' COMMAND
 - 4.4.5 'R' COMMAND
 - 4.4.6 'WT' COMMAND
 - 4.4.7 GENERAL COMMAND INFORMATION
- 5. PERFORMANCE SUMMARY TYPEOUT
 - 5.1 PERFORMANCE SUMMARY TYPEOUT EXPLANATION
 - 5.2 HARD/SOFT ERROR DEFINITIONS
 - 5.2.1 HARD ERRORS
 - 5.2.2 SOFT ERRORS
- 6. DATA CHECKING & ERROR RECOVERY

58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76

- 6.1 DATA BUFFER COMPARISON
- 6.2 VERIFICATION OF DATA WRITTEN
- 6.3 SECTOR REFORMATTING
- 6.4 BAD ADDRESS FLAGGING

7. ERROR MESSAGES

- 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 HOW VARIABLES ARE SELECTED FOR EACH OPERATION
- 8.4 DATA PATTERNS

9. RM SOFTWARE DRIVER DOCUMENT

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 EITHER AN RH11 OR AN RH70. 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 DRIVES ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE DRIVE IS PERFORMING A DATA TRANSFER OR WRITE CHECK OPERATION). 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 ARE USED (I.E., WRITE DATA, WRITE HEADER & DATA, READ DATA, AND READ HEADER & DATA) AS WELL AS WRITE CHECK DATA AND WRITE CHECK HEADER & DATA COMMANDS. 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 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 TELETYPE.

ALL COMMANDS, DATA PATTERNS, AND DATA BUFFER SIZES ARE SELECTED RANDOMLY BY THE PROGRAM. ADDITIONALLY THE ADDRESSES (EG, CYLINDER, TRACK, AND SECTOR) FOR EACH OPERATION ARE SELECTED RANDOMLY.

1.1 GENERAL NOTES

A. IN REFERENCE TO NUMBERS. 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.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
16K MEMORY (20K IF THE PROGRAM IS INCLUDED IN AN 'XXDP' CHAIN)
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 DIAGNOSTIC, PART 1 & 2
RM05/3/2 FUNCTIONAL TEST, PART 1, 2 & 3

3. OPERATING PROCEDURE

3.1 LOADING

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

THE PROGRAM STARTS AT LOCATION 200

THE PROGRAM WRITES A DATA PATTERN TO ALL ON-LINE DRIVES IN A SEQUENTIAL SEEK MODE. UPON COMPLETION OF THE WRITE, THE PROGRAM GOES INTO A RANDOM TESTING MODE.

NOTE: PARAMETERS NOT INCLUDED IN THE TELETYPE DIAGLOGUE GROUP MUST BE CHANGED BEFORE THE PROGRAM IS STARTED.

3.3 START THE PROGRAM AT LOCATION 204 IF THE RH11 OR THE RH70 IS

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

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

NOT AT THE DESIRED ADDRESS. (DEFAULT IS 176700)

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

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.

3.5 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 SPECIFICATIONS FOR THE RM DRIVE SPECIFY NO MORE THAN 1 SOFT ERROR (NON-PACK RELATED) IN 1×10^9 BITS READ OR NO MORE THAN 1 SEEK ERROR IN 1×10^6 SEEKS. THE NUMBER OF BITS OR SEEKS DETERMINING A PASS WERE SELECTED TO PROVIDE A 90% CONFIDENCE LEVEL THAT THE DRIVE IS PERFORMING TO THE APPLICABLE SPECIFICATION.

A PASS IN 'W' OR 'R' COMMAND MODE IS RELATED TO THE MAXIMUM DISK ADDRESS LIMITS SETUP BY THE USER.

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 1.875×10^8 WORDS (3×10^9 BITS).
- B. IF PARAMETER 'ENDING' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 3×10^6 SEEKS.

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 USER, 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 100..

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

C. A FATAL ERROR OCCURS: EM12 OR EM14.
D. OPERATOR DEASSIGNS THE DRIVE

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. THE MODE IS DETERMINED BY THE VALUE IN PARAMETER 'SIZE'. IF 'SIZE' IS ONE SECTOR, THE PROGRAM RUNS IN A SEEK HEAVY MODE; IF 'SIZE' APPROACHES 1/2 TRACK IN SIZE (4096 WORDS) THE PROGRAM RUNS IN A DATA TRANSFER HEAVY MODE. THE PROGRAM RUN TIME VARIES GREATLY DEPENDING ON THE OPERATION MODE SELECTED, THE MEMORY AVAILABLE OVER 16K, THE READ/WRITE RATIO PARAMETER - 'RATIO', AND BY SWITCHES 0, 1, & 2.

3.6.1 DATA TRANSFER MODE

1 DRIVE - APPROX. 3.6 HRS (TO REACH 1.875×10^8 WORDS)

TO

8 DRIVES - APPROX. 16 HRS (FOR ALL DRIVES TO REACH 1.875×10^8 WORDS)

NOTE: IF THE PROGRAM IS RUN WITH BOTH SW<00> AND SW<01> SET, THE RUN TIMES SHOULD BE ABOUT 20% FASTER.

3.6.2 SEEK VERIFICATION MODE

PARAMETER 'SIZE' = 1 SECTOR (256 WORDS)
PARAMETER 'MAXTRK' = 'MINTRK'
PARAMETER 'MAXSEC' = 'MINSEC'
SW<00> = 1 (READ ONLY MODE)

1 DRIVE - APPROXIMATELY 25 HRS (3×10^6 SEEKS)

TO

8 DRIVES - APPROXIMATELY 40 HRS (3×10^6 SEEKS FOR ALL DRIVES)

3.7 UNIBUS & VECTOR ADDRESSES

THE PROGRAM ASSUMES THE FOLLOWING UNIBUS AND VECTOR ADDRESSES. (REFER TO SECTION 4.2.2 FOR THE LOCATIONS AT WHICH TO CHANGE THESE ADDRESSES.)

UNIT	UNIBUS ADDRESS	VECTOR ADDRESS
----	-----	-----
RH11 OR RH70	176700	254
TTY PRINTER	177564	NOT USED
TTY KEYBOARD	177560	60
KW11-L	177546	100
KW11-P	172542	104

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

3.8 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.

3.9 APT

THIS PROGRAM IS APT COMPATIBLE TO THE EXTENT THAT APT HOOKS WILL BE IN THE PROGRAM AND WILL WORK THRU THE 'OPTION INTERFACE'.

FOR OTHER INTERFACES, APT MAY ONLY LOAD AND START THE PROGRAM. I.E. LOAD AND DUMP MODE.

AUTOMATIC MODE (MONITOR)

1. THE INPUT DIALOGUE IS BYPASSED.
2. THE BUSS ADDRESS AND CONTROLLER INTERRUPT VECTOR IS DEFAULTED.

DUMP MODE: INPUT DIALOGUE AFTER PROGRAM STARTS

3.10 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 CONSOLE OUTPUT
- = 0 ALLOW 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 CONSOLE SWITCH REGISTER.

4. SWITCH 2 (USER SWITCH REGISTER)
NOT USED

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

5. 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
 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 KEYBOARD ENTRY, THE PROGRAM WILL RETURN TO THE BEGINNING OF THE GROUP BEING ENTERED.
- 4.1 DATE & OPERATOR IDENTIFICATION

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

ASSUMING THE DIAGNOSTIC HAS BEEN LOADED BY ANY OTHER MODE OTHER THAN THE APT SCRIPT MODE THE PROGRAM WHEN IT IS INITIALLY STARTED, WILL ASK FOR DATE AND OPERATOR I.D. ENTRIES. (THE REQUEST FOR THESE ENTRIES OCCURS ONLY WHEN THE PROGRAM IS FIRST STARTED AND WILL NOT APPEAR WHEN THE PROGRAM IS RESTARTED.) THESE ENTRIES ARE OPTIONAL AND MAY BE BYPASSED BY ENTERING A 'CARRIAGE RETURN' IN RESPONSE TO THE REQUEST. THE PROGRAM DOES NOT EDIT OR CHECK EITHER ENTRY. UP TO 8 CHARACTERS OF DATE INFORMATION AND UP TO 6 CHARACTERS OF OPERATOR IDENTIFICATION MAY BE ENTERED. BOTH THE DATE AND THE OPERATOR I.D. WILL BE TYPED WHEN THE 'SA' COMMAND IS PREFORMED (SEE SFCTION 4.4.3).

4.2 PARAMETERS

WHEN THE PROGRAM IS STARTED FROM LOCATION 204, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

CHANGE PARAMETERS ?

THE OPERATOR MUST ENTER A 'Y' IF PARAMETER ENTRIES ARE TO BE MADE. ANY OTHER CHARACTER IS ACCEPTED AS A 'NO' ENTRY. 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.

NOTE: WHEN THE DIAGNOSTIC IS LOADED VIA APT SCRIPT MODE ALL PARAMETERS ARE LOADED FROM THE APT SYSTEM E TABLE. THIS INCLUDES THE SOFTWARE SWITCH REGISTER. THEREFORE A WORST CASE CONDITION WILL BE SET IN THE E TABLE FOR NORMAL AUTOMATIC OPERATION.

4.2.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
SIZE	10	(SEE NOTE)		CONTROLS THE MAXIMUM BUFFER SIZE USED FOR DATA TRANSFERS
PASSES	10	1	1 - 999.	NUMBER OF PASSES TO END OF TEST.
MINUTE	10	120	0 - 256.	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS
RANDOM	8	000000	0 OR 1	ERRORS PRINTED OUT IF SW<07>=0 IF PARAMETER = 0, THE DATA TRANSFER WORD COUNT IS RANDOMLY SELECTED BETWEEN 4 AND THE VALUE IN 'SIZE'. IF PARAMETER = 1, THE DATA TRANSFER WORD COUNT WILL BE THE VALUE IN 'MAXDL'
ENDING	8	000001	0 OR 1	IF PARAMETER = 1, END OF PASS DETERMINED BY THE 'WORDS READ' COUNT.

400					IF PARAMETER = 0, END OF PASS
401					IS DETERMINED BY THE NUMBER
402					OF SEKS.
403		FORMAT	8	000001	0 OR 1
404					IF PARAMETER = 0; DO NOT
405					PERFORM WRITE HEADER & DATA
406					ORDERS; IF PARAMETER > 0,
407					PERFORM WRITE HEADER & DATA
408					ORDERS
409		RATIO	8	000003	0 - 7
410					CONTROLS THE APPROXIMATE
411					RATIO OF READ TO WRITE
412					ORDERS.
413					VALUE R/W RATIO
414					0 15/1
415					1 7/1
416					2 6/2
417					3 5/3
418					4 4/4
419					5 3/5
420					6 2/6
421					7 1/7
422		MESSAGE	8	000001	0 OR 1
423					IF PARAMETER = 1, DO NOT PRINT
424					ERROR MESSAGES FOR DATA ERRORS
425					OCCURRING AT LOCATIONS DEFINED
426					BY THE OPERATOR AS BAD PACK
427					LOCATION.
428					IF PARAMETER = 0, PRINT ERROR
429					MESSAGES ASSOCIATED WITH BAD
430					PACK LOCATIONS.
431		PATTERN	10	000000	0 - 15.
432					IF PARAMETER=0, DATA
433					PATTERN IS RANDOMLY
434					SELECTED.
435					IF PARAMETER>0, SPECIFIES
436					ONE OF THE 15 PATTERNS.
437					THE SELECTED DATA PATTERN
438					IS POINTED BY THE PARAMETER
439					'PATTERN'.
440		HEADER	8	000001	0 OR 1
441					IF PARAMETER=0, RANDOM
442					DATA BLOCK ADDRESS IS
443					USED IN 'T' COMMAND
444					IF PARAMETER=1, SEQUENTIAL
445					DATA BLOCK IS USED IN
446					'T' COMMAND.

NOTE: THE PROGRAM WILL SELECT A MAXIMUM BUFFER SIZE WHICH IS DETERMINED BY THE MEMORY AVAILABLE. THE MAXIMUM BUFFER SIZE ASSIGNED BY THE PROGRAM IS 8192.(ONE TRACK) WORDS. THE OPERATOR MAY SPECIFY ANY OTHER MAXIMUM SIZE AS LONG AS THE VALUE SPECIFIED IS AT LEAST 4 WORDS BUT NO LARGER THAN THE INITIAL VALUE OF 'MAXDL' DETERMINED BY THE PROGRAM.

4.2.2 PERIPHERAL ADDRESSES AND OTHER LOCATIONS OF INTEREST

TO ALTER THESE LOCATIONS, THE OPERATOR MUST MAKE MANUAL ENTRIES BEFORE THE PROGRAM IS STARTED. THE KEYBOARD ENTRY ROUTINE DOES

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

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

NOT PROVIDE ACCESS TO THESE LOCATIONS.

LOC	TAG	CONTENTS	FUNCTION
1160	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1162	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1164	\$TPS	177564	TTY PRINTER STATUS REGISTER
1166	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1274	\$LKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1276	\$LKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1300	\$LPVEC	104	KW11-P VECTOR ADDRESS
1302	\$LKS	177546	ADDRESS OF KW11-L STATUS REGISTER
1304	\$LLVEC	100	KW11-L VECTOR ADDRESS
1312	HZ	74	74 (60 DECIMAL) IF SYSTEM IS 60 HZ; 62 (50 DECIMAL) IF SYSTEM IS 50 HZ.

THE RH11-RH70 ADDRESS AND VECTOR MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204 OR IF THE PROGRAM DOES NOT RECEIVE A RESPONSE WHEN IT ACCESSES THE DEFAULT RH11-RH70 ADDRESS.

4.2.3 PARAMETERS FOR THE FIRST OPERATION

THE FOLLOWING PARAMETERS ARE USED FOR THE INITIAL OPERATION (IN ADDITION TO THE 'MINIMUM' ADDRESS VALUES).

LOC	TAG	INITIAL VALUE	VALUE RANGE	FUNCTION
1514	BEGPAT	10	1 - 15	THE CODE FOR THE STARTING PATTERN. (IF A WRITE ORDER OR A WRITE CHECK ORDER IS SPECIFIED IN 'BEGCOD')
1516	BEGCOD	5	0 - 5	THE INITIAL COMMAND FOR EACH DRIVE EXERCISED. 0 = WRITE CHECK DATA 1 = WRITE CHECK HEADER & DATA 2 = WRITE DATA 3 = WRITE HEADER & DATA 4 = READ DATA 5 = READ HEADER & DATA
1520	BEGSIZ	402	2 - SIZE	THE BUFFER SIZE FOR THE FIRST DATA TRANSFER OPERATION.

4.3 SWITCH REGISTER SETTINGS

SW <15>	= 1	HALT ON ERROR
SW <13>	= 1	INHIBIT ERROR TYPEOUT
SW <10>	= 1	RING THE TELETYPE BELL IF ERROR
SW <7>	= 1	DISPLAY ALL DATA COMPARE ERRORS
SW <6>	= 1	DO NOT ALTER THE CURRENT OPERATION PARAMETERS
SW <5>	= 1	PARTIAL REGISTER DISPLAY IF ERROR; DO NOT DISPLAY ECC CORRECTION RESULTS
SW <4>	= 1	INHIBIT MAXIMUM ERROR COUNT CHECK; DO NOT DEASSIGN DRIVES WHEN END OF TEST IS REACHED.
SW <3>	= 1	DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS)

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

IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH
RETRY IF UNCORRECTABLE 'DCK' ERROR.
IF DATA COMPARE ERRORS & SW<7> SET, DISPLAY REST
OF BUFFER
SW <2> = 1 INHIBIT SUBSYSTEM STATUS TIMEOUT DURING STARTUP.
INHIBIT PERFORMANCE REPORT AFTER SPECIFIED TIME
SW <1> = 1 INHIBIT DATA COMPARSION AFTER READ ORDERS
SW <0> = 1 READ ONLY MODE

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34)
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 16 SWITCHES IN THE
'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE
'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST
BE FOLLOWED.

4.4 KEYBOARD COMMANDS

NOTE: ALL KEYBOARD COMMANDS WILL BE DISABLED IF DIAGNOSTIC IS
RUNNING IN THE APT SCRIPT MODE.

THROUGH THE KEYBOARD COMMANDS, THE OPERATOR MAY ASSIGN DRIVES
FOR TEST ('T' COMMAND), WRITE AND CHECK DATA PACKS ('W' COMMAND),
PERFORM WRITE DATA AND FOLLOWED BY TEST ('WT' COMMAND),
PERFORM A SEQUENTIAL READ OF A PACK ('R' COMMAND), REQUEST A DRIVE
PERFORMANCE SUMMARY ('S' COMMAND), OR DEASSIGN A DRIVE WHICH IS
BEING TESTED, READING, OR WRITING ('D' COMMAND).

IF THE START ADDRESS WAS 204, THE FOLLOWING MESSAGE WILL BE TYPED
AFTER THE PROGRAM HAS BEEN INITIALIZED.

'PROGRAM INITIALIZATION COMPLETE
'TYPE A CONTROL C TO ENTER COMMANDS'

KEYBOARD ENTRIES WILL NOT BE RECOGNIZED UNTIL THE OPERATOR TYPES
A 'CONTROL C'. WHEN THE PROGRAM SEES A 'CONTROL C' ENTRY, IT WILL
SUSPEND THE SCHEDULING OF FURTHER DEVICE OPERATIONS AND WAIT UNTIL
ALL OUTSTANDING ORDERS HAVE TERMINATED. THE PROGRAM WILL ENTER
COMMAND ENTRY MODE AND TYPE THE FOLLOWING PROMPTING MESSAGE:

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

'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; THE OPERATOR MUST TYPE ANOTHER 'CONTROL C' TO RETURN THE PROGRAM TO COMMAND MODE. IF THE COMMAND ENTERED SPECIFIED AN 'A' DRIVE NUMBER, THE PROGRAM WILL REMAIN IN COMMAND MODE UNTIL ALL AVAILABLE DRIVES HAVE BEEN PROCESSED.

THE 'WT', 'T', 'W' AND 'R' COMMANDS REQUIRE DRIVE I.D., ADDRESS LIMITS AND BAD LOCATION ADDRESS ENTRIES FOR THE DRIVE BEING REFERENCED.

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

***** DRIVE # N

THE PROGRAM WILL THEN ASK FOR A DRIVE IDENTIFICATION NUMBER WITH THE FOLLOWING TYPEOUT:

'ENTER DRIVE I.D.:'

THE OPERATOR MAY ENTER AN I.D. NUMBER FOR THE DRIVE OF UP TO 6 CHARACTERS IN LENGTH. THIS I.D. WILL BE DISPLAYED, ALONG WITH THE DATE AND OPERATOR I.D. ENTRIES (SEE SECTION 4.1), WHEN THE 'SA' COMMAND IS EXECUTED. THE OPERATOR MAY ENTER ANY CHARACTER STRING, TERMINATED BY A 'CARRIAGE RETURN', OR A 'PERIOD' ONLY (NULL ENTRY) IN RESPONSE TO THE I.D. REQUEST.

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 - 822.	THE MINIMUM CYLINDER ADDRESS
MAXCYL	822.	0 - 822.	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.

NOTE: WHEN THE 'T' COMMAND IS SELECTED, THE MINIMUM CYLINDER, TRACK, OR SECTOR ADDRESS MAY BE SPECIFIED AS BEING LARGER THAN THE MAXIMUM ADDRESS. WHEN THESE VALUES ARE INVERTED, THE PROGRAM WILL SELECT ADDRESSES BETWEEN THE 'MAX' ADDRESS AND THE UPPER PHYSICAL LIMIT.

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 ASK FOR BAD SPOT ADDRESSES WITH THE FOLLOWING TYPEOUTS:

'ENTER BAD SPOT ADDRESSES:'

THIS ROUTINE ALLOWS THE USER TO MANUALLY ENTER THE BAD SECTORS INTO A BAD SECTOR TABLE FOR THAT PARTICULAR DRIVE.

THE PROMPT USED TO ENTER THE CYLINDER ADDRESS IS AS FOLLOWS:

'CYLNDR -1 / '

TO ENTER A BAD CYLINDER ADDRESS INTO THE BAD SPOT TABLE, TYPE ANY NUMBER FROM 0 - 822., FOLLOWED BY A <CR>. HOWEVER, TO ENTER ALL TRACKS/SECTORS ON THE CURRENT CYLINDER AS BEING BAD AND EXIT THE ROUTINE, JUST TYPE A NUMBER AS ABOVE, FOLLOWED BY PERIOD(.) AND <CR>. TO EXIT WITH NO ENTRY, TYPE A <CR> OR TYPE A PERIOD(.) FOLLOWED BY A <CR>.

THE PROMPT USED TO ENTER THE TRACK ADDRESS IS AS FOLLOWS:

'TRACK -1 / '

TO ENTER THE BAD TRACK ADDRESS INTO THE BAD SPOT TABLE, TYPE ANY NUMBER FROM 0 - NN, FOLLOWED BY A <CR>. TO ENTER ALL TRACKS ON THE PREVIOUSLY ENTERED CYLINDER AS BEING BAD, JUST TYPE A <CR> WITH NO ENTRY, WHICH WILL BRING YOU TO THE NEXT PROMPT OR JUST TYPE A PERIOD(.) FOLLOWED BY A <CR>, WHICH WILL CAUSE YOU TO EXIT THE ROUTINE.

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

THE PROMPT USED TO ENTER THE SECTOR ADDRESS IS AS FOLLOWS:

'SECTOR -1 / '

TO ENTER THE BAD SECTOR ADDRESS INTO THE BAD SPOT TABLE, TYPE ANY NUMBER FROM 0 - 31., FOLLOWED BY A <CR>. TO ENTER ALL SECTORS ON THE PREVIOUSLY ENTERED CYLINDER/TRACK AS BEING BAD, JUST TYPE A <CR> WITH NO ENTRY, WHICH WILL BRING YOU TO THE NEXT CYLINDER PROMPT OR JUST TYPE A PERIOD(.) FOLLOWED BY A <CR>, WHICH WILL CAUSE YOU TO EXIT THE ROUTINE.

FOR MORE INFORMATION (SEE SECTIONS 6.3 AND 6.4)

EXAMPLE #1

CYLNDR -1 / <CR> ;NO ENTRIES AND EXIT ROUTINE

EXAMPLE #2

CYLNDR -1 / 820<CR> ;ENTERED 820. AS BAD CYLINDER.
TRACK -1 / 3<CR> ;ENTERED 3 AS BAD TRACK.
SECTOR -1 / 20.<CR> ;ENTERED 20. AS BAD SECTOR AND
;EXIT ROUTINE

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

```
CYLNR -1 / 820.<CR> :ENTERED 820. AS BAD CYLINDER.
                          :INDICATE ALL TRACKS AND SECTORS
                          :ON THAT CYLINDER AS BEING BAD.
                          :EXIT ROUTINE
```

EXAMPLE #4

```
CYLNR -1 / 820<CR> :SAME AS EXAMPLE #3
TRACK -1 / .<CR>
```

EXAMPLE #5

```
CYLNR -1 / 820<CR> :SAME AS EXAMPLE #3
TRACK -1 / <CR>
SECTOR -1 / .<CR>
```

EXAMPLE #6

```
CYLNR -1 / 820<CR> :ENTERED 820. AS BAD CYLINDER.
TRACK -1 / <CR> :INDICATE ALL TRACKS AS BEING BAD.
SECTOR -1 / 20<CR> :ENTERED 20. AS BAD SECTOR.
CYLNR -1 / <CR> :EXIT ROUTINE
```

EXAMPLE #7

```
CYLNR -1 / 820<CR> :ENTERED 820. AS BAD CYLINDER.
TRACK -1 / 3<CR> :ENTERED 3 AS BAD TRACK.
SECTOR -1 / .<CR> :INDICATE ALL SECTORS OF THE ABOVE
                  :CYLINDER/TRACK AS BEING BAD AND
                  :EXIT ROUTINE
```

4.4.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR A RANDOM TEST. THIS COMMAND IS REQUIRED TO PERFORM THE TEST OF THE DRIVE(S). THE OTHER COMMANDS ARE CONVIENCE COMMANDS OR SUPPORT COMMANDS.

FORMAT: TN<CR>

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

EXAMPLE: T0<CR> - ASSIGN DRIVE 0 FOR TEST
TA<CR> - ASSIGN ALL AVAILABLE DRIVES FOR TEST

NOTE: DRIVE OPERATION BEGINS IMMEDIATELY AFTER COMMAND IS ENTERED.

4.4.2 'D' COMMAND

USED TO DEASSIGN A DRIVE(S) BEING EXERCISED.

FORMAT: DN<CR>

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

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

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

- NOTES: 1. IF THE 'D' COMMAND REFERENCES A DRIVE NOT ASSIGNED THE PROGRAM WILL TYPEOUT '?DRIVE NOT ASSIGNED'
2. THE DRIVES WILL BE DEASSIGNED AS THEIR OPERATIONS COMPLETE.
3. IF 'DA' IS USED, ALL DRIVES BEING TESTED WILL BE DEASSIGNED; THE MESSAGE IN (1) WILL BE DISPLAYED FOR ALL DRIVES NOT BEING TESTED.

4.4.3 'S' COMMAND

USED TO REQUEST A PERFORMANCE SUMMARY TYPEOUT FOR THE REFERENCED DRIVE(S).

FORMAT: SN<CR>

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

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

- NOTES: 1. IF PARAMETER 'MINUTE' IS NOT ZERO, THE PROGRAM WILL AUTOMATICALLY DISPLAY A PERFORMANCE SUMMARY FOR EACH DRIVE BEING TESTED AT A RATE DETERMINED BY 'MINUTE'.
2. IF THE 'SA' COMMAND IS USED, THE PROGRAM WILL TYPEOUT THE OPERATOR ENTERED DATE, OPERATOR I.D., AND THE DRIVE I.D. FOR EACH DRIVE BEING TESTED. THE DATE AND OPERATOR I.D. WILL NOT BE TYPED OUT IF NO DRIVES ARE BEING TESTED.

4.4.4 'W' COMMAND

USED TO WRITE A DATA PACK WITH DATA ACCEPTABLE TO THE PERFORMANCE EXERCISER PROGRAM.

FORMAT: WN<CR>

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

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

- NOTES: 1. DATA PATTERNS GENERATED BY THE FORMATTER PROGRAM

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

(CZRML-) ARE COMPATIBLE.

2. THE 'W' COMMAND MUST BE USED TO WRITE A NEW PACK UNDER TEST BEFORE OTHER COMMANDS ARE ISSUED; IF THE DISK PACK HAS BEEN WRITTEN BY ANY PROGRAM OTHER THAN THE FORMATTER.
3. THE 'W' COMMAND PERFORMS A SEQUENTIAL WRITE OF THE PACK USING A 'WRITE DATA' COMMAND. THE DATA PATTERN USED FOR EACH WRITE IS SELECTED RANDOMLY. HOWEVER, THE OPERATION OF THE COMMAND IS SEQUENTIAL, BEGINNING AT 'MINCYL', 'MINTRK' AND CONTINUING TO 'MAXCYL', 'MAXTRK'.
4. THE 'W' COMMAND DOES NOT WRITE HEADERS AND ASSUMES THAT THE FORMAT OF THE PACK IS GOOD.
5. THE 'W' COMMAND CANNOT BE STARTED IF SWITCH 0 (READ ONLY MODE) IS SET. IF SWITCH 0 IS SET DURING THE OPERATION OF THE 'W' COMMAND, THE DRIVE PERFORMING THE 'W' COMMAND WILL IGNORE THE SWITCH.

4.4.5 'R' COMMAND

USED TO PERFORM A SEQUENTIAL READ OF THE PACK.

FORMAT: RN<CR>

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

EXAMPLE: R0<CR> - READ THE PACK ON DRIVE 0.
RA<CR> - READ THE PACKS ON ALL OF THE ONLINE DRIVES.

- NOTES:
1. THE PROGRAM WILL PERFORM A NORMAL CHECK OF ALL DATA READ. HOWEVER, ALL OPERATIONS WILL BE SEQUENTIAL.
 2. THE PROGRAM WILL READ THE PACK STARTING AT THE ADDRESS SPECIFIED BY 'MINCYL', 'MINTRK' TO THE ADDRESS SPECIFIED BY 'MAXCYL', 'MAXTRK'. THE READ WILL BE SEQUENTIAL.

4.4.6 'WT' COMMAND

USED TO PERFORM A SEQUENTIAL WRITE PACK AND 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: W0<CR> - WRITE PACK 0 AND TEST PACK 0
WTA<CR> - WRITE ALL PACKS AND TEST ALL PACKS

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

4.4.7 GENERAL COMMAND INFORMATION

- A. CONTROL-C MUST BE ENTERED BEFORE ISSUING ANY COMMAND.
- B. T, R, W AND WT COMMANDS ARE EXCLUSIVE TO ONE ANOTHER ON THE SAME DRIVE UNDER TESTING. D COMMAND MUST BE ENTERED IN ORDER TO SWITCH COMMANDS AMONG T, R, W AND WT.
- C. S COMMAND CAN BE ENTERED ANY TIME DURING THE TEST.
- D. THE ERROR RESPONSES FROM THE PROGRAM ARE AS FOLLOWS

<u>RESPONSE</u>	<u>COMMAND(S)</u>
?DRIVE N IS 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, W

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 OR CAN BE DISPLAYED ON REQUEST BY THE OPERATOR THROUGH THE USE OF THE 'S' COMMAND. THE PERFORMANCE SUMMARY TYPEOUT CONTAINS THE FOLLOWING FIELDS:

'DRV'	THE DRIVE NUMBER
'PASS'	THE PRESENT PASS COUNT FOR THE DRIVE
'ORDERS'	THE NUMBER OF ORDERS PERFORMED BY THE DRIVE
'SEEKS'	THE NUMBER OF SEEK OPERATIONS THE DRIVE PERFORMED
'WRDS XFER'	THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY THE DRIVE
'WRDS READ'	THE TOTAL NUMBER OF WORDS READ BY THE DRIVE
'SOFT'	THE NUMBER OF SOFT DATA ERRORS
'HARD'	THE NUMBER OF HARD DATA ERRORS
'SKI'	THE NUMBER OF 'SKI' ERRORS
'MISP'	THE NUMBER OF POSITIONING ERRORS
'OTHER'	THE TOTAL ERRORS OF OTHER TYPES

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)

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

WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION AND IS NOT CORRECTABLE OR DOES NOT BECOME 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 ATTEMPTS 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 ORDERS.
- 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 ORDER TERMINATED WITH NO ERROR.
- 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 WRITTEN DATA BACK AND MATCHING THE DATA WITH ONE OF THE 15 PATTERNS OR THROUGH ISSUING WRITE CHECK COMMANDS.

6.3 SECTOR REFORMATTING

THE PROGRAM WILL REFORMAT AN UNCORRECTABLE ERROR SECTOR IN THE FOLLOWING CASES (PARAMETER 'FORMAT' MUST BE SET AND SW<00> = 0). THIS PREVENTS THE SAME ERROR FROM BEING CONTINUOUSLY REPORTED.

- A. DATA CHECK ERRORS - EM21
- B. HEADER READ ERRORS - EM20, EM24, EM25, EM26, EM27
- C. DRIVE TIMING ERRORS - EM31
- D. OPERATION INCOMPLETE ERRORS - EM32
- E. WRITE CHECK ERRORS - EM22, EM23

NOTE: THE SECTOR WILL NOT BE REFORMATTED IF THAT PARTICULAR ADDRESS IS CONTAINED IN THE BAD SECTOR TABLE. (SEE SECTION 6.4)

6.4 BAD ADDRESS FLAGGING

970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026

SINCE THE RM05/3/2 SUB-SYSTEM HAS AN AUTOMATIC BAD SPOT HANDLING CAPABILITY, THE PERFORMANCE EXERCISER ALLOWS THE USER TO IDENTIFY UP TO 252. BAD SECTOR LOCATIONS FOR EACH DRIVE, WHEN ASSIGNING THE DRIVES FOR TESTING.

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

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

WHEN A DRIVE IS ASSIGNED TO BE TESTED, THE PROGRAM READS THE BAD SECTOR FILE FROM THE LAST TRACK OF THE PACK AND THEN ALLOWS THE ADDITIONAL BAD SPOTS TO BE ENTERED MANUALLY.

THE MAXIMUM NUMBER OF BAD SPOTS ALLOWED FOR EACH DRIVE, BOTH READING FROM THE LAST TRACK AND ENTERING FROM KEYBOARD, IS 252..

THE MANUALLY ENTERED BAD SECTORS ARE NOT RECORDED TO THE BAD SECTOR FILE OF THE PACK CURRENT UNDER TESTING. IF IT IS DESIRED TO RECORD THE BAD SECTORS TO THE BAD SECTOR FILE, USE THE FORMAT PROGRAM CZRML-. (STARTING AT LOCATION 204)

7. ERROR MESSAGES

DRIVE ERRORS ARE REPORTED ON THE TELETYPE OR (IF AVAILABLE) A LINE PRINTER. ALL ERROR CONDITIONS ARE REPORTED IN ERROR MESSAGES; 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

MESSAGES EM1, EM2, EM3, EM4, EM5, EM10, EM11, & EM12 ARE ALWAYS DISPLAYED ON THE TTY. THE OTHER MESSAGES ARE DISPLAYED ON EITHER THE LINE PRINTER (IF AVAILABLE) OR THE TTY.

(THE MESSAGE TAGS ARE GIVEN FOR REFERENCE.)

MESSAGE
TAG

TEXT

EM1 RH11 INTERRUPT OCCURRED (RMAS=0)

1027 THE RH11 INTERRUPTED AND THE ATTENTION SUMMARY REGISTER
1028 (PMAS) WAS CLEARED.
1029
1030 EM2 UNEXPECTED ATTENTION OCCURRED
1031
1032 THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT
1033 PERFORMING AN OPERATION.
1034
1035 EM3 MASSBUS PARITY ERROR (MCPE=1)
1036
1037 THE RH11 DETECTED A CONTROL BUS PARITY ERROR WHEN READING
1038 THE INDICATED REGISTER FROM THE INDICATED DRIVE.
1039
1040 EM4 MASSBUS PARITY ERROR (PAR=1)
1041
1042 THE INDICATED RM DETECTED A CONTROL BUS PARITY ERROR
1043 WHEN THE RH11 LOADED THE SPECIFIED REGISTER.
1044
1045 EM5 ADDRESS PLUG CHANGE BIT SET
1046
1047 THE 'OPE' BIT WAS SET WHEN THE INDICATED DRIVE INTERRUPTED.
1048
1049 EM6 RH11 DIDN'T RESPOND TO ADDRESSING
1050
1051 WHEN THE PROGRAM ADDRESSED THE RH11, NO RESPONSE WAS RECEIVED
1052 FROM THE INDICATED ADDRESS.
1053
1054 EM10 UNCORRECTABLE MASSBUS PARITY ERROR
1055
1056 THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED
1057 REGISTER.
1058
1059 EM11 FATAL MASSBUS PARITY ERROR
1060
1061 A CONTROL BUS PARITY ERROR OCCURRED WHEN THE RH11 ATTEMPTED
1062 TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.
1063
1064 EM12 PERSISTENT DEVICE UNSAFE
1065
1066 THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID
1067 NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL
1068 AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT
1069 BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN
1070 CLEARED BY MANUAL INTERVENTION.
1071
1072 EM13 OPERATION NOT COMPLETED WITHIN TIME LIMIT
1073
1074 THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 1 SECOND
1075 AFTER THE OPERATION WAS INITIATED.
1076
1077 EM14 UNIT WENT OFFLINE
1078
1079 THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION.
1080 (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY
1081 DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE
1082 WITH THE 'T' COMMAND TO RE-INITIATE TESTING.
1083

1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140

EM15 NO RESPONSE TO PORT REQUEST
THE PROGRAM IS TESTING A DUAL PORT DRIVE WHICH HAS NOT SWITCHED TO THE REQUESTING PORT WITHIN 10 SECONDS AFTER PORT REQUEST TO THE DRIVE FROM THE REPORTING PORT.

EM20 HEADER CRC ERROR
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.

EM21 DATA CHECK ('DCK') ERROR
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.

EM22 WRITE CHECK ERROR - DATA CHECK ('DCK') SET
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 16 TIMES.

EM23 WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET
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

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

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.
- EM36 WRITE LOCK ('WLE') ERROR
A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE LOCKED.
- EM40 RH11 OR UNIBUS TRANSFER ERROR
'IRE' IS SET IN THE RH11 CONTROL REGISTER AND NO DRIVE ERROR HAS OCCURRED. THE OPERATION WILL BE RETRIED 3 TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF', OR 'MDPE'.
- EM41 BUS ADDRESS OR WORD COUNT INCORRECT
NO DRIVE ERROR OCCURRED BUT EITHER THE BUS ADDRESS INDICATES THAT AN INCORRECT NUMBER OF WORDS WERE TRANSFERED OR THE WORD COUNT REGISTER IS NOT ZERO.
- EM42 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED
NO SUBSYSTEM ERROR WAS SIGNALLED; HOWEVER, THE DATA DOES NOT COMPARE.
- EM43 CAN'T MATCH DATA READ WITH A PATTERN
THE DATA IN THE BUFFER DOES NOT MATCH ANY OF THE STANDARD PATTERNS.
- EM44 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11

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 OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RM SET OR ERROR BITS IN THE RH11 SET.

EM45 ECC LOGIC FAILURE

THE CONTENTS OF EITHER THE ECC POSITION REGISTER (RMEC1) OR THE CONTENTS OF ECC PATTERN REGISTER (RMEC2) ARE NOT VALID. THE POSITION REGISTER IS EITHER A 0 OR > 040066 OR THE PATTERN REGISTER CONTAINS ZEROS.

EM46 BUS ADDRESS OR WORD COUNT NOT CONSISTENT

THE PROGRAM WAS PROCESSING AN ERROR AND FOUND THAT THE NUMBER OF WORDS TRANSFERED AS INDICATED BY THE BUS ADDRESS REGISTER DOES NOT AGREE WITH THE TRANSFER COUNT FROM THE WORD COUNT REGISTER.

EM50 SEEK INCOMPLETE ERROR

THE DRIVE SIGNALLED EITHER 'SKI' OR 'OCYL' ERROR BITS.

EM51 NOT USED

EM60 DEVICE UNSAFE

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

'PRESENT ORDER = XXXX PREVIOUS ORDER = YYYY'

MNEMONICS USED FOR THE ORDERS ARE DEFINED BELOW:

UNLOAD - UNLOAD (OCTAL 3)
SEEK - SEEK (OCTAL 5)
RECAL - RECALIBRATE (OCTAL 7)
DRVCLR - DRIVE CLEAR (OCTAL 11)
RELSE - RELEASE (OCTAL 13)
OFFSET - OFFSET (OCTAL 15)

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

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)

(DISPLAY OF THE RH/RM REGISTERS IN TWO GROUPS:
 RMCS1, RMCS2, RMD51, RMER1, RMER2, RMER3, RMEC1, & RMEC2 FROM 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'

THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURS AT AN ADDRESS
 ON THE PACK 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. THE BITS IN THIS WORD ARE ENCODED
 AS FOLLOWS:

BIT #	MEANING IF BIT IS '1'
-----	-----
15	ERROR OCCURRED DONE (BIT07=0), BITS 14-9, 2, 1 SPECIFY TYPE DONE (BIT07=1), BITS 6-3 SPECIFY TYPE
14	DRIVE IS OFFLINE
12	PERSISTENT UNSAFE CONDITION EXISTS
11	UNCORRECTABLE PARITY ERROR OCCURRED
10	FATAL PARITY ERROR OCCURRED. MASSBUS CLEAR WAS PERFORMED
9	OPERATION NOT COMPLETED WITHIN 1 SECOND MASSBUS CLEAR PERFORMED. ALL OTHER OUTSTANDING OPERATIONS WERE RESTARTED.
7	DONE - OPERATION COMPLETED

312	6	DATA ERROR OCCURRED DURING THE TRANSFER
1313		
1314	5	ERROR OCCURRED WHILE SEARCHING FOR THE 'TRANSFER' SECTOR OR DURING RECALIBRATE OR OFFSET COMMANDS
1315		
1316		
1317	4	CORRECTABLE UNSAFE CONDITION OCCURRED
1318		
1319	3	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC RECALIBRATE SEQUENCE
1320		
1321		
1322	2	PORT REQUEST TIMEOUT
1323		
1324	1	NON-EXISTENT DRIVE REQUESTED
1325		

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

PRESENT 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 RH11 BUFFER ADDRESS REGISTER AND THE RH11 WORD COUNT REGISTER. THIS LINE IS

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

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

NOT PRINTED IF SW<05> IS NOT SET.

LINE 8

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

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 SIZE = YYYY ACTUAL NUMBR WRDS XFRD = ZZZZ

THIS LINE GIVES THE STARTING ADDRESS OF THE BUFFER USED FOR THE CURRENT DATA TRANSFER OPERATION, ITS SIZE, AND THE ACTUAL NUMBER OF WORD TRANSFERED. THE STARTING ADDRESS OF THE BUFFER IS IN OCTAL, THE SIZE AND WORD 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

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

ECC CORRECTABLE WITHOUT OFFSET

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

LINE 15

READ CORRECTLY AT (NEG OR POS) OFFSET

THE SECTOR IN ERROR WAS READ WITHOUT ERROR AT THE INDICATED OFFSET VALUE.

LINE 16

ECC CORRECTABLE AT (NEG OR POS) OFFSET

THE SECTOR IN ERROR BECAME ECC CORRECTABLE AT THE INDICATED OFFSET.

LINE 17

CORRECTED ON X RETRY

THE OPERATION WAS PERFORMED ERROR FREE ON THE INDICATED RETRY ATTEMPT.

LINE 18

UNCORRECTABLE AFTER X RETRIES

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

LINE 19

DIFFERENT ERROR DURING RETRY

WHILE 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

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

TOTAL COMPARE ERRORS = XXXX

THIS LINE GIVES THE TOTAL DATA COMPARISON ERROR COUNT. THE VALUE GIVEN IS IN DECIMAL.

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

ORDERS: WWW ERRORS: X WRDS XFR: YYYY WRDS READ: ZZZZ

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

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

'ORDERS' IS THE TOTAL NUMBER OF COMMANDS GIVEN TO THE DRIVE WHICH REPORTED THE ERROR.

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

'WRDS XFR' IS THE TOTAL NUMBER OF WORDS WRITTEN AND READ BY THE DRIVE.

'WRDS READ' IS THE TOTAL NUMBER OF WORD READ BY THE DRIVE.

LINE 28

ORDERS: WWW TOTAL SEEKS: XXX TOTAL POS ERR = YYY TOTAL SKI ERR = Z

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

'ORDERS' IS THE TOTAL NUMBER OF ORDERS GIVEN TO THE DRIVE WHICH REPORTED THE ERROR.

'TOTAL SEEKS' IS THE TOTAL NUMBER OF SEEK OPERATIONS PERFORMED 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. RH11 INTERRUPT ENABLE ('IE') IS SET, TTY KEYBOARD INTERRUPT ENABLE IS SET, AND THE KW11-L OR KW11-P CLOCK IS STARTED. WHEN THESE ACTIONS HAVE BEEN COMPLETED, THE PROGRAM TYPES OUT 'PROGRAM INITIALIZE COMPLETE'. 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

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

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 ORDER, THE ASSIGNED BUFFER WILL BE FILLED WITH THE SELECTED PATTERN. (WRITE CHECK ORDERS ARE ISSUED AFTER EACH WRITE ORDER. THE WRITE CHECK ORDER USES THE PARAMETERS SELECTED FOR THE PRECEDING WRITE ORDER.) CONTROL IS THEN PASSED TO THE COMMAND INITIATION ROUTINE.

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 8 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 READS THE LOOK AHEAD REGISTER (RMLA) OF THE INTERRUPTING DRIVE AND COMPARES THE POSITION OF THE DISK WITH THAT OF THE DESIRED SECTOR.

IF OTHER DRIVES ARE WAITING ON CYLINDER, THEY ARE ALSO CHECKED. THE PROGRAM THEN ISSUES THE REQUESTED ORDER TO THE DRIVE NEAREST ITS TRANSFER SECTOR. THE DRIVES NOT SELECTED WILL HAVE ANOTHER SEARCH INITIATED. IF A DRIVE IS NOT SELECTED FOR TRANSFER AFTER THREE REVOLUTIONS OF ITS DISK, IT IS GIVEN PRIORITY OVER DRIVES WHICH HAVE NOT BEEN ON CYLINDER AS LONG.

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 RH11 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 ORDER WAS A READ ORDER, 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.

PERSISTENT UNSAFE CONDITION - EM12
UNCORRECTABLE MASSBUS PARITY ERROR - EM10
FATAL MASSBUS PARITY ERROR - EM11
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

1654 DATA CHECK ERROR - EM21
 1655 WRITE CHECK WITH DCK SET - EM22
 1656 HEADER CRC ERRORS - EM20
 1657 FORMAT ERRORS - EM24, EM26
 1658 HEADER COMPARE ERRORS - EM25, EM27
 1659 PROGRAM DETECTED POSITIONING ERROR - EM51
 1660 SEEK INCOMPLETE OR OFF CYLINDER ERROR - EM50
 1661 WRITE CHECK WITHOUT 'DCK' SET - EM23
 1662 RH11 OR UNIBUS TRANSFER ERROR - EM40
 1663 'OPI' ERROR - EM31
 1664 'PAR' ERROR - EM33
 1665 'WCF' ERROR - EM34
 1666 'IAE' ERROR - EM35
 1667 'WLE' ERROR - EM36
 1668 MISCELLANEOUS DRIVE ERROR - EM30
 1669

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

1670
 1671
 1672 BUS ADDRESS OR WORD COUNT INCORRECT - EM41
 1673 DATA COMPARE ERRORS - NO DRIVE ERROR DETECTED - EM42
 1674 CAN'T MATCH DATA READ WITH A PATTERN - EM43
 1675 ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11 - EM44
 1676 ECC LOGIC FAILURE - EM45
 1677 BUS ADDRESS OR WORD COUNT NOT CONSISTENT - EM46
 1678

8.2 DUAL PORT OPERATION

1679 DUAL PORT OPERATION IS NEARLY IDENTICAL TO THE OPERATION DESCRIBED
 1680 IN SECTION 8.1. THE DIFFERENCES ARE IN COMMAND SEQUENCE INITIATION
 1681 AND ORDER TERMINATION.
 1682

1683 WHEN THE DUAL PORT HANDLER ROUTINE IN THE EXERCISER PROGRAM RECEIVES
 1684 A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS
 1685 ONLINE. THE DRIVE IS SELECTED AND READ THE RMCS1 REGISTER BY
 1686 TEST THE 'DVA' BIT.
 1687

1688 IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE
 1689 DRIVE IS SEIZED BY THE OTHER PORT, READING 'RMDS1' WILL SET
 1690 'PORT REQUEST'. THE PROGRAM CHECKS 'DVA' IN 'RMCS1'. IF THE DRIVE
 1691 IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE
 1692 WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE).
 1693 IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE
 1694 IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND
 1695 STARTS A 10 SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS
 1696 NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 10 SECOND INTERVAL,
 1697 THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY
 1698 THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL
 1699 LOGIC IN THE DRIVE BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS
 1700 (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM
 1701 ON A MOD33 TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE
 1702 AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF
 1703 WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT
 1704 RELATED ERROR MESSAGES PROPERLY.
 1705

1706 AFTER A DRIVE HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE
 1707 THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE; IF THE OPERATION
 1708 TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL
 1709 ERROR PROCESSING HAS BEEN COMPLETED.
 1710

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

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 ORDER 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 EXCLUDE ALL ADDRESSES BETWEEN 'MAX' AND 'MIN' FROM THE SELECTION. FOR EXAMPLE: IF 'MINTRK' IS 5 AND 'MAXTRK' IS 2, THEN TRACK ADDRESS SELECTION WILL EXCLUDE TRACKS 3 - 4 FROM THE SELECTION AND SELECT AN ADDRESS FROM AMONG ADDRESSES 5, 0, 1, 2.
- B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 - AND THE VALUE IN 'SIZE'. THE SIZE SELECTED IS WEIGHTED TO ENSURE 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.
- 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; OTHERWISE, THE DATA PATTERN INDEXED BY THE VALUE 'PATTERN' IS SELECTED.
- D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE CHECK HEADER & DATA ORDERS ARE PERFORMED ONLY IF THE PREVIOUS ORDER WAS THE APPROPRIATE DATA ORDER. IF THE 'FORMAT' PARAMETER IS ZERO, THE PROGRAM WILL NOT SELECT WRITE HEADER & DATA (AND WRITE CHECK HEADER & DATA) ORDERS. WHEN THE PROGRAM SELECTS A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 258. THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.
- E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'W', OR 'R' COMMAND IS NOT RANDOMLY SELECTED. THE PARAMETERS FOR THE FIRST OPERATION ARE THE MINIMUM OR STARTING VALUES OF THE VARIABLES.

8.4 DATA PATTERNS

THE PROGRAM SELECTS ONE OF THE FOLLOWING DATA PATTERNS TO WRITE WHEN A WRITE ORDER 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: FROM THE FIRST 4 DATA WORDS OF EACH SECTOR, THE PROGRAM MATCHES THE DATA TO ONE OF THE FOLLOWING PATTERNS.

PAT 1 PAT 2 PAT 3 PAT 4 PAT 5 PAT 6 PAT 7 PAT 8

1768	000001	177776	000000	133331	052525	155554	026455	066666
1769	000003	177774	000000	133331	052525	155554	026455	066666
1770	000007	177770	000000	133331	052525	155554	026455	066666
1771	000017	177760	177777	133331	125252	155554	151322	066666
1772	000037	177740	177777	133331	125252	155554	151322	066666
1773	000077	177700	177777	133331	125252	155554	151322	066666
1774	000177	177600	000000	133331	052525	155554	026455	066666
1775	000377	177400	000000	133331	052525	155554	026455	066666
1776	000777	177000	177777	133331	125252	155554	151322	066666
1777	001777	176000	177777	133331	125252	155554	151322	066666
1778	003777	174000	000000	133331	052525	155554	026455	066666
1779	007777	170000	177777	133331	125252	155554	151322	066666
1780	017777	160000	000000	133331	052525	155554	026455	066666
1781	037777	140000	177777	133331	125252	155554	151322	066666
1782	077777	100000	000000	133331	052525	155554	026455	066666
1783	177777	000000	177777	133331	125252	155554	151322	066666

1786	PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15
1787	000001	177776	172666	077777	153333	000000	177777
1788	000002	177775	155555	137777	066667	177777	000000
1789	000004	177773	172666	157777	153333	177777	000000
1790	000010	177767	155555	167777	066667	177777	000000
1791	000020	177757	172666	173777	153333	177777	000000
1792	000040	177737	155555	175777	066667	177777	000000
1793	000100	177677	172666	176777	153333	177777	000000
1794	000200	177577	155555	177377	066667	177777	000000
1795	000400	177377	172666	177577	153333	177777	000000
1796	001000	176777	155555	177677	066667	177777	000000
1797	002000	175777	172666	177737	153333	177777	000000
1798	004000	173777	155555	177757	066667	177777	000000
1799	010000	167777	172666	177767	153333	177777	000000
1800	020000	157777	155555	177773	066667	177777	000000
1801	040000	137777	172666	177775	153333	177777	000000
1802	100000	077777	155555	177776	066667	177777	000000

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

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

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

```
=0      OFFLINE, DRIVE
        IS NOT AN RM05/3/2, OR
        NONEXISTENT DRIVE
<0      UNSAFE
```

THE DRIVE TYPF 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
```

*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
```

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

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.
THE DRIVER WILL SET THE TIMEOUT TO 1 SECOND FOR ALL POSITIONING
AND DATA TRANSFER OPERATIONS AND WILL SET THE TIMEOUT TO 30
SECONDS FOR ERROR RECOVERY OPERATIONS.

9.4.1 EXAMPLE - WRITE 1000. WORDS

```
1$:   JSR    R0,RM05    ;CALL THE DRIVER
      WRTDPB    ;DPB ADDRESS
      BR      1$       ;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
        .BYTE 3        ;SECTOR
        .BYTE 5        ;TRACK
        .WORD 400      ;CYLINDER
        .WORD ERRTB5   ;ERROR TABLE
        .WORD 0        ;STATUS/ERROR INDICATOR
```

ALTERNATE DPB SETUP

```
WRTDPB: .WORD 5          ;THIS SETUP ACHIEVED
        .WORD WRITE     ;EVERYTHING THE
        .WORD -1000.    ;ABOVE TABLE DID, BUT
        .WORD WRTBUF    ;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

1939	RMCS2	10
1940	RMDS	12
1941	RMER1	14
1942	RMA5	16
1943	RMLA	20
1944	RMDB	22
1945	RMMR1	24
1946	RMDT	26
1947	RMSN	30
1948	RMOF	32
1949	RMDC	34
1950	RMHR	36
1951	RMMR2	40
1952	RMER2	42
1953	RMEC1	44
1954	RMEC2	46

9.6 COMMANDS PERFORMED BY THE DRIVER

	COMMAND	CODE	COMMAND TYPE
	-----	----	-----
1961	NO OPERATION	101	N
1962	UNLOAD	103	N
1963	SEEK	105	P
1964	RECALIRATE	107	P
1965	DRIVE CLEAR	111	P
1966	RELEASE	113	N
1967	OFFSET	115	N
1968	RETURN TO CENTER	117	P
1969	READIN PRESET	121	P
1970	PACK ACKNOWLEDGE	123	N
1971	SEARCH	131	P
1972	GET REGISTER(S)	141	S
1973	SET FORMAT	143	S
1974	SELECT DRIVE	145	S
1975	WRITE CHECK DATA	151	S
1976	WRITE CHK HEADER & DATA	153	D
1977	WRITE DATA	161	D
1978	WRITE HEADER & DATA	163	D
1979	READ DATA	171	D
1980	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'
-----	-----

1995

1996		
1997		
1998		
1999		
2000		
2001	15	ERROR OCCURRED DONE (BIT07=0); BITS 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
2002	14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
2003		
2004	13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
2005		
2006		
2007		
2008	12(2)	PERSISTENT UNSAFE CONDITION EXIST.
2009		
2010	11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
2011		
2012	10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED, ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
2013		
2014		
2015		
2016	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
2017		
2018	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
2019		
2020	7	DONE
2021		
2022	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
2023		
2024	5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
2025		
2026		
2027	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
2028		
2029	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC 'RECALIBRATE' SEQUENCE
2030		
2031		
2032	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
2033		
2034		
2035		
2036	1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
2037		
2038		
2039	(1) =>	REQUEST WASN'T PUT IN QUEUE. (RH/RM REGISTERS WERE NOT SAVED)
2040		
2041		
2042	(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'.
2043		
2044		
2045		
2046		
2047	(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.
2048		
2049		
2050		
2051		
2052	(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	RH70 INTERRUPT OCCURRED (RHAS=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
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WR. IN 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 =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097

a

1
62
63

:PROGRAM REVISION #001

.TITLE CZRMJAO RM05/3/2 PERF EXER
:*COPYRIGHT (C) 1980
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754

:*PROGRAM BY MIKE LEAVITT

:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-(4)), 1980.

54

.SBTTL OPERATIONAL SWITCH SETTINGS

65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81

SWITCH	USE
15	HALT ON ERROR
13	INHIBIT ERROR TYPEOUTS
10	BELL ON ERROR
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 COMPARISON AFTER READ ORDERS
0	READ ONLY MODE

.SBTTL BASIC DEFINITIONS

```

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

```

```

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

```

```

000000  : *GENERAL PURPOSE REGISTER DEFINITIONS
000001  R0 = %0          ;;GENERAL REGISTER
        R1 = %1          ;;GENERAL REGISTER

```

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

;*PRIORITY LEVEL DEFINITIONS

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

;*SWITCH REGISTER SWITCH DEFINITIONS

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

;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

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

```

000100 BIT06 = 100
000040 BIT05 = 40
000020 BIT04 = 20
000010 BIT03 = 10
000004 BIT02 = 4
000002 BIT01 = 2
000001 BIT00 = 1
001000 BIT9=BIT09
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00

```

```

;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC = 4 ;:TIME OUT AND OTHER ERRORS
000010 RESVEC = 10 ;:RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC = 14 ;: 'T' BIT
000014 TRTVEC = 14 ;:TRACE TRAP
000014 BPTVEC = 14 ;:BREAKPOINT TRAP (BPT)
000020 IOTVEC = 20 ;:INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC = 24 ;:POWER FAIL
000030 EMTVEC = 30 ;:EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC = 34 ;:'TRAP' TRAP
000060 TKVEC = 60 ;:TTY KEYBOARD VECTOR
000064 TPVEC = 64 ;:TTY PRINTER VECTOR
000240 PIRQVEC = 240 ;:PROGRAM INTERRUPT REQUEST VECTOR

```

.SBTTL RH11/RH70 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 ;:MASSBUSS 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)
000010 BAI = 10 ;:BUS ADDRESS INCREMENT INHIBIT (BIT #3)

```

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      000020      PAT      = 20      ;MASSBUS PARITY TEST (BIT #4)
109      000040      CLR      = 40      ;CLEAR (BIT #5)
110      000100      IR       = 100     ;INPUT READY (BIT #6)
111      000200      OR       = 200     ;OUTPUT READY (BIT #7)
112      000400      MDPE     = 400     ;MASS BUS PARITY ERROR (BIT #8)
113      001000      MXF      = 1000    ;MISSED TRANSFER ERROR (BIT #9)
114      002000      PGE      = 2000    ;PROGRAM ERROR (BIT #10)
115      004000      NEM      = 4000    ;NON EXISTENT MEMORY (BIT #11)
116      010000      NED      = 10000   ;NON EXISTENT DRIVE (BIT #12)
117      020000      UPE      = 20000   ;UNIBUS PARITY ERROR (BIT #13)
118      040000      WCE      = 40000   ;WRITE CHECK ERROR (BIT #14)
119      100000      DLT      = 100000  ;DATA LATE (BIT #15)
120
121      ;DATA BUFFER REGISTER (RMDDB)
122      ;(EACH BIT IS CALLED BY BIT NUMBER)
123
124      .SBTTL  RM REGISTERS
125
126      ;CONTROL AND STATUS 1 REGISTER. (#00)
127
128      000001      GO       = 1       ;GO BIT (BIT #0)
129      000002      F0       = 2       ;FUNCTION CODE BIT #1
130      000004      F1       = 4       ;FUNCTION CODE BIT #2
131      000010      F2       = 10      ;FUNCTION CODE BIT #3
132      000020      F3       = 20      ;FUNCTION CODE BIT #4
133      000040      F4       = 40      ;FUNCTION CODE BIT #5
134      004000      DVA      = 4000    ;DEVICE AVAILABLE (BIT #11)
135
136      ;DRIVE STATUS REGISTER (RMDS1) (#01)
137
138      000001      OFFON    = 1       ;OFFSET ON (BIT #0)
139      000100      VV       = 100     ;VOLUME VALID (BIT #6)
140      000200      DRY      = 200     ;DRIVE READY (BIT #7)
141      000400      DPR      = 400     ;DRIVE PRESENT (BIT #8)
142      001000      PGM      = 1000    ;PROGRAMABLE (BIT #9)
143      002000      LBT      = 2000    ;LAST SECTOR TRANSFERRED (BIT #10)
144      004000      WRL      = 4000    ;WRITE LOCK (BIT #11)
145      010000      MOL      = 10000   ;MEDIUM ON-LINE (BIT #12)
146      020000      PIP      = 20000   ;POSITIONING OPERATION IN PROGRESS (BIT #13)
147      040000      ERR      = 40000   ;COMPOSITE ERROR (BIT #14)
148      100000      ATA      = 100000  ;ATTENTION ACTIVE (BIT #15)
149
150      ;ERROR REGISTER #01 (RMER1) (#02)
151
152      000001      ILF      = 1       ;ILLEGAL FUNCTION (BIT #0)
153      000002      ILR      = 2       ;ILLEGAL REGISTER (BIT #1)
154      000004      RMR      = 4       ;REGISTER MODIFICATION REFUSED (BIT #2)
155      000010      PAR      = 10      ;PARITY ERROR (BIT #3)
156      000020      FER      = 20      ;FORMAT ERROR (BIT #4)
157      000040      WCF      = 40      ;WRITE CLOCK FAIL (BIT #5)
158      000100      ECH      = 100     ;ECC HARD ERROR (BIT #6)
159      000200      HCE      = 200     ;HEADER COMPARE ERROR (BIT #7)
160      000400      HCRC     = 400     ;HEADER CRC ERROR (BIT #8)
161      001000      AOE      = 1000    ;ADDRESS OVERFLOW ERROR (BIT #9)
162      002000      IAE      = 2000    ;INVALID ADDRESS ERROR (BIT #10)
163      004000      WLE      = 4000    ;WRITE LOCK ERROR (BIT #11)
164      010000      DTE      = 10000   ;DRIVE TIMING ERROR (BIT #12)
```

165	020000	OPI = 20000	:OPERATION INCOMPLETE (BIT #13)
166	040000	UNS = 40000	:DRIVE UNSAFE (BIT #14)
167	100000	DCK = 100000	:DATA CHECK ERROR (BIT 15)
168			
169			
170			
171			
172			
173			
174	000001	ATO = 1	:DEVICE 0 (BIT #0)
175	000002	AT1 = 2	:DEVICE 1 (BIT #1)
176	000004	AT2 = 4	:DEVICE 2 (BIT #2)
177	000010	AT3 = 10	:DEVICE 3 (BIT #3)
178	000020	AT4 = 20	:DEVICE 4 (BIT #4)
179	000040	AT5 = 40	:DEVICE 5 (BIT #5)
180	000100	AT6 = 100	:DEVICE 6 (BIT #6)
181	000200	AT7 = 200	:DEVICE 7 (BIT #7)
182			
183			
184			
185			
186			
187			
188	000001	DT00 = 1	:DRIVE TYPE NUMBER BIT 1
189	000002	DT01 = 2	:DRIVE TYPE NUMBER BIT 2
190	000004	DT02 = 4	:DRIVE TYPE NUMBER BIT 3
191	000010	DT03 = 10	:DRIVE TYPE NUMBER BIT 4
192	000020	DT04 = 20	:DRIVE TYPE NUMBER BIT 5
193	000040	DT05 = 40	:DRIVE TYPE NUMBER BIT 6
194	000100	DT06 = 100	:DRIVE TYPE NUMBER BIT 7
195	000200	DT07 = 200	:DRIVE TYPE NUMBER BIT 8
196	000400	DT08 = 400	:DRIVE TYPE NUMBER BIT 9
197	004000	DRQ = 4000	:DRIVE REQUEST REQUIRED (BIT #11)
198	020000	MOH = 20000	:MOVING HEAD (BIT #13)
199	040000	TAP = 40000	:TAPE DRIVE (BIT #14)
200	100000	NSA = 100000	:NOT SECTOR ADDRESSED (BIT #15)
201			
202			
203			
204	000100	SC1 = 100	:SECTOR COUNT FIELD 0 (BIT #6)
205	000200	SC2 = 200	:SECTOR COUNT FIELD 1 (BIT #7)
206	000400	SC04 = 400	:SECTOR COUNT FIELD 2 (BIT #8)
207	001000	SC10 = 1000	:SECTOR COUNT FIELD 3 (BIT #9)
208	002000	SC20 = 2000	:SECTOR COUNT FIELD 4 (BIT #10)
209			
210			
211			
212			
213			
214	000001	OFFDIR = 1	:OFFSET DIRECTION
215	002000	HCI = 2000	:HEADER COMPARE INHIBIT (BIT #10)
216	004000	ECI = 4000	:ERROR CORRECTION CODE INHIBIT (BIT #11)
217	010000	FMT16 = 10000	:FORMAT BIT (BIT #12)
218			
219			
220			
221			

```
222 ;CURRENT CYLINDER ADDRESS (RMCC) (#13)
223 ;(REGISTER CURRENTLY NOT USED)
224
225 ;RM ERROR REGISTER #02 (RMER2) (#15)
226
227 000010 OPE = 10
228 000200 DVC = 200
229 002000 LBC = 2000
230 004000 LSC = 4000
231 010000 IVC = 10000
232 020000 DPE = 20000
233 040000 SKI = 40000 ;SEEK INCOMPLETE (BIT #14)
234
235 ;ECC POSITION REGISTER (RMEC1) (#16)
236 ;(EACH BIT IS CALLED BY BIT NUMBER)
237
238 ;ECC PATTERN REGISTER (RMEC2) (#17)
239 ;(EACH BIT IS CALLED BY BIT NUMBER)
240
241 .SBTTL RM DRIVER COMMANDS
242
243 000101 RNOP = 101 ;NO OPERATION
244 000105 SEEK = 105 ;SEEK
245 000107 RECAL = 107 ;RECALIBRATE
246 000111 DRVCLR = 111 ;DRIVE CLEAR
247 000113 RELSE = 113 ;RELEASE
248 000115 OFFSET = 115 ;OFFSET
249 000117 RTC = 117 ;RETURN TO CENTER LINE
250 000121 READIN = 121 ;READ IN PRESET
251 000123 ACK = 123 ;PACK ACKNOWLEDGE
252 000131 SEARCH = 131 ;SEARCH
253 000141 GETREG = 141 ;GET REGISTERS
254 000143 SETFMT = 143 ;SET FORMAT (& ECI OR HCI)
255 000145 SELDRV = 145 ;SELECT DRIVE
256 000151 WCKD = 151 ;WRITE CHECK DATA
257 000153 WCKHD = 153 ;WRITE CHECK HEADER & DATA
258 000161 WRTDAT = 161 ;WRITE DATA
259 000163 WRTHD = 163 ;WRITE HEADER & DATA
260 000171 RDDAT = 171 ;READ DATA
261 000173 RDHD = 173 ;READ HEADER & DATA
262
263 176700 ABASE = 176700
264 000254 AVECT1 = 254
265
266
267
268 .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)
```


269 000200 000137 003636
270 000204 000137 003626
271
272

JMP @#START1 ;:JUMP TO STARTING ADDRESS OF PROGRAM
JMP @#START ;CHANGE THE RH ADDRESS

.SBTTL ACT11 HOOKS

:HOOKS REQUIRED BY ACT11

000210 \$SVPC= ;SAVE PC
000046 .=46
000046 030536 \$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
000052 000052 .=52
000052 C40000 .WORD 40000 ;:2)SET LOC.52 TO 40000
000210 .= \$SVPC ;: RESTORE PC
273 001100
274
275 .=1100
.SBTTL APT PARAMETER BLOCK

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

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

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

001100 \$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 000264 \$STMT: .WORD 180. ;:RUN TIM OF LONGEST TEST
001106 000264 \$PASTM: .WORD 180. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110 000264 \$UNITM: .WORD 180. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112 000032 .WORD \$ETEND-\$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
276 001114
277 TAB.XY= ;CMTAGSTARING ADDRESS

0

.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			\$SWR: .WORD DSWR	:: ADDRESS OF SWITCH REGISTER
001156	177570			\$DISPLAY: .WORD DDISP	:: ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS: 177560	:: TTY KBD STATUS
001162	177562			\$TKB: 177562	:: TTY KBD BUFFER
001164	177564			\$TPS: 177564	:: TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB: 177566	:: TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL: .BYTE 0	:: CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS: .BYTE 2	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC: .BYTE 12	:: INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TPFLG: .BYTE 0	:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMP0: .WORD 0	:: USER DEFINED
001176	207	377	377	\$BELL: .ASCIZ <207><377><377>	:: CODE FOR BELL
001202	077			\$QUES: .ASCII /?/	:: QUESTION MARK
001203	015			\$CRLF: .ASCII <15>	:: CARRIAGE RETURN
001204	012	000		\$LF: .ASCIZ <12>	:: LINE FEED

.SBTTL APT MAILBOX-ETABLE

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

001226	000	\$ENV:	.BYTE	AENV	::ENVIRONMENT BYTE
001227	000	\$ENVM:	.BYTE	AENVM	::ENVIRONMENT MODE BITS
001230	000000	\$SWREG:	.WORD	ASWREG	::APT SWITCH REGISTER
001232	000000	\$USWR:	.WORD	AUSWR	::USER SWITCHES
001234	000000	\$CPUOP:	.WORD	ACPUOP	::CPU TYPE,OPTIONS
		*			BITS 15-11=CPU TYPE
		*			11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		*			11/70=06,PDQ=07,Q=10
		*			BIT 10=REAL TIME CLOCK
		*			BIT 9=FLOATING POINT PROCESSOR
		*			BIT 8=MEMORY MANAGEMENT
001236	000	\$MAMS1:	.BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001237	000	\$MTYP1:	.BYTE	AMTYP1	::MEM. TYPE,BLK#1
		*			MEM.TYPE BYTE -- (HIGH BYTE)
		*			900 NSEC CORE=001
		*			300 NSEC BIPOLAR=002
		*			500 NSEC MOS=003
001240	000000	\$MADR1:	.WORD	AMADR1	::HIGH ADDRESS,BLK#1
		*			MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE
001242	000	\$MAMS2:	.BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001243	000	\$MTYP2:	.BYTE	AMTYP2	::MEM. TYPE,BLK#2
001244	000000	\$MADR2:	.WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001246	000	\$MAMS3:	.BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001247	000	\$MTYP3:	.BYTE	AMTYP3	::MEM. TYPE,BLK#3
001250	000000	\$MADR3:	.WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001252	000	\$MAMS4:	.BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001253	000	\$MTYP4:	.BYTE	AMTYP4	::MEM. TYPE,BLK#4
001254	000000	\$MADR4:	.WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001256	000254	\$VECT1:	.WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001260	000000	\$VECT2:	.WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001262	176700	\$BASE:	.WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001264	000000	\$DEVN:	.WORD	ADEVN	::DEVICE MAP
001266	000000	\$CDW1:	.WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001270	000000	\$CDW2:	.WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001272		\$ETEND:			
		.MEXIT			

0

.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.	:74(8) IF 60 HZ SYSTEM, 62(8) IF 50 HZ SYSTEM
001316	000000			STATIN: .WORD	0	: 'TYPE STATISTICS' INDICATOR
001320	000000			PACK: .WORD	0	: 'W' COMMAND INDICATOR
001322	000000	000000	000000	DATE: .WORD	0,0,0,0,0	:OPERATOR ENTERED DATE
001334	000000	000000	000000	OPERID: .WORD	0,0,0,0,0	:OPERATOR ID
	001220			DRIVE	=\$UNIT	:DRIVE # STORAGE: ERRORS 1-5 & 10
						:SAME AS USED IN APT
001344	000000			ATTN: .WORD	0	:ATTN REG STORAGE: ERRORS 1-5 & 10
001346	000000			UNIT: .WORD	0	:DRIVE # STORAGE FOR PRINTOUT
001350	000000			MASK: .WORD	0	:ERROR RETRY REGISTER MASK
001352	000	000		RETRY: .BYTE	0,0	:ERROR RETRY LIMIT IN THE LOWER BYTE
						:RETRY COUNT IN THE UPPER BYTE
001354	000003			FAIRNS: .WORD	3	:MAXIMUM TIME IN QUEUE VALUE
001356	000000			LSTAD: .WORD	0	:STORE LAST MEMORY ADDRESS HERE
001360	000000			CHGADR: .WORD	0	:CHANGE RH/RM UNIBUS ADDRESS FLAG
001362	000000			CFLAG: .WORD	0	: 'CONTROL C' FLAG
001364	000000			BADSEC: .WORD	0	:BAD SECTOR/TRACK FLAG
001366	000000			HOUR: .WORD	0	:HOUR COUNT STORED HERE (MAXIMUM - 999.)
001370	000000			MINUTE: .WORD	0	:MINUTE'S COUNT STORED HERE
001372	000000			SECOND: .WORD	0	:SECOND'S COUNT STORED HERE
001374	000000			SIXTEE: .WORD	0	:TIMER ROUTINE COUNTER (FOR ONE SECOND)
001376	177777			ZROIND: .WORD	-1	:ZERO INDICATOR FOR THE DATA COMPARE ROUTINE
001400	000			FRSTER: .BYTE	0	:DATA COMPARE ERROR FLAG
						:IF > 0, PROCESSING 'DCKER' OR CAN'T MATCH PATTERN
						:IF < 0, MISCOMPARSION FOUND
001401	000			.BYTE	0	:MISCOMPARSION OR CAN'T MATCH PATTERN FLAG
						:IF < 0, ERROR IN BUFFER
001402	000000			SAVER1: .WORD	0	:SAVE R1 HERE
001404	000000			SAVER5: .WORD	0	:SAVE R5 HERE
001406	000000			ERCTR: .WORD	0	:NUMBER OF ERRORS
001410	000000			LIMIT: .WORD	0	:DISPLAY LIMIT
001412	000000			CMCNT: .WORD	0	:WORD COUNT
001414	000000			CMCYL: .WORD	0	:CYLINDER ADDRESS
001416	000			CMSEC: .BYTE	0	:SECTOR ADDRESS
001417	000			CMTRK: .BYTE	0	:TRACK ADDRESS
001420	000000			ECBIT: .WORD	0	:ERROR BURST BIT OFFSET
001422	000000			ECSEC: .WORD	0	:ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
001424	000000			ECMSK0: .WORD	0	:CORRECTION MASK FOR FIRST ERROR WORD
001426	000000			ECMSK1: .WORD	0	:CORRECTION MASK FOR SECOND ERROR WORD
001430	000000			ECWRD: .WORD	0	:LOCATION OF FIRST ERROR WORD
001432	000000			ECGD: .WORD	0	:GOOD DATA, FIRST WORD
001434	000000			ECBAD0: .WORD	0	:BAD DATA, FIRST WORD
001436	000000			ECWRD1: .WORD	0	:LOCATION OF SECOND ERROR WORD
001440	000000			ECGD1: .WORD	0	:GOOD DATA, SECOND WORD
001442	000000			ECBAD1: .WORD	0	:BAD DATA, SECOND WORD

001444	001465	CYLMT: .WORD	821.	:CYLINDER ADDRESS LIMIT
001446	000037	SECLMT: .WORD	31.	:SECTOR ADDRESS LIMIT
001450	000004	TRKLMT: .WORD	4.	:TRACK ADDRESS LIMIT, RM02/3 = 4. AND RM05 = 18.
001452	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.

.SB*TL COMMON PARAMETERS

001454	002740	ENDCON: .WORD	002740	:1.875 X 10^8 WORDS (10) (3 X 10^9 BITS)
001456	005455		005455	:MSW
001460	143300	ENDSEK: .WORD	143300	:3 X 10^6 SEEKS (LSW)
001462	000055		55	:MSW
001464	000001	PASCNT: .WORD	1	:NUMBER OF PASSES TO END OF TEST
001466	000000	MAXDL: .WORD	0	:MAXIMUM DATA TRANSFER SIZE IN WORDS :(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR :DURING PARAMETER ENTRY DIALOG.)
001470	000144	MAXER: .WORD	100.	:MAXIMUM ERRORS - 100(10)
001472	000170	INTRVL: .WORD	120.,0	:FIRST WORD IS THE PERFORMANCE TIMEOUT INTERVAL :(IN MINUTES). SECOND WORD IS THE INTERVAL COUNTER.
001476	000004	CPLMT: .WORD	4	:NUMBER OF COMPARE ERRORS TYPED OUT
001500	000001	FORMAT: .WORD	1	:IF NOT EQ 0, ALLOW WRITE HEADER & DATA ORDERS
001502	000000	WCSEL: .WORD	0	:IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
				:IF EQ TO 0, GENERATE A RANDOM WORD COUNT : FOR THE OPERATION.
				:IF NOT EQ TO 0, USE THE VALUE IN 'MAXDL' FOR : THE WORD COUNT
001504	000003	RATIO: .WORD	3	:READ/WRITE RATIO [RANGE 0 - 7] :0 - 0/8 (READ/WRITE) :1 - 7/1 :2 - 6/2 :3 - 5/3 :4 - 4/4 :5 - 3/5 :6 - 2/6 :7 - 1/7
001506	000001	AUTOCK: .WORD	1	:IF NOT EQ 0, DO AN APPROPRIATE WRITE :CHECK AFTER EACH WRITE ORDER.
001510	000001	NOTPRT: .WORD	1	:IF EQ 0, SELECT WRITE CHECK ORDERS :RANDOMLY.
				:IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES :ASSOCIATED WITH OPERATOR SPECIFIED :BAD PACK AREAS.
001512	000001	ENDET: .WORD	1	:IF NOT EQ 0, PRINT ERROR MESSAGES RELATING TO :THESE AREAS.
				:IF NOT EQ 0, END OF PASS DETERMINED :BY THE 'WORDS READ' COUNT.
				:IF EQ 0, END OF PASS DETERMINED :BY THE SEEK COUNT.
001514	000000	PATTEN: .WORD	0	:IF EQ 0, RANDOMLY SELECT DATA PATTERN
				:IF NOT EQ 0, SELECT ONE SET OF PATTERN :POINTED BY THE 'PATTEN'.
001516	000000	HEADER: .WORD	0	:IF EQ TO 0, RANDOMLY SELECT DATA BLOCK :ADDRESS. IF NOT EQU 0, SEQUENTIALLY :SELECT DATA BLOCK ADDRESS

.SBTTL VALUES FOR FIRST OPERATION

```
001520 000010      BEGPAT: .WORD 10      ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
001522 000004      BEGCOD: .WORD 4       ;STARTING COMMAND CODE [RANGE 0 - 5]
                                ;0 = WRITE CHECK DATA ('WCKD')
                                ;1 = WRITE CHECK HEADER & DATA ('WCHKHD')
                                ;2 = WRITE DATA ('WRDAT')
                                ;3 = WRITE HEADER & DATA ('WRTHD')
                                ;4 = READ DATA ('RDDAT')
                                ;5 = READ HEADER & DATA ('RDHD')
001524 000400      BEGSIZ: .WORD 400    ;STARTING RECORD SIZE [RANGE 4 - MAXMEM]
```

.SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS

;LIST OF DRIVES PERFORMING COMMANDS

```
001526 000000      ORDERQ: .WORD 0
001530 000000      .WORD 0
001532 000000      .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      ASNLS1: .WORD 0      ;A BIT SET IS AN ASSIGNED DRIVE
```

;ADDRESSES OF DRIVES TO BE DEASSIGNED

```
001552 000000      DUNIT: .WORD 0
001554 000000      .WORD 0
001556 000000      .WORD 0
001560 000000      .WORD 0
001562 000000      .WORD 0
001564 000000      .WORD 0
001566 000000      .WORD 0
001570 000000      .WORD 0
001572 000000      .WORD 0
```

;ADDRESSES OF NEWLY ASSIGNED DRIVES

```
001574 000000      NEWUNT: .WORD 0
001576 000000      .WORD 0
001600 000000      .WORD 0
001602 000000      .WORD 0
001604 000000      .WORD 0
001606 000000      .WORD 0
001610 000000      .WORD 0
001612 000000      .WORD 0
001614 000000      .WORD 0
```

;LIST OF DRIVES WAITING FOR BUFFERS/PARAMETERS

```
001616 000000      AVAIL: .WORD 0
001620 000000      .WORD 0
001622 000000      .WORD 0
001624 000000      .WORD 0
001626 000000      .WORD 0
001630 000000      .WORD 0
001632 000000      .WORD 0
001634 000000      .WORD 0
```

```

001636 000000 .WORD 0

;LIST OF DRIVES WAITING FOR BUFFERS
WAIT:
001640 000000 .WORD 0
001642 000000 .WORD 0
001644 000000 .WORD 0
001646 000000 .WORD 0
001650 000000 .WORD 0
001652 000000 .WORD 0
001654 000000 .WORD 0
001656 000000 .WORD 0
001660 000000 .WORD 0
    
```

```

;LIST OF DRIVES WAITING FOR NEXT PARAMETERS
PARQ:
001662 000000 .WORD 0
001664 000000 .WORD 0
001666 000000 .WORD 0
001670 000000 .WORD 0
001672 000000 .WORD 0
001674 000000 .WORD 0
001676 000000 .WORD 0
001700 000000 .WORD 0
001702 000000 .WORD 0
    
```

```

;BUFFER ALLOCATION TABLE ENTRY COUNT
BUFTBL:
001704 000000 .WORD 0
001706 000000 000000 .WORD 0.0
001712 000000 000000 .WORD 0.0
001716 000000 000000 .WORD 0.0
001722 000000 000000 .WORD 0.0
001726 000000 000000 .WORD 0.0
001732 000000 000000 .WORD 0.0
001736 000000 000000 .WORD 0.0
001742 000000 000000 .WORD 0.0
001746 000000 000000 .WORD 0.0
001752 000000 000000 .WORD 0.0
001756 000000 000000 .WORD 0.0
001762 000000 000000 .WORD 0.0
001766 000000 000000 .WORD 0.0
001772 000000 000000 .WORD 0.0
001776 000000 000000 .WORD 0.0
002002 000000 000000 .WORD 0.0
002006 000000 000000 .WORD 0.0
002012 000000 000000 .WORD 0.0
002016 000000 000000 .WORD 0.0
002022 000000 000000 .WORD 0.0
002026 000000 000000 .WORD 0.0
002032 000000 000000 .WORD 0.0
002036 000000 000000 .WORD 0.0
002042 000000 000000 .WORD 0.0
002046 000000 000000 .WORD 0.0
002052 000000 000000 .WORD 0.0
002056 000000 000000 .WORD 0.0
002062 000000 000000 .WORD 0.0
002066 000000 000000 .WORD 0.0
002072 000000 000000 .WORD 0.0
002076 000000 000000 .WORD 0.0
    
```

```

002102 000000 000000          .WORD 0,0
002106 044210          BI KADR: .WORD DRIVE0 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 0
002110 046376          .WORD DRIVE1 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 1
002112 050564          .WORD DRIVE2 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 2
002114 052752          .WORD DRIVE3 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 3
002116 055140          .WORD DRIVE4 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 4
002120 057326          .WORD DRIVE5 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 5
002122 061514          .WORD DRIVE6 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 6
002124 063702          .WORD DRIVE7 ;ADDRESS OF THE PARAMETER BLOCK FOR DRIVE 7
    
```

```

002126 151          COMTBL: .BYTE WCKD ;WRITE CHECK DATA
002127 153          .BYTE WCKHD ;WRITE CHECK HEADER AND DATA
002130 161          .BYTE WRTDAT ;WRITE DATA
002131 163          .BYTE WRTHD ;WRITE HEADER AND DATA
002132 171          .BYTE RDDAT ;READ DATA
002133 173          .BYTE RDHD ;READ HEADER AND DATA
    
```

```

002134 002          OPTBL: .BYTE 2 ;UNLOAD
002135 004          .BYTc 4 ;SEEK
002136 006          .BYTE 6 ;RECAL
002137 010          .BYTE 10 ;DRIVE CLEAR
002140 012          .BYTE 12 ;RELEASE
002141 014          .BYTE 14 ;OFFSET
002142 016          .BYTE 16 ;RETURN TO CENTERLINE
002143 020          .BYTE 20 ;READIN PRESET
002144 022          .BYTE 22 ;PACK ACKNOWLEDGE
002145 030          .BYTE 30 ;SEARCH
002146 050          .BYTE 50 ;WRITE CHECK DATA
002147 052          .BYTE 52 ;WRITE CHECK HEADER AND DATA
002150 060          .BYTE 60 ;WRITE DATA
002151 062          .BYTE 62 ;WRITE HEADER AND DATA
002152 070          .BYTE 70 ;READ DATA
002153 072          .BYTE 72 ;READ HEADER AND DATA
002154 377          .BYTE -1 ;TERMINATOR
    
```

.EVEN

```

002156 125 116 114 MNTBL: .ASCIZ /UNLOAD /
002166 123 105 105 .ASCIZ /SEEK /
002176 122 105 103 .ASCIZ /RECAL /
002206 104 122 126 .ASCIZ /DRVCLR /
002216 122 105 114 .ASCIZ /RELSE /
002226 117 106 106 .ASCIZ /OFFSET /
002236 122 124 103 .ASCIZ /RTC /
002246 122 105 101 .ASCIZ /READIN /
002256 120 101 103 .ASCIZ /PACK /
002266 123 105 101 .ASCIZ /SEARCH /
002276 127 103 113 .ASCIZ /WCKD /
002306 127 103 113 .ASCIZ /WCKHD /
002316 127 122 124 .ASCIZ /WRTDAT /
002326 127 122 124 .ASCIZ /WRTHD /
002336 122 104 104 .ASCIZ /RDDAT /
002346 122 104 110 .ASCIZ /RDHD /
002356 116 117 116 .ASCIZ /NONE /
    
```


002366 101 106 124 OFMSG0: .ASCIZ /AFTER RETRY WITHOUT OFFSET/
 002421 101 124 040 OFMSG1: .ASCIZ /AT NEGATIVE OFFSET/
 002444 101 124 040 OFMSG2: .ASCIZ /AT POSITIVE OFFSET/

.EVEN

002470 000 OFFCOD: .BYTE 0 ;OFFSET CODE TABLE
 002471 001 .BYTE 1 ;NUMBER FOR NEGATIVE OFFSET (DIR = OUT)
 002472 000 .BYTE 0 ;NUMBER FOR POSITIVE OFFSET (DIR = IN)

.EVEN

002474 002366 OFMTBL: .WORD OFMSG0 ;1ST OFFSET MESSAGE
 002476 002421 .WORD OFMSG1 ;2ND OFFSET MESSAGE
 002500 002444 .WORD OFMSG2 ;3RD OFFSET MESSAGE

.SBTTL DATA PATTERNS

:STANDARD DATA PATTERN POINTER TABLE

002502 002546 STNDAT: .WORD DATA0 ;STANDARD DATA PATTERN 0
 002504 002606 .WORD DATA1 ;STANDARD DATA PATTERN 1
 002506 002646 .WORD DATA2 ;STANDARD DATA PATTERN 2
 002510 002706 .WORD DATA3 ;STANDARD DATA PATTERN 3
 002512 002746 .WORD DATA4 ;STANDARD DATA PATTERN 4
 002514 003006 .WORD DATA5 ;STANDARD DATA PATTERN 5
 002516 003046 .WORD DATA6 ;STANDARD DATA PATTERN 6
 002520 003106 .WORD DATA7 ;STANDARD DATA PATTERN 7
 002522 003146 .WORD DATA8 ;STANDARD DATA PATTERN 8
 002524 003206 .WORD DATA9 ;STANDARD DATA PATTERN 9
 002526 003246 .WORD DATA10 ;STANDARD DATA PATTERN 10
 002530 003306 .WORD DATA11 ;STANDARD DATA PATTERN 11
 002532 003346 .WORD DATA12 ;STANDARD DATA PATTERN 12
 002534 003406 .WORD DATA13 ;STANDARD DATA PATTERN 13
 002536 003446 .WORD DATA14 ;STANDARD DATA PATTERN 14
 002540 003506 .WORD DATA15 ;STANDARD DATA PATTERN 15
 002542 002546 .WORD ZEROS ;ALL 0'S PATTERN
 002544 003450 .WORD ONES ;ALL 1'S PATTERN

ZEROS:
 DATA0: .WORD 0 ;ALL 0'S DATA PATTERN

002546
 002546 000000
 002550 000000
 002552 000000
 002554 000000
 002556 000000
 002560 000000
 002562 000000
 002564 000000
 002566 000000
 002570 000000
 002572 000000
 002574 000000
 002576 000000
 002600 000000
 002602 000000
 002604 000000

DATA1: .WORD 000001 ;STANDARD PATTERN 1

002606 000001

002610	000003	.WORD	000003
002612	000007	.WORD	000007
002614	000017	.WORD	000017
002616	000037	.WORD	000037
002620	000077	.WORD	000077
002622	000177	.WORD	000177
002624	000377	.WORD	000377
002626	000777	.WORD	000777
002630	001777	.WORD	001777
002632	003777	.WORD	003777
002634	007777	.WORD	007777
002636	017777	.WORD	017777
002640	037777	.WORD	037777
002642	077777	.WORD	077777
002644	177777	.WORD	177777

002646	177776	DATA2: .WORD	177776	; STANDARD PATTERN 2
002650	177774	.WORD	177774	
002652	177770	.WORD	177770	
002654	177760	.WORD	177760	
002656	177740	.WORD	177740	
002660	177700	.WORD	177700	
002662	177600	.WORD	177600	
002664	177400	.WORD	177400	
002666	177000	.WORD	177000	
002670	176000	.WORD	176000	
002672	174000	.WORD	174000	
002674	170000	.WORD	170000	
002676	160000	.WORD	160000	
002700	140000	.WORD	140000	
002702	100000	.WORD	100000	
002704	000000	.WORD	000000	

002706	000000	DATA3: .WORD	000000	; STANDARD PATTERN 3
002710	000000	.WORD	000000	
002712	000000	.WORD	000000	
002714	177777	.WORD	177777	
002716	177777	.WORD	177777	
002720	177777	.WORD	177777	
002722	000000	.WORD	000000	
002724	000000	.WORD	000000	
002726	177777	.WORD	177777	
002730	177777	.WORD	177777	
002732	000000	.WORD	000000	
002734	177777	.WORD	177777	
002736	000000	.WORD	000000	
002740	177777	.WORD	177777	
002742	000000	.WORD	000000	
002744	177777	.WORD	177777	

002746	133331	DATA4: .WORD	133331	; STANDARD PATTERN 4
002750	133331	.WORD	133331	
002752	133331	.WORD	133331	
002754	133331	.WORD	133331	
002756	133331	.WORD	133331	
002760	133331	.WORD	133331	
002762	133331	.WORD	133331	

002764	133331	.WORD	133331
002766	133331	.WORD	133331
002770	133331	.WORD	133331
002772	133331	.WORD	133331
002774	133331	.WORD	133331
002776	133331	.WORD	133331
003000	133331	.WORD	133331
003002	133331	.WORD	133331
003004	133331	.WORD	133331

003006	052525	DATA5: .WORD	052525	; STANDARD PATTERN 5
003010	052525	.WORD	052525	
003012	052525	.WORD	052525	
003014	125252	.WORD	125252	
003016	125252	.WORD	125252	
003020	125252	.WORD	125252	
003022	052525	.WORD	052525	
003024	052525	.WORD	052525	
003026	125252	.WORD	125252	
003030	125252	.WORD	125252	
003032	052525	.WORD	052525	
003034	125252	.WORD	125252	
003036	052525	.WORD	052525	
003040	125252	.WORD	125252	
003042	052525	.WORD	052525	
003044	125252	.WORD	125252	

003046	155554	DATA6: .WORD	155554	; STANDARD PATTERN 6
003050	155554	.WORD	155554	
003052	155554	.WORD	155554	
003054	155554	.WORD	155554	
003056	155554	.WORD	155554	
003060	155554	.WORD	155554	
003062	155554	.WORD	155554	
003064	155554	.WORD	155554	
003066	155554	.WORD	155554	
003070	155554	.WORD	155554	
003072	155554	.WORD	155554	
003074	155554	.WORD	155554	
003076	155554	.WORD	155554	
003100	155554	.WORD	155554	
003102	155554	.WORD	155554	
003104	155554	.WORD	155554	

003106	026455	DATA7: .WORD	026455	; STANDARD PATTERN 7
003110	026455	.WORD	026455	
003112	026455	.WORD	026455	
003114	151322	.WORD	151322	
003116	151322	.WORD	151322	
003120	151322	.WORD	151322	
003122	026455	.WORD	026455	
003124	026455	.WORD	026455	
003126	151322	.WORD	151322	
003130	151322	.WORD	151322	
003132	026455	.WORD	026455	
003134	151322	.WORD	151322	
003136	026455	.WORD	026455	

003140 151322 .WORD 151322
003142 026455 .WORD 026455
003144 151322 .WORD 151322

003146 066666
003150 066666
003152 066666
003154 066666
003156 066666
003160 066666
003162 066666
003164 066666
003166 066666
003170 066666
003172 066666
003174 066666
003176 066666
003200 066666
003202 066666
003204 066666

DATA8: .WORD 066666 ;STANDARD PATTERN 8
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666
.WORD 066666

003206 000001
003210 000002
003212 000004
003214 000010
003216 000020
003220 000040
003222 000100
003224 000200
003226 000400
003230 001000
003232 002000
003234 004000
003236 010000
003240 020000
003242 040000
003244 100000

DATA9: .WORD 000001 ;STANDARD PATTERN 9
.WORD 000002
.WORD 000004
.WORD 000010
.WORD 000020
.WORD 000040
.WORD 000100
.WORD 000200
.WORD 000400
.WORD 001000
.WORD 002000
.WORD 004000
.WORD 010000
.WORD 020000
.WORD 040000
.WORD 100000

003246 177776
003250 177775
003252 177773
003254 177767
003256 177757
003260 177737
003262 177677
003264 177577
003266 177377
003270 176777
003272 175777
003274 173777
003276 167777
003300 157777
003302 137777
003304 077777

DATA10: .WORD 177776 ;STANDARD PATTERN 10
.WORD 177775
.WORD 177773
.WORD 177767
.WORD 177757
.WORD 177737
.WORD 177677
.WORD 177577
.WORD 177377
.WORD 176777
.WORD 175777
.WORD 173777
.WORD 167777
.WORD 157777
.WORD 137777
.WORD 077777

003306 172666
003310 155555

DATA11: .WORD 172666 ;STANDARD PATTERN 11
.WORD 155555

003312	172666	.WORD	172666
003314	155555	.WORD	155555
003316	172666	.WORD	172666
003320	155555	.WORD	155555
003322	172666	.WORD	172666
003324	155555	.WORD	155555
003326	172666	.WORD	172666
003330	155555	.WORD	155555
003332	172666	.WORD	172666
003334	155555	.WORD	155555
003336	172666	.WORD	172666
003340	155555	.WORD	155555
003342	172666	.WORD	172666
003344	155555	.WORD	155555

003346	077777	DATA12: .WORD	077777	; STANDARD PATTERN 12
003350	137777	.WORD	137777	
003352	157777	.WORD	157777	
003354	167777	.WORD	167777	
003356	173777	.WORD	173777	
003360	175777	.WORD	175777	
003362	176777	.WORD	176777	
003364	177377	.WORD	177377	
003366	177577	.WORD	177577	
003370	177677	.WORD	177677	
003372	177737	.WORD	177737	
003374	177757	.WORD	177757	
003376	177767	.WORD	177767	
003400	177773	.WORD	177773	
003402	177775	.WORD	177775	
003404	177776	.WORD	177776	

003406	153333	DATA13: .WORD	153333	; STANDARD PATTERN 13
003410	066667	.WORD	066667	
003412	153333	.WORD	153333	
003414	066667	.WORD	066667	
003416	153333	.WORD	153333	
003420	066667	.WORD	066667	
003422	153333	.WORD	153333	
003424	066667	.WORD	066667	
003426	153333	.WORD	153333	
003430	066667	.WORD	066667	
003432	153333	.WORD	153333	
003434	066667	.WORD	066667	
003436	153333	.WORD	153333	
003440	066667	.WORD	066667	
003442	153333	.WORD	153333	
003444	066667	.WORD	066667	

003446	000000	DATA14: .WORD	000000	; STANDARD PATTERN 14
003450	177777	ONES: .WORD	177777	; ALL 1'S DATA PATTERN
003452	177777	.WORD	177777	
003454	177777	.WORD	177777	
003456	177777	.WORD	177777	
003460	177777	.WORD	177777	
003462	177777	.WORD	177777	
003464	177777	.WORD	177777	

003466	177777	.WORD	177777
003470	177777	.WORD	177777
003472	177777	.WORD	177777
003474	177777	.WORD	177777
003476	177777	.WORD	177777
003500	177777	.WORD	177777
003502	177777	.WORD	177777
003504	177777	.WORD	177777

003506	177777	DATA15: .WORD	177777	; STANDARD PATTERN 15
003510	000000	.WORD	000000	
003512	000000	.WORD	000000	
003514	000000	.WORD	000000	
003516	000000	.WORD	000000	
003520	000000	.WORD	000000	
003522	000000	.WORD	000000	
003524	000000	.WORD	000000	
003526	000000	.WORD	000000	
003530	000000	.WORD	000000	
003532	000000	.WORD	000000	
003534	000000	.WORD	000000	
003536	000000	.WORD	000000	
003540	000000	.WORD	000000	
003542	000000	.WORD	000000	
003544	000000	.WORD	000000	

0

.SBTTL ERROR POINTER TABLE

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

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

\$ERRTB:

:ERROR 1

1				
2				
3				
4	003546	066160	EM1	:RH70 INTERRUPT OCCURRED (RMAS = 0)
5	003550	070613	DH1	
6	003552	071244	DT1	
7	003554	000000	0	

:ERROR 2

8				
9				
10				
11	003556	066230	EM2	:UNEXPECTED ATTENTION OCCURRED
12	003560	070620	DH2	
13	003562	071250	DT2	
14	003564	000000	0	

:ERROR 3

15				
16				
17				
18	003566	066266	EM3	:MASSBUS PARITY ERROR (MCPE=1)
19	003570	070674	DH3	
20	003572	071266	DT3	
21	003574	000000	0	

:ERROR 4

22				
23				
24				
25	003576	066324	EM4	:MASSBUS PARITY ERROR (PAR=1)
26	003600	070722	DH4	
27	003602	071276	DT4	
28	003604	000000	0	

:ERROR 5

29				
30				
31				
32	003606	066361	EM5	:ADDRESS PLUG BIT CHANGED
33	003610	070620	DH2	
34	003612	071250	DT2	
35	003614	000000	0	

:ERROR 6

36				
37				
38				
39	003616	066415	EM6	:RH/RM DIDN'T RESPOND TO ADDRESSING
40	003620	070761	DH6	
41	003622	071310	DT6	
42	003624	000000	0	


```

1          .SBTTL PROGRAM START
2
3 003626 012737 177777 001360 START: MOV    #-1,CHGADR    ;SET RH/RM ADDRESS CHANGE FLAG
4 003634 000407                                BR      START2    ;START THE PROGRAM
5
6 003636 012737 000400 001360 START1: MOV    #400,CHGADR    ;CLEAR THE RH/RM ADDRESS CHANGE FLAG
7          ;*****
8          003644 000240 TST1:  NOP
9          003646 012737 000001 001212 MOV    #1,$TESTN    ;;SET TEST NUMBER IN APT MAIL BOX
10         003654 005227 000000 START2: INC    #0      ;TTY LOOP, WAIT FOR INCREMENT
11         003660 001375 BNE     .-4      ;OF WORD
12         003662 000005 RESET   .-4      ;CLEAR THE WORLD
13
          .SBTTL INITIALIZE THE COMMON TAGS
          ;;CLEAR THE COMMON TAGS ($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    # $TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
          MOV    #340,@TRAPVEC+2 ;;LEVEL 7
          MOV    # $PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
          MOV    #340,@PWRVEC+2 ;;LEVEL 7
          MOV    #176543,$SHNUM  ;;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
          65$: MOV    #SWREG,SWR  ;;POINT TO SOFTWARE SWR
          MOV    #DISPREG,DISPLAY
          66$: MOV    (SP)+,@ERRVEC ;;RESTORE ERROR VECTOR
          CLR    $PASS         ;;CLEAR PASS COUNT
          BITB  #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
          BEQ  67$           ;;YES,USE NON-APT SWITCH
          MOV  # $SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
          67$:
14         .SBTTL TYPE PROGRAM NAME
          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
          INC    #-1          ;;FIRST TIME?
          BNE   68$         ;;BRANCH IF NO
          TYPE  ,69$        ;;TYPE ASCIZ STRING
          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER

```

```

004106 005737 000042.          TST      @#42          ;;ARE WE RUNNING UNDER XXDP/ACT?
004112 001012                    BNE      70$          ;;BRANCH IF YES
004114 123727 001226 000001    CMPB    $ENV,#1      ;;ARE WE RUNNING UNDER APT?
004122 001406                    BEQ      70$          ;;BRANCH IF YES
004124 023727 001154 000176    CMP     SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
004132 001005                    BNE      71$          ;;BRANCH IF NO
004134 104406                    GTSWR                    ;;GET SOFT-SWR SETTINGS
004136 000403                    BR       71$
004140 112737 000001 001150 70$:  MOVB    #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
004146 000426                    71$:  BR       68$
004224                    ;;69$: .ASCIZ <CRLF>@CZRMUAO - RM05/3/2 PERFORMANCE EXERCISER@<CRLF>
004224                    68$:

15
16                                ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
17                                ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
18
19 004224 005037 001452          CLR      XXDP          ;CLEAR 'XXDP' LOAD DEVICE STORAGE
20 004230 122737 000016 000041  CMPB    #16,@#41     ;LOADED FROM AN RM05/3/2 ?
21 004236 001160                    BNE      3$           ;BR IF NOT
22 004240 013737 000040 001452  MOV     @#40,XXDP    ;GET DEVICE INDICATOR AND NUMBER
23 004246 122737 000007 001452  CMPB    #7,XXDP     ;IS IT A VALID NUMBER ?
24 004254 103002                    BHIS    1$           ;YES
25 004256 105037 001452          CLRB    XXDP          ;NO, DEFAULT TO DRIVE 0
26 004262 005737 000042          TST     @#42          ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
27 004266 001425                    BEQ     2$           ;BR IF NEITHER
28 004270 104401 004276          TYPE    ,73$         ;;TYPE ASCIZ STRING
004274 000412                    BR      72$         ;;GET OVER THE ASCIZ
004322                    ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
004322                    72$:

29 004322 005046                    CLR      -(SP)        ;CLEAR WORD ON STACK
30 004324 113716 001452          MOVB    XXDP,(SP)   ;GET DRIVE ADDRESS
31 004330 104403                    TYPOS                    ;TYPE THE ADDRESS
32 004332 001                    .BYTE  1            ;ONLY 1 CHARACTER
33 004333 000                    .BYTE  0            ;SUPRESS LEADING ZEROS
34 004334 104401 001203          TYPE    ,5$CRLF     ;CR-LF
35 004340 000517                    BR      3$           ;GET NUMBER OF DRIVES
36
37 004342 005227 177777          2$:  INC     #-1        ;FIRST TIME THRU HERE ?
38 004346 001114                    BNE     3$           ;NO
39 004350 104401 004356          TYPE    ,75$         ;;TYPE ASCIZ STRING
004354 000410                    BR      74$         ;;GET OVER THE ASCIZ
004376                    ;;75$: .ASCIZ <CRLF>/TO TEST DRIVE /
004376                    74$:

40 004376 005046                    CLR      -(SP)        ;CLEAR WORD ON STACK
41 004400 113716 001452          MOVB    XXDP,(SP)   ;GET DRIVE ADDRESS
42 004404 104403                    TYPOS                    ;TYPE DRIVE ADDRESS
43 004406 001                    .BYTE  1            ;ONLY 1 CHARACTER
44 004407 000                    .BYTE  0            ;SUPRESS LEADING ZEROS
45 004410 104401 004416          TYPE    ,77$         ;;TYPE ASCIZ STRING
004414 000431                    BR      76$         ;;GET OVER THE ASCIZ
004500                    ;;77$: .ASCIZ /, HALT PROGRAM, REMOVE RRDP PACK AND REPLACE IT/<CRLF>
004500                    76$:
46 004500 104401 004506          TYPE    ,78$         ;;TYPE ASCIZ STRING
004504 000435                    BR      3$           ;;GET OVER THE ASCIZ
004600                    ;;78$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
004600                    3$:

```

```

50 004600 004737 031604
51 .SBTTL JSR PC,$TKINT ;TURN ON THE KEYBOARD INTERRUPT
      GET VALUE FOR SOFTWARE SWITCH REGISTER
      TST #42 ;:ARE WE RUNNING UNDER XXDP/ACT?
      BNE 79$ ;:BRANCH IF YES
      CMPB $ENV,#1 ;:ARE WE RUNNING UNDER APT?
      BEQ 79$ ;:BRANCH IF YES
      CMP SWR,#SWREG ;:SOFTWARE SWITCH REG SELECTED?
      BNE 80$ ;:BRANCH IF NO
      GTSWR ;:GET SOFT-SWR SETTINGS
      BR 80$
      MOV #1,$AUTOB ;:SET AUTO-MODE INDICATOR
79$:
80$:
52 004644 105737 001226 TSTB $ENV ;:RUN UNDER APT MODE
53 004650 001423 BEQ 4$ ;:NO,DO NOT BOTHER
54 004652 105737 001256 TSTB $VECT1 ;:NEW VECTOR ?
55 004656 001403 BEQ .+10 ;:NOT LOAD IF = 0
56 004660 113737 001256 001274 MOVB $VECT1,$RMVEC ;:NEW VECTOR
57 004666 005737 001262 TST $BASE ;:NEW BASE ADDRESS ?
58 004672 001403 BEQ .+10 ;:NO
59 004674 013737 001262 001272 MOV $BASE,$RMADR ;:NEW BASE ADDRESS
60 004702 013737 001272 035762 MOV $RMADR,$RMADR ;:LOAD ADDRESS INTO DRIVER
61 004710 013737 001274 035764 MOV $RMVEC,$RMVEC ;:LOAD VECTOR INTO DRIVER
62 004716 000420 BR 5$
63
64 004720 105737 001150 4$: TSTB $AUTOB ;:AUTO MODE ?
65 004724 001015 BNE 5$ ;:YES
66 004726 004737 076500 JSR PC,BUSADR ;:CHECK RH/RM BUS ADDRESS
67 004732 013737 001272 035762 MOV $RMADR,$RMADR ;:RH/RM ADDRESS
68 004740 013737 001274 035764 MOV $RMVEC,$RMVEC ;:RH/RM VECTOR ADDRESS
69 004746 005227 177777 INC #-1 ;:FIRST TIME THRU HERE ?
70 004752 001002 BNE 5$ ;:BR IF NO
71 004754 004737 076160 JSR PC,OPRDAT ;:GET THE DATE AND OPERATOR ID
72 004760 005037 001316 5$: CLR STATIN ;:CLEAR PERFORMANCE SUMMARY TYPEOUT FLAG
73 004764 012705 001526 MOV #ORDERQ,$R5 ;:START OF AREA TO CLEAR
74 004770 005025 6$: CLR (R5)+
75 004772 022705 002106 CMP #BLKADR,$R5 ;:LOOK FOR END OF CLEAR AREA
76 004776 001374 BNE 6$ ;:BR IF NOT FINISHED
77 005000 012706 001100 MOV #STACK,$SP ;:SETUP THE STACK POINTER
78 005004 005037 177776 CLR PS ;:CLEAR THE PROCESSOR STATUS WORD
79 005010 013737 001314 001374 MOV HZ,$SIXTEE ;:1/60 TH OR 1/50 TH SECOND COUNTER VALUE
80 005016 005037 001366 CLR HOUR ;:CLEAR THE HOUR'S COUNTER
81 005022 005037 001370 CLR MINUTE ;:CLEAR THE MINUTE'S COUNTER
82 005026 005037 001372 CLR SECOND ;:CLEAR THE SECOND'S COUNTER
83 005032 005037 001474 CLR INTRVL+2 ;:CLEAR INTERVAL COUNTER
84 005036 005037 001320 CLR PACK ;:CLEAR THE 'R' OR 'W' COMMAND FLAG
85 005042 005037 001362 CLR CFLAG ;:CLEAR THE 'CONTROL C' FLAG
86 005046 042737 170000 001470 BIC #170000,$MAXER ;:MAKE SURE ERROR LIMITS ARE NOT TOO HIGH
87
88 ;ROUTINE TO DETERMINE BUFFER AREA SIZE
89
90 005054 005227 177777 SIZMEM: INC #-1 ;:FIRST TIME THRU HERE ?
91 005060 001005 BNE 1$ ;:BR IF NO
92 005062 004737 076346 JSR PC,$SIZE ;:SEE HOW MUCH MEMORY ON SYSTEM
93 005066 013737 076476 001356 MOV $LSTAD,$LSTAD ;:SAVE THE LAST ADDRESS
94 005074 012737 000001 001704 1$: MOV #1,$BUFTBL ;:LOAD NUMBER OF BUFFERS
95 005102 012737 077752 001706 MOV #ENDPGM,$BUFTBL+2 ;:STARTING ADDRESS OF BUFFER
96 005110 013737 001356 001710 MOV $LSTAD,$BUFTBL+4 ;:LAST ADDR TO BUFFER ALLOCATION TABLE

```

```

97 005114 023727 001356 160000      CMP      LSTAD,#160000      :OVER 28K ?
98 005124 101403                    BLOS     2$              :NO
99 005126 012737 160000 001710      MOV      #160000,BUFTBL+4 :XXDP MAX MEMORY 28K
100 005134 162737 077752 001710 2$:  SUB      #ENDPGM,BUFTBL+4 :SUBTRACT PROGRAM SPACE
101 005142 000241                    CLC                      :CLEAR THE 'C' BIT
102 005144 006037 001710            RGR      BUFTBL+4        :CONVERT TO WORD COUNT
103 005150 162737 000144 001710      SUB      #100.,BUFTBL+4  :SAVE ROOM FOR THE 'ABS' LOADER
104 005156 005737 000042            TST      42              :LOAD FROM XXDP OR OTHER MONITOR ?
105 005162 001403                    BEQ      3$              :BR IF LOADED BY PAPER TAPE
106 005164 162737 002570 001710      SUB      #1400.,BUFTBL+4 :SUBTRACT 'XXDP' LOADER SIZE
107 005172 005737 001466            3$:  TST      MAXDL        :VALUE IN 'MAXDL' ?
108 005176 001003                    BNE      4$              :BR IF VALUE IS
109 005200 012737 020000 001466      MOV      #8192.,MAXDL    :ASSUME FULL TRACK MAXIMUM
110 005206 023737 001466 001710 4$:  CMP      MAXDL,BUFTBL+4  :IS THAT TOO LARGE ?
111 005214 103403                    BLO      5$              :BR IF NOT
112 005216 013737 001710 001466      MOV      BUFTBL+4,MAXDL  :USE MAX AVAIL MEMORY AS MAX BUFFER SIZE
113 005224 013737 001710 075122 5$:  MOV      BUFTBL+4,PARLST+2 :VALUE FOR THE PARAMETER TABLE
114
115                                     :SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS
116
117 005232 105737 001150      L&PAR: TSTB     $AUTOB      :'XXDP' CHAIN MODE OR 'ACT11' OPERATION ?
118 005236 001044                    BNE      SETVEC         :BR IF YES
119 005240 022737 007070 001266      CMP      #7070,$CDW1    :FROM THE POWER FAIL RT. ?
120 005246 001440                    BEQ      SETVEC         :BRANCH IF SO
121 005250 105737 001226      TSTB     $ENV           :APT STAND ALONE MODE ?
122 005254 001035                    BNE      SETVEC         :NO
123 005256 104401 075222            TYPE     ,ASKPAR        :ASK FOR PARAMETERS
124 005262 104410                    RDCHR                    :READ THE ENTRY
125 005264 012637 001174            MOV      (SP)+,$TMP0    :SAVE RESPONSE
126 005270 122737 000131 001174      CMPB     #'Y',$TMP0     :WAS IT A YES RESPONSE ?
127 005276 001405                    BEQ      1$              :BR IF YES
128 005300 104401 074244            TYPE     ,N             :TYPE 'N'
129 005304 104401 001203            TYPE     ,$CRLF        :CR-LF
130 005310 000417                    BR                      :
131 005312 104401 001174            1$:  TYPE     , $TMP0      :TYPE 'Y'
132 005316 104401 001203            TYPE     , $CRLF        :CR-LF
133
134 005322 012703 075120      ENTPR: MOV      #PARLST,R3 :PARAMETER TABLE ADDRESS
135 005326 004737 027702            JSR      PC,PARENT     :GET THE PARAMETER ENTRY
136 005332 023727 001466 000004      CMP      MAXDL,#4      :IS THE 'MAXDL' VALUE OK ?
137 005340 103003                    BHS      SETVEC         :BR IF IT IS
138 005342 012737 000004 001466      MOV      #4,MAXDL      :SET 'MAXDL' TO THE MINIMUM VALUE
139
140                                     :DISPLAY DRIVE STATUS AND SET UP THE OTHER SYSTEM DEVICES THAT
141                                     :THE PROGRAM WILL USE
142
143 005350 004737 023326      SETVEC: JSR      PC,CKCLK :START THE CLOCK
144 005354 004737 036000            JSR      PC,RMINIT     :INITIALIZE THE RM DRIVER
145 005360 012737 177777 035722      MOV      #-1,SAVEFG    :SET THE SAVE REGISTERS FLAG
146 005366 005227 177777            INC      #-1           :FIRST TIME THRU ?
147 005372 001404                    BEQ      1$              :BR IF YES
148 005374 032777 000004 173552      BIT      #SW02,@SWR    :TYPEOUT THE DRIVE STATUS TABLE ?
149 005402 001104                    BNE      12$            :BR IF NOT
150 005404 005004            1$:  CLR      R4            :DRIVE TABLE POINTER
151 005406 104401 073535            TYPE     ,SYSTAT       :TYPE STATUS HEADING
152 005412 104401 001203            2$:  TYPE     , $CRLF        :CR-LF
153 005416 010446            MOV      R4,-(SP)      :SAVE R4 FOR TYPEOUT
    
```

```

005420 104403          TYPOS          ::TYPE DRIVE NUMBER
005422          002      .BYTE 2          ::GO TYPE--OCTAL ASCII
005423          000      .BYTE 0          ::TYPE 2 DIGIT(S)
154 005424 104401 073266 TYPE ,BLNKS4    ::SUPPRESS LEADING ZEROS
155 005430 105764 035634 TSTB DRVSTA(R4)  ::TYPE 4 BLANKS
156 005434 100416      BMI 5$          ::CHECK DRIVE'S STATUS
157 005436 001020      BNE 6$          ::BR IF UNSAFE
158                                     ::BR IF ONLINE
159 005440 105764 035644 TSTB DRV TYP(R4)  ::SEE IF OFFLINE OR NONEXISTENT
160 005444 001404      BEQ 3$          ::BR IF NONEXISTENT
161 005446 100006      BPL 4$          ::BR IF OFFLINE
162 005450 104401 073430 TYPE ,NOTRM      ::DRIVE NOT AN RM05/3/2
163 005454 000451      BR 11$         ::CHECK NEXT DRIVE
164
165 005456 104401 073451 3$: TYPE ,NOTPRS   ::DRIVE NOT PRESENT
166 005462 000446      BR 11$         ::CHECK NEXT DRIVE
167
168 005464 104401 073337 4$: TYPE ,UNTOFF   ::DRIVE OFFLINE
169 005470 000416      BR 8$          ::PRINT DRIVE TYPE
170
171 005472 104401 073505 5$: TYPE ,NOTSAF   ::DRIVE UNSAFE
172 005476 000413      BR 8$          ::PRINT DRIVE TYPE
173
174 005500 005737 001452 6$: TST XXDP      ::LOADED FROM THIS DEVICE ?
175 005504 001406      BEQ 7$          ::BR IF NO
176 005506 123704 001452 CMPB XXDP,R4     ::LOADED FROM THIS DRIVE ?
177 005512 001003      BNE 7$          ::BR IF NO
178 005514 104401 073515 TYPE ,LODEV     ::DRIVE IS LOAD DEVICE
179 005520 000427      BR 11$         ::
180 005522 104401 073350 7$: TYPE ,UNTON    ::DRIVE ONLINE
181 005526 104401 073270 8$: TYPE ,BLNKS2   ::TYPE 2 BLANKS
182 005532 012737 073560 005576 MOV #SRM03,10$  ::ADDRESS OF RM03 MESSAGE
183 005540 122764 000004 035644 CMPB #4,DRV TYP(R4) ::IS DEVICE AN RM03 ?
184 005546 001412      BEQ 9$          ::BR IF YES
185 005550 012737 073553 005576 MOV #SRM02,10$  ::ADDRESS OF RM02 MESSAGE
186 005556 122764 000005 035644 CMPB #5,DRV TYP(R4) ::IS DEVICE AN RM02 ?
187 005564 001403      BEQ 9$          ::BR IF YES
188 005566 012737 073565 005576 MOV #SRM05,10$  ::ADDRESS OF RM05 MESSAGE
189
190 005574 104401 9$: TYPE          ::TYPE THE DRIVE TYPE MESSAGE
191 005576 000000 10$: .WORD 0      ::MESSAGE ADDRESS HERE
192
193 005600 005204 11$: INC R4          ::INCREMENT DRIVE NUMBER/TABLE POINTER
194 005602 020427 000010 CMP R4,#8.     ::FINISHED ?
195 005606 001301      BNE 2$          ::BR IF NOT
196 005610 104401 001203 TYPE ,$CRLF    ::CR-LF
197 005614
198
199 :SETUP IF 'XXDP' OR 'ACT11' OPERATION
200
201 005614 105737 001150 MONTR: TSTB $AUTOB  ::'XXDP' CHAIN MODE OR 'ACT11' AUTO ACCEPT
202 005620 001407      BEQ 1$          ::BR IF NEITHER
203 005622 005737 001360 TST CHGADR     ::200 START ?
204 005626 003004      BGT 1$          ::YES
205 005630 004737 025260 JSR PC,ASGN2   ::ASSIGN DRIVES
206 005634 104401 001203 TYPE ,$CRLF    ::CR-LF
    
```

```

207 005640 004737 031604      1$:   JSR   PC,$TKINT      ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
208
209                               ;FORCE TO TEST PACK FOR 200 START
210 005644 012737 002740 001454 FOWT1: MOV   #2740,ENDCON    ;INITIAL XTRANS COUNT
211 005652 012737 005455 001456      MOV   #5455,ENDCON+2    ;
212 005660 105737 001226      TSTB  $ENV              ;APT SCRIPT MODE, THEN MAKE IT RUN 2 MIN.
213 005664 001412              BEQ   1$                ;NO
214 005666 012737 000001 001504      MOV   #1,RATIO         ;SPEED UP TEST
215 005674 012737 077777 001454      MOV   #77777,ENDCON    ;SET LSW OF ENDING VALUE
216 005702 012737 000027 001456      MOV   #27,ENDCON+2     ;SET MSW OF ENDING VALUE
217 005710 000403              BR    FOWT2            ;START TO WRITE AND TEST
218 005712 005737 001360      1$:   TST   CHGADR        ;START AT 200 ?
219 005716 003464              BLE   FOWT3            ;NO
220
221 005720 005001      FOWT2: CLR   R1              ;DRIVE #
222 005722 005002              CLR   R2              ;AVAIL TABLE INDEX
223 005724 005003              CLR   R3              ;DRIVE# * 2
224 005726 005737 001452      1$:   TST   XXDP         ;LOADED FROM THIS DEVICE ?
225 005732 001403              BEQ   2$                ;BR IF NO
226 005734 123701 001452      CMPB  XXDP,R1          ;LOADED FROM THIS DRIVE ?
227 005740 001442              BEQ   4$                ;BR IF YES
228 005742 105761 035634      2$:   TSTB  DRVSTA(R1)   ;DRIVE ON LINE ?
229 005746 003437              BLE   4$                ;NO
230 005750 016300 002106      MOV   BLKADR(R3),RO    ;LOAD DPB ADDRESS
231 005754 004737 026174      JSR   PC,CLRDPB        ;CLEAR DPB BLOCK
232 005760 004537 026700      JSR   R5,GETADR        ;RETRIEVE BAD SPOT FILE
233 005764 000430              BR    4$                ;UNSUCCESSFUL !
234 005766 010062 001574      MOV   RO,NEWUNT(R2)    ;LOAD DPB ADDRESS TO ABAIL QUEUE
240 005772 112760 177776 000026      MOVB  #-2,$SPACK(RO)   ;WRITE PATTERN THEN TEST
241 006000 013760 001444 000106      MOV   CYLMT,MAXCYL(RO) ;UP CYLINDER LIMIT
242 006006 004737 026546      JSR   PC,GETLMT        ;GET ADDRESS LIMITS
243 006012 013760 001450 000112      MOV   TRKLMT,MAXTRK(RO);UPPER TRACK LIMIT
244 006020 013760 001446 000116      MOV   SECLMT,MAXSEC(RO);UPPER SECTOR LIMIT
245 006026 005060 000110      CLR   MINCYL(RO)       ;CLEAR LOWER LIMIT
246 006032 005060 000114      CLR   MINTRK(RO)       ;CLEAR LOWER LIMIT
247 006036 005060 000120      CLR   MINSEC(RO)       ;CLEAR LOWER LIMIT
248 006042 004737 020374      JSR   PC,WRTPK         ;SET UP PARAMETERS
249
250 006046 022322      4$:   CMP   (R3)+,(R2)+     ;INCREMENT INDEX
251 006050 005201              INC   R1                ;NEXT DRIVE
252 006052 020127 000007      CMP   R1,#7            ;ALL DRIVE ASSIGN ?
253 006056 003723              BLE   1$                ;NO
254 006060 005037 001320      CLR   PACK             ;TEST PACK FLAG
255 006064 000137 006260      JMP   MAIN1            ;JUMP TO WAIT PARAMETER AND BUFFER LOOP
256
257 006070 104401 074610      FOWT3: TYPE  ,INTDON    ;TYPE 'INITIALIZE COMPLETE'
258 006074 000137 006260      JMP   MAIN1            ;START THE PROGRAM
262
    
```

```

1          .SBTTL  MAIN PROGRAM
2
3 006100  005737  001362  MAIN:  TST      CFLAG      ;KEYBOARD INTERRUPTED ?
4 006104  001407                BEQ      3$           ;BRANCH IF NOT
5 006106  005737  001526  1$:   TST      ORDERQ     ;DON'T DEASSIGN ANY DRIVE,IF SYSTEM NOT IDLE
6 006112  001402                BEQ      2$           ;
7 006114  000137  006676                JMP      IDLE        ;LET ALL DRIVE FINISH ORDER
8 006120  004737  024624  2$:   JSR      PC,KSR     ;SERVICE THE KEYBOARD
9 006124  000240                3$:   NOP             ;EXIT
10
11 006126  012703  000010  MAINDA: MOV     #8.,R3     ;DRIVE COUNTER
12 006132  012705  001552                MOV     #DUNIT,R5    ;ADDRESS OF 'DROP DRIVE' TABLE
13 006136  005715  1$:   TST      (R5)       ;SEE IF ENTRY AT PRESENT POSITION
14 006140  001011                BNE     3$           ;BR IF THERE IS ONE
15 006142  062705  000002  2$:   ADD     #2,R5       ;INCREMENT TO NEXT TABLE POSITION
16 006146  005303                DEC     R3           ;DECREMENT DRIVE COUNTER
17 006150  001372                BNE     1$           ;BR IF MORE TO CHECK
18 006152  105737  001550                TSTB   ASNLST       ;ANY DRIVES ACTIVE ?
19 006156  001040                BNE     MAIN1        ;BR IF YES
20 006160  000137  030526                JMP     $GET42       ;GIVE CONTROL TO MONITOR
21
22 006164  012701  001616  3$:   MOV     #AVAIL,R1    ;ADDRESS OF 'AVAILABLE DRIVES' TABLE
23 006170  005711  4$:   TST      (R1)       ;SEE IF AT END OF TABLE
24 006172  001405                BEQ     5$           ;BR IF AT END: GO CHECK 'WAIT' TABLE
25 006174  021115                CMP     (R1),(R5)    ;IS DRIVE IN 'AVAIL' THE ONE TO BE DROPPED
26 006176  001414                BEQ     7$           ;BR IF YES
27 006200  062701  000002  ADD     #2,R1         ;INCREMENT 'AVAIL' TABLE ADDRESS
28 006204  000771                BR      4$           ;CONTINUE LOOKING
29 006206  012701  001640  5$:   MOV     #WAIT,R1     ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
30 006212  005711  6$:   TST      (R1)       ;AT THE END OF THE 'WAIT' TABLE ?
31 006214  001752                BEQ     2$           ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
32 006216  021115                CMP     (R1),(R5)    ;DRIVE IN THE 'WAIT' TABLE ?
33 006220  001403                BEQ     7$           ;BR IF IT IS
34 006222  062701  000002  ADD     #2,R1         ;INCREMENT 'WAIT' TABLE ADDRESS
35 006226  000771                BR      6$           ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
36 006230  011100  7$:   MOV     (R1),R0      ;PUT THE DRIVE'S BLOCK ADDRESS IN R0
37 006232  104401  001203  TYPE   ,SCLF         ;CR-LF
38 006236  104401  074114  TYPE   ,DEASSG       ;TYPE 'DRIVE DEASSIGNED'
39 006242  004737  023616  JSR    PC,SUMARY     ;TYPE THE DRIVE'S PERFORMANCE SUMMARY
40 006246  005015                CLR     (R5)        ;CLEAR THE 'DROP DRIVE' TABLE ENTRY
41 006250  005011                CLR     (R1)        ;REMOVE THE DRIVE FROM THE 'AVAIL' OR 'WAIT' TABLE
42 006252  004737  020356  JSR    PC,COMPRES    ;COMPRESS THE RESPECTIVE TABLE
43 006256  000731                BR      2$           ;SEE IF ANY MORE DRIVES
44
45          ;LOOK FOR DRIVES TO BE ASSIGNED
46
47 006260  012703  000010  MAIN1: MOV     #8.,R3     ;DRIVE COUNT
48 006264  005002                CLR     R2          ;'AVAIL' INDEX
49 006266  005004                CLR     R4          ;ASSIGN LIST INDEX
50 006270  005005                CLR     R5          ;NEW DRIVE INDEX
51 006272  005755  001574  1$:   TST      NEWUNT(R5)   ;NEW DRIVE IN THIS POSITION
52 006276  001006                BNE     3$           ;BR IF THERE IS
53 006300  062705  000002  2$:   ADD     #2,R5       ;INCREMENT R5
54 006304  005204                INC     R4          ;INCREMENT ASSIGN INDEX
55 006306  005303                DEC     R3          ;DECREMENT DRIVE COUNT
56 006310  001770                BNE     1$           ;BR IF MORE DRIVES
57 006312  000432                BR      MAIN2       ;START OPERATIONS FOR THE AVAILABLE DRIVES

```

58	006314	104401	001203	3\$:	TYPE	,\$CR LF	:CR-LF
59	006320	104401	073331		TYPE	,UNTMSG	: 'DRIVE'
60	006324	010446			MOV	R4,-(SP)	::SAVE R4 FOR TYPEOUT
	006326	104403			TYPOS		::TYPE DRIVE NUMBER
	006330	002			.BYTE	2	::GO TYPE--OCTAL ASCII
	006331	000			.BYTE	0	::TYPE 2 DIGIT(S)
61	006332	104401	074162		TYPE	,ASGND	::SUPPRESS LEADING ZEROS
62	006336	005762	001616	4\$:	TST	AVAIL(R2)	: 'STARTED'
63	006342	001403			BEQ	5\$:AT END OF AVAILABLE TABLE
64	006344	062702	000002		ADD	#2,R2	:BR IF YES
65	006350	000772			BR	4\$:INCREMENT AVAILABLE TABLE INDEX
66	006352	016562	001574	001616	5\$:	MOV	:CONTINUE LOOKING FOR END OF TABLE
67	006360	005065	001574		CLR	NEWUNT(R5),AVAIL(R2)	:MOVE ADDR OF DRIVE INTO AVAIL LST
68	006364	156437	035750	001550	CLR	NEWUNT(R5)	:TAKE DRIVE OUT OF NEW DRIVE TABLE
69	006372	062702	000002		BISB	ATABIT(R4),ASNLST	:SET DRIVE ASSIGNED INDICATOR
70	006376	000740			ADD	#2,R2	:INCREMENT AVAILABLE TABLE POINTER
71					BR	2\$:LOOK FOR MORE DRIVES
72							
73							:GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
74							:THE 'AVAILABLE' QUEUE
75	006400				MAIN2:		:SET UP THE WAIT QUEUE FOR
76							:ALL THE DRIVES THAT WAITING
77							:FOR BUFFER IN THREE TIMES OF
78							:SEARCHING
79	006400	005002			CLR	R2	:START FROM THE FIRST LOCATION
80	006402	005762	001640	1\$:	TST	WAIT(R2)	:WAIT FOR THE BUFFER ?
81	006406	001434			BEQ	MAIN3	:EXIT IF NONE
82	006410	016200	001640		MOV	WAIT(R2),R0	:LOAD R0 WITH THE DPB ADDRESS
83	006414	005046			CLR	-(SP)	:CLEAR THE STACK FOR BUFFER REQ
84	006416	004737	016312		JSR	PC,GETBUF	:CALL TO GET THE BUFFER RT.
85	006422	012660	000006		MOV	(SP)+,\$BUF(R0)	:IF 0,BUFFER : STILL NOT AVAILABLE
86	006426	001421			BEQ	2\$:BRANCH IF NO BUFFER AVAILABLE
87	006430	005060	000072		CLR	\$FAIR(R0)	:CLEAR THE FAIR FLAG
88	006434	004737	016676		JSR	PC,FILBUF	:FILL THE BUFFER
89	006440	004737	017024		JSR	PC,GODRIV	:SET COMMAND AND GO
90	006444	012705	001526		MOV	#ORDERQ,R5	:PUT THE WAIT QUEUE INTO ORDER QUEUE
91	006450	005725			TST	(R5)+	:QUEUE AVAILABLE ?
92	006452	001376			BNE	-2	:BRANCH IF NOT
93	006454	010045			MOV	R0,-(R5)	:LOAD THE DPB ADDRESS INTO THE ORDER QUEUE
94	006456	012701	001640		MOV	#WAIT,R1	:REMOVE THE QUEUE FROM THE WAITING QUEUE
95	006462	060201			ADD	R2,R1	:OFFSET THE QUEUE POSITION
96	006464	004737	020356		JSR	PC,COMPRES	:COMPRESS THE QUEUE
97	006470	000744			BR	1\$:BRANCH IF DONE
98	006472	062702	000002	2\$:	ADD	#2,R2	:CHECK THE NEXT QUEUE
99	006476	000741			BR	1\$:LOOPING BACK
100							
101	006500	005737	001526	MAIN3:	TST	ORDERQ	:OUTSTANDING BUFFER REQUESTS
102	006504	001074			BNE	IDLE	:BR IF THERE ARE
103	006506	005002			CLR	R2	:CLEAR DRIVE TABLE POINTER
104	006510	005762	001616	1\$:	TST	AVAIL(R2)	:ANY DRIVES WAITING FOR PARAMETERS
105	006514	001002			BNE	+6	:BRANCH IF ANY
106	006516	000137	006676		JMP	IDLE	:BRANCH IF NONE
107	006522	016200	001616		MOV	AVAIL(R2),R0	:CONTROL BLOCK ADDR IN R0
108	006526	005760	000104		TST	\$NEXT(R0)	:PARAMETERS BEEN SELECTED ?
109	006532	001010			BNE	6\$:BR IF THEY HAVE
110	006534	105760	000026		TSTB	\$PACK(R0)	: 'R' OR 'W' COMMAND FOR THE DRIVE ?


```

111 006540 001403          BEQ      5$          ;BR IF NOT
112 006542 004737 020374   JSR      PC,WRTPK    ;GET DATA PACK PARAMETERS
113 006546 000404          BR       7$          ;GET THE BUFFER
114 006550 004737 017102   5$: JSR      PC,SELPAR ;SELECT THE PARAMETERS
115 006554 004737 020064   6$: JSR      PC,GETPAR ;LOAD NEW PARAMETERS
116 006560 005046          7$: CLR      -(SP)     ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
117 006562 004737 016312   JSR      PC,GETBUF   ;GET BUFFER
118 006566 012660 000006   MOV      (SP)+,$BUF(R0) ;MOVE BUFFER ADDR TO DPB
119 006572 001414          BEQ      8$          ;BR IF '0' ADDR (NO BUFFER)
120 006574 004737 016676   JSR      PC,FILBUF   ;FILL THE BUFFER
121 006600 005060 000072   CLR      $FAIR(R0)   ;CLEAR THE 'FAIRNESS' COUNT
122 006604 004737 017024   JSR      PC,GODRIV   ;PUT CURRENT DPB IN DRIVER
123 006610 012705 001526   MOV      #ORDERQ,R5 ;ADDRESS OF ORDER QUEUE IN R5
124 006614 005725          TST      (R5)+       ;END OF QUEUE ?
125 006616 001376          BNE      -2          ;BR IF NOT
126 006620 010045          MOV      R0,-(R5)    ;PUT BLOCK ADDRESS INTO QUEUE
127 006622 000417          BR       10$         ;CONTINUE LOOKING
128 006624          8$:
129 006624 005260 000072   INC      $FAIR(R0)   ;INCREMENT THE FAIR COUNT
130 006630 022760 000003 000072  CMP      #3,$FAIR(R0) ;THREE TIMES,BUFFER IS NOT AVAILABLE?
131 006636 101006          BHI      9$          ;BRANCH IF NOT OVER THREE TIMES
132 006640 012705 001640   MOV      #WAIT,R5    ;LOAD INTO THE WAIT QUEUE
133 006644 005725          TST      (R5)+       ;AN AVAILABLE LOCATION ?
134 006646 001376          BNE      -2          ;BRANCH IF NOT
135 006650 010045          MOV      R0,-(R5)    ;LOAD INTO WAIT QUE
136 006652 000403          BR       10$         ;REMOVE THE DPB FROM AVAILABLE QUE
137 006654          9$:
138 006654 062702 000002   ADD      #2,R2       ;INCREMENT INDEX
139 006660 000713          BR       1$          ;BRANCH BACK TO FIRE NEXT DRIVE
140 006662 012701 001616   10$: MOV      #AVAIL,R1  ;'AVAILABLE' TABLE ADDRESS
141 006666 060201          ADD      R2,R1       ;FORM ADDRESS OF LAST ENTRY
142 006670 004737 020356   JSR      PC,COMPRES ;COMPRESS THE TABLE
143 006674 000705          BR       1$          ;CONTINUE LOOKING
144
145          ;GET BUFFER ASSIGNMENTS FOR DRIVES IN THE 'BUFFER WAIT' QUEUE
146
147
148          ;WAIT FOR AN ORDER TO FINISH
149
150 006676 012701 001526   IDLE: MOV      #ORDERQ,R1 ;ADDRESS OF THE ORDER QUEUE IN R1
151 006702 012100          1$: MOV      (R1)+,R0   ;PUT BLOCK ADDRESS INTO R0
152 006704 001433          BEQ      IDLE1       ;BR IF END OF QUEUE
153 006706 005760 000016   16$: TST      $STATUS(R0) ;SEE IF DRIVE FINISHED
154 006712 001775          BEQ      16$         ;BR IF DRIVE NOT FINISHED
155 006714 162701 000002   SUB      #2,R1       ;CORRECT THE QUEUE POINTER
156 006720 010146          MOV      R1,-(SP)    ;SAVE THE QUEUE ADDRESS
157 006722 004737 016154   JSR      PC,STATIS   ;ACCUMULATE STATISTICS FOR DRIVE IN R0
158 006726 000240          NOP                 ;DEBUGGING AID
159 006730 004737 007162   JSR      PC,PROCES   ;PROCESS END OF ORDER
160 006734 005037 001364   CLR      BADSEC      ;CLEAR THE BAD TRK/SEC ERROR INDICATOR
161 006740 004737 030152   JSR      PC,ABNRML   ;SEE IF ANY DRIVES HAVE TOO MANY ERRORS
162 006744 004737 030200   JSR      PC,$EOP     ;IS IT END OF PASS ?
163 006750 012601          MOV      (SP)+,R1    ;RESTORE THE ORDER TABLE INDEX
164 006752 012705 001616   MOV      #AVAIL,R5   ;FIND THE END OF THE 'AVAILABLE' TABLE
165 006756 005725          2$: TST      (R5)+       ;END OF THE TABLE ?
166 006760 001376          BNE      2$          ;BR IF NOT AT END OF LIST
167 006762 011145          MOV      (R1),-(R5) ;MOVE THE BLOCK ADDRESS INTO THE TABLE

```

```

168 006764 004737 020356      JSR    PC,CMPRES      ;COMPRESS THE ORDER QUEUE
169 006770 004737 016446      JSR    PC,RELBUF      ;RESTORE BUFFER
170 006774                                IDLE1:
171 006774 032777 000004 172152 1$:  BIT    #SW02,@SWR      ;TYPE PERFORMANCE SUMMARY
172 007002 001014                                BNE    2$              ;BR IF NOT
173 007004 005737 001316      TST    STATIN         ;TIME TO TYPE THE PERFORMANCE SUMMARY ?
174 007010 001411                                BEQ    2$              ;BR IF NOT
175 007012 005037 001316      CLR    STATIN         ;CLEAR THE INDICATOR
176 007016 005737 001550      TST    ASNLST         ;ANY DRIVES ASSIGNED ?
177 007022 001404                                BEQ    2$              ;BR IF NO
178 007024 104401 073572      TYPE   ,REPHD         ;TYPE PERFORMANCE REPORT HEADING
179 007030 004737 023530      JSR    PC,STATPR      ;TYPE THE SUMMARY
180 007034 000137 006100      2$:  JMP    MAIN        ;CONTINUE THE LOOP
181
182                                ;SETUP TO REFORMAT AN ERROR SECTOR
183
184 007040 032777 000001 172106  REFMT:  BIT    #SW0,@SWR      ;READ ONLY SWITCH SET ?
185 007046 001044                                BNE    REFMTX         ;BR IF IT IS
186 007050 032777 000200 172076      BIT    #SW7,@SWR      ;SWITCH 7 SET ?
187 007056 001040                                BNE    REFMTX         ;BR IF IT IS
188 007060 005737 001500      TST    FORMAT         ;WRITE HEADER & DATA ORDERS ALLOWED :
189 007064 001435                                BEQ    REFMTX         ;BR IF NOT
190 007066 022760 001465 002152      CMP    #821,,$RMDC(RO) ;LEGAL VALUE ?
191 007074 101431                                BLOS   REFMTX         ;NO, NOT FORMAT
192 007076 004737 023176      JSR    PC,READDR      ;GET CORRECTED SECTOR-TRACK ADDRESSES
193 007102 012660 000100                                MOV    (SP)+,$NCYL(RO) ;CYLINDER
194 007106 112660 000077                                MOV    (SP)+,$NTRK(RO) ;TRACK ADDR TO DPB
195 007112 112660 000076                                MOV    (SP)+,$NSEC(RO) ;SECTOR ADDR TO DPB
196 007116 012760 000402 000102      MOV    #258,,$NWRDL(RO) ;WORD COUNT FOR FORMAT
197 007124 112760 000003 000074 1$:  MOV    #3,$NCODE(RO)   ;COMMAND CODE
198 007132 004537 017656      JSR    R5,CHKADR      ;AVOID REFORMAT BAD SPOT
199 007136 000401                                BR     2$              ;BRANCH IF NOT ON BAD SPOT
200 007140 000407                                BR     REFMTX         ;BRANCH IF ON BAD SPOT
201 007142 004737 020022      2$:  JSR    PC,GETPAT    ;GET A PATTERN
202 007146 110560 000075                                MOV    R5,$NPATC(RO)  ;PATTERN CODE TO CONTROL BLOCK
203 007152 012760 177777 000104      MOV    #-1,$NEXT(RO)  ;SET PARAMETERS SELECTED INDICATOR
204 007160 000207      REFMTX: RTS    PC     ;RETURN
205
206                                ;PROCESS THE ORDER TERMINATION
207
208 007162 111037 001346      PROCES: MOV    (RO),UNIT ;DRIVE NUMBER FOR ANY ERROR MESSAGES
209 007166 005760 000016      TST    $STATUS(RO)    ;SEE IF DRIVER SIGNALLED AN ERROR
210 007172 100427                                BMI    ERPROC         ;BR IF ERROR
211 007174 032760 100000 002116      BIT    #BIT15,$RMCS1(RO) ;SEE IF 'SC' SET
212 007202 001410                                BEQ    1$              ;BR IF NOT SET
213 007204 032760 040000 002116      BIT    #BIT14,$RMCS1(RO) ;SEE IF 'TRE' SET
214 007212 001017                                BNE    ERPROC         ;BR IF SET
215 007214 032760 040000 002130      BIT    #BIT14,$RMDS(RO) ;SEE IF 'ERR' SET
216 007222 001013                                BNE    ERPROC         ;BR IF SET
217 007224 004737 013306      1$:  JSR    PC,CKERR      ;NO ERROR, CHECK ERROR BITS ANYWAY
218 007230 004737 013400      JSR    PC,CKBUS       ;NO ERROR, CHECK BUS ADDR & WC
219 007234 032777 000002 171712      BIT    #SW01,@SWR      ;DATA COMPARE ?
220 007242 001002                                BNE    2$              ;BR IF NOT
221 007244 004737 013464      JSR    PC,CMPAR       ;NO ERROR, COMPARE DATA
222 007250 000207      2$:  RTS    PC         ;RETURN
223
224                                ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR

```

```

225
226 007252 032760 000200 000016 ERPROC: BIT #BIT07,$STATUS(R0) ;DONE BIT SET ?
227 007260 001402 BEQ ERPRC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
228 007262 000137 007646 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
229
230 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
231
232 007266 032760 010000 000016 ERPRC1: BIT #BIT12,$STATUS(R0) ;SEE IF DRIVE WAS UNSAFE
233 007274 001025 BNE PUNSAF ;BR IF YES
234 007276 032760 004000 000016 BIT #BIT11,$STATUS(R0) ;PARITY ERROR OCCURRED
235 007304 001043 BNE UCPAR ;BR IF IT DID
236 007306 032760 002000 000016 BIT #BIT10,$STATUS(R0) ;FATAL PARITY ERROR?
237 007314 001046 BNE FALPAR ;BR IF THERE IS ONF
238 007316 032760 001000 000016 BIT #BIT09,$STATUS(R0) ;TIMEOUT?
239 007324 001070 BNE SWTIM ;BR IF YES
240 007326 032760 040002 000016 BIT #BIT14!BIT01,$STATUS(R0) ;DRIVE WENT OFFLINE ?
241 007334 001103 BNE OFLIN ;BR IF IT DID
242 007336 032760 000004 000016 BIT #BIT2,$STATUS(R0) ;PORT REQUEST TIME OUT ?
243 007344 001121 BNE PRTIM ;BR IF IT DID
244 007346 000207 RTS PC ;ERROR. RETURN
245
246 ;DRIVE IS PERSISTENTLY UNSAFE
247
248 007350 104401 001203 PUNSAF: TYPE , $CRLF ;CR-LF
249 007354 004737 021112 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
250 007360 104414 066562 DISPLY ,EM12 ;PERSISTENT DEVICE UNSAFE MESSAGE
251 007364 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
252 007370 004737 021574 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
253 007374 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
254 007400 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
255 007404 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
256 007410 000137 030056 JMP DROP ;DROP THE DRIVE
257
258 ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURRED
259
260 007414 104401 001203 UCPAR: TYPE , $CRLF ;CR-LF
261 007420 104401 001203 TYPE , $CRLF ;CR-LF
262 007424 104401 066464 TYPE ,EM10 ;'UNCORRECTABLE PARITY ERROR' MESSAGE
263 007430 000406 BR FALPR1 ;FINISH PROCESSING THE ERROR
264
265 ;'FATAL' MASSBUS PARITY ERROR OCCURRED
266
267 007432 104401 001203 FALPAR: TYPE , $CRLF ;CR-LF
268 007436 104401 001203 TYPE , $CRLF ;CR-LF
269 007442 104401 066527 TYPE ,EM11 ;'FATAL PARITY ERROR' MESSAGE
270
271 007446 104401 074064 FALPR1: TYPE ,MSGON ;TYPE 'ON'
272 007452 104401 073331 TYPE ,UNTMSG ;TYPE 'DRIVE'
273 007456 013746 001346 MOV UNIT,-(SP) ;;SAVE UNIT FOR TYPEOUT
;;TYPE DRIVE NUMBER
;;GO TYPE--OCTAL ASCII
;;TYPE 2 DIGIT(S)
;;SUPPRESS LEADING ZEROS
007462 104403 TYPOS ;TYPE 'ON'
007464 002 .BYTE 2 ;;TYPE 2 DIGIT(S)
007465 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
274 007466 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
275 007472 032777 100000 171454 BIT #SW15,@SWR ;HALT ON ERROR ?
276 007500 001401 BEQ 1$ ;BR IF NOT
277 007502 000000 HALT ;ERROR HALT

```

```

278 007504 000207      1$:   RTS   PC
279
280                    ;SOFTWARE TIMEOUT OCCURRED
281
282 007506 004737 021112  SWTIM: JSR   PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
283 007512 104414 066613      DISPLY  ,EM13      ;PRINT THE TIME OUT MESSAGE
284 007516 004737 021166      JSR   PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
285 007522 004737 021574      JSR   PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
286 007526 004737 022244      JSR   PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
287 007532 004737 024340      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
288 007536 004737 022654      JSR   PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
289 007542 000207      RTS   PC            ;RETURN
290
291                    ;DRIVE WENT OFFLINE
292
293 007544 104401 001203  OFLIN: TYPE  ,%CRLF      ;CR-LF
294 007550 004737 021112      JSR   PC,LINE1      ;PRINT LINE 1 OF THE ERROR MESSAGE
295 007554 104414 066665      DISPLY  ,EM14      ;PRINT OFFLINE MESSAGE
296 007560 004737 021166      JSR   PC,LINE2      ;PRINT LINE 2 OF THE ERROR MESSAGE
297 007564 004737 021574      JSR   PC,LINE3      ;PRINT LINE 3 OF THE ERROR MESSAGE
298 007570 004737 022244      JSR   PC,LINE4      ;PRINT LINE 4 OF THE ERROR MESSAGE
299 007574 004737 024340      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
300 007600 004737 022654      JSR   PC,LINE7      ;PRINT LINE 7 OF THE ERROR MESSAGE
301 007604 000137 030056      JMP   DROP          ;DROP THE DRIVE
302
303                    ;PORT REQUEST TIMEOUT ERROR
304
305 007610 004737 021112  PRTIM: JSR   PC,LINE1      ;TYPE LINE 1 OF THE ERROR MESSAGE
306 007614 104414 066710      DISPLY  ,EM15      ;PRINT PORT TIME OUT MESSAGE
309 007620 004737 021166      JSR   PC,LINE2      ;TYPE LINE 2 OF THE ERROR MESSAGE
      007624 004737 021574      JSR   PC,LINE3      ;TYPE LINE 3 OF THE ERROR MESSAGE
      007630 004737 022244      JSR   PC,LINE4      ;TYPE LINE 4 OF THE ERROR MESSAGE
310 007634 004737 024340      JSR   PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
311 007640 004737 022654      JSR   PC,LINE7      ;TYPE LINE 7 OF THE ERROR MESSAGE
312 007644 000207      RTS   PC            ;RETURN
313
314                    ;PROLESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
315
316 007646 032760 000030 000016  DONE:  BIT   #BIT04,!BIT03,$STATUS(R0) ;UNSAFE OCCURRED
317 007654 001402      BEQ   .+6          ;BR IF NOT
318 007656 000137 012754      JMP   UNSAF        ;REPORT UNSAFE
319 007662 032760 040000 002126  BIT   #BIT14,$RMCS2(R0) ;IS 'WCE' SET ?
320 007670 001402      BEQ   .+6          ;BRANCH IF NOT SET
321 007672 000137 010630      JMP   WCKER        ;WRITE CHECK ERROR
322 007676 032760 040000 002130  BIT   #BIT14,$RMDS(R0) ;CHECK 'ERR'
323 007704 001002      BNE   1$          ;BR IF SET
324 007706 000137 012514      JMP   TRFER        ;PROCESS 'TRE'
325 007712 032760 000400 002132  1$:  BIT   #BIT08,$RMER1(R0) ;'HCRC' SET?
326 007720 001402      BEQ   .+6          ;BR IF NOT
327 007722 000137 011156      JMP   HRCRCR       ;PROCESS 'HCRC'
328 007726 032760 000020 002132  BIT   #BIT04,$RMER1(R0) ;'FMT' SET?
329 007734 001402      BEQ   .+6          ;BR IF NOT SET
330 007736 000137 011354      JMP   CKFMT        ;CHECK FORMAT ERROR
331 007742 032760 000200 002132  BIT   #BIT07,$RMER1(R0) ;'HCE' SET?
332 007750 001402      BEQ   .+6          ;BR IF NOT SET
333 007752 000137 011550      JMP   CKHCE        ;CHECK 'HCE' ERROR
334 007756 032760 020000 002132  BIT   #BIT13,$RMER1(R0) ;'OPI' SET?

```

```

335 007764 001402          BEQ      .+6          ;BR IF NOT SET
336 007766 000137 012050  JMP      OPIER        ;REPORT 'OPI'
337 007772 032760 000010 002132  BIT      #BIT3,$RMER1(R0) ;'PAR' SET?
338 010000 001402          BEQ      .+6          ;BR IF NOT SET
339 010002 000137 012202  JMP      PARER        ;REPORT 'PAR'
340 010006 032760 000040 002132  BIT      #BIT5,$RMER1(R0) ;'WCF' SET?
341 010014 001402          BEQ      .+6          ;BR IF NOT SET
342 010016 000137 012656  JMP      WCFER        ;REPORT 'WCF'
343 010022 032760 002000 002132  BIT      #BIT10,$RMER1(R0) ;'IAE' SET?
344 010030 001402          BEQ      .+6          ;BR IF NOT SET
345 010032 000137 012274  JMP      IAEER        ;REPORT 'IAE'
346 010036 032760 004000 002132  BIT      #BIT11,$RMER1(R0) ;'WLE' SET?
347 010044 001402          BEQ      .+6          ;BR IF NOT SET
348 010046 000137 012326  JMP      WLEER        ;REPORT 'WLE'
349 010052 032760 001000 002132  BIT      #BIT9,$RMER1(R0) ;'AOE' SET?
350 010060 001405          BEQ      2$          ;BR IF NOT SET
351 010062 032760 002000 002130  BIT      #BIT10,$RMDS(R0) ;'LST' SET?
352 010070 001401          BEQ      2$          ;BR IF NOT SET
353 010072 000207          RTS      PC          ;'AOE' & 'LST' SET, EXIT
354 010074 032760 010000 002132 2$:  BIT      #BIT12,$RMER1(R0) ;SEE IF 'DTE' SET
355 010102 001402          BEQ      .+6          ;BR IF NOT
356 010104 000137 012160  JMP      DTEER        ;REPORT 'DTE' ERROR
357 010110 005760 002132  TST      $RMER1(R0)   ;SEE IF 'DCK' SET
358 010114 100002          BPL      .+6          ;BR IF NOT
359 010116 000137 010154  JMP      DCKER        ;PROCESS 'DCK'
360 010122 032760 060000 002160  BIT      #BIT14!BIT13,$RMER2(R0) ;'SKI' OR 'OCYL' SET
361 010130 001006          BNE      3$          ;BRANCH IF SKI, OCYL SET
362 010132 032760 100000 002160  BIT      #BIT15,$RMER2(R0) ;BAD SPOT ?
363 010140 001004          BNE      4$          ;BRANCH IF SO (NO, OTHER ERROR)
364 010142 000137 011322  JMP      DRIVER      ;REPORT ERROR
365 010146 000137 012614 3$:  JMP      SKIER        ;REPORT SKI ERROR
366 010152 000207 4$:  RTS      PC          ;EXIT FROM ERROR ANALYSIS ROUT.
367
368 ;PROCESS DATA ('DCK') CHECK ERROR
369
370 010154 022760 010041 002162  DCKER:  CMP      #10041,$RMEC1(R0) ;VALID POSITION COUNT ?
371 010162 101407          BLOS     1$          ;BR IF NOT VALID
372 010164 005760 002162  TST      $RMEC1(R0)   ;POSITION COUNT 0 ?
373 010170 001404          BEQ      1$          ;BR IF 0'S
374 010172 005760 002164  TST      $RMEC2(R0)   ;VALUE IN PATTERN REGISTER ?
375 010176 001027          BNE      4$          ;BR IF YES
376 010200 000431          BR       6$          ;DATA CHECK ERROR
377 010202 004737 021112 1$:  JSR      PC,LINE1    ;TYPE FIRST LINE OF ERROR MESSAGE
378 010206 104414 070327  DISPLY  ,EM45        ;TYPE 'ECC LOGIC ERROR'
379 010212 004737 021166  JSR      PC,LINE2    ;TYPE LINE 2 OF ERROR MESSAGE
380 010216 004737 024340  JSR      PC,INCTOT   ;INCREMENT TOTAL ERROR COUNT
381 010222 012737 000003 001352  MOV      #3,RETRY    ;RETRY COUNT
382 010230 004737 016026  JSR      PC,$RETRY   ;RETRY THE ORDER
383 010234 000403          BR       2$          ;RETRY WAS NOT SUCCESSFUL
384 010236 004737 022572  JSR      PC,LINE6C   ;TYPE LINE 6C OF ERROR MESSAGE
385 010242 000402          BR       3$          ;FINISH THE ERROR REPORT
386 010244 004737 022600 2$:  JSR      PC,LINE6D   ;TYPE LINE 6D OF ERROR MESSAGE
387 010250 004737 022654 3$:  JSR      PC,LINE7    ;TYPE LINE 7 OF ERROR MESSAGE
388 010254 000402          BR       5$          ;EXIT
389 010256 004737 020760 4$:  JSR      PC,SPOTCK   ;SEE IF ERROR AT A BAD SPOT ON THE PACK
390 010262 000207 5$:  RTS      PC          ;IT IS, DON'T REPORT IT
391 010264 004737 021112 6$:  JSR      PC,LINE1    ;PRINT LINE 1 OF ERROR MESSAGE

```

```

392 010270 104414 066765          DISPLY ,EM21          ;DATA CHECK ERROR
393 010274 004737 021166          DCKER1: JSR PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
394 010300 004737 021574          JSR PC,LINE3          ;PRINT LINE 3 OF ERROR MESSAGE
395 010304 004737 022244          JSR PC,LINE4          ;PRINT LINE 4 OF ERROR MESSAGE
396 010310 004737 015450          JSR PC,PRTBAD         ;SEE IF BAD SECTOR TO BE PRINTED
397 010314 012737 110100 001350  MOV #BIT15!BIT12!BIT06,MASK ;LOAD ERROR MASK
398 010322 032760 010100 002132  BIT #BIT12!BIT06,$RMER1(RO) ;CHECK 'DTE' & 'ECH'
399 010330 001003          BNE 1$              ;BR IF SET
400 010332 004737 022544          JSR PC,LINE6          ;PRINT LINE 6 OF ERROR MESSAGE
401 010336 000477          BR 8$              ;FINISH THE ERROR REPORT
402 010340 012737 000020 001352 1$: MOV #16.,RETRY        ;RETRY COUNT
403 010346 005001          CLR R1              ;R1 IS OFFSET CODE POINTER
404
405
406 010350 004737 017024          2$: JSR PC,GODRIV      ;RETRY
407 010354 005760 000016          3$: TST $STATUS(RO)    ;TEST FOR DONE
408 010360 001775          BEQ 3$             ;BR IF NOT DONE
409 010362 100075          BPL 10$            ;BR IF NOT ERROR
410 010364 032760 000200 000016  BIT #BIT7,$STATUS(RO) ;SEE IF ORDER TERMINIATED NORMALLY
411 010372 001006          BNE 14$            ;BR IF NOT
412 010374 004737 024340          JSR PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
413 010400 104414 072455          DISPLY ,LIN8M        ;'DIFFERENT ERROR DURING RETRY'
414 010404 000137 007266          JMP ERPRC1           ;SEE WHICH ERROR
415 010410 033760 001350 002132 14$: BIT MASK,$RMER1(RO) ;LOOK AT CURRENT ERROR
416 010416 001430          BEQ 5$             ;BR IF DIFFERENT ERROR
417 010420 032760 010100 002132  BIT #BIT12!BIT6,$RMER1(RO) ;'ECH' OR 'DTE' STILL SET ?
418 010426 001437          BEQ 7$             ;BR IF NEITHER SET
419 010430 105237 001353          INCB RETRY+1        ;INCREMENT RETRY COUNT
420 010434 123737 001352 001353  CMPB RETRY,RETRY+1   ;DONE ?
421 010442 001342          BNE 2$             ;BR IF NOT
422 010444 005201          INC R1              ;INCREMENT TABLE INDEX
423 010446 116137 002470 066071  MOVB OFFCOD(R1),GENDPB+$FMT ;OFFSET CODE
424 010454 001435          BEQ 9$             ;BR IF END OF OFFSET TABLE
425 010456 062737 000002 001352  ADD #2,RETRY         ;NEW RETRY LIMIT
426 010464 004737 015710          JSR PC,OFFST         ;OFFSET
427 010470 005737 066106          4$: TST GENDPB+$STATUS ;SEE IF FINISHED WITH OFFSET
428 010474 001775          BEQ 4$             ;BR IF NOT
429 010476 100324          BPL 2$             ;BR IF NO ERROR PERFORMING OFFSET
430 010500 004737 023112          5$: JSR PC,LINE8      ;PRINT LINE 8 OF ERROR MESSAGE
431 010504 004737 024244          6$: JSR PC,INCHRD     ;INCREMENT 'HARD' ERROR COUNT
432 010510 004737 024340          JSR PC,INCTOT        ;INCREMENT TOTAL ERROR COUNT
433 010514 004737 022654          JSR PC,LINE7         ;PRINT LINE 7 OF ERROR MESSAGE
434 010520 004737 015450          JSR PC,PRTBAD        ;PRINT THE BAD SECTOR
435 010524 000436          BR 13$             ;CLEAN UP AND RETURN
436 010526 004737 022564          7$: JSR PC,LINE6B     ;PRINT LINE 6B OF ERROR MESSAGE
437 010532 004737 022502          JSR PC,LINE5B        ;PRINT LINE 5B OF THE ERROR MESSAGE
438 010536 004737 024220          8$: JSR PC,INCSOF     ;INCREMENT 'SOFT' ERROR COUNT
439 010542 004737 014710          JSR PC,ECC           ;CORRECT THE ERROR USING ECC AND CHECK IT
440 010546 000407          BR 11$             ;COMPARE THE BUFFER
441 010550 004737 022600          9$: JSR PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
442 010554 000753          BR 6$             ;INCREMENT ERROR COUNT
443 010556 004737 022556          10$: JSR PC,LINE6A    ;PRINT LINE 6A OF ERROR MESSAGE
444 010562 004737 024220          JSR PC,INCSOF        ;INCREMENT 'SOFT' ERROR COUNT
445 010566 012737 000001 001400 11$: MOV #1,FRSTER       ;SET PROCESSING 'DCKER' INDICATOR
446 010574 004737 013502          JSR PC,CMPARD        ;COMPARE THE BUFFER
447 010600 105737 001401          TSTB FRSTER+1        ;ERROR IN COMPARE ?
448 010604 100406          BMI 13$            ;BRANCH IF ERROR

```

```

449 010606 004737 024340      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
450 010612 104414 072655      DISPLY   ,LIN9G        ;'DATA COMPARE OK' MESSAGE
451 010616 004737 022654      12$:    JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
452 010622 004737 007040      13$:    JSR      PC,REFMT      ;REFORMAT THE ERROR SECTOR
453 010626 000207              RTS      PC              ;RETURN
454
455      ;WRITE CHECK ERROR PROCESSING
456
457 010630 032760 100000 002132 WCKER:  BIT      #BIT15,$RMER1(R0) ;SEE IF 'DCK' SET ALSO
458 010636 001034              BNE     2$              ;BR IF IT IS
459 010640 004737 021112      JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
460 010644 104414 067071      DISPLY   ,EM23        ;PRINT WCE & DCK NOT
461 010650 005037 001350      CLR     MASK          ;CLEAR ERROR MASK
464 010654 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
      010660 004737 021574      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
      010664 004737 022244      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
      010670 004737 022334      JSR      PC,LINE5      ;PRINT LINE 5 OF ERROR MESSAGE
465 010674 004737 024340      JSR      PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
466 010700 012737 000003 001352  MOV     #3,RETRY      ;RETRY LIMIT
467 010706 004737 016026      JSR      PC,$RETRY     ;RETRY THE OPERATION
468 010712 000403              BR      1$              ;RETRY UNSUCCESSFUL
469 010714 004737 022572      JSR      PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
470 010720 000502              BR      8$              ;FINISH PROCESSING THE ERROR
471 010722 004737 022600      1$:    JSR      PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
472 010726 000506              BR      10$             ;FINISH PROCESSING THE ERROR
473 010730 004737 020760      2$:    JSR      PC,SPOTCK    ;SEE IF ERROR AT BAD SPOT ON THE PACK
474 010734 000507              BR      11$             ;EXIT IF AT BAD SPOT ON PACK
475 010736 004737 021112      JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
476 010742 012737 067016 010770  MOV     #EM22,13$     ;ASSUME THAT EM22 WILL BE PRINTED
477 010750 032760 040000 002126  BIT     #BIT14,$RMCS2(R0) ;DID 'WCK' ALSO SET ?
478 010756 001003              BNE     12$             ;BR IF IT DID
479 010760 012737 067717 010770  MOV     #EM37,13$     ;MESSAGE FOR 'DCK' AND 'WCK' NOT DURING
480
481 010766 104414              12$:    DISPLY
482 010770 000000              13$:    .WORD 0
485 010772 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
      010776 004737 021574      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
      011002 004737 022244      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
      011006 004737 022334      JSR      PC,LINE5      ;PRINT LINE 5 OF ERROR MESSAGE
486 011012 032760 000100 002132  BIT     #BIT06,$RMER1(R0) ;ECH SET ALSO ?
487 011020 001442              BEQ     8$              ;FINISH PROCESSING THE ERROR
488 011022 012737 000020 001352  MOV     #16,RETRY     ;RETRY LIMIT - 16 (10)
489 011030 004737 017024      3$:    JSR      PC,GODRIV    ;RETRY THE ORDER
490 011034 005760 000016      4$:    TST     $STATUS(R0) ;ORDER FINISHED ?
491 011040 001775              BEQ     5$              ;BR IF NOT
492 011042 100405              BMI     6$              ;BR IF ERROR ON ORDER
493 011044 105237 001353      INCB   RETRY+1        ;INCREMENT RETRY COUNT
494 011050 004737 022572      JSR      PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
495 011054 000431              BR      9$              ;FINISH ERROR PROCESSING
496 011056 105237 001353      6$:    INCB   RETRY+1        ;INCREMENT RETRY COUNT
497 011062 123737 001352 001353  CMPB   RETRY,RETRY+1  ;DONE ?
498 011070 001714              BEQ     1$              ;BR IF AT RETRY LIMIT
499 011072 032760 100000 002132  BIT     #BIT15,$RMER1(R0) ;'DCK' SET
500 011100 001407              BEQ     7$              ;BR IF NOT - DIFFERENT ERROR
501 011102 032760 000100 002132  BIT     #BIT06,$RMER1(R0) ;'ECH' ALSO SET ?
502 011110 001347              BNE     4$              ;BR IF IT IS, RETRY ORDER
503 011112 004737 022572      JSR      PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE

```

```

504 011116 000403
505 011120 004737 023112 7$: BR 8$ ;FINISH PROCESSING ERROR
506 011124 000405 JSR PC,LINE8 ;PRINT LINE 8 - 'DIFFERENT ERROR '
507 011126 004737 024340 BR 9$ ;FINISH PROCESSING ERROR
508 011132 004737 022654 8$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
509 011136 000406 JSR PC,LINE7 ;FINISH THE ERROR MESSAGE
510 011140 004737 024340 BR 11$ ;EXIT
511 011144 004737 022654 9$: JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
512 011150 004737 007040 10$: JSR PC,LINE7 ;FINISH THE ERROR MESSAGE
513 011154 000207 JSR PC,REFMT ;REFORMAT THE SECTOR IN ERROR
514 RTS PC ;RETURN
515 ;REPORT 'HCRC' ERROR
516
517 011156 004737 020760 HRCRCR: JSR PC,SPOTCK ;SEE IF ERROR AT PACK BAD SPOT
518 011162 000456 BR 3$ ;EXIT IF IT IS
519 011164 004737 023176 JSR PC,READDR ;READ ERROR SECTOR HEADER
520 011170 004737 015734 JSR PC,READHD ;GET THE HEAD INFORMATION
521 011174 004737 021112 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
522 011200 104414 066744 DISPLY ,EM20 ;REPORT 'HCRC'
523 011204 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
524 011210 004737 021574 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
525 011214 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
526 011220 004737 022434 JSR PC,LINESA ;PRINT THE HEADER INFORMATION
527 011224 032760 040000 002126 BIT #BIT14,$RMCS2(R0) ;'WCE' ERROR ALSO ?
528 011232 001402 BEQ 1$ ;BR IF NOT
529 011234 004737 022334 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
530 011240 004737 024220 1$: JSR PC,INCSOF ;INCREMENT 'SOFT' ERROR COUNT
531 011244 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
532 011250 012737 000400 001350 MOV #BIT8,MASK ;SET ERROR MASK
533 011256 012737 000003 001352 MOV #3,RETRY ;RETRY LIMIT
534 011264 004737 016026 JSR PC,$RETRY ;RETRY ORDER
535 011270 000405 BR 2$ ;RETRY NOT SUCCESSFUL
536 011272 004737 022572 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
537 011276 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
538 011302 000406 BR 3$ ;EXIT
539 011304 004737 022600 2$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
540 011310 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
541 011314 004737 007040 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
542 011320 000207 3$: RTS PC ;RETURN
543
544 ;REPORT DRIVE ERROR
545
546 011322 004737 021112 DRVER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
547 011326 104414 067361 DISPLY ,EM30 ;REPORT DRIVE ERROR
548 011332 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
549 011336 004737 021574 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
550 011342 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
551 011346 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
552 011352 000207 RTS PC ;RETURN
553
554 ;PROCESS FORMAT ('FER') ERROR
555
556 011354 032760 000400 002132 CKFMT: BIT #BIT8,$RMER1(R0) ;'HCRC' SET ON ORIGINAL ERROR ?
557 011362 001402 BEQ 1$ ;BR IF NOT SET
558 011364 000137 011156 JMP HRCRCR ;REPORT HCRC ERROR
559
560 011370 004737 023176 1$: JSR PC,READDR ;GET CORRECTED TRACK & SECTOR ADDRSES

```



```

561 011374 004737 015734      JSR    PC,READHD      ;READ HEADER
562 011400 032737 000400 066124 BIT    #BIT8,GENREG+RMER1 ;'HCRC' SET WHEN HEADER READ?
563 011406 001002          BNE    2$             ;BR IF 'HCRC' SET
564 011410 000137 012354      JMP    FMTER          ;NO, ERROR IS 'FMT' ONLY
565
566 011414 004737 020760      2$:   JSR    PC,SPOTCK    ;SEE IF ERROR AT BAD SPOT ON THE PACK
567 011420 000452          BR     5$             ;EXIT IF IT IS
568 011422 004737 021112      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
569 011426 104414 067150      DISPLY ,EM24         ;HEADER READ ERROR - FMT BIT DROPPED UP
570 011432 004737 021166      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
571 011436 004737 021574      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
572 011442 004737 022244      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
573 011446 032760 040000 002126 BIT    #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
574 011454 001402          BEQ    3$             ;BR IF NOT
575 011456 004737 022334      JSR    PC,LINES      ;DISPLAY WORDS WHICH CAUSED 'WCE'
576 011462 004737 022434      3$:   JSR    PC,LINESA    ;DISPLAY HEADER
577 011466 004737 024220      JSR    PC,INCSOF     ;INCREMENT SOFT ERROR COUNT
578 011472 004737 024340      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
579 011476 012737 000020 001350 MOV    #BIT4,MASK     ;SET ERROR MASK
580 011504 012737 000003 001352 MOV    #3,RETRY       ;RETRY LIMIT
581 011512 004737 016026      JSR    PC,$RETRY     ;RETRY THE ORDER
582 011516 000405          BR     4$             ;RETRY NOT SUCCESSFUL
583 011520 004737 022572      JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
584 011524 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
585 011530 000406          BR     5$             ;EXIT
586 011532 004737 022600      4$:   JSR    PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
587 011536 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
588 011542 004737 007040      JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
589 011546 000207      5$:   RTS    PC          ;RETURN
590
591      ;PROCESS HEADER COMPARE ('HCE') ERROR
592
593 011550 032760 000400 002132 CKHCE: BIT    #BIT8,$RMER1(RO) ;HCRC SET ON ORIGINAL ERROR ?
594 011556 001402          BEQ    1$             ;BR IF NOT SET
595 011560 000137 011156          JM     HRCRCR        ;REPORT HEADER CRC ERROR
596 011564 004737 023176      1$:   JSR    PC,READDR    ;GET CURRENT SECTOR & TRACK ADDRS
597 011570 004737 015734      JSR    PC,READHD     ;READ HEADER OF CURRENT SECTOR
598 011574 032737 000400 066124 BIT    #BIT8,GENREG+RMER1 ;'HCRC' SET ?
599 011602 001016          BNE    3$             ;BR IF SET
600 011604 042737 150000 076746 BIC    #150000,CYLNDR ;CLEAR FORMAT,MFG,USER BITS FROM HEADER
601 011612 026037 002152 076746 CMP    $RMDC(RO),CYLNDR ;CORRECT CYLINDER ?
602 011620 001402          BEQ    2$             ;BR IF IT IS
603 011622 000137 011774          JMP    POSER         ;REPORT POSITIONING ERROR
604 011626 052737 150000 076746 2$:   BIS    #150000,CYLNDR ;RESTORE THE FORMAT,MFG,USER BITS
605 011634 000137 012432          JMP    HCEER        ;REPORT 'HCE' ERROR
606
607 011640 004737 020760      3$:   JSR    PC,SPOTCK    ;SEE IF ERROR AT BAD SPOT
608 011644 000452          BR     6$             ;EXIT IF IT IS
609 011646 004737 021112      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
610 011652 104414 067216      DISPLY ,EM25         ;HEADER READ ERROR - 'HCE' SET
611 011656 004737 021166      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
612 011662 004737 021574      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
613 011666 004737 022244      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
614 011672 032760 040000 002126 BIT    #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
615 011700 001402          BEQ    4$             ;BR IF NOT
616 011702 004737 022334      JSR    PC,LINES      ;DISPLAY WORDS WHICH CAUSED 'WCE'
617 011706 004737 022434      4$:   JSR    PC,LINESA    ;PRINT LINE 5 OF ERROR MESSAGE

```

```

618 011712 004737 024220      JSR    PC,INCSOF      ;INCREMENT SOFT ERROR COUNT
619 011716 004737 024340      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
620 011722 012737 000200      MOV    #BIT7,MASK    ;SET ERROR MASK
621 011730 012737 000003      MOV    #3,RETRY      ;RETRY LIMIT
622 011736 004737 016026      JSR    PC,$RETRY     ;RETRY THE ORDER
623 011742 000405              BR     5$            ;RETRY NOT SUCCESSFUL
624 011744 004737 022572      JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
625 011750 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
626 011754 000406              BR     6$            ;EXIT
627 011756 004737 022600      5$:   JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
628 011762 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
629 011766 004737 007040      JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
630 011772 000207      6$:   RTS     PC      ;RETURN
631
632
633      ;REPORT POSSIBLE POSITIONING ERROR
634 011774 004737 021112      POSER: JSR    PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
635 012000 104414 070525      DISPLY ,EM51         ;PROGRAM DETECTED POSITIONING ERROR
636 012004 004737 021166      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
637 012010 004737 021622      JSR    PC,LINE3C     ;PRINT LINE 3C OF ERROR MESSAGE
638 012014 052737 150000      BIS    #150000,CYLNR ;RESTORE THE FORMAT BIT
639 012022 004737 022434      JSR    PC,LINE5A     ;PRINT LINE 5A OF THE ERROR MESSAGE
640 012026 004737 024314      JSR    PC,INCMIS     ;INCREMENT MISPOSITIONING COUNT
641 012032 004737 024340      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
642 012036 004737 022776      JSR    PC,LINE7A     ;PRINT LINE 7A OF ERROR MESSAGE
643 012042 004737 015646      JSR    PC,RECALT     ;RECALIBRATE
644 012046 000207      RTS     PC           ;EXIT
645
646      ;REPORT 'OPI' ERROR
647
648 012050 004737 020760      OPIER: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
649 012054 000207              RTS     PC           ;RETURN IF IT IS
650 012056 004737 021112      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
651 012062 104414 067413      DISPLY ,EM31         ;'OPI' ERROR
652 012066 004737 021166      JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
653 012072 004737 021574      JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
654 012076 004737 022244      JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
655 012102 004737 024340      JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
656 012106 012737 020000      MOV    #BIT13,MASK   ;ERROR MASK
657 012114 012737 000003      OPIER1: MOV    #3,RETRY ;RETRY LIMIT
658 012122 004737 016026      JSR    PC,$RETRY     ;RETRY THE ORDER
659 012126 000405              BR     1$            ;RETRY UNSUCCESSFUL
660 012130 004737 022572      JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
661 012134 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
662 012140 000207              RTS     PC           ;EXIT
663 012142 004737 022600      1$:   JSR    PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
664 012146 004737 022654      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
665 012152 004737 007040      JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
666 012156 000207      RTS     PC           ;RETURN
667
668      ;REPORT 'DTE' ERROR
669
670 012160 004737 020760      DTEER: JSR    PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
671 012164 000207              RTS     PC           ;RETURN IF IT IS
672 012166 004737 021112      JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
673 012172 104414 067456      DISPLY ,EM32         ;'DTE' ERROR
674 012176 000137 010274      JMP    DCKER1        ;FINISH PROCESSING THE 'DTE' ERROR

```

```
675
676 ;REPORT 'PAR' ERROR
677
678 012202 004737 021112 PARER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
679 012206 104414 067511 DISPLY ,EM33 ;REPORT 'PAR'
680 012212 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
681 012216 004737 021700 JSR PC,LINE3E ;PRINT LINE 3E OF ERROR MESSAGE
682 012222 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
683 012226 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
684 012232 012737 000010 001350 MOV #BIT03,MASK ;ERROR MASK
685 012240 012737 000003 001352 MOV #3,RETRY ;RETRY LIMIT
686 012246 004737 016026 JSR PC,$RETRY ;RETRY ORDER
687 012252 000405 BR 2$ ;RETRY UNSUCCESSFUL
688 012254 004737 022572 JSR PC,LINE6C ;RETRY SUCCESSFUL
689 012260 004737 022654 1$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
690 012264 000207 RTS PC ;EXIT
691 012266 004737 022600 2$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
692 012272 000772 BR 1$ ;FINISH ERROR MESSAGE
693
694 ;REPORT 'IAE' ERROR
695
696 012274 004737 021112 IAEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
697 012300 104414 067630 DISPLY ,EM35 ;REPORT 'IAE'
698 012304 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
699 012310 004737 021766 JSR PC,LINE3F ;PRINT LINE 3F OF ERROR MESSAGE
700 012314 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
701 012320 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
702 012324 000207 RTS PC ;RETURN
703
704 ;REPORT 'WLE' ERROR
705
706 012326 004737 021112 WLFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
707 012332 104414 067666 DISPLY ,EM36 ;REPORT 'WLE'
708 012336 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
709 012342 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
710 012346 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
711 012352 000207 RTS PC ;RETURN
712
713 ;REPORT FORMAT ERROR
714
715 012354 004737 021112 FMTER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
716 012360 104414 067277 DISPLY ,EM26 ;FORMAT ERROR
717 012364 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
718 012370 004737 021574 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
719 012374 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF ERPOR MESSAGE
720 012400 032760 040000 002126 BIT #BIT14,$RMCS2(RO) ;'WCE' ERROR ALSO ?
721 012406 001402 BEQ 1$ ;BR IF NOT
722 012410 004737 022334 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
723 012414 004737 022434 1$: JSR PC,LINESA ;PRINT LINE 5A OF ERROR MESSAGE
724 012420 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
725 012424 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
726 012430 000207 RTS PC
727
728 ;REPORT HEADER COMPARE ERROR
729
730 012432 004737 021112 HCEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
731 012436 104414 067324 DISPLY ,EM27 ;HEADER COMPARE ERROR
```

```

732 012442 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
733 012446 004737 021574      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
734 012452 004737 022244      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
735 012456 032760 040000 002126 BIT      #BIT14,$RMCS2(R0) ;'WCE' ERROR ALSO ?
736 012464 001402              BEQ      1$            ;BR IF NOT
737 012466 004737 022334      JSR      PC,LINE5      ;DISPLAY WORDS WHICH CAUSED 'WCE'
738 012472 004737 022434      1$: JSR      PC,LINE5A    ;PRINT LINE 5A OF ERROR MESSAGE
739 012476 004737 024340      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
740 012502 004737 022654      JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
741 012506 004737 007040      JSR      PC,REFMT      ;REFORMAT THE ERROR SECTOR
742 012512 000207              RTS      PC            ;RETURN
743
744
745      ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
746 012514 004737 021112  TRFER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
747 012520 104414 070001      DISPLY  ,EM40          ;RH11/RH70 OR UNIBUS TRANSFER ERROR
748 012524 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
749 012530 004737 021574      JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
750 012534 004737 022244      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
751 012540 004737 024340      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
752 012544 032760 121400 002126 BIT      #BIT15!BIT13!BIT9!BIT8,$RMCS2(R0) ;'DLT','UPE','MXF','MDPE' SET ?
753 012552 001415              BEQ      2$            ;BR IF NONE SET
754 012554 012737 000003 001352 MOV      #3,RETRY      ;RETRY LIMIT
755 012562 005037 001350      CLR      MASK          ;CLEAR ERROR MASK
756 012566 004737 016026      JSR      PC,$RETRY     ;RETRY THE OPERATION
757 012572 000403              BR       1$            ;RETURN HERE IF RETRY UNSUCCESSFUL
758 012574 004737 022572      JSR      PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
759 012600 000402              BR       2$            ;FINISH THE ERROR REPORT
760 012602 004737 022600      1$: JSR      PC,LINE6D    ;PRINT LINE 6D OF ERROR MESSAGE
761 012606 004737 022654      2$: JSR      PC,LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
762 012612 000207              RTS      PC
763
764      ;PROCESS 'SKI' OR 'OCYL' ERRORS
765
766 012614 004737 021112  SKIER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
767 012620 104414 070467      DISPLY  ,EM50          ;'SKI' OR 'OCYL' ERROR
768 012624 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
769 012630 004737 021610      JSR      PC,LINE3B     ;PRINT LINE 3B OF ERROR MESSAGE
770 012634 004737 024340      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
771 012640 004737 024270      JSR      PC,INCSKI     ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
772 012644 004737 022776      JSR      PC,LINE7A     ;PRINT LINE 7A OF ERROR MESSAGE
773 012650 004737 015646      JSR      PC,RECALT     ;RECALIBRATE
774 012654 000207              RTS      PC
775
776      ;REPORT WRITE CLOCK FAILURE ('WCF')
777
778 012656 004737 021112  WCFER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
779 012662 104414 067566      DISPLY  ,EM34          ;REPORT WRITE CLOCK FAILURE
780 012666 004737 021166      JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
781 012672 004737 021602      JSR      PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
782 012676 004737 022244      JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
783 012702 004737 024340      JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
784 012706 004737 015450      'SR      PC,PRTBAD     ;SEE IF BAD SECTOR TO BE PRINTED
785 012712 012737 000003 001352 MOV      #3,RETRY      ;RETRY COUNT
786 012720 012737 000040 001330 MOV      #BIT05,MASK    ;ERROR MASK
787 012726 004737 016026      JSR      PC,$RETRY     ;RETRY THE ORDER
788 012732 000405              BR       2$            ;RETURN HERE IF RETRY UNSUCCESSFUL
  
```

789	012734	004737	022572		JSR	PC,LINE6C	:PRINT LINE 6C OF ERROR MESSAGE
790	012740	004737	022654	1\$:	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
791	012744	000207			RTS	PC	
792	012746	004737	022600	2\$:	JSR	PC,LINE6D	:PRINT LINE 6D OF ERROR MESSAGE
793	012752	000772			BR	1\$	
794							
795							:PROCESS DRIVE UNSAFE ERROR
796							
797	012754	004737	021112	UNSAF:	JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
798	012760	104414	070570		DISPLY	,EM60	:REPORT DRIVE UNSAFE
799	012764	004737	021166		JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
800	012770	004737	021574		JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
801	012774	004737	024340		JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
802	013000	012737	000003	001352	MOV	#3,RETRY	:RETRY COUNT
803	013006	004737	016026		JSR	PC,\$RETRY	:RETRY THE ORDER
804	013012	000403			BR	1\$:RETRY WAS UNSUCCESSFUL
805	013014	004737	022572		JSR	PC,LINE6C	:PRINT LINE 6C OF ERROR MESSAGE
806	013020	000402			BR	2\$:CONTINUE WITH ERROR REPORT
807	013022	004737	022600	1\$:	JSR	PC,LINE6D	:PRINT LINE 6D OF ERROR MESSAGE
808	013026	004737	022654	2\$:	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
809	013032	000207			RTS	PC	:RETURN
810							
811							:REPORT AN 'UNKNOWN' DATA PATTERN
812							
813	013034	105737	001400	NOMTCH:	TSTB	FRSTER	:FIRST ERROR IN THE SECTOR ?
814	013040	001013			BNE	1\$:BR IF NOT OR IF PROCESSING 'DCKER'
815	013042	004737	021112		JSR	PC,LINE1	:TYPE LINE 1 OF ERROR MESSAGE
816	013046	104414	070171		DISPLY	,EM43	: 'CAN'T MATCH DATA WITH PATTERN'
817	013052	004737	021166		JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
818	013056	004737	021602		JSR	PC,LINE3A	:PRINT LINE 3A OF ERROR MESSAGE
819	013062	004737	022244		JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
820	013066	000404			BR	2\$:CONTINUE PROCESSING ERROR
821	013070	104414	070171	1\$:	DISPLY	,EM43	: 'CAN'T MATCH DATA WITH PATTERN'
822	013074	104414	001203		DISPLY	,\$CRLF	:CR-LF
823	013100	104414	072605	2\$:	DISPLY	,LIN9I	:HEADER FOR DATA PRINTOUT
831	013104	010146			MOV	R1,-(SP)	:ADDRESS OF WORD 1
	013106	004737	023124		JSR	PC,LINOC	:TYPE WORD 1
	013112	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS
	013116	012146			MOV	(R1)+,-(SP)	:ADDRESS OF WORD 1
	013120	004737	023124		JSR	PC,LINOC	:TYPE WORD 1
	013124	104414	001203		DISPLY	,\$CRLF	:CR-LF
	013130	010146			MOV	R1,-(SP)	:ADDRESS OF WORD 2
	013132	004737	023124		JSR	PC,LINOC	:TYPE WORD 2
	013136	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS
	013142	012146			MOV	(R1)+,-(SP)	:ADDRESS OF WORD 2
	013144	004737	023124		JSR	PC,LINOC	:TYPE WORD 2
	013150	104414	001203		DISPLY	,\$CRLF	:CR-LF
	013154	010146			MOV	R1,-(SP)	:ADDRESS OF WORD 3
	013156	004737	023124		JSR	PC,LINOC	:TYPE WORD 3
	013162	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS
	013166	012146			MOV	(R1)+,-(SP)	:ADDRESS OF WORD 3
	013170	004737	023124		JSR	PC,LINOC	:TYPE WORD 3
	013174	104414	001203		DISPLY	,\$CRLF	:CR-LF
	013200	010146			MOV	R1,-(SP)	:ADDRESS OF WORD 4
	013202	004737	023124		JSR	PC,LINOC	:TYPE WORD 4
	013206	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS
	013212	012146			MOV	(R1)+,-(SP)	:ADDRESS OF WORD 4

```
013214 004737 023124 JSR PC,LIN0CT ;TYPE WORD 4
013220 104414 001203 DISPLY ,SCRLF ;CR-LF
832 013224 062701 000770 ADD #<252.*2.>,R1 ;INCREMENT BUFFER POINTER
833 013230 162737 000400 001412 SUB #256,CMCNT ;ADJUST WORD COUNT FOR MATCH
834 013236 132760 000001 000024 BITB #BIT00,$CODE(R0) ;HEADER OPERATION INVOLVED ?
835 013244 001405 BEQ 3$ ;NO
836 013246 062701 000004 ADD #4,R1 ;ADJUST BUFFER ADDRESS
837 013252 162737 000002 001412 SUB #2,CMCNT ;ADJUST WORD COUNT
838 013260 005002 3$: CLR R2 ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
839 013262 112737 000001 001400 MOVB #1,FRSTER ;SET 'NOT FIRST ERROR' INDICATOR
840 013270 112737 177777 001401 MOVB #-1,FRSTER+1 ;SET ERROR FOUND INDICATOR
841 013276 013737 001476 001410 MOV CMLMT,LIMIT ;RESET THE COMPARE ERROR TYPEOUT LIMIT
842 013304 000207 RTS PC ;RETURN
843
844 ;CHECK ERROR BITS IN THE RH/RM REGISTERS
845
846 013306 032760 060000 002116 CKERR: BIT #60000,$RMCS1(R0) ;SEE IF 'TRE' OR 'MCPE' SET
847 013314 001012 BNE 1$ ;BR IF EITHER SET
848 013316 032760 177400 002126 BIT #177400,$RMCS2(R0) ;SEE IF ERROR BITS IN CS2 SET
849 013324 001006 BNE 1$ ;BR IF ANY SET
850 013326 005760 002132 TST $RMER1(R0) ;ANY BITS SET IN ER1
851 013332 001003 BNE 1$ ;BR IF ANY SET
852 013334 005760 002160 TST $RMER2(R0) ;ANY BITS SET IN ER2 ?
853 013340 001416 BEQ 2$ ;BR IF NONE SET
854 013342 004737 021112 1$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
855 013346 104414 070236 DISPLY ,EM44 ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SE
856 013352 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
857 013356 004737 021574 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
858 013362 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
859 013366 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
860 013372 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
861 013376 000207 2$: RTS PC ;RETURN
862
863 ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
864
865 013400 005760 002120 CKBUS: TST $RMWC(R0) ;CHECK WORD COUNT
866 013404 001010 BNE 1$ ;BR IF NOT ZERO
867 013406 016046 000020 MOV $WRDL(R0),-(SP) ;WORD LENGTH
868 013412 006316 ASL (SP) ;CHANGE INTO BYTE COUNT
869 013414 066016 000006 ADD $BUF(R0),(SP) ;ADD THE STARTING LOCATION
870 013420 022660 002122 CMP (SP)+,$RMBA(R0) ;BUFFER ADDRESS PROPER ?
871 013424 001416 BEQ 2$ ;BR IF OK
872 013426 004737 021112 1$: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
873 013432 104414 070044 DISPLY ,EM41 ;BUS ADDRESS OR WORD COUNT INCORRECT
874 013436 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
875 013442 004737 021632 JSR PC,LINE3D ;PRINT LINE 3D OF ERROR MESSAGE
876 013446 004737 022244 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
877 013452 004737 024340 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
878 013456 004737 022654 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
879 013462 000207 2$: RTS PC ;RETURN
880
881 ;COMPARE THE BUFFER
882
883 013464 132760 000004 000024 CMPAR: BITB #BIT02,$CODE(R0) ;SEE IF READ ORDER
884 013472 001001 BNE 1$ ;BR IF IT IS
885 013474 000207 RTS PC ;RETURN
886 013476 005037 001400 1$: CLR FRSTER ;CLEAR 'FIRST ERROR' INDICATOR
```

```

887 013502 032777 000002 165444 CMPARD: BIT #SW01,@SWR ;IS SWITCH 1 SET?
888 013510 001401 BEQ 1$ ;BR IF NOT
889 013512 000207 RTS PC ;YES, DON'T COMPARE
890 013514 005037 001406 1$: CLR ERCTR ;CLEAR THE ERROR COUNTER
891 013520 016001 000006 MOV $BUF(R0),R1 ;BUFFER ADDRESS
892 013524 016037 000020 001412 MOV $WRDL(R0),CMCNT ;WORD COUNT TO WORKING LOCATION
893 013532 066037 002120 001412 ADD $RMWC(R0),CMCNT ;CALCULATE ACTUAL WORDS TRANSFERED
894 013540 016037 000012 001414 MOV $CYL(R0),CMCYL ;CYLINDER ADDRESS WORKING LOCATION
895 013546 052737 010000 001414 BIS #BIT12,CMCYL ;SET FORMAT BIT
896 013554 052737 140000 001414 BIS #140000,CMCYL ;SET MFG AND USER BITS
897 013562 016037 000010 001416 MOV $SEC(R0),CMSEC ;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
898 013570 013737 001476 001410 MOV CMPLMT,LIMIT ;DISPLAY LIMIT
899 013576 005237 001410 INC LIMIT ;CONVERT PARAMETER INTO LIMIT VALUE
900 013602 012737 177777 001376 CMSTR: MOV #-1,ZROIND ;CLEAR THE 'ZERO'S' INDICATOR
901 013610 005037 001402 CLR SAVER1 ;CLEAR THE R1 SAVE WORD
902 013614 005037 001404 CLR SAVER5 ;CLEAR THE R5 SAVE WORD
903 013620 023760 001412 000022 CMP CMCNT,$SSEC(R0) ;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
904 013626 101005 BHI 1$ ;BR IF IT IS
905 013630 013702 001412 MOV CMCNT,R2 ;LESS THAN, USE REMAINING BUFFER
906 013634 005037 001412 CLR CMCNT ;SET COUNTER TO 0
907 013640 000405 BR 2$ ;
908 013642 016002 000022 1$: MOV $SSEC(R0),R2 ;COMPARE SECTOR
909 013646 166037 000022 001412 SUB $SSEC(R0),CMCNT ;DECREMENT WORD COUNT
910 013654 126027 000024 000005 2$: CMPB $CODE(R0),#5 ;READ HEADER & DATA?
911 013662 001025 BNE CMDAT ;BR IF NOT
912 013664 052711 140000 CMHED: BIS #BIT15!BIT14,(R1) ;SET BIT14,BIT15 IN CASE BAD SPOT ENCOUNTER
913 013670 023721 001414 CMP CMCYL,(R1)+ ;CHECK CYLINDER
914 013674 001402 BEQ 1$ ;BR IF COMPARE OK
915 013676 004737 013724 JSR PC,CMSTR2 ;REPORT ERROR
916 013702 023721 001416 1$: CMP CMSEC,(R1)+ ;COMPARE SECTOR & TRACK
917 013706 001402 BEQ 4$ ;BR IF EQ
918 013710 004737 013724 JSR PC,CMSTR2 ;REPORT ERROR
919 013714 162702 000002 4$: SUB #2,R2 ;SUBTRACT HEADER LENGTH FROM SIZE
920 013720 003570 BLE CMPRX ;BR IF FINISHED
921 013722 000405 BR CMDAT ;COMPARE THE DATA PORTION
922 013724 005237 001406 CMSTR2: INC ERCTR ;INCREMENT THE ERROR COUNT
923 013730 004737 014310 JSR PC,CMPRT ;REPORT THE COMPARISON ERROR
924 013734 000207 RTS PC ;CHECK THE REST OF THE HEADER
925
926 013736 004737 014632 CMDAT: JSR PC,MATCH ;FIND THE PATTERN
927 013742 000403 BR 1$ ;FOUND A PATTERN
928 013744 004737 013034 JSR PC,NOMTCH ;RETURN HERE IF NO MATCH WITH PATTERN MADE
929 013750 000456 BR 7$ ;BYPASS COMPARE ROUTINE
930 013752 011405 1$: MOV (R4),R5 ;ADDRESS OF PATTERN ADDRESS IN R4
931 013754 012703 000020 MOV #20,R3 ;R3 IS PATTERN POS COUNTER
932 013760 022125 2$: CMP (R1)+,(R5)+ ;COMPARE BUFFER WITH PATTERN
933 013762 001016 BNE 4$ ;BR IF NOT EQUAL
934 013764 005737 001406 TST ERCTR ;ERRORS DETECTED ?
935 013770 001406 BEQ 3$ ;BR IF NO ERRORS
936 013772 032777 000010 165154 BIT #SW3,@SWR ;SWITCH 3 SET ?
937 014000 001402 BEQ 3$ ;BR IF NOT SET
938 014002 004737 014310 JSR PC,CMPRT ;DISPLAY THE WORD
939 014006 005302 3$: DEC R2 ;DECREMENT SIZE COUNT
940 014010 003436 BLE 7$ ;BR WHEN AT END
941 014012 005303 DEC R3 ;DECREMENT PATT POS COUNT
942 014014 001361 BNE 2$ ;BR IF NOT AT END OF PATT
943 014016 000755 BR 1$ ;RESTART THE PATTERN

```

```

944 014020 005761 177776      4$:  TST      -2(R1)      ;IS MISCOMPARED CHARACTER=0
945 014024 001410              BEQ      5$          ;BR IF YES
946 014026 012737 177777 001376  MOV      #-1,ZROIND  ;SET NON-ZERO MISCOMPARED INDCATOR
947 014034 005237 001406              INC      ERCTR      ;INCREMENT THE ERROR COUNTER
948 014040 004737 014310              JSR      PC,CMPRT   ;REPORT ERROR
949 014044 000760              BR       3$          ;CONTINUE COMPARE
950 014046 105737 001400      5$:  TSTB     FRSTER     ;FIRST ERROR?
951 014052 100407              BMI     6$          ;BR IF NOT
952 014054 005037 001376              CLR     ZROIND     ;SET THE ZERO INDICATOR
953 014060 010137 001402              MOV     R1,SAVER1  ;SAVE CURRENT R1
954 014064 010537 001404              MOV     R5,SAVER5  ;SAVE CURRENT R5
955 014070 000746              BR     3$          ;CONTINUE COMPARE
956 014072 005737 001376      6$:  TST     ZROIND     ;ANY MISCOMPARISONS NOT ZEROS ?
957 014076 001743              BEQ     3$          ;BR IF NONE-ALL ERRORS=ZERO
958 014100 004737 014310              JSR     PC,CMPRT   ;REPORT ERROR
959 014104 000740              BR     3$          ;CONTINUE COMPARING
960 014106 023727 001412 000004  7$:  CMP     CMCNT,#4    ;LAST 3WORDS ?
961 014114 002472              BLT     CMPRX      ;YES
962 014116 126027 000024 000005  CMPB    $CODE(R0),#5 ;READ HEAD AND DATA ?
963 014124 001414              BEQ     9$          ;YES
964 014126 013702 001412      8$:  MOV     CMCNT,R2    ;SET COUNTER = REMAIN BUFFER LENGTH
965 014132 020227 000004              CMP     R2,#4      ;LAST 3 WORDS ?
966 014136 002461              BLT     CMPRX      ;YES,EXIT
967 014140 162737 000400 001412  SUB     #256.,CMCNT ;GREATER THAN A SECTOR ?
968 014146 003673              BLE     CMDAT      ;NO,RETURN TO COMPARE LOOP
969 014150 012702 000400              MOV     #256.,R2   ;SET COUNTER =SECTOR SIZE
970 014154 000670              BR     CMDAT      ;RETURN TO COMPARE LOOP
971 014156 105237 001416      9$:  INCB   CMSEC      ;INCREMENT COUNTER
972 014162 123727 001416 000040  CMPB    CMSEC,#32.  ;MAX SECTOR # ?
973 014170 103423              BLO     11$        ;NO
974 014172 105037 001416              CLRB   CMSEC      ;RESET SECTOR #
975 014176 105237 001417              INCB   CMTRK      ;INCREMENT TRACK #
976 014202 004737 026546              JSR     PC,GETLMT  ;GET ADDRESS LIMITS
977 014206 123737 001417 001450  CMPB    CMTRK,TRKLMT ;MAX TRACK # ?
978 014214 101411              BLOS   11$        ;NO
979 014216 105037 001417              CLRB   CMTRK      ;RESET TRACK #
980 014222 005237 001414              INC    CMCYL      ;INCREMENT CYLINDER NUMBER
981 014226 023727 001414 151466  CMP     CMCYL,#150000+822. ;LAST CYLINDER ?
982 014234 103401              BLO     11$        ;NO
983 014236 000421              BR     CMPRX      ;NORMAL RETURN,NOT WRAP AROUND
984 014240 052711 140000      11$:  BIS     #BIT15!BIT14,(R1) ;SET BIT15/14 INCASE BAD SPOT ENCOUNTER
985 014244 023721 001414              CMP     CMCYL,(R1)+ ;COMPARE 1ST HEADER WORD
986 014250 001402              BEQ     12$        ;MATCH
987 014252 004737 013724              JSR     PC,CMSTR2  ;NOT MATCH
988 014256 023721 001416      12$:  CMP     CMSEC,(R1)+ ;SECOND WORD OF HEADER
989 014262 001402              BEQ     13$        ;MATCH
990 014264 004737 013724              JSR     PC,CMSTR2  ;NOT MATCH
991 014270 162737 000002 001412  13$:  SUB     #2,CMCNT   ;ADJUST WORD COUNT
992 014276 003401              BLE     CMPRX      ;COMPARE IS DONE
993 014300 000712              BR     8$          ;RETURN TO COMPARE LOOP
994
995 014302 004737 014304      CMPRX: JSR     PC,ENDCMP ;PRINT LAST LINE IF ERRORS
996 014306 000207              RTS     PC
997
998 ;TYPE DATA COMPARE ERRORS
999
1000 014310 005737 001402      CMPRT: TST     SAVER1 ;PRINT SAVED VALUES ?

```



```

1001 014314 001010          BNE      2$          ;BR IF NOT
1002 014316 105737 001400  TSTB   FRSTER      ;FIRST ERROR?
1003 014322 100402          BMI     1$          ;BR IF NOT
1004 014324 004737 014404  JSR    PC,4$       ;PRINT INITIAL MESSAGE INFO
1005 014330 004737 014466  1$:   JSR    PC,8$       ;PRINT REMAINDER OF MESSAGE
1006 014334 000422          BR     3$          ;EXIT
1007 014336          2$:   MOV    R1,-(SP)     ;;PUSH R1 ON STACK
      014336 010146          MOV    R5,-(SP)     ;;PUSH R5 ON STACK
      014340 010546          MOV    SAVER1,R1    ;DISPLAY SAVED R1
1008 014342 013701 001402  MOV    SAVER5,R5    ;DISPLAY SAVED R5
1009 014346 013705 001404  JSR    PC,4$       ;PRINT INITIAL MESSAGE INFO
1010 014352 004737 014404  JSR    PC,8$       ;PRINT SAVED VALUES
1011 014356 004737 014466  CLR    SAVER1      ;CLEAR SAVED REGISTER INDICATORS
1012 014362 005037 001402  CLR    SAVER5      ;CLEAR THE OTHER ONE
1013 014366 005037 001404  MOV    (SP)+,R5    ;;POP STACK INTO R5
1014 014372 012605          MOV    (SP)+,R1    ;;POP STACK INTO R1
      014374 012601          JSR    PC,8$       ;PRINT REMAINDER OF MESSAGE
1015 014376 004737 014466  3$:   RTS    PC        ;RETURN
1016 014402 000207          4$:   TSTB   FRSTER      ;FIRST ERROR ?
1017 014404 105737 001400  BMI     7$          ;BR IF NOT
1018 014410 100425          BNE     5$          ;BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
1019 014412 001013          JSR    PC,LINE1    ;PRINT LINE 1 OF ERROR MESSAGE
1020 014414 004737 021112  DISPLY ,EM42       ;DATA COMPARE ERROR
1021 014420 104414 070110  JSR    PC,LINE2    ;PRINT LINE 2 OF ERROR MESSAGE
1022 014424 004737 021166  JSR    PC,LINE3A   ;PRINT LINE 3A OF ERROR MESSAGE
1023 014430 004737 021602  JSR    PC,LINE4    ;PRINT LINE 4 OF ERROR MESSAGE
1024 014434 004737 022244  BR     6$          ;GO TO TYPE HEADER
1025 014440 000404          5$:   DISPLY ,LIN9B     ;HEADER MESSAGE OF PROCESSING 'DCK' ERROR
1026 014442 104414 072500  DISPLY ,$CRLF      ;CR-LF
1027 014446 104414 001203  6$:   DISPLY ,LIN9H     ;DISPLAY HEADER
1028 014452 104414 072527  MOV    #-1,FRSTER  ;SET ERROR FLAG
1029 014456 012737 177777 001400  7$:   RTS    PC        ;RETURN
1030 014464 000207          8$:   TST    LIMIT      ;TIMEOUT LIMIT REACHED ?
1031 014466 005737 001410  BEQ    9$          ;BR IF IT HAS
1032 014472 001403          DEC    LIMIT      ;DECREMENT LIMIT COUNTER
1033 014474 005337 001410  BNE    10$         ;BR IF NOT AT LIMIT
1034 014500 001005          9$:   BIT    #SW07,@SWR ;PRINT ALL DATA COMPARE ERRORS ?
1035 014502 032777 000200 164444 BNE    10$         ;BR IF YES
1036 014510 001001          RTS    PC        ;RETURN
1037 014512 000207          10$:  MOV    R1,-(SP)   ;BUFFER ADDRESS
1038 014514 010146          SUB    #2,(SP)    ;ADJUST ADDRESS
1039 014516 162716 000002  JSR    PC,LINOC   ;TYPE IT
1040 014522 004737 023124  DISPLY ,BLNKS2    ;TYPE 2 BLANKS
1041 014526 104414 073270  MOV    -2(R5),-(SP);PUT GOOD DATA ON THE STACK
1042 014532 016546 177776  JSR    PC,LINOC   ;TYPE IT
1043 014536 004737 023124  DISPLY ,BLNKS2    ;TYPE 2 BLANKS
1044 014542 104414 073270  MOV    -2(R1),-(SP);BAD DATA
1045 014546 016146 177776  JSR    PC,LINOC   ;TYPE IT
1046 014552 004737 023124  DISPLY ,$CRLF      ;CR-LF
1047 014556 104414 001203  RTS    PC        ;RETURN
1048 014562 000207
1049
1050          ;LAST LINE OF COMPARE ERROR REPORTING
1051
1052 014564 105737 001401  ENDCMP: TSTB   FRSTER+1 ;ANY COMPARE ERRORS FOUND ?
1053 014570 001417          BEQ    2$          ;BR IF NOT
1054 014572 005737 001406  TST    ERCTR      ;SEE HOW MANY ERRORS

```

```

1055 014576 001410          BEQ      1$          ;BR IF ONLY CAN'T MATCH PATTERN
1056 014600 104414 072625  DISPLY   ,LIN9     ;'NUMBER OF ERRORS='
1057 014604 013746 001406  MOV      ERCTR,-(SP) ;NUMBER OF ERRORS
1058 014610 004737 023156  JSR      PC,LINDEC ;TYPE IT
1059 014614 104414 001203  DISPLY   ,%CRLF    ;CR-LF
1060 014620 004737 024340  1$:     JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
1061 014624 004737 022654  JSR      PC,LINE7  ;PRINT LINE 7 OF ERROR MESSAGE
1062 014630 000207          2$:     RTS       PC   ;RETURN
1063
1064
1065          ;ROUTINE TO MATCH THE DATA WITH A PATTERN
1066          ;CALL:
1067          ;     MOV      #BUFFER,R1      ;BUFFER ADDRESS
1068          ;     JSR      PC,MATCH
1069          ;     RETURN1                     ;PATTERN ADDRESS IN R4
1070          ;     RETURN2                     ;COULDN'T MATCH PATTERN
1071
1072 014632 010146          MATCH:  MOV      R1,-(SP)   ;SAVE R1 ON THE STACK
1073 014634 012704 000044  MOV      #4,R4      ;PATTERN TABLE INDEX
1074 014640 011601          1$:     MOV      (SP),R1   ;RELOAD R1
1075 014642 162704 000002  SUB      #2,R4      ;DECREMENT INDEX
1076 014646 001413          BEQ      3$          ;BR IF PATTERN NOT MATCH
1077 014650 016405 002502  MOV      STNDAT(R4),R5 ;ADDRESS OF PATTERN ADDRESS
1078 014654 012703 000004  MOV      #4,R3      ;NUMBER OF LOCATIONS TO CHECK
1079 014660 022125          2$:     CMP      (R1)+,(R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN
1080 014662 001366          BNE      1$          ;BR IF NOT EQUAL, TRY NEXT PATTERN
1081 014664 005303          DEC      R3         ;FINISHED CHECKING?
1082 014666 001374          BNE      2$          ;BR IF NOT FINISHED
1083 014670 062704 002502  ADD      #STNDAT,R4 ;MAKE PATTERN ADDRESS ABSOLUTE
1084 014674 000403          BR       4$          ;EXIT
1085 014676 062766 000002 000002  3$:     ADD      #2,2(SP) ;INCREMENT RETURN ADDRESS
1086 014704 012601          4$:     MOV      (SP)+,R1 ;RESTORE R1
1087 014706 000207          RTS       PC   ;RETURN
1088
1089          ;USE ECC TO CORRECT THE DATA ERROR
1090
1091 014710 016037 002122 001422  ECC:    MOV      $RMBA(RO),ECSEC ;ADDRESS OF LAST LOCN XFERED
1092 014716 016046 002120  MOV      $RMWC(RO),-(SP) ;ACT WORDS XFERED (2'S COMP)
1093 014722 066016 000020  ADD      $WRDL(RO),(SP) ;ADD WORDS REQUESTED
1094 014726 005046          CLR      -(SP)      ;CLEAR NEXT STACK LOCN
1095 014730 016046 000022  MOV      $SSEC(RO),-(SP) ;SECTOR SIZE
1096 014734 004737 030610  JSR      PC,LINKDV  ;DIVIDE WORDS XFERED BY SECTOR SIZE
1097 014740 005716          TST      (SP)      ;PARTIAL SECTOR XFERED ?
1098 014742 001413          BEQ      1$          ;BR IF NOT
1099 014744 006316          ASL      (SP)      ;CONVERT INTO NUMBER OF BYTES
1100 014746 161637 001422  SUB      (SP),ECSEC ;SUBTRACT SECTOR RESIDUE
1101 014752 126027 000024 000005  CMPB    $CODE(RO),#5 ;WAS OP READ HEAD & DATA
1102 014760 001007          BNE      2$          ;BR IF NOT
1103 014762 062737 000004 001422  ADD      #4,ECSEC   ;ADD HEADER SIZE (IN BYTES) BACK IN
1104 014770 000403          BR       2$          ;GO ADJUST THE STACK POINTER
1105 014772 162737 001000 001422  1$:     SUB      #1000,ECSEC ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
1106 015000 062706 000004 000004  2$:     ADD      #4,SP  ;ADJUST THE STACK POINTER
1107 015004 016037 002162 001420  MOV      $RMEC1(RO),ECBIT ;ECC POSITION COUNT
1108 015012 005337 001420  DEC      ECBIT     ;ADJUST THE POSITION COUNT
1109 015016 013737 001420 001430  MOV      ECBIT,ECWRD ;LOAD THE WORD COUNT LOCATION
1110 015024 042737 177760 001420  BIC     #^C17,ECBIT ;SAVE THE BIT OFFSET COUNT
1111 015032 042737 000017 001430  BIC     #17,ECWRD  ;CLEAR THE BIT OFFSET

```

1112	015040	006237	001430		ASR	ECWRD	:CHANGE TO BYTE COUNT	
1113	015044	006237	001430		ASR	ECWRD	:CHANGE TO BYTE COUNT	
1114	015050	006237	001430		ASR	ECWRD	:CHANGE TO BYTE COUNT	
1115	015054	104414	072704		DISPLY	,LIN10A	: 'ERROR BURST BEGINS AT '	
1116	015060	013746	001430		MOV	ECWRD,-(SP)	:PUT THE WORD COUNT ON THE STACK	
1117	015064	006216			ASR	(SP)	:CONVERT TO WORD COUNT FOR MESSAGE	
1118	015066	004737	031524		JSR	PC,\$SB2D	:CONVERT THE WORD COUNT	
1119	015072	004737	031124		JSR	PC,\$SUPRS	:PRINT IT	
1120	015076	104414	072740		DISPLY	,LIN10B	: ' IN DATA FIELD OF ERROR SECTOR'	
1121	015102	063737	001422	001430	ADD	ECSEC,ECWRD	:FIND THE BEGINNING OF THE ERROR BURST	
1122	015110	026037	002122	001430	CMP	\$RMB(A(R0),ECWRD	:SEE IF BURST WAS IN DATA READ	
1123	015116	101002			BHI	+.6	:BR IF IN DATA READ	
1124	015120	000137	015436		JMP	ECC2	:NOT IN DATA READ - REPORT IT	
1125	015124	016037	002164	001424	MOV	\$RMEC2(R0),ECMSK0	:GET THE ERROR MASK	
1126	015132	005037	001426		CLR	ECMSK1	:CLEAR THE UPPER MASK WORD	
1127	015136	005737	001420	3\$:	TST	ECBIT	:BIT OFFSET EQUAL ZERO	
1128	015142	001407			BEQ	4\$:BR IF IT IS	
1129	015144	005337	001420		DEC	ECBIT	:DECREMENT THE BIT OFFSET COUNT	
1130	015150	006337	001424		ASL	ECMSK0	:SHIFT THE ERROR MASK	
1131	015154	006137	001426		ROL	ECMSK1	:SHIFT THE LOWER INTO THE UPPER	
1132	015160	000766			BR	3\$:CONTINUE THE SHIFT	
1133	015162	017737	164242	001434	4\$:	MOV	@ECWRD,ECBADO	:SAVE THE INCORRECT WORD
1134	015170	005037	001436		CLR	ECWRD1	:CLEAR SECOND INCORRECT WORD ADDRESS	
1135	015174	013746	001424		MOV	ECMSK0,-(SP)	:PUT LOWER MASK ON STACK	
1136	015200	047716	164224		BIC	@ECWRD,(SP)	:CLEAR ERRONEOUS ONE BITS FROM MASK	
1137	015204	043777	001424	164216	BIC	ECMSK0,@ECWRD	:CLEAR ERRONEOUS ONE BITS FROM BAD WORD	
1138	015212	052677	164212		BIS	(SP)+,@ECWRD	:SET DROPPED BITS	
1139	015216	005737	001426		TST	ECMSK1	:DOES BURST GO INTO NEXT WORD ?	
1140	015222	001431			BEQ	ECC1	:BR IF BURST ONLY IN ONE WORD	
1141	015224	013737	001430	001436	MOV	ECWRD,ECWRD1	:DUPLICATE ADDRESS	
1142	015232	062737	000002	001436	ADD	#2,ECWRD1	:INCREMENT ERROR ADDRESS	
1143	015240	026037	002122	001436	CMP	\$RMB(A(R0),ECWRD1	:IS NEXT WORD IN THE BUFFER	
1144	015246	101003			BHI	5\$:BR IF IT IS	
1145	015250	005037	001436		CLR	ECWRD1	:CLEAR 2ND WORD ADDRESS	
1146	015254	000414			BR	ECC1	:PRINT WORD CORRECTED	
1147	015256	017737	164154	001442	5\$:	MOV	@ECWRD1,ECBAD1	:SAVE THE SECOND BAD WORD
1148	015264	013746	001426		MOV	ECMSK1,-(SP)	:PUT THE UPPER MASK ON THE STACK	
1149	015270	047716	164142		BIC	@ECWRD1,(SP)	:CLEAR ERRONEOUS ONE BITS FROM UPPER MASK	
1150	015274	043777	001426	164134	BIC	ECMSK1,@ECWRD1	:CLEAR ERRONEOUS ONE BITS FROM DATA WORD	
1151	015302	052677	164130		BIS	(SP)+,@ECWRD1	:SET DROPPED BITS	
1152	015306	104414	073106		ECC1:	DISPLY	,LIN10H	:HEADER
1157	015312	013746	001430		MOV	ECWRD,-(SP)	:PUT ECWRD ON THE STACK	
	015316	004737	023124		JSR	PC,LIN0CT	:TYPE ECWRD	
	015322	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS	
	015326	013746	001434		MOV	ECBADO,-(SP)	:PUT ECBADO ON THE STACK	
	015332	004737	023124		JSR	PC,LIN0CT	:TYPE ECBADO	
	015336	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS	
	015342	017746	164062		MOV	@ECWRD,-(SP)	:PUT @ECWRD ON THE STACK	
	015346	004737	023124		JSR	PC,LIN0CT	:TYPE @ECWRD	
	015352	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS	
1158	015356	005737	001436		TST	ECWRD1	:PRINT THE NEXT WORD ?	
1159	015362	001427			BEQ	ECCX	:BR IF NOT	
1160	015364	104414	001203		DISPLY	,\$CRLF	:CR-LF	
1165	015370	013746	001436		MOV	ECWRD1,-(SP)	:PUT ECWRD1 ON THE STACK	
	015374	004737	023124		JSR	PC,LIN0CT	:TYPE ECWRD1	
	015400	104414	073270		DISPLY	,BLNKS2	:TYPE 2 BLANKS	
	015404	013746	001442		MOV	ECBAD1,-(SP)	:PUT ECBAD1 ON THE STACK	

```

015410 004737 023124      JSR    PC,LINOC      ;TYPE ECBAD1
015414 104414 073270      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
015420 017746 164012      MOV    @ECWRD1,-(SP) ;PUT @ECWRD1 ON THE STACK
015424 004737 023124      JSR    PC,LINOC      ;TYPE @ECWRD1
015430 104414 073270      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
1166 015434 000402      BR     ECCX          ;EXIT
1167 015436 104414 073001  ECC2:  DISPLY ,LIN10C  ;ERROR BURST WAS NOT TRANSFERED TO MEMORY
1168 015442 104414 001203  ECCX:  DISPLY ,$CRLF  ;CR-LF
1169 015446 000207      RTS     PC           ;RETURN
1170
1171      ;ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
1172
1173 015450 032777 000010 163476 PRTBAD: BIT    #SW3,@SWR    ;PRINT THE BAD SECTOR ?
1174 015456 001460      BEQ    6$           ;BR IF NOT
1175 015460 016001 002122      MOV    $RMB(R0),R1  ;PUT THE END ADDRESS INTO R1
1176 015464 016046 000020      MOV    $WRDL(R0),-(SP) ;FIND THE BEGINNING OF THE SECTOR
1177 015470 066016 002120      ADD    $RMWC(R0),(SP) ;SUBTRACT THE WORDS NOT TRANSFERED
1178 015474 005046      CLR    -(SP)        ;MAKE THE UPPER DIVIDEND 0
1179 015476 016046 000022      MOV    $SSEC(R0),-(SP) ;DIVIDE THE WORDS TRANSFERED BY THE SECTOR SIZE
1180 015502 004737 030610      JSR    PC,LINKDV    ;DIVIDE
1181 015506 005716      TST   (SP)          ;REMAINDER = 0 ?
1182 015510 001403      BEQ    1$           ;BR IF IT IS - COMPLETE SECTOR TRANSFERED
1183 015512 006316      ASL   (SP)          ;CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
1184 015514 161601      SUB   (SP),R1       ;SUBTRACT IT FROM THE END ADDRESS
1185 015516 000410      BR     2$           ;FINISH THE SIZING
1186 015520 162701 001000 1$:    SUB   #1000,R1      ;SUBTRACT FULL SECTOR SIZE FROM END ADDR
1187 015524 126027 000024 000005  CMBP  $CODE(R0),#5  ;WAS OPERATION READ HEADER & DATA ?
1188 015532 001002      BNE   2$           ;BR IF NOT
1189 015534 162701 000004      SUB   #4,R1        ;SUBTRACT HEADER SIZE FROM ADDR
1190 015540 062706 000004 2$:    ADD   #4,SP        ;RESTORE THE STACK POINTER
1191 015544 104414 073173      DISPLY ,LIN11H      ;PRINT THE HEADER
1192 015550 012702 000007 3$:    MOV   #7,R2        ;R2 CONTAINS THE WORDS/LINE COUNT
1193 015554 010146      MOV   R1,-(SP)     ;PUT THE ADDRESS ON THE STACK
1194 015556 004737 023124      JSR    PC,LINOC      ;TYPE THE ADDRESS
1195 015562 020160 002122 4$:    CMP   R1,$RMB(R0)  ;PRINTED ALL THE SECTOR ?
1196 015566 001412      BEQ   5$           ;BR IF ALL PRINTED
1197 015570 104414 073270      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
1198 015574 012146      MOV   (R1)+,-(SP)  ;PUT THE DATA ON THE STACK
1199 015576 004737 023124      JSR    PC,LINOC      ;TYPE THE DATA
1200 015602 005302      DEC   R2           ;DECREMENT THE HORIZONTAL COUNT
1201 015604 001366      BNE   4$          ;BR IF NOT AT THE END OF THE LINE
1202 015606 104414 001203      DISPLY , $CRLF      ;CR-LF
1203 015612 000756      BR    3$          ;RESTORE THE WORDS/LINE COUNT
1204 015614 104414 001203 5$:    DISPLY , $CRLF      ;PRINT WHAT REMAINS IN THE BUFFER
1205 015620 000207 6$:    RTS     PC           ;RETURN
1206
1207      ;ROUTINE TO DO AN RTC - DRIVE SELECTED IN R0
1208      ;CALL:
1209      ;
1210      ;     MOV    #DPB,R0      ;DPB ADDRESS
1211      ;     JSR    PC,RTNCTR
1212      ;     RETURN
1213 015622 111037 066070 066072 RTNCTR: MOVB  (R0),GENDPB  ;MOVE THE DRIVE # TO THE GENERAL DPB
1214 015626 112737 000117      MOVB  #RTC,GENDPB+$COMND ;COMMAND CODE
1215 015634 004037 036550 1$:    JSR    R0,RM05      ;DRIVER ENTRANCE
1216 015640 066070      GENDPB  ;DPB ADDRESS FOR ORDER
1217 015642 000774      BR    1$          ;DRIVER DIDN'T ACCEPT ORDER

```

```

1218 015644 000207          RTS      PC          ;RETURN
1219
1220          ;ROUTINE TO DO A RECALIBRATE - DRIVE SELECTED IN RO
1221          ;CALL:
1222          ;      MOV      #DPB,RO          ;DPB ADDRESS
1223          ;      JSR      PC,RECALT
1224          ;      RETURN
1225          ;OR
1226          ;      MOV      #DPB,RO          ;DPB ADDRESS
1227          ;      MOVB    #DRIVE,GENDPB    ;DRIVE ADDRESS
1228          ;      JSR      PC,RECALTO
1229          ;      RETURN
1230
1231 015646 111037 066070          RECALT: MOVB    (RO),GENDPB    ;MOVE THE DRIVE # TO THE GENERAL DPB
1232 015652 112737 000107 066072 RECALO: MOVB    #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
1233 015660 005060 002124          CLR      $RMDA(RO)          ;FAKE OUT REGISTERS FOR RECAL
1234 015664 005060 002152          CLR      $RMDC(RO)
1235 015670 004037 036550          1$:      JSR      RO,RM05          ;DRIVER ENTRANCE
1236 015674 066070          GENDPB          ;DPB ADDRESS FOR ORDER
1237 015676 000774          BR       1$          ;DRIVER DIDN'T ACCEPT THE ORDER
1238 015700 005737 066106          2$:      TST      GENDPB+$STATUS    ;SEE IF FINISHED
1239 015704 001775          BEQ     2$          ;BR IF NOT FINISHED
1240 015706 000207          RTS      PC          ;RETURN
1241
1242          ;OFFSET THE DRIVE IN RO (OFFSET CODE PRELOADED INTO 'RMOF')
1243          ;CALL:
1244          ;      MOVB    #OFFSET,GENDPB+$FMT ;OFFSET CODE
1245          ;      MOV      #DPB,RO          ;DPB ADDRESS
1246          ;      JSR      PC,OFFST
1247          ;      RETURN
1248
1249 015710 111037 066070          OFFST: MOVB    (RO),GENDPB    ;DRIVE # TO GENERAL DPB
1250 015714 112737 000115 066072 MOVB    #OFFSET,GENDPB+$COMND ;COMMAND
1251 015722 004037 036550          1$:      JSR      RO,RM05          ;DRIVER ENTRANCE
1252 015726 066070          GENDPB          ;DPB ADDRESS FOR ORDER
1253 015730 000774          BR       1$          ;DRIVER DIDN'T ACCEPT ORDER
1254 015732 000207          RTS      PC
1255
1256          ;UTILITY READ HEADER ROUTINE
1257          ;CALL:
1258          ;      MOV      #DPB,RO          ;DPB ADDRESS
1259          ;      MOV      #SECTOR,-(SP)    ;SECTOR ADDRESS
1260          ;      MOV      #TRACK,-(SP)    ;TRACK ADDRESS
1261          ;      MOV      #CYLINDER,-(SP) ;CYLINDER ADDRESS
1262          ;      JSR      PC,READDR
1263          ;      RETURN
1264
1265 015734 116637 000004 066101 READHD: MOVB    4(SP),GENDPB+$STRK ;TRACK ADDRESS
1266 015742 116637 000006 066100 MOVB    6(SP),GENDPB+$SEC ;SECTOR ADDRESS
1267 015750 016637 000002 066102 MOV      2(SP),GENDPB+$CYL ;CYLINDER ADDRESS
1268 015756 111037 066070          MOVB    (RO),GENDPB          ;DRIVE NUMBER
1269 015762 112737 000173 066072 MOVB    #RDHD,GENDPB+$COMND ;COMMAND
1270 015770 012737 177776 066074 MOV      #-2,GENDPB+$WRDM          ;WORD CTR = 2
1271 015776 004037 036550          1$:      JSR      RO,RM05          ;DRIVER ENTRANCE
1272 016002 066070          GENDPB          ;DPB ADDRESS FOR ORDER
1273 016004 000774          BR       1$          ;DRIVER DIDN'T ACCEPT COMMAND
1274 016006 005737 066106          2$:      TST      GENDPB+$STATUS    ;FINISHED?

```

```

1275 016012 001775          BEQ      2$          ;BR IF NOT
1276 016014 011666 000006  MOV      (SP),6(SP)  ;ADJUST STACK FOR RETURN
:277 016020 062706 000006  ADD      #6,SP       ;ADJUST RETRUN POINTER
1278 016024 000207          RTS       PC          ;RETURN
1279
1280          ;RETRY THE PRESENT OPERATION
1281          ;CALL:
1282          ;
1283          ;      MOV      #COUNT,RETRY ;RETRY COUNT
1284          ;      JSR      PC,$RETRY
1285          ;      RETURN1 ;RETRY UNSUCCESSFUL
1286          ;      RETURN2 ;SUCCESSFUL RETRY
1287          ;      ;NOTE: IF A DIFFERENT ERROR OCCURS DURING
1288          ;      ;RETRY, THE ROUTINE EXITS TO 'ERPRC1'
1289 016026 004737 017024  $RETRY: JSR      PC,GODRIV ;RE-START ORDER
1290 016032 005760 000016  1$:     TST      $STATUS(R0) ;ORDER FINISHED?
1291 016036 001775          BEQ      1$          ;BR IF NOT
1292 016040 100405          BMI      2$          ;BR IF ERROR
1293 016042 105237 001353  INCB    RETRY+1      ;INCREMENT RETRY COUNT
1294 016046 062716 000002  ADD     #2,(SP)      ;INCREMENT RETURN
1295 016052 000425          BR       5$          ;GO TO EXIT
1296 016054 032760 000200 000016  2$:     BIT      #BIT7,$STATUS(R0) ;DID ORDER TERMINATE NORMALLY ?
1297 016062 001430          BEQ      7$          ;BR IF NOT
1298 016064 005737 001350  TST     MASK         ;IS ERROR MASK 0 ?
1299 016070 001004          BNE     3$          ;BR IF NOT
1300 016072 005760 002132  TST     $RMER1(R0)  ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
1301 016076 001014          BNE     6$          ;BR IF NOT
1302 016100 000404          BR      4$          ;CONTINUE RETRY
1303 016102 033760 001350 002132  3$:     BIT      MASK,$RMER1(R0) ;SAME ERROR?
1304 016110 001407          BEQ      6$          ;BR IF NOT
1305 016112 105237 001353  4$:     INCB    RETRY+1      ;INCREMENT RETRY COUNT
1306 016116 123737 001352 001353  CMPB   RETRY,RETRY+1 ;DONE ?
1307 016124 001340          BNE     $RETRY      ;BR IF NOT DONE
1308 016126 000207          5$:     RTS       PC          ;RETURN
1309 016130 004737 023112  6$:     JSR      PC,LINE8  ;REPORT DIFFERENT ERROR
1310 016134 004737 022654  JSR     PC,LINE7    ;PRINT LINE 7
1311 016140 005726          TST     (SP)+        ;ADJUST STACK POINTER FOR DIRECT RETURN
1312 016142 000207          RTS       PC          ;RETURN
1313 016144 104414 072455  7$:     DISPLY  ,LIN8M    ;'DIFFERENT ERROR DURING RETRY'
1314 016150 000137 007266  JMP     ERPRC1      ;REPORT THE ERROR
1315
1316          ;ROUTINE TO UPDATE THE PERFORMANCE SUMMARY STATISTICS
1317          ;CALL:
1318          ;
1319          ;      MOV      #DPB,R0      ;DPB ADDRESS
1320          ;      JSR      PC,STATIS
1321          ;      RETURN
1322 016154 032760 000300 000016  STATIS: BIT      #BIT07:BIT06,$STATUS(R0) ;CHECK FOR DATA TERMINATION
1323 016162 001451          BEQ      3$          ;BR IF NOT DATA TERMINATION
1324 016164 016037 002122 016310  MOV     $RMB(A(R0),FACTOR ;STORE THE FINAL BUFFER ADDRESS
1325 016172 166037 000006 016310  SUB     $BUF(R0),FACTOR ;SUBTRACT THE INITIAL ADDRESS
1326 016200 001431          BEQ      2$          ;BR IF NO DATA TRANSFER
1327 016202 006237 016310  ASR     FACTOR        ;CONVERT TO A WORD COUNT
1328 016206 063760 016310 000046  ADD     FACTOR,$TRANS(R0) ;UPDATE WORD COUNT
1329 016214 005560 000050  ADC     $TRANS+2(R0)  ;ADD ANY CARRY
1330 016220 132760 000002 000024  BITB   #BIT01,$CODE(R0) ;SEE IF ORDER READ OR WRITE
1331 016226 001016          BNE     2$          ;BRANCH IF ORDER WRITE

```

```
1332 016230 126027 000024 000001      CMPB   $CODE(R0),#1      ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
1333 016236 101005                BHI    1$                ;BR IF NOT
1334 016240 066060 000020 000046      ADD    $WRDL(R0),$TRANS(R0) ;ADD WORDS WRITTEN
1335 016246 005560 000050                ADC    $TRANS+2(R0)      ;ADD A CARRY
1336 016252 063760 016310 000052 1$:    ADD    FACTOR,$READ(R0)   ;UPDATE THE READ WORD COUNT
1337 016260 005560 000054                ADC    $READ+2(R0)      ;ADD ANY CARRY
1338 016264 026060 000012 002152 2$:    CMP    $CYL(R0),$RMDC(R0) ;DID MID-TRANSFER SEEK OCCUR
1339 016272 001405                BEQ    3$                ;BR IF NOT
1340 016274 062760 000001 000042      ADD    #1,$POSIT(R0)     ;INCREMENT SEEK COUNT
1341 016302 005560 000044                ADC    $POSIT+2(R0)     ;ADD CARRY TO UPPER WORD
1342 016306 000207                3$:    RTS    PC
1343
1344 016310 000000      FACTOR: .WORD 0          ;USED FOR WORDS TRANSFERED
1345
1346      ;ROUTINE TO GET A BUFFER
1347      ;CALL:
1348      :      MOV    #DPB,R0          ;DPB ADDRESS
1349      :      CLR    -(SP)           ;CLEAR THE STACK
1350      :      JSR    PC,GETBUF
1351      :      RETURN
1352      :      ;BUFFER ADDRESS WILL BE ON THE STACK
1353      :      ;STACK WILL BE ZERO IF NO BUFFER AVAILABLE
1354 016312 010146      GETBUF: MOV    R1,-(SP)        ;SAVE R1
1355 016314 010246      MOV    R2,-(SP)        ;SAVE R2
1356 016316 010346      MOV    R3,-(SP)        ;SAVE R3
1357 016320 013702 001704      MOV    BUFTBL,R2       ;NUMBER OF SEPARATE BUFFERS
1358 016324 001444      BEQ    6$              ;BR IF NONE AVAILABLE
1359 016326 012701 001706      MOV    #BUFTBL+2,R1    ;FIRST ADDRESS OF ALLOCATION TABLE
1360 016332 026061 000020 000002 1$:    CMP    $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
1361 016340 101405      BLOS  3$              ;BRANCH IF IT IS
1362 016342 005302      DEC    R2              ;DECREMENT TABLE COUNT
1363 016344 001434      BEQ    6$              ;BR IF THROUGH TABLE
1364 016346 062701 000004      ADD    #4,R1           ;INCREMENT TABLE POINTER
1365 016352 000767      BR     1$             ;CONTINUE LOOKING
1366 016354 011166 000010 000002 3$:    MOV    (R1),10(SP)     ;BUFFER ADDRESS TO STACK
1367 016360 166061 000020      SUB    $WRDL(R0),2(R1) ;ADJUST BUFFER SIZE
1368 016366 001407      BEQ    4$              ;IF DIFFERENCE IS ZERO
1369 016370 006360 000020      ASL    $WRDL(R0)       ;CONVERT # WORDS TO BYTES
1370 016374 066011 000020      ADD    $WRDL(R0),(R1)  ;MAKE NEW STARTING ADDRESS
1371 016400 006260 000020      ASR    $WRDL(R0)       ;RETURN # BYTES TO WORDS
1372 016404 000414      BR     6$             ;RETURN
1373 016406 005337 001704      4$:    DEC    BUFTBL        ;DECREMENT ENTRIES COUNT
1374 016412 001411      BEQ    6$              ;BR IF ALLOCATION TABLE EMPTY
1375 016414 005302      DEC    R2              ;DECREMENT TABLE COUNT
1376 016416 001407      BEQ    6$              ;BR IF ITEM WERE LAST ENTRY
1377 016420 010103      MOV    R1,R3          ;MOVE TABLE POINTER
1378 016422 062703 000004      ADD    #4,R3          ;POINT TO NEXT ENTRY
1379 016426 012321 000004      5$:    MOV    (R3)+,(R1)+    ;MOVE ITEMS
1380 016430 012321      MOV    (R3)+,(R1)+
1381 016432 005302      DEC    R2              ;DECREMENT TABLE COUNT
1382 016434 001374      BNE  5$              ;CONTINUE IF NOT AT END OF TABLE
1383 016436 012603      6$:    MOV    (SP)+,R3      ;RESTORE R3
1384 016440 012602      MOV    (SP)+,R2      ;RESTORE R2
1385 016442 012601      MOV    (SP)+,R1      ;RESTORE R1
1386 016444 000207      RTS    PC            ;RETURN
1387
1388
```

```
1389 ;ROUTINE TO PUT BUFFER BACK IN TABLE
1390 ;CALL:
1391 ;
1392 ;     MOV     #DPB,R0      ;DPB ADDRESS
1393 ;     JSR     PC,RELBUF
1394 ;     RETURN
1395 016446 010146 RELBUF: MOV     R1,-(SP)      ;SAVE R1
1396 016450 010246     MOV     R2,-(SP)      ;SAVE R2
1397 016452 010446     MOV     R4,-(SP)      ;SAVE R4
1398 016454 010546     MOV     R5,-(SP)      ;SAVE R5
1399 016456 012701 001706     MOV     #BUFTBL+2,R1    ;BEGINNING OF TABLE
1400 016462 013702 001704     MOV     BUFTBL,R2      ;ENTRY COUNT
1401 016466 001424     BEQ     2$              ;BR IF EMPTY TABLE
1402 016470 016003 000020     MOV     $WRDL(R0),R3    ;TRIAL ADDRESS
1403 016474 006303     ASL     R3              ;CHANGE TO BYTE COUNT
1404 016476 066003 000006     ADD     $BUF(R0),R3     ;ADDRESS OF HIGHER ADJACENT BLOCK
1405 016502 021103 1$:     CMP     (R1),R3        ;UPPER ADJACENT BLOCK
1406 016504 001424     BEQ     4$              ;BR IF YES
1407 016506 062701 000004     ADD     #4,R1          ;INCREMENT POINTER
1408 016512 005302     DEC     R2              ;DECREMENT ENTRY COUNT
1409 016514 001372     BNE     1$              ;CONTINUE SEARCHING
1410 016516 016011 000006     MOV     $BUF(R0),(R1)   ;PUT THE BUFFER BLOCK INTO THE TABLE
1411 016522 016061 000020 000002     MOV     $WRDL(R0),2(R1) ;BLOCK SIZE
1412 016530 005237 001704     INC     BUFTBL         ;INCREMENT ENTRY COUNT
1413 016534 005202     INC     R2              ;INCREMENT R2 FOR USE LATER
1414 016536 000414     BR     5$              ;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
1415 016540 016021 000006 2$:     MOV     $BUF(R0),(R1)+  ;BLOCK ADDRESS TO TABLE
1416 016544 016021 000020     MOV     $WRDL(R0),(R1)+ ;SIZE TO TABLE
1417 016550 005237 001704     INC     BUFTBL         ;INCREMENT ENTRY COUNT
1418 016554 000443     BR     10$             ;EXIT
1419 016556 016011 000006 4$:     MOV     $BUF(R0),(R1)   ;RELEASED BUFFER IS LOWER ADJACENT
1420 016562 066061 000020 000002     ADD     $WRDL(R0),2(R1) ;INCREMENTED SIZE
1421 016570 010246 5$:     MOV     R2,-(SP)      ;SAVE R2
1422 016572 013702 001704     MOV     BUFTBL,R2      ;ENTRY COUNT
1423 016576 012705 001706     MOV     #BUFTBL+2,R5    ;BEGINNING OF TABLE
1424 016602 016504 000002 6$:     MOV     2(R5),R4        ;BLOCK SIZE (IN WORDS)
1425 016606 006304     ASL     R4              ;CHANGE TO BYTE COUNT
1426 016610 061504     ADD     (R5),R4        ;ADD BLOCK BEGINNING ADDRESS
1427 016612 020411     CMP     R4,(R1)        ;R1 STILL POINTS TO INSERTED ENTRY
1428 016614 001406     BEQ     8$              ;LOWER ADJACENT IN TABLE
1429 016616 062705 000004     ADD     #4,R5          ;INCREMENT POINTER
1430 016622 005302     DEC     R2              ;DECREMENT ENTRY COUNT
1431 016624 001366     BNE     6$              ;CONTINUE LOOKING
1432 016626 005726     TST     (SP)+          ;RESTORE STACK POINTER
1433 016630 000415     BR     10$             ;END
1434 016632 012602 8$:     MOV     (SP)+,R2       ;RESTORE R2
1435 016634 066165 000002 000002     ADD     2(R1),2(R5)    ;INCREMENT LOWER BLOCK LENGTH
1436 016642 005337 001704     DEC     BUFTBL         ;DECREMENT ENTRY COUNT
1437 016646 010105     MOV     R1,R5          ;GET READY TO COMPRESS
1438 016650 062705 000004     ADD     #4,R5          ;INCREMENT TO NEXT ENTRY
1439 016654 012521 9$:     MOV     (R5)+,(R1)+    ;COMPRESS TABLE
1440 016656 012521     MOV     (R5)+,(R1)+    ;MOVE SIZE FIELD DOWN
1441 016660 005302     DEC     R2              ;DECREMENT ENTRY COUNT
1442 016662 001374     BNE     9$              ;BR IF NOT FINISHED
1443 016664 012605 10$:    MOV     (SP)+,R5       ;RESTORE R5
1444 016666 012604     MOV     (SP)+,R4       ;RESTORE R4
1445 016670 012602     MOV     (SP)+,R2       ;RESTORE R2
```



```
1446 016672 012601      MOV      (SP)+,R1      ;RESTORE R1
1447 016674 000207      RTS       PC           ;RETURN
1448
1449
1450      ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
1451      ;CALL:
1452      :      MOV      #DPB,R0      ;DPB ADDRESS
1453      :      MOV      #BUFADR,$BUF(R0) ;LOAD BUFFER ADDRESS INTO THE DPB
1454      :      MOV      #PATTERN,$PATT(R0) ;PATTERN CODE
1455      :      JSR      PC,FILBUF
1456      :
1457
1458 016676 104412      FILBUF: SAVREG      ;SAVE THE REGISTERS
1459 016700 132760 000004 000024  BITB     #BIT02,$CODE(R0) ;SEE IF READ ORDER
1460 016706 001044      RNE      4$          ;BR IF READ
1461 016710 016001 000006      1$:     MOV      $BUF(R0),R1      ;BUFFER ADDRESS
1462 016714 016002 000020      MOV      $WRDL(R0),R2      ;POSITIVE WORD COUNT
1463 016720 132760 000001 000024  BITB     #BIT00,$CODE(R0) ;SEE IF WRITE HEADER TYPE ORDER
1464 016726 001413      BEQ      2$          ;BR IF NOT
1465 016730 016011 000012      MOV      $CYL(R0),(R1)      ;CYLINDER ADDRESS
1466 016734 052711 010000      BIS      #BIT12,(R1)      ;SET FMT22 BIT
1467 016740 052721 140000      BIS      #140000,(R1)+     ;SET MFG AND USER BITS
1468 016744 016021 000010      MOV      $SEC(R0),(R1)+     ;MOVE SECTOR & TRACK
1469 016750 162702 000002      SUB      #2,R2            ;ADJUST THE WORD COUNT
1470 016754 003421      BLE      4$          ;BR IF END OF PATTERN
1471 016756 005004      2$:     CLR      R4            ;CLEAR R4
1472 016760 116004 000030      MOV      $PATT(R0),R4      ;RELATIVE PATTERN ADDRESS
1473 016764 016405 002502      MOV      STNDAT(R4),R5     ;PATTERN ADDRESS
1474 016770 012703 000020      MOV      #20,R3           ;PATTERN COUNT
1475 016774 012521      3$:     MOV      (R5)+,(R1)+     ;MOVE THE PATTERN INTO THE BUFFER
1476 016776 005302      DEC      R2            ;DECREMENT THE WORD COUNT
1477 017000 003407      BLE      4$          ;BR IF DONE (WORD COUNT = 0)
1478 017002 005303      DEC      R3            ;DECREMENT THE PATTERN COUNT
1479 017004 001373      BNE      3$          ;BR IF MORE PATTERN
1480 017006 012703 000020      MOV      #20,R3           ;RESTORE PATTERN COUNT
1481 017012 016405 002502      MOV      STNDAT(R4),R5     ;RESTORE THE ADDRESS
1482 017016 000766      BR       3$          ;CONTINUE DISTRIBUTING THE PATTERN
1483 017020 104413      4$:     RESREG
1484 017022 000207      RTS       PC           ;RESTORE THE REGISTERS
1485      ;RETURN
1486
1487      ;START THE ORDER FOR THE DPB IN R0
1488      ;CALL:
1489      :      MOV      #DPB,R0      ;DPB ADDRESS
1490      :      JSR      PC,GODRIV
1491      :      RETURN
1492
1492 017024 010046      GODRIV: MOV      R0,-(SP)      ;SAVE R0
1493 017026 010037 017036      MOV      R0,2$          ;CURRENT DPB ADDRESS
1494 017032 004037 036550      1$:     JSR      R0,RM05      ;CALL THE DRIVE HANDLER
1495 017036 000000      2$:     .WORD     0          ;DRIVE BLOCK ADDRESS GOES HERE
1496 017040 000000      HALT
1497 017042 012600      ;DRIVER REJECTED REQUEST
1498 017044 062760 000001 000036  MOV      (SP)+,R0      ;RESTORE R0
1499 017052 005560 00 0      ADD      #1,$OPERC(R0)     ;INCREMENT THE OPERATION COUNT
1500 017056 026060 000004 000012  ADC      $OPERC+2(R0)
1501 017064 001405      CMP      $PREVA+2(R0),$CYL(R0) ;DID ORDER REQUIRE A CYLINDER CHANGE
1502 017066 062760 000001 000042  BEQ      3$          ;BR IF NOT
1503      ADD      #1,$POSIT(R0) ;INCREMENT SEEK COUNT
```

```

1503 017074 005560 000044      ADC    $POSIT+2(R0)    ;ADD ANY CARRY
1504 017100 000207      3$:   RTS    PC
1505
1506      ;GENERATE PARAMETERS FOR THE OPERATION
1507      ;CALL:
1508      ;     MOV    #DPB,R0      ;DPB ADDRESS
1509      ;     JSR    PC,SELPAR
1510      ;     RETURN
1511
1512 017102 004737 035014      SELPAR: JSR    PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
1513 017106 032777 000001 162040  BIT    #SWO,@SWR      ;SEE IF SWO SET
1514 017114 001012      BNE    2$            ;BR IF SET - READ ONLY
1515 017116 012705 000010      1$:   MOV    #10,R5      ;READ/WRITE SELECTION DIVISOR
1516 017122 004737 030562      JSR    PC,GETREM     ;GET SELECTION VALUE
1517 017126 020537 001504      CMP    R5,RATIO     ;DETERMINE IF READ OR WRITE
1518 017132 103003      BHIS   2$            ;BR IF READ
1519 017134 004737 017766      JSR    PC,RANWRT     ;SELECT A WRITE ORDER
1520 017140 000410      BR     THEAD        ;SELECT ADDRESS
1521 017142 013705 035114      2$:   MOV    $LONUM,R5  ;SELECT READ OPERATION CODE
1522 017146 042705 177776      BIC    #^C1,R5      ;MASK OUT ALL BUT BIT 0
1523 017152 062705 000004      ADD    #4,R5        ;TABLE OFFSET FOR READ CODE
1524 017156 110560 000074      MOVB   R5,$NCODE(R0) ;ORDER SELECTION CODE TO CONTROL BLOCK
1525 017162 005737 001516      THEAD: TST    HEADER  ;ENABLE RANDOM ADDRESS SELECT ?
1526 017166 001425      BEQ    RANSEC       ;YES
1527 017170 016060 002124 000076  MOV    $RMDA(R0),$NSEC(R0) ;SECTOR AND TRACK
1528 017176 016060 002152 000100  MOV    $RMDC(R0),$NCYL(R0) ;CYLINDER NUMBER
1529 017204 026060 000100 000106  CMP    $NICYL(R0),$MAXCYL(R0) ;OVER MAX CYLINDER # ?
1530 017212 103521      BLO    XWCNT        ;NO
1531 017214 016060 000110 000100  MOV    MINCYL(R0),$NICYL(R0) ;RESET CYLINDER NUMBER
1532 017222 116060 000120 000076  MOVB   MINSEC(R0),$NSEC(R0) ;RESET SECTOR NUMBER
1533 017230 116060 000114 000077  MOVB   MINTRK(R0),$NTRK(R0) ;RESET TRACK NUMBER
1534 017236 000137 017456      JMP    XWCNT        ;SELECT BUFFER SIZE
1535
1536      ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
1537
1538 017242 016005 000116      RANSEC: MOV    MAXSEC(R0),R5 ;GET MAXIMUM SECTOR ADDRESS
1539 017246 026005 000120      CMP    MINSEC(R0),R5 ;'MINSEC' AND 'MAXSEC' THE SAME ?
1540 017252 001417      BEQ    2$            ;BR IF THEY ARE
1541 017254 166005 000120      SUB    MINSEC(R0),R5 ;SUBTRACT MINIMUM SECTOR ADDRESS
1542 017260 100002      BPL    1$            ;BR IF MAX LARGER THAN MIN
1543 017262 062705 000040      ADD    #32.,R5      ;CORRECT THE NUMBER
1544 017266 005205      1$:   INC    R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
1545 017270 004737 030562      JSR    PC,GETREM     ;GET THE RANDOM AUGMENT
1546 017274 066005 000120      ADD    MINSEC(R0),R5 ;NEW ADDRESS
1547 017300 020527 000037      CMP    R5,#31.      ;IS VALUE TOO LARGE ?
1548 017304 101402      BLOS   2$            ;BR IF NOT
1549 017306 162705 000040      SUB    #32.,R5      ;CORRECT VALUE
1550 017312 110560 000076      2$:   MOVB   R5,$NSEC(R0) ;STORE SECTOR ADDRESS IN DPB
1551
1552      ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
1553
1554 017316 004737 026546      RANTRK: JSR    PC,GETLMT  ;GET ADDRESS LIMITS
1555 017322 016005 000112      MOV    MAXTRK(R0),R5 ;GET MAXIMUM TRACK ADDRESS
1556 017326 026005 000114      CMP    MINTRK(R0),R5 ;'MINTRK' AND 'MAXTRK' THE SAME ?
1557 017332 001421      BEQ    3$            ;BR IF THEY ARE
1558 017334 166005 000114      SUB    MINTRK(R0),R5 ;SUBTRACT MINIMUM TRACK ADDRESS
1559 017340 100003      BPL    2$            ;BR IF MAX LARGER THAN MIN

```

```

1560 017342 063705 001450      ADD    TRKLMT,R5      ;CORRECT THE NUMBER
1561 017346 005205              INC    R5
1562 017350 005205      2$:   INC    R5      ;INCREMENT DIFFERENCE TO USE AS DIVISOR
1563 017352 004737 030562      JSR    PC,GETREM     ;GET THE RANDOM AUGMENT
1564 017356 066005 000114      ADD    MINTRK(R0),R5 ;NEW TRACK ADDRESS
1565 017362 020537 001450      CMP    R5,TRKLMT    ;IS VALUE TOO LARGE ?
1566 017366 003403              BLE    3$           ;BR IF NO
1567 017370 163705 001450      SUB    TRKLMT,R5    ;CORRECT VALUE
1568 017374 005305              DEC    R5
1569 017376 110560 000077      3$:   MOVB   R5,$NTRK(R0) ;STORE TRACK ADDRESS IN DPB
1570
1571      ;GENERATE A RANDGM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
1572
1573 017402 016005 000106      RANCYL: MOV   MAXCYL(R0),R5 ;GET MAXIMUM CYLINDER ADDRESS
1574 017406 026005 000110      CMP    MINCYL(R0),R5 ;'MINCYL' AND 'MAXCYL' THE SAME ?
1575 017412 001417              BEQ    2$           ;BR IF THEY ARE
1576 017414 166005 000110      SUB    MINCYL(R0),R5 ;SUBTRACT MINIMUM CYLINDER ADDRESS
1577 017420 100002              BPL    1$           ;BR IF MAX LARGER THAN MIN
1578 017422 062705 001466      ADD    #822.,R5     ;CORRECT THE NUMBER
1579 017426 005205      1$:   INC    R5      ;INCREMENT DIFFERENCE TO USE AS DIVISOR
1580 017430 004737 030562      JSR    PC,GETREM     ;GET THE RANDOM AUGMENT
1581 017434 066005 000110      ADD    MINCYL(R0),R5 ;NEW CYLINDER ADDRESS
1582 017440 020527 001465      CMP    R5,#821.    ;IS VALUE TOO LARGE ?
1583 017444 003402              BLE    2$           ;BR IF NO
1584 017446 162705 001466      SUB    #822.,R5     ;CORRECT VALUE
1585 017452 010560 000100      2$:   MOV    R5,$NCYL(R0) ;STORE CYLINDER ADDRESS IN DPB
1586
1587 017456 132760 000001 000074 XWCNT: BITB  #BIT00,$NCODE(R0) ;HEADER OPERATION INVOKED ?
1588 017464 001414              BEQ    RANSIZ      ;NO
1589 017466 012760 000402 000102 MOV    #258.,$NWRDL(R0) ;CHANGE WORD LENGTH TO 258 FOR WRTHD ORDER
1590 017474 122760 000005 000074 3$:   CMPB  #5,$NCODE(R0) ;READ HEADER AND DATA ?
1591 017502 001461              BEQ    RANXIT      ;YES
1592 017504 004537 017656      JSR    R5,CHKADR   ;IF WRITE HEAD AND DATA COMMAND
1593      ;AVOID WRITING BAD SPOT HEADER
1594 017510 000452              BR    RANPAT      ;BRANCH IF NOT ON BAD SPOT
1595 017512 000137 017102      JMP    SELPAR     ;SELECT THE PARAMETERS AGAIN
1596
1597      ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
1598
1599 017516 013705 001466      RANSIZ: MOV   MAXDL,R5 ;GET BUFFER SIZE
1600 017522 005737 001502      TST   WCSEL      ;SELECT A RANDOM WORD COUNT ?
1601 017526 001010              BNE    1$         ;BR IF NOT
1602 017530 005205              INC    R5         ;INCREMENT THE MAXIMUM SIZE
1603 017532 004737 030562      JSR    PC,GETREM   ;DIVIDE BY MAX VALUE
1604 017536 005705              TST   R5         ;IS THE REMAINDER 0 ?
1605 017540 001003              BNE    1$         ;NOT 0, CONTINUE
1606 017542 004737 035014      JSR    PC,$RAND    ;CYCLE THE RANDOM NUMBER GENERATOR
1607 017546 000763              BR    RANSIZ      ;TRY AGAIN
1608 017550 022705 000004      1$:   CMP    #4,R5   ;LESS THAN 4 ?
1609 017554 003403              BLE    2$         ;NO
1610 017556 012705 000004      MOV    #4,R5     ;SET SIZE TO 4
1611 017562 000405              BR    3$
1612 017564 023705 001466      2$:   CMP    MAXDL,R5 ;LARGE THAN MAX ALLOWED ?
1613 017570 101002              BHI    3$         ;NO
1614 017572 013705 001466      MOV    MAXDL,R5  ;RESET COUNTER
1615 017576 132760 000004 000074 3$:   BITB  #BIT02,$NCODE(R0) ;READ OPERATION ?
1616 017604 001006              BNE    4$         ;YES

```

```

1617 017606 042705 000377      BIC    #377,R5      ;SECTOR BOUNDARY FOR WRITE OP
1618 017612 005705              TST    R5           ;NONE
1619 017614 003002              BGT    4$          ;NO
1620 017616 012705 000400      MOV    #256.,R5    ;AT LEAST ONE SECTOR
1621 017622 010560 000102      MOV    R5,$NWRDL(RU) ;WORD COUNT
1622 017626 132760 000004 000074 4$: BITB  #BIT02,$NCODE(R0) ;READ OP ?
1623 017634 001004              BNE    RANXIT      ;YES
1624
1625      ;GET A RANDOM PATTERN NUMBER
1626 017636 004737 020022      RANPAT: JSR    PC,GETPAT ;GET PATTERN CODE
1627 017642 110560 000075      MOV    R5,$NPATC(R0) ;MOVE PATTERN CODE TO CONTROL BLOCK
1628 017646 012760 177777 000104 RANXIT: MOV  #-1,$NEXT(R0) ;SET PARAMETERS SELECTED INDICATOR
1629 017654 000207              RTS    PC          ;RETURN
1630
1631      ;ROUTINE TO CHECK THE SELECTED ADDRESS IS NOT ON THE BAD SPOT
1632      ;CALLING SEQ
1633      ;      MOV    #DPB,R0      ;DPB ADDRESS
1634      ;      JSR    R5,CHKADR
1635      ;      RET1   NORMAL RETURN
1636      ;      RET2   ERROR RET
1637      ;
1638      ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
1639
1640 017656      CHKADR:
1641 017656 010246      MOV    R2,-(SP)    ;;PUSH R2 ON STACK
1642 017660 010346      MOV    R3,-(SP)    ;;PUSH R3 ON STACK
1643 017662 010002      MOV    R0,R2      ;TABLE ADDRESS
1644 017664 062702 000124      ADD    #SBDSEC,R2 ;DBP ADDRESS + TABLE ADDRESS OFFSET
1645 017670 022712 177777 1$: CMP    #-1,(R2)   ;EMPTY ENTRY OR TERMINATOR ?
1646 017674 001431      BEQ    5$          ;BRANCH IF SO
1647 017676 026012 000100      CMP    $NCYL(R0),(R2) ;ON THE SAME CYLINDER ?
1648 017702 001023      BNE    4$          ;BRANCH IF NOT
1649 017704 122762 177777 000003  CMPB  #-1,3(R2)   ;ON ALL TRACKS ?
1650 017712 001404      BEQ    2$          ;BR IF YES
1651 017714 126062 000011 000003  CMPB  $TRK(R0),3(R2) ;ON THE SAME TRACK ?
1652 017722 001013      BNE    4$          ;BRANCH IF NOT
1653 017724 122762 177777 000002 2$: CMPB  #-1,2(R2)   ;ON ALL SECTORS ?
1654 017732 001404      BEQ    3$          ;BR IF YES
1655 017734 126062 000010 000002  CMPB  $SEC(R0),2(R2) ;ON THE SAME SECTOR ?
1656 017742 001003      BNE    4$          ;BRANCH IF NOT
1657 017744 062705 000002 3$: ADD    #2,R5      ;FOUND BAD ADDRESS(ES) IN TABLE
1658 017750 000403      BR     5$          ;EXIT
1659 017752 062702 000004 4$: ADD    #4,R2      ;ADJUST TO NEXT TABLE ENTRY
1660 017756 000744      BR     1$          ;LOOP BACK
1661 017760 5$:
1662 017760 012603      MOV    (SP)+,R3    ;;POP STACK INTO R3
1663 017762 012602      MOV    (SP)+,R2    ;;POP STACK INTO R2
1664 017764 000205      RTS    R5         ;RETURN
1665
1666      ;ROUTINE TO SELECT A WRITE (OR WRITE HEAD AND DATA) OPERATION
1667 017766 012705 000002      RANWRT: MOV   #2,R5 ;SET WRITE DATA COMMAND
1668 017772 005737 001500      TST    FORMAT     ;ALLOW FORMAT OPERATION ?
1669 017776 001406              BEQ    1$          ;NO
1670 020000 122760 000004 000024  CMPB  #4,$CODE(R0) ;PREVIOUS A READ DATA COMMAND ?
1671 020006 001002              BNE    1$          ;NO
1672 020010 012705 000003      MOV    #3,R5      ;SET A WRITE HEAD AND DATA COMMAND

```

```

1670 020014 110560 000074 1$:   MOVB   R5,$NCODE(R0)  ;SELECT THE WRITE COMMAND
1671 020020 000207          RTS      PC                ;EXIT
1672
1673          ;ROUTINE TO SELECT A PATTERN
1674
1675 020022 012705 000020  GETPAT: MOV    #20,R5          ;SELECT PATTERN
1676 020026 005737 001514      TST    PATTEN          ;ENABLE RANDOM PATTERN SELECTION ?
1677 020032 001403          BEQ    2$              ;YES
1678 020034 013705 001514      MOV    PATTEN,R5       ;USE INDEXED PATTERN
1679 020040 000407          BR     1$              ;NO
1680 020042 004737 030562  2$:   JSR    PC,GETREM      ;GET CODE
1681 020046 005705          TST    R5              ;WAS PATTERN ZERO SELECTED ?
1682 020050 001003          BNE   1$              ;BR IF NOT ZERO
1683 020052 004737 035014      JSR    PC,$RAND        ;CYCLE THE RANDOM NUMBER GENERATOR
1684 020056 000761          BR     GETPAT          ;TRY AGAIN
1685 020060 006305 1$:   ASL    R5              ;MAKE CODE INTO TABLE INDEX
1686 020062 000207          RTS      PC
1687
1688          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
1689          ;CALL:
1690          ;
1691          ;   MOV    #DPB,R0          ;DPB ADDRESS
1692          ;   JSR    PC,SELPAR       ;SELECT THE PARAMETERS
1693          ;   JSR    PC,GETPAR
1694          ;   RETURN
1695 020064 010546  GETPAR: MOV    R5,-(SP)          ;SAVE R5
1696 020066 116060 002116 000027  MOVB   $RMC(S1(R0),$PREVO(R0) ;SAVE CURRENT PARAMETERS
1697 020074 142760 177701 000027  BICB  #^C76,$PREVO(R0)        ;STRIP GO,AND IE BITS
1698 020102 032760 000006 000074  BIT    #6,$NCODE(R0)          ;SEE IF NEXT OPERATION IS READ OR WRITE
1699 020110 001007          BNE   1$              ;BR IF EITHER
1700 020112 016060 000012 000034  MOV    $CYL(R0),$PREVA+2(R0)   ;SAVE STARTING CYLINDER
1701 020120 016060 000010 000032  MOV    $SEC(R0),$PREVA(R0)     ;SAVE STARTING SECTOR AND TRACK
1702 020126 000410          BR     22$
1703 020130 004737 023176 1$:   JSR    PC,READDR          ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
1704 020134 012660 000034          MOV    (SP)+,$PREVA+2(R0)      ;CYLINDER ADDRESS
1705 020140 112660 000033          MOVB  (SP)+,$PREVA+1(R0)      ;TRACK ADDRESS
1706 020144 112660 000032          MOVB  (SP)+,$PREVA(R0)       ;SECTOR ADDRESS
1710 020150 032777 000100 160776 22$:  BIT    #SW06,@SWR            ;SWITCH 6 SET ?
1711 020156 001073          BNE   5$              ;BR IF SET
1712 020160 116060 000074 000024  MOVB  $NCODE(R0),$CODE(R0)    ;LOGICAL CODE FOR OPERATION
1713 020166 116005 000074          MOVB  $NCODE(R0),R5          ;LOAD R5 FOR USE AS TABLE INDEX
1714 020172 116560 002126 000002  MOVB  COMTBL(R5),$COMND(R0)   ;COMMAND CODE
1715 020200 122760 000151 000002  CMPB  #151,$COMND(R0)        ;WRITE CHECK DATA COMMAND ?
1716 020206 001013          BNE   3$              ;NO,DO NOT CARE
1717 020210 122760 000060 000027  CMPB  #60,$PREVO(R0)        ;PREVIOUS A WRITE DATA COMMAND ?
1718 020216 001420          BEQ   4$              ;YES,OK
1719 020220 112760 000171 000002 2$:  MOVB  #171,$COMND(R0)        ;CHANG TO READ DATA COMMAND
1720 020226 112760 000004 000024  MOVB  #4,$CODE(R0)          ;CODE NUMBER CHANGED TO READ DATA
1721 020234 000411          BR     4$              ;EXIT
1722 020236 122760 000153 000002 3$:  CMPB  #153,$COMND(R0)        ;WRITE CHECK HEAD AND DATA COMMAND ?
1723 020244 001005          BNE   4$              ;NO,THEN EXIT
1724 020246 122760 000062 000077  CMPB  #62,$PREVO(R0)        ;PREVIOUS A WRITE HEAD AND DATA COMMAND?
1725 020254 001401          BEQ   4$              ;YES,EXIT
1726 020256 000760          BR     2$              ;SET TO READ DATA COMMAND
1727 020260
1728 020260 116060 000075 000030 4$:  MOVB  $NPATC(R0),$PATTC(R0)  ;PATTERN CODE
1729 020266 016060 000076 000010  MOV    $NSEC(R0),$SEC(R0)     ;TRACK AND SECTOR ADDRESSES

```

```

1730 020274 016060 000100 000012      MOV      $NCYL(R0), $CYL(R0)      ;CYLINDER ADDRESS
1731 020302 016060 000102 000020      MOV      $NWRDL(R0), $WRDL(R0)   ;BUFFER SIZE
1732 020310 016060 000102 000004      MOV      $NWRDL(R0), $WRDM(R0)   ;WORD COUNT
1733 020316 005460 000004      NEG      $WRDM(R0)                ;COMPLEMENT IT
1734 020322 012760 000400 000022      MOV      #256, $SSEC(R0)         ;INITIAL VALUE OF SECTOR SIZE
1735 020330 032760 000001 000024      BIT      #1, $CODE(R0)           ;HEADER OPERATION ?
1736 020336 001403      BEQ      5$                      ;BR IF NOT
1737 020340 062760 000002 000022      ADD      #2, $SSEC(R0)           ;ADD HEADER SIZE
1738 020346 005060 000104      5$:    CLR      $NEXT(R0)          ;RESET 'PARAMETERS LOADED' INDICATOR
1739 020352 012605      MOV      (SP)+, R5                ;RESTORE R5
1740 020354 000207      RTS      PC                       ;RETURN
1741
1742      ;ROUTINE TO COMPRESS A LIST
1743      ;CALL:
1744      ;      MOV      #ADDRS, R1      ;COMPRESS LIST STARTING AT THIS ADDRESS
1745      ;      JSR      PC, CMPRES
1746      ;      RETURN
1747
1748 020356 016111 000002      CMPRES: MOV      2(R1), (R1)       ;COMPRESS THE TABLE IN R1
1749 020362 001403      BEQ      1$                      ;BR WHEN ZERO FOUND
1750 020364 062701 000002      ADD      #2, R1                  ;INCREMENT R1
1751 020370 000772      BR      CMPRES                   ;CONTINUE COMPRESSING TABLE
1752 020372 000207      1$:    RTS      PC               ;RETURN
1753
1754      ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
1755      ;CALL:
1756      ;      MOV      #DPB, R0       ;DPB ADDRESS
1757      ;      MOV      #-1, $PACK(R0) ;'WRITE PACK' FLAG
1758      ;      OR
1759      ;      MOV      #1, $PACK(R0) ;'READ PACK' FLAG
1760      ;      JSR      PC, WRTPK
1761      ;      RETURN
1762
1763 020374 004737 035014      WRTPK:  JSR      PC, $RAND          ;CYCLE THE RANDOM NUMBER GENERATOR
1764 020400 005760 000040      TST      $OPERC+2(R0)            ;SEE IF FIRST OPERATION
1765 020404 001007      BNE      WRTPK1                  ;BR IF UPPER WORD OF COUNTER NOT ZERO
1766 020406 005760 000036      TST      $OPERC(R0)              ;LOWER WORD ZERO ?
1767 020412 001004      BNE      WRTPK1                  ;BR IF NOT 1ST OPERATION
1768 020414 105760 000026      TSTB     $PACK(R0)               ;SEE WHICH - 'R' OR 'W'
1769 020420 100530      BMI      WRTPK3                  ;BR IF 'W'
1770 020422 000515      BR      WRTPK2                  ;'R' OPERATION
1771 020424 116060 002116 000027      WRTPK1: MOVB     $RMCS1(R0), $PREV0(R0) ;SAVE CURRENT PARAMETERS
1772 020432 004737 023176      JSR      PC, READR              ;GET THE DECREMENTED SECTOR AND TRACK ADDRESSES
1773 020436 012660 000034      MOV      (SP)+, $PREVA+2(R0)     ;CYLINDER ADDRESS
1774 020442 112660 000033      MOVB     (SP)+, $PREVA+1(R0)     ;TRACK ADDRESS
1775 020446 112660 000032      MOVB     (SP)+, $PREVA(R0)       ;SECTOR ADDRESS
1776 020452 016060 002124 000010      MOV      $RMDC(R0), $SEC(R0)     ;NEW SECTOR & TRACK ADDRESS
1777 020460 016060 002152 000012      MOV      $RMDC(R0), $CYL(R0)     ;NEW CYLINDER ADDRESS
1778 020466 026060 000012 000106      CMP      $CYL(R0), $MAXCYL(R0)   ;SEE IF AT END
1779 020474 103436      BLO      2$                      ;BR IF LESS THAN 'MAXCYL'
1780 020476 101004      BHI      1$                      ;BR IF GREATER THAN 'MAXCYL'
1781 020500 126060 000011 000112      CMPB     $TRK(R0), $MAXTRK(R0)   ;SEE IF AT MAX TRACK
1782 020506 103431      BLO      2$                      ;BR IF NOT GREATER
1783 020510 116060 000114 000011      1$:    MOVB     $MINTRK(R0), $TRK(R0) ;RESET TRACK ADDRESS
1784 020516 116060 000120 000010      MOVB     $MINSEC(R0), $SEC(R0)   ;RESET SECTOR ADDRESS
1785 020524 016060 000110 000012      MOV      $MINCYL(R0), $CYL(R0)   ;RESET CYLINDER ADDRESS
1786 020532 112760 000004 000024      MOVB     #4, $CODE(R0)           ;SET CODE TO READ DATA

```

```

1787 020540 122760 177776 000026      CMPB   #-2,$PACK(RO)  ;WT OPERATION IN PROCESSING
1788 020546 001475                BEQ    WRTPK5          ;YES
1789 020550 004737 030254                JSR    PC,EOP2        ;DROP THE DRIVE (NORMAL TERMINATION)
1790
1791 020554 032777 000020 160372      BIT    #SW04,@SWR     ;IS SWITCH 4 SET ?
1792 020562 001003                BNE    2$             ;BR IF SET
1793 020564 005726                TST    (SP)+          ;INCREMENT THE STACK POINTER
1794 020566 000137 006100                JMP    MAIN           ;RETURN DIRECTLY TO 'MAIN'
1795 020572 013760 001466 000020 2$:      MOV    MAXDL,$WRDL(RO) ;BUFFER SIZE IS MAXIMUM
1796 020600 042760 000377 000020      BIC    #377,$WRDL(RO) ;SECTOR BOUNDARY FOR WRITTING
1797 020606 013760 001466 000004      MOV    MAXDL,$WRDM(RO) ;WORD COUNT
1798 020614 042760 000377 000004      BIC    #377,$WRDM(RO) ;SECTOR BOUNDARY FOR WRITTING
1799 020622 005760 000004                TST    $WRDM(RO)     ;SIZE=0 ?
1800 020626 003006                BGT    3$             ;NO
1801 020630 012760 000400 000004      MOV    #256.,$WRDM(RO) ;SET TO ONE SECTOR
1802 020636 012760 000400 000020      MOV    #256.,$WRDL(RO) ;SET ONE SECTOR
1803 020644 005460 000004 3$:      NEG    $WRDM(RO)      ;CHANGE WORD COUNT TO 2'S COMPLEMENT
1804 020650 105760 000026                TSTB   $PACK(RO)     ;READ OR WRITE ?
1805 020654 100412                BMI    WRTPK3         ;BR IF WRITE
1806 020656 012760 000402 000022  WRTPK2: MOV    #258.,$SSEC(RO) ;SECTOR SIZE FOR READ
1807 020664 112760 000005 000024      MOVB   #5,$CODE(RO)   ;CODE FOR READ HEADER & DATA
1808 020672 112760 000173 000002      MOVB   #RDHD,$COMND(RO) ;DRIVE CODE FOR OPERATION
1809 020700 000415                BR     WRTPK4         ;SET UP FOR EXIT
1810 020702 012760 000400 000022  WRTPK3: MOV    #256.,$SSEC(RO) ;SECTOR SIZE
1811 020710 112760 000002 000024      MOVB   #2,$CODE(RO)   ;CODE FOR WRDAT
1812 020716 112760 000161 000002      MOVB   #WRDAT,$COMND(RO) ;OP CODE
1813 020724 004737 020022                JSR    PC,GETPAT     ;GET PATTERN CODE
1814 020730 110560 000030                MOVB   R5,$PATT(RO)  ;PATTERN CODE
1815 020734 005060 000104  WRTPK4: CLR    $NEXT(RO)  ;CLEAR 'PARAMETER SELECTED' INDICATOR
1816 020740 000207                RTS    PC             ;RETURN
1817 020742 005037 001320  WRTPK5: CLR    PACK      ;CLEAR WT FLAG
1818 020746 105060 000026                CLRB   $PACK(RO)     ;CLEAR WT FLAG
1819 020752 005726                TST    (SP)+         ;CLEAR STACK LEVEL
1820 020754 000137 006100                JMP    MAIN           ;JUMP TO MAIN BACKGROUND LOOP
1821
1822 ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
1823 ; IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
1824 ;CALL:
1825 ;      JSR    PC,SPOTCK
1826 ;      RETURN1
1827 ;      RETURN2
1828 ;      ;ERROR AT AN ADDRESS IN TABLE
1829 ;      ;NO TABLE ENTRY FOR ERROR ADDRESS OR
1830 ;      ;PARAMETER 'NOTPRT' IS 0
1830 020760 010146  SPOTCK: MOV    R1,-(SP)      ;:PUSH R1 ON STACK
1831 020762 012701 000124      MOV    #SBDSEC,R1    ;:INCREMENT FOR BAD SECTOR TABLE
1832 020766 060001      ADD    R0,R1         ;:ADD THE BLOCK'S STARTING ADDRESS
1833 020770 004737 023176  1$:      JSR    PC,READDR     ;:DECREMENT THE SECTOR/TRACK ADDRESS
1834 020774 021126      CMP    (R1),(SP)+    ;:ON THE SAME CYLINDER ?
1835 020776 001023      BNE    5$            ;:BRANCH IF NOT
1836 021000 122761 177777 000003      CMPB   #-1,3(R1)     ;:ALL BAD TRACKS ?
1837 021006 001002      BNE    2$            ;:BR IF NO
1838 021010 005726      TST    (SP)+         ;:ADJUST STACK AND
1839 021012 000403      BR     3$            ;:GO CHECK SECTORS
1840 021014 122661 000003  2$:      CMPB   (SP)+,3(R1)   ;:COMPARE THE TRACK ADDRESS
1841 021020 001013      BNE    6$            ;:BR IF IT IS NOT EQUAL
1842 021022 122761 177777 000002  3$:      CMPB   #-1,2(R1)    ;:ALL BAD SECTORS ?
    
```

```

1843 021030 001002      BNE      4$      ;BR IF NO
1844 021032 005726      TST      (SP)+   ;ADJUST STACK AND
1845 021034 000413      BR       8$      ;CHECK 'NOTPRT'
1846 021036 122661 000002 4$:  CMPB    (SP)+,2(R1) ;COMPARE THE SECTOR ADDRESS
1847 021042 001003      BNE      7$      ;BR IF NOT EQUAL
1848 021044 000407      BR       8$      ;CHECK 'NOTPRT'
1849 021046 005726      5$:  TST      (SP)+   ;CLEAR OFF THE STACK
1850 021050 005726      6$:  TST      (SP)+   ;INCREMENT THE STACK POINTER
1851 021052 062701 000004 7$:  ADD     #4,R1    ;GO TO THE NEXT LOCATION IN THE TABLE
1852 021056 005711      TST      (R1)    ;EMPTY ENTRY OR TERMINATOR ?
1853 021060 100407      BMI     9$      ;BR IF YES
1854 021062 000742      BR       1$      ;TRY NEXT SECTOR
1855 021064 005737 001510 8$:  TST     NOTPRT  ;PRINT THE ERROR ANYWAY ?
1856 021070 001006      BNE     10$     ;BR IF NOT
1857 021072 012737 177777 001364 9$:  MOV     #-1,BADSEC ;SET THE INDICATOR FOR THE IDENTIFICATION LINE
1858 021100 062766 000002 000002 10$: ADD     #2,2(SP)  ;INCREMENT THE RETURN
1859 021106      10$:  MOV     (SP)+,R1  ;;POP STACK INTO R1
1860 021110 000207      RTS     PC      ;RETURN

```

```

1861
1862 ;:*****
1863
1864 .SBTTL  ERROR MESSAGE GENERATION ROUTINES
1865
1866 ;:*****
1867

```

```

1868 ;PRINT LINE 1 OF ERROR MESSAGE:
1869 ;'HH:MM:SS'
1870

```

```

1871 021112 032777 002000 160034 LINE1: BIT     #SW10,@SWR  ;SWITCH 10 SET ?
1872 021120 001402      BEQ     1$      ;BR IF NOT
1873 021122 104401 001176      TYPE    ,SBELL  ;RING THE BELL
1874 021126 032777 020000 160020 1$:  BIT     #SW13,@SWR  ;INHIBIT TYPEOUT ?
1875 021134 001405      BEQ     2$      ;BR IF NOT
1876 021136 104414 001203      DISPLY  ,$CRLF  ;CR-LF
1877 021142 104414 001203      DISPLY  ,$CRLF  ;CR-LF
1878 021146 000406      BR       3$      ;EXIT
1879 021150 104414 001203 2$:  DISPLY  ,$CRLF  ;CR-LF
1880 021154 004737 024364      JSR     PC,$TIME ;TYPE THE TIME
1881 021160 104414 073271      DISPLY  ,BLNKS1 ;TYPE 1 BLANK
1882 021164 000207 3$:  RTS     PC      ;RETURN & TYPE DESCRIPTION

```

```

1883
1884 ;PRINT LINE 2 OF ERROR MESSAGE
1885 ;'PRESENT ORDER = XXXX PREVIOUS ORDER = XXXX'
1886 ;'* ERROR AT BAD TRACK/SECTOR'
1887 ;'DRV RMCS1  RMCS2  RMDS1  RMER1  RMMR2  RMER2  RMEC1  RMEC2'
1888 ;'RMWC  RMBA  RMDA  RMAS  RMLA  RMDB  RMMR1  RMDT'
1889 ;'RMSN  RMOF  RMDC  RMCC  STATUS'
1890 ;'BUS ADDRESS OR WORD COUNT NOT CONSISTENT'
1891 ;'RMBA = XXXXXX  RMWC = XXXXXX'
1892 ;'BUFFER ADR = XXXXXX  SIZE = XXXX  ACTUAL NMBR WRDS XFRD = XXX'
1893

```

```

1894 021166      LINE2:  MOV     R3,-(SP)  ;;PUSH R3 ON STACK
      021166 010346      MOV     R4,-(SP)  ;;PUSH R4 ON STACK
      021170 010446      MOV     R5,-(SP)  ;;PUSH R5 ON STACK
1895 021174 104414 001203  DISPLY  ,$CRLF  ;CR-LF

```


1896	021200	005037	021326		CLR	4\$:CLEAR MESSAGE ADDRESS STORAGE
1897	021204	005004			CLR	R4	:WORKING REGISTER
1898	021206	012737	071374	021326	MOV	#LIN2C,4\$:ADDRESS OF 'PRESENT ORDER = ' MSG
1899	021214	116004	002116		MOVB	\$RMCS1(R0),R4	:GET THE OPCODE
1900	021220	042704	177701		BIC	#^C76,R4	:SAVE ONLY SIGNIFICANT BITS
1901	021224	004737	021262		JSR	PC,1\$:TYPE THE FIRST MNEMONIC
1902	021230	005737	021332		TST	5\$:SEE IF MNEMONIC ENTRY FOUND
1903	021234	001440			BEQ	LINE2A	:BR IF NOT
1904	021236	012737	071415	021326	MOV	#LIN2P,4\$:ADDRESS OF 'PREVIOUS ORDER = ' MSG
1905	021244	116004	000027		MOVB	\$PREVC(R0),R4	:PREVIOUS OPERATION CODE
1906	021250	042704	177701		BIC	#^C76,R4	:SAVE ONLY SIGNIFICANT BITS
1907	021254	004737	021262		JSR	PC,1\$:TYPE THE PREVIOUS MNEMONIC
1908	021260	000426			BR	LINE2A	:CONTINUE
1909	021262	005005		1\$:	CLR	R5	:CLEAR THE TABLE INDEX
1910	021264	126504	002134	2\$:	CMPB	OPTBL(R5),R4	:LOOK FOR THE OPCODE
1911	021270	001405			BEQ	3\$:BR WHEN OPCODE COUNT EQUALS OPCODE
1912	021272	105765	002134		TSTB	OPTBL(R5)	:LOOK FOR END OF TABLE
1913	021276	100402			BMI	3\$:BR IF END
1914	021300	005205			INC	R5	:INCREMENT THE POINTER
1915	021302	000770			BR	2\$:CONTINUE - NOT END OF TABLE
1916	021304	006305		3\$:	ASL	R5	:SHIFT INDEX
1917	021306	006305			ASL	R5	:SHIFT THE INDEX
1918	021310	006305			ASL	R5	:SHIFT THE INDEX
1919	021312	012737	002156	021332	MOV	#MNTBL,5\$:ADDRESS OF ASCII TEXT TABLE
1920	021320	060537	021332		ADD	R5,5\$:ADD THE INDEX
1921	021324	104414			DISPLY		:TYPE IT
1922	021326	000000		4\$:	.WORD	0	:ADDRESS OF 'PRESENT' OR 'PREVIOUS' MESSAGE
1923	021330	104414			DISPLY		:TYPE THE OPERATION MNEMONIC
1924	021332	000000		5\$:	.WORD	0	:ADDRESS OF MESSAGE
1925	021334	000207			RTS	PC	:RETURN TO MAIN ROUTINE
1926	021336	005737	001364	LINE2A:	TST	BADSEC	:PRINT THE BAD SECTOR LINE ?
1927	021342	001404			BEQ	LINE2B	:BR IF NOT
1928	021344	104414	001203		DISPLY	,\$CRLF	:CR-LF
1929	021350	104414	071441		DISPLY	,\$LIN2S	:ERROR ADDRESS DEFINED AS BAD AREA
1930	021354	104414	001203	LINE2B:	DISPLY	,\$CRLF	:CR-LF
1931	021360	104414	070770		DISPLY	,\$DH14	:STANDARD RM REGISTER HEADER
1932	021364	104414	073271		DISPLY	,\$BLNKS1	:TYPE 1 BLANK
1933	021370	013746	001346		MOV	UNIT,-(SP)	:PUT THE DRIVE NUMBER ON THE STACK
1934	021374	004737	023156		JSR	PC,LINDEC	:TYPE DRIVE NUMBER
1935	021400	104414	073270		DISPLY	,\$BLNKS2	:TYPE 2 BLANKS
1936	021404	012705	071314		MOV	#DT14,R5	:REGISTER INDEXES
1937	021410	004737	021540		JSR	PC,3\$:PRINT THE REGISTERS
1938	021414	032777	000040	157532	BIT	#SW05,@SWR	:PRINT THE OPTIONAL REGISTERS ?
1939	021422	001014			BNE	1\$:BR IF NOT
1940	021424	104414	071073		DISPLY	,\$DH15	
1941	021430	012705	071336		MOV	#DT15,R5	:SECOND DATA LINE
1942	021434	004737	021540		JSR	PC,3\$:PRINT THEM
1943	021440	104414	071172		DISPLY	,\$DH16	
1944	021444	012705	071360		MOV	#DT16,R5	:THIRD DATA LINE
1945	021450	004737	021540		JSR	PC,3\$:PRINT THE REGISTERS
1946	021454	032760	000100	000016	1\$:	BIT	#BIT6,\$STATUS(R0)
1947	021462	001422			BEQ	2\$:DATA ERROR ?
1948	021464	016046	000020		MOV	\$WRDL(R0),-(SP)	:BR IF NOT
1949	021470	066016	002120		ADD	\$RMWC(R0),(SP)	:TRANSFER SIZE
1950	021474	006316			ASL	(SP)	:ADD REMAINING WORD COUNT
1951	021476	066016	000006		ADD	\$BUF(R0),(SP)	:CONVERT TO AN BYTE INCREMENT
1952	021502	022660	002122		CMP	(SP)+,\$RMBA(R0)	:BUFFER STARTING ADDRESS
							:CORRECT BUFFER ADDRESS ?

```

1953 021506 001410          BEQ      2$          ;BR IF YES
1954 021510 104414 070415  DISPLY  ,EM46        ;'BUS ADDRESS AND WORD COUNT ARE NOT CONSISTENT'
1955 021514 104414 001203  DISPLY  ,$CRLF       ;CR-LF
1956 021520 004737 021632  JSR     PC,LIN3D     ;PRINT LINE 3D OF ERROR MESSAGE
1957 021524 004737 022244  JSR     PC,LIN4      ;PRINT LINE 4 OF ERROR MESSAGE
1958 021530          2$:  MOV     (SP)+,R5      ;:POP STACK INTO R5
      021530 012605          MOV     (SP)+,R4      ;:POP STACK INTO R4
      021532 012604          MOV     (SP)+,R3      ;:POP STACK INTO R3
      021534 012603          RTS     PC          ;RETURN TO ERROR PROCESSING ROUTINE
1959 021536 000207          3$:  MOV     (R5)+,-(SP)    ;PUT THE REGISTER INDEX ON THE STACK
1960 021540 012546          ADD     R0,(SP)      ;ADD DRIVE'S TABLE ADDRESS
1961 021542 060016          MOV     @ (SP),-(SP) ;VALUE
1962 021544 017646 000000  JSR     PC,LINOC     ;TYPE IT
1963 021550 004737 023124  TST     (SP)+       ;CORRECT THE STACK POINTER
1964 021554 005726          DISPLY  ,BLNKS2     ;TYPE 2 BLANKS
1965 021556 104414 073270  TST     (R5)        ;AT END OF LINE ?
1966 021562 005715          BNE     3$          ;BR IF NOT
1967 021564 001365          4$:  DISPLY  , $CRLF   ;CR-LF
1968 021566 104414 001203  RTS     PC          ;RETURN
1969 021572 000207
1970
1971          ;PRINT LINE 3 OF ERROR MESSAGE
1972          ;'ERROR AT CCC TT SS  PREVIOUS ADR = CCC TT SS'
1973
1974 021574 104414 071475  LINE3:  DISPLY  ,LINM3 ;LINE 3 ENTRANCE
1975 021600 000517          BR      LIN3.1      ;FINISH PRINTOUT
1976
1977          ;PRINT LINE 3A OF ERROR MESSAGE
1978          ;'START CYL = CCC  END CYL = CCC'
1979
1980 021602 104414 071513  LINE3A: DISPLY  ,LINM3 ;LINE 3A ENTRANCE
1981 021606 000514          BR      LIN3.1      ;FINISH ERROR LINE
1982
1983          ;PRINT LINE 3B OF ERROR MESSAGE
1984          ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC'
1985
1986 021610 004737 022146  LINE3B: JSR     PC,LIN3.3 ;LINE 3B ENTRANCE
1987 021614 104414 001203  DISPLY  , $CRLF
1988 021620 000207          RTS     PC
1989
1990          ;PRINT LINE 3C OF ERROR MESSAGE
1991          ;'START CYL = CCC  END CYL = CCC  ACTUAL CYL = CCC  TRK = TT'
1992
1993 021622 004737 022146  LINE3C: JSR     PC,LIN3.3 ;LINE 3C ENTRANCE
1994 021626 000137 022200  JMP     LIN3.4      ;FINISH MESSAGE
1995
1996          ;PRINT LINE 3D OF ERROR MESSAGE
1997          ;'RMBA = XXXXXX  RMWC = XXXXXX'
1998
1999 021632 032777 000040 157314 LINE3D: BIT     #SW05,@SWR ;SWITCH 5 SET ?
2000 021640 001416          BEQ     1$          ;BR IF IT IS
2001 021642 104414 071664  DISPLY  ,LINB3      ;'RMBA = '
2002 021646 016046 002122  MOV     $RMBA(R0),-(SP) ;BUFFER ADDR REG CONTENTS
2003 021652 004737 023124  JSR     PC,LINOC     ;CONVERT TO OCTAL AND TYPE IT
2004 021656 104414 071674  DISPLY  ,LINW3      ;' RMWC = '
2005 021662 016046 002120  MOV     $RMWC(R0),-(SP) ;WORD COUNT REGISTER CONTENTS
2006 021666 004737 023124  JSR     PC,LINOC     ;CONVERT TO OCTAL AND TYPE IT

```

2007 021672 104414 001203
 2008 021676 000207
 2009
 2010
 2011
 2012
 2013 021700 104414 071560
 2014 021704 016046 000012
 2015 021710 004737 023156
 2016 021714 104414 073270
 2017 021720 104414 071706
 2018 021724 005046
 2019 021725 116016 000011
 2020 021732 004737 023156
 2021 021736 104414 073270
 2022 021742 104414 071723
 2023 021746 005046
 2024 021750 116016 000010
 2025 021754 004737 023156
 2026 021760 104414 001203
 2027 021764 000207

```

1$:   DISPLY , $CRLF
      RTS      PC

:PRINT LINE 3E OF ERROR MESSAGE
:'START CYL = CCC   START TRK = TT   START SEC = SS'

LINE3E:  DISPLY , LINS3           ;'START CYL = '
          MOV    $CYL(RO), -(SP)  ;MOVE CYL TO STACK
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , BLNKS2         ;TYPE 2 BLANKS
          DISPLY , LINST3        ;'START TRK = '
          CL    -(SP)            ;CLEAR STACK
          MOVB  $TRK(RO), (SP)    ;TRACK TO STACK
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , BLNKS2         ;TYPE 2 BLANKS
          DISPLY , LINS3         ;'START SEC = '
          CLR   -(SP)            ;CLEAR STACK
          MOVB  $SEC(RO), (SP)    ;SECTOR ADDR TO STACK
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , $CRLF
          RTS      PC
    
```

2028
 2029
 2030
 2031

```

:PRINT LINE 3F OF ERROR MESSAGE
:'RMDA = XXXXXX   RMCA = XXXXXX'
    
```

2032 021766 032777 000040 157160
 2033 021774 001420
 2034 021776 104414 071654
 2035 022002 016046 002124
 2036 022006 004737 023124
 2037 022012 104414 073270
 2038 022016 104414 071643
 2039 022022 016046 002152
 2040 022026 004737 023124
 2041 022032 104414 001203
 2042 022036 000207

```

LINE3F:  BIT    #SW5, @SWR       ;SWITCH 5 SET ?
          BEQ   1$              ;BR IF NOT
          DISPLY , LINDA3        ;'RMDA = '
          MOV   $RMDA(RO), -(SP) ;PUT SECTOR/TRACK ADDRESS ON THE STACK
          JSR   PC, LINOCT       ;TYPE IT
          DISPLY , BLNKS2         ;TYPE 2 BLANKS
          DISPLY , LINCA3        ;' RMDC = '
          MOV   $RMDC(RO), -(SP) ;PUT DESIRED CYLINDER ADDRESS ON THE STACK
          JSR   PC, LINOCT       ;TYPE IT
          DISPLY , $CRLF
1$:      RTS      PC
    
```

2043
 2044
 2045

```

:'CCC TT SS   PREV ADR = CCC TT SS'
    
```

2049 022040 004737 023176
 2050 022044 004737 023156
 2051 022050 104414 071510
 2055 022054 004737 023156
 2056 022060 104414 071534
 2057 022064 004737 023156
 2058 022070 104414 071537
 2059 022074 016046 000034
 2060 022100 004737 023156
 2061 022104 104414 071510
 2062 022110 005046
 2063 022112 116016 000033
 2064 022116 004737 023156
 2065 022122 104414 071534
 2066 022126 005046
 2067 022130 116016 000032
 2068 022134 004737 023156
 2069 022140 104414 001203

```

LIN3.1:  JSR   PC, READDR        ;DECREMENT TRACK AND SECTOR ADDRESS
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , T              ;PRINT ' T '
          JSR   PC, LINDEC        ;TYPE TRACK IN DECIMAL
          DISPLY , S              ;PRINT ' S '
          JSR   PC, LINDEC        ;TYPE SECTOR ADDRESS
          DISPLY , LIMP3         ;PRINT 'PREV ADDR'
          MOV   $PREVA+2(RO), -(SP) ;PREVIOUS CYLINDER
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , T              ;PRINT ' T '
          CLR   -(SP)            ;MAKE ROOM ON THE STACK
          MOVB  $PREVA+1(RO), (SP) ;PREVIOUS TRACK ADDRESS
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , S              ;PRINT ' S '
          CLR   -(SP)            ;MAKE ROOM ON THE STACK
          MOVB  $PREVA(RO), (SP)  ;PREVIOUS SECTOR DDRESS
          JSR   PC, LINDEC        ;TYPE IT IN DECIMAL
          DISPLY , $CRLF
    
```

```

2070 022144 000207          RTS      PC
2071
2072          ;'START CYL = CCC  END CYL = CCC'
2073
2074 022146 104414 071560  LIN3.3: DISPLY  ,LINS3          ;LINE '3B & 3C' ENTRANCE
2075 022152 016046 000034          MOV      $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
2076 022156 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2077 022162 104414 071575          DISPLY  ,LINS3          ;PRINT 'END CYL'
2078 022166 016046 002152          MOV      $RMDC(RO),-(SP) ;PRESENT CYLINDER
2079 022172 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2080 022176 000207          RTS      PC
2081
2082          ;'ACTUAL CYL = CCC  TRK = TT'
2083
2084 022200 104414 071612  LIN3.4: DISPLY  ,LINA3          ;PRINT 'ACTUAL'
2085 022204 013746 076746          MOV      (CYLNR,-(SP)      ;ACTUAL CYLINDER
2086 022210 042716 010000          BIC      #BIT12,(SP)      ;CLEAR THE FORMAT BIT
2087 022214 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2088 022220 104414 071632          DISPLY  ,LINT3          ;PRINT TRACK
2089 022224 005046          CLR      -(SP)            ;CLEAR STACK WORD
2090 022226 116016 002125          MOV      $RMDA+1(RO),(SP) ;PUT TRACK ON STACK
2091 022232 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2092 022236 104414 001203          DISPLY  , $CRLF
2093 022242 000207          RTS      PC
2094
2095          ;PRINT LINE 4 OF ERROR MESSAGE
2096          ;'BUFFER ADR = XXXXXX  SIZE = XXXX  ACTUAL NMBR WRDS XFRD = XXX'
2097
2098 022244 032760 000100 000016 LINE4:  BIT      #BIT06,$STATUS(RO) ;DATA ERROR ?
2099 022252 001427          BEQ      1$              ;BR IF NOT
2100 022254 104414 071740          DISPLY  ,LINA4          ;'PRINT BUFFER'
2101 022260 016046 000006          MOV      $BUF(RO),-(SP)  ;BUFFER ADDR ON STACK
2102 022264 004737 023124          JSR      PC,LINOCT        ;CONVERT TO OCTAL & PRINT
2103 022270 104414 071757          DISPLY  ,LINS4          ;PRINT 'SIZE'
2104 022274 016046 000020          MOV      $WRDL(RO),-(SP) ;BUFFER SIZE
2105 022300 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2106 022304 104414 071771          DISPLY  ,LINA4          ;'ACTUAL NMBR WRDS XFRD = '
2107 022310 016046 002122          MOV      $RMB A(RO),-(SP) ;VALUE IN BUFFER ADDR REGISTER
2108 022314 166016 000006          SUB      $BUF(RO),(SP)   ;SUBTRACT STARTING ADDRESS
2109 022320 006216          ASR      (SP)            ;CONVERT INTO A WORD COUNT
2110 022322 004737 023156          JSR      PC,LINDEC          ;TYPE IT IN DECIMAL
2111 022326 104414 001203          DISPLY  , $CRLF          ;CR-LF
2112 022332 000207          1$:      RTS      PC          ;RETURN
2113
2114          ;PRINT LINE 5 OF ERROR MESSAGE
2115          ;'GOOD DATA = XXXXXX  BAD DATA = XXXXXX  SECT POS = XXX'
2116
2117 022334 104414 072024          LINE5: DISPLY  ,LIND5          ;PRINT 'GOOD DATA'
2118 022340 162760 000002 002122          SUB      #2,$RMB A(RO)   ;BACK THE ADDRESS UP
2119 022346 017046 002122          MOV      @ $RMB A(RO),-(SP) ;'GOOD' DATA - AT THE BUFFER LOCATION
2120 022352 004737 023124          JSR      PC,LINOCT        ;TYPE IT
2121 022356 104414 072041          DISPLY  ,LIND5          ;PRINT 'BAD DATA'
2122 022362 016046 002140          MOV      $RMD B(RO),-(SP) ;BAD DATA FROM BUFFER
2123 022366 004737 023124          JSR      PC,LINOCT        ;TYPE IT
2124 022372 016046 002120          MOV      $RMW C(RO),-(SP) ;WORD LENGTH ON STACK
2125 022376 066016 000020          ADD      $WRDL(RO),(SP)  ;MAKE INTO A POSITIVE NUMBER
2126 022402 005046          CLR      -(SP)          ;UPPER DIVIDEND TO ZERO
    
```

```

2127 022404 016046 000022      MOV    $SSEC(RO),-(SP)  ;SECTOR SIZE ON THE STACK
2128 022410 004737 030610      JSR    PC,LINKDV       ;DIVIDE WORDS XFERED BY SECTOR SIZE
2129 022414 012616              MOV    (SP)+(SP)       ;MOVE REMAINDER UP THE STACK
2130 022416 104414 072057      DISPLY ,LINP5         ;PRINT 'SECT POS'
2131 022422 004737 023156      JSR    PC,LINDEC      ;TYPE THE POSITION
2132 022426 104414 001203      DISPLY ,$CRLF
2133 022432 000207      RTS    PC
2134
2135      ;PRINT LINE 5A OF THE ERROR MESSAGE
2136      ;'HEADER FROM ERROR ,ECTOR XXXXXX XXXXXX XXXXXX XXXXXX'
2137
2138 022434
2139 022434 104414 072075      LINE5A:
2144 022440 013746 076746      2$:  DISPLY ,LINS5      ;'HEADER CONTENTS OF ERROR SECTOR'
      MOV    CYLNDR,-(SP)  ;HEADER POSITION
      JSR    PC,LINOCT     ;TYPE IT
      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
      MOV    CYLNDR+2,-(SP);HEADER POSITION +2
      JSR    PC,LINOCT     ;TYPE IT
      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
2145 022470 104414 073273      3$:  DISPLY ,LINX5      ;APPENDING INFO 1/23/77
2146 022474 104414 001203      DISPLY ,LINS5
2147 022500 000207      RTS    PC
2148
2149      ;PRINT LINE 5B OF ERROR MESSAGE
2150      ;'RMEC1 = XXXXXX RMEC2 = XXXXXX'
2151
2152 022502 104414 072131      LINE5B: DISPLY ,LINEP5  ;'RMEC1 = '
2153 022506 016046 002162      MOV    $RMEC1(RO),-(SP);PUT REGISTER CONTENTS ON THE STACK
2154 022512 004737 023124      JSR    PC,LINOCT     ;TYPE IT
2155 022516 104414 073270      DISPLY ,BLNKS2      ;TYPE 2 BLANKS
2156 022522 104414 072142      DISPLY ,LINE05      ;' RMEC2 = '
2157 022526 016046 002164      MOV    $RMEC2(RO),-(SP);PUT REGISTER CONTENTS ON THE STACK
2158 022532 004737 023124      JSR    PC,LINOCT     ;TYPE IT
2159 022536 104414 001203      DISPLY ,LINS5
2160 022542 000207      RTS    PC           ;RETURN
2161
2162      ;PRINT LINE 6 OF ERROR MESSAGE
2163      ;'SECTOR IS ECC CORRECTABLE'
2164
2165 022544 104414 072154      LINE6:  DISPLY ,LINB6      ;ECC CORRECTABLE
2166 022550 104414 001203      DISPLY ,LINS5
2167 022554 000207      RTS    PC
2168
2169      ;PRINT LINE 6A OF THE ERROR MESSAGE
2170      ;'SECTOR READ CORRECTLY AT OFFSET N'
2171
2172 022556 104414 072207      LINE6A: DISPLY ,LINC6      ;PRINT 'READ CORRECTLY AT OFFSET N'
2173 022562 000411              BR     LIN6.1         ;TYPE THE REST OF THE LINE
2174
2175      ;PRINT LINE 6B OF THE ERROR MESSAGE
2176      ;'SECTOR IS ECC CORRECTABLE AT OFFSET N'
2177
2178 022564 104414 072154      LINE6B: DISPLY ,LINB6      ;PRINT 'SECTOR IS ECC CORRECTABLE '
2179 022570 000406              BR     LIN6.1
2180
2181      ;PRINT LINE 6C OF THE ERROR MESSAGE
2182      ;'CORRECTED ON NTH RETRY'

```

```
2183
2184 022572 104414 072236 LINE6C: DISPLY ,LING6 ;'CORRECTED ON NTH RETRY'
2185 022576 000414 BR LIN6.2 ;TYPE THE REST OF THE LINE
2186
2187 ;PRINT LINE 6D OF THE ERROR MESSAGE
2188 ;'UNCORRECTABLE AFTER N RETRIES'
2189
2190 022600 104414 072265 LINE6D: DISPLY ,LINU06 ;'UNCORRECTABLE AFTER N RETRIES'
2191 022604 000411 BR LIN6.2 ;FINISH
2192
2193 ;TYPE THE OFFSET VALUE IN MICRO-INCHES
2194
2195 022606 006301 LIN6.1: ASL R1 ;DOUBLE THE OFFSET TABLE INDEX
2196 022610 016137 002474 022620 MOV OFMTBL(R1),1$ ;ADDRESS OF OFFSET POSITION MESSAGE
2197 022616 104414 DISPLY
2198 022620 000000 1$: .WORD 0 ;OFFSET VALUE
2199 022622 104414 001203 DISPLY , $CRLF
2200 022626 000207 RTS PC
2201
2202 ;RETRY COUNT TYPEOUT
2203
2204 022630 005046 LIN6.2: CLR -(SP) ;CLEAR STACK
2205 022632 113716 001353 MOVB RETRY+1,(SP) ;RETRY COUNT
2206 022636 004737 023156 JSR PC,LINDEC ;TYPE IT IN DECIMA
2207 022642 104414 072254 DISPLY ,LINR6 ;'RETRY'
2208 022646 104414 001203 DISPLY , $CRLF
2209 022652 000207 RTS PC
2210
2211 ;PRINT LINE 7 OF THE ERROR MESSAGE
2212 ;'ORDERS:XXXXX TOTAL ERRORS:XXX WRDS XFRD:XXXXXXXX WRDS READ:XXXXXXXX'
2213
2214 022654 104414 072340 LINE7: DISPLY ,LIN70 ;PRINT ORDER COUNT
2215 022660 012746 000036 MOV # $OPERC,-(SP) ;TO STACK
2216 022664 060016 ADD RO,(SP) ;ADD THE BASE ADDRESS
2217 022666 004737 035212 JSR PC,$DB2D ;CONVERT IT
2218 022672 004737 031124 JSR PC,$SUPRS ;PRINT IT
2219 022676 104414 072412 DISPLY ,LIN7T ;TOTAL ERRORS
2220 022702 016046 000056 MOV $TOTAL(RO),-(SP) ;TO STACK
2221 022706 004737 023156 JSR PC,LINDEC ;TYPE IT IN DECIMAL
2222 022712 104414 072424 DISPLY ,LIN7X ;PRINT 'WRDS XFR'
2223 022716 012746 000046 MOV # $TRANS,-(SP) ;ADDRESS OF LOW WORD ON STACK
2224 022722 060016 ADD RO,(SP)
2225 022724 004737 035212 JSR PC,$DB2D ;CONVERT
2226 022730 004737 031124 JSR PC,$SUPRS ;PRINT
2227 022734 104414 072440 DISPLY ,LIN7R ;'BITS READ'
2228 022740 012746 000052 MOV # $READ,-(SP) ;LOW WORD ADDRESS
2229 022744 060016 ADD RO,(SP)
2230 022746 004737 035212 JSR PC,$DB2D ;CONVERT
2231 022752 004737 031124 JSR PC,$SUPRS ;PRINT IT
2232 022756 104414 001203 DISPLY , $CRLF ;CR-LF
2233 022762 032777 100000 156164 BIT #SW15,@SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15
2234 022770 001401 BEQ 1$ ;BR IF NOT
2235 022772 000000 HALT ;SWITCH 15 HALT
2236 022774 000207 1$: RTS PC
2237
2238 ;PRINT LINE 7A OF ERROR MESSAGE
2239 ;'ORDERS:XXXXX TOTAL SEEKS=XXXXX TOTAL MISPOS ERR = XXX TOTAL SKI= XXX'
```

```

2240
2241 022776 104414 072340 LINE7A: DISPLY ,LIN70 ;'ORDERS = '
2242 023002 012746 000036 MOV #SOPERC,-(SP) ;ORDER COUNT INCREMENT
2243 023006 060016 ADD RO,(SP) ;ADD BASE ADDRESS
2244 023010 004737 035212 JSR PC,$DB2D ;CONVERT THE COUNT
2245 023014 004737 031124 JSR PC,$SUPRS ;PRINT IT
2246 023020 104414 072350 DISPLY ,LIN7P ;'TOTAL SEEKS = '
2247 023024 012746 000042 MOV #SPOSIT,-(SP) ;TOTAL SEEKS
2248 023030 060016 ADD RO,(SP) ;DEVICE TABLE ADDRESS;
2249 023032 004737 035212 JSR PC,$DB2D ;CONVERT THE SEEK COUNT
2250 023036 004737 031124 JSR PC,$SUPRS ;PRINT IT
2251 023042 104414 072312 DISPLY ,LIN7M ;' TOTAL MISPOS ERR = '
2252 023046 016046 000066 MOV $MISPO(RO),-(SP) ;TOTAL ERRORS
2253 023052 004737 023156 JSR PC,LINDEC ;TYPE IT IN DECIMAL
2254 023056 104414 072370 DISPLY ,LIN7S ;' TOTAL SKI,OCYL ERR = '
2255 023062 016046 000064 MOV $SKI(RO),-(SP) ;CONVERT & PRINT IT
2256 023066 004737 023156 JSR PC,LINDEC ;TYPE IT IN DECIMAL
2257 023072 104414 001203 DISPLY ,$CRLF ;CR-LF
2258 023076 032777 100000 156050 BIT #SW15,@SWR ;SEE IF HALT ON ERROR - SWITCH 15 SET
2259 023104 001401 BEQ 1$ ;BR IF NOT
2260 023106 000000 HALT ;SWITCH 15 HALT
2261 023110 000207 1$: RTS PC
2262
2263 ;PRINT LINE 8 OF THE ERROR MESSAGE
2264 ;'DIFFERENT ERROR DURING RETRY'
2265
2266 023112 104414 072455 LINE8: DISPLY ,LIN8M
2267 023116 004737 021166 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
2268 023122 000207 RTS PC
2269
2270 ;OCTAL TYPEOUT ROUTINE
2271 ;CALL:
2272 ;
2273 ; MOV NUM,-(SP) ;PUT THE NUMBER ON THE STACK
2274 ; JSR PC,LINOCT
2275 ; RETURN
2276 023124 016646 000000 LINOCT: MOV 2(SP),-(SP) ;PUT NUMBER IN PROPER LOCATION ON STACK
2277 023130 004737 031554 JSR PC,$SB2D ;CONVERT THE NUMBER TO OCTAL
2278 023134 012637 023150 MOV (SP)+,1$ ;GET THE ADDRESS OF THE ASCII STRING
2279 023140 062737 000005 023150 ADD #5.,1$ ;ADDRESS THE LAST 6 ASCII DIGITS
2280 023146 104414 DISPLY ;TYPE IT
2281 023150 000000 1$: .WORD 0 ;ADDRESS
2282 023152 012616 MOV (SP)+,(SP) ;CORRECT THE STACK
2283 023154 000207 RTS PC ;RETURN
2284
2285 ;ROUTINE TO CONVERT THE INPUT NUMBER TO DECIMAL AND TYPE IT WITH
2286 ;LEADING ZERO SUPPRESSION
2287 ;CALL:
2288 ;
2289 ; MOV NUM,-(SP) ;PUT THE NUMBER ON THE STACK
2290 ; JSR PC,LINDEC
2291 ; RETURN
2292 023156 016646 000002 LINDEC: MOV 2(SP),-(SP) ;SET UP STACK FOR CONVERT
2293 023162 004737 031524 JSR PC,$SP2D ;CONVERT IT TO DECIMAL
2294 023166 004737 031124 JSR PC,$SUPRS ;TYPE IT (WITH LEADING ZEROS SUPRESSED)
2295 023172 012616 MOV (SP)+,(SP) ;RESTORE STACK POINTER
2296 023174 000207 RTS PC

```

```

2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315 023176 162706 000006
2316 023202 016616 000006
2317 023206 005066 000006
2318 023212 005066 000004
2319 023216 116066 002124 000006
2320 023224 116066 002125 000004
2321 023232 016066 002152 000002
2322 023240 005766 000006
2323 023244 001403
2324 023246 105366 000006
2325 023252 000423
2326 023254 005766 000004
2327 023260 001406
2328 023262 112766 000037 000006
2329 023270 105366 000004
2330 023274 000412
2331 023276 112766 000037 000006
2332 023304 004737 026546
2333 023310 113766 001450 000004
2334 023316 005366 000002
2335 023322 000240
2336 023324 000207
2337
2338
2339
2340 023326 012737 177777 001312
2341 023334 012737 177777 001310
2342 023342 012737 023422 000004
2343 023350 005037 000006
2344 023354 005777 155716
2345 023360 005037 001312
2346 023364 005037 001310
2347 023370 013701 001302
2348 023374 012721 024462
2349 023400 012711 000300
2350 023404 012777 174575 155666
2351 023412 012777 000131 155656
2352 023420 000437
2353 023422 062706 000004

```

```

;*****
.SBTTL GENERAL SUPPORT SUBROUTINES
;*****
;DECREMENT THE SECTOR-TRACK ADDRESS
;CALL
;      MOV      #DPB,RO      ;DPB ADDRESS
;      JSR      C,READDR
;      RETURN
;ON RETURN THE STACK CONTAINS THE FOLLOWING:
;      4(SP) = SECTOR ADDRESS
;      2(SP) = TRACK ADDRESS
;      (SP) = CYLINDER ADDRESS
;
READDR: 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(RO),6(SP) ; SECTOR ON STACK
        MOVB     $RMDA+1(RO),4(SP) ;TRACK ADDRESS
        MOV      $RMDC(RO),2(SP) ;CYLINDER ADDRESS
        TST      6(SP)      ;SECTOR 0 ?
        BEQ      1$
        DECB     6(SP)      ;DECREMENT ONE SECTOR
        BR       4$
1$:     TST      4(SP)      ;ALSC ON TRACK 0 ?
        BEQ      2$
        MOVB     #31,6(SP)   ;LAST SECTOR
        DECB     4(SP)      ;DECREMENT ONE TRACK
        BR       4$
2$:     MOVB     #31,6(SP)   ;LAST SECTOR
        JSR      PC,GETLMT   ;GET ADDRESS LIMITS
        MOVB     TRKLMT,4(SP) ;GET LAST TRACK
        DEC      2(SP)      ;DECREMENT ONE CYLINDER COUNT
4$:     NOP
        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      #CKCLK1,ERRVEC ;SET UP VECTOR FOR CLOCK CHECK
        CLR      @#ERRVEC+2   ;NEW PSW
        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: ADD     #4,SP      ;RESTORE THE STACK POINTER

```



```

2354 023426 012737 023470 000004      MOV      #CKCLK2,@#ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
2355 023434 005777 155644                TST      @SLKS             ;LOOK FOR KW11-L
2356 023440 005037 001312                CLR      CLKFLG           ;SET CLOCK FLAG
2357 023444 013701 001306                MOV      $LLVEC,R1        ;KW11-L VECTOR ADDRESS
2358 023450 012721 024462                MOV      #CLOCK,(R1)+    ;SET UP KW11-L VECTOR
2359 023454 012711 000300                MOV      #300,(R1)       ;PSW - PRI 6
2360 023460 012777 000100 155616      MOV      #100,@SLKS      ;SET KW11-L INTERRUPT
2361 023466 000414                BR       CKCLK3
2362 023470 062706 000004      CKCLK2: ADD      #4,SP      ;RESTORE THE STACK POINTER
2363 023474 104401 074173                TYPE     ,NEDCLK         ;'P OR L CLOCK MUST BE ON SYSTEM'
2364 023500 005737 000042                TST      42              ;UNDER MONITOR CONTROL ?
2365 023504 001402                BEQ      1$              ;BR IF NOT
2366 023506 000137 030526                JMP      $GET42          ;ABORT PROGRAM
2367 023512 000000      1$:      HALT           ;HALT
2368 023514 000137 003626                JMP      START          ;TRY AGAIN
2369 023520 012737 000006 000004      CKCLK3: MOV      #6,@#ERRVEC ;RESTORE THE ERROR VECTOR
2370 023526 000207                RTS      PC
2371
2372                ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
2373                ;CALL.
2374                ;      JSR      PC,STATPR
2375                ;      RETURN
2376
2377 023530 010046      STATPR: MOV      R0,-(SP)      ;SAVE R0
2378 023532 010446                MOV      R4,-(SP)      ;SAVE R4
2379 023534 005737 001550                TST      ASNLST        ;ANY DRIVES ASSIGNED ?
2380 023540 001423                BEQ      3$            ;BR IF NOT
2381 023542 004737 023644                JSR      PC,SHDTYP     ;TYPE THE HEADING
2382 023546 005004                CLR      R4            ;CLEAR THE DRIVE INDEX
2383 023550 006304      1$:      ASL      R4            ;CHANGE TO INDEX WORDS
2384 023552 016400 002106                MOV      BLKADR(R4),R0 ;GET THE DRIVE'S BLOCK ADDRESS
2385 023556 006204                ASR      R4            ;RESTORE R4
2386 023560 136437 035750 001550      BITB    ATABIT(R4),ASNLST ;IS THIS DRIVE ASSIGNED ?
2387 023566 001404                BEQ      2$            ;BR IF NOT
2388 023570 004737 023666                JSR      PC,SDETAL     ;TYPE THE PERFORMANCE SUMMARY
2389 023574 104401 001203                TYPE     ,$CRLF        ;CR-LF
2390 023600 005204      2$:      INC      R4            ;INCREMENT THE INDEX
2391 023602 020427 000010                CMP      R4,#8         ;FINISHED ?
2392 023606 001360                BNE      1$            ;BR IF NO
2393 023610 012604      3$:      MOV      (SP)+,R4      ;RESTORE R4
2394 023612 012600                MOV      (SP)+,R0      ;RESTORE R0
2395 023614 000207                RTS      PC            ;RETURN
2396
2397                ;ROUTINE TO TYPE THE PERFORMANCE SUMMARY (STATISTICS) FOR AN INDIVIDUAL
2398                ;DRIVE.
2399                ;CALL:
2400                ;      MOV      #DPB,R0      ;DPB ADDRESS
2401                ;      JSR      PC,SUMARY
2402                ;      RETURN
2403
2404 023616 010046      SUMARY: MOV      R0,-(SP)      ;SAVE R0
2405 023620 010446                MOV      R4,-(SP)      ;SAVE R4
2406 023622 004737 023644                JSR      PC,SHDTYP     ;TYPE THE HEADING
2407 023626 005004                CLR      R4            ;CLEAR R4 FOR DRIVE NUMBER
2408 023630 111004                MOV      (R0),R4       ;DRIVE NUMBER
2409 023632 004737 023666                JSR      PC,SDETAL     ;TYPE THE STATISTICS
2410 023636 012604                MOV      (SP)+,R4      ;RESTORE R4

```

```

2411 023640 012600          MOV    (SP)+,R0      ;RESTORE R0
2412 023642 000207          RTS     PC           ;RETURN
2413
2414          ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
2415          ;CALL:
2416          ;      JSR     PC,SHDTYP
2417          ;      RETURN
2418
2419 023644 004737 024364    SHDTYP: JSR     PC,$TIME      ;TYPE THE TIME OF DAY
2420 023650 004537 031164    JSR     R5,TYPRI4      ;TYPE AT PRIORITY 4
2421 023654 001203          $CRLF          ;CR-LF
2422 023656 004537 031164    JSR     R5,TYPRI4      ;TYPE THE HEADER
2423 023662 073644          STATHD         ;HEADER
2424 023664 000207          RTS     PC           ;RETURN
2425
2426          ;TYPE THE PERFORMANCE SUMMARY DATE LINE
2427          ;CALL:
2428          ;      MOV    #DRIVE,R4      ;DRIVE NUMBER
2429          ;      MOV    #DPB,R0       ;DPB ADDRESS
2430          ;      RETURN
2431
2432 023666 010246          SDETAL: MOV    R2,-(SP)      ;SAVE R2
2433 023670 010002          MOV    R0,R2         ;DPB ADDRESS
2434 023672 062702 000036    ADD    #$OPERC,R2    ;FIRST STATISTICAL FIELD
2435 023676 010446          MOV    R4,-(SP)      ;SAVE R4 FOR TYPEOUT
                        ;      ;TYPE DRIVE NUMBER
                        ;      ;GO TYPE--OCTAL ASCII
                        ;      ;TYPE 2 DIGIT(S)
                        ;      ;SUPPRESS LEADING ZEROS
                        ;      ;TYPE 2 BLANKS
2436          023700 104403          TYPOS          ;TYPE 2 BLANKS
2437          023702 002          .BYTE 2          ;PUT THE PASS COUNT ON THE STACK
2438          023703 000          .BYTE 0          ;CONVERT IT
2439 023704 104401 073270    TYPE ,BLNKS2        ;TYPE 3 DIGITS
2440 023710 016046 000070    MOV    $PASSC(R0),-(SP) ;TYPE 2 BLANKS
2441 023714 004737 031524    JSR    PC,$SB2D      ;PUT $OPERC ON THE STACK
2442 023720 004537 031034    JSR    R5,REPLZ      ;CONVERT IT
2443 023724 000003          .WORD 3            ;TYPE $OPERC
2444 023726 104401 073270    TYPE ,BLNKS2        ;TYPE 6 DIGITS
2445 023732 010246          MOV    R2,-(SP)      ;TYPE 2 BLANKS
2446 023734 004737 035212    JSR    PC,$DB2D      ;PUT $POSIT ON THE STACK
2447 023740 004537 031034    JSR    R5,REPLZ      ;CONVERT IT
2448 023744 000006          .WORD 6            ;TYPE $POSIT
2449 023746 104401 073270    TYPE ,BLNKS2        ;TYPE 6 DIGITS
2450 023752 062702 000004    ADD    #4,R2         ;TYPE 2 BLANKS
2451 023756 010246          MOV    R2,-(SP)      ;INCREMENT R2
2452 023760 004737 035212    JSR    PC,$DB2D      ;PUT $STRANS ON THE STACK
2453 023764 004537 031034    JSR    R5,REPLZ      ;CONVERT $STRANS
2454 023770 000006          .WORD 6            ;TYPE IT
2455 023772 104401 073270    TYPE ,BLNKS2        ;TYPE 10 DIGITS
2456 023776 062702 000004    ADD    #4,R2         ;TYPE 2 BLANKS
2457 024002 010246          MOV    R2,-(SP)      ;INCREMENT R2
2458 024004 004737 035212    JSR    PC,$DB2D      ;PUT $READ ON THE STACK
2459 024010 004537 031034    JSR    R5,REPLZ      ;CONVERT $READ
2460 024014 000012          .WORD 10          ;TYPE IT
2461 024016 104401 073270    TYPE ,BLNKS2        ;TYPE 10 DIGITS
2462 024022 062702 000004    ADD    #4,R2         ;INCREMENT R2
2463 024026 010246          MOV    R2,-(SP)      ;PUT $READ ON THE STACK
2464 024030 004737 035212    JSR    PC,$DB2D      ;CONVERT $READ
2465 024034 004537 031034    JSR    R5,REPLZ      ;TYPE IT
2466 024040 000012          .WORD 10          ;TYPE 10 DIGITS

```

```

024042 104401 073271      TYPE      ,BLNKS1      ;TYPE 1 BLANK
024046 062702 000004      ADD      #4,R2        ;INCREMENT R2
2466 024052 062702 000002  ADD      #2,R2        ;INCREMENT R2 AGAIN
2473 024056 012246      MOV      (R2)+,-(SP)  ;PUT $SOFT ON THE STACK
024060 004737 031524      JSR      PC,$SB2D    ;CONVERT $SOFT
024064 004537 031034      JSR      R5,REPLZ    ;TYPEOUT $SOFT
024070 000004      .WORD    4          ;TYPE 4 DIGITS
024072 104401 073271      TYPE      ,BLNKS1      ;TYPE 1 BLANK
024076 012246      MOV      (R2)+,-(SP)  ;PUT $HARD ON THE STACK
024100 004737 031524      JSR      PC,$SB2D    ;CONVERT $HARD
024104 004537 031034      JSR      R5,REPLZ    ;TYPEOUT $HARD
024110 000004      .WORD    4          ;TYPE 4 DIGITS
024112 104401 073271      TYPE      ,BLNKS1      ;TYPE 1 BLANK
024116 012246      MOV      (R2)+,-(SP)  ;PUT $SKI ON THE STACK
024120 004737 031524      JSR      PC,$SB2D    ;CONVERT $SKI
024124 004537 031034      JSR      R5,REPLZ    ;TYPEOUT $SKI
024130 000004      .WORD    4          ;TYPE 4 DIGITS
024132 104401 073271      TYPE      ,BLNKS1      ;TYPE 1 BLANK
024136 012246      MOV      (R2)+,-(SP)  ;PUT $MISPO ON THE STACK
024140 004737 031524      JSR      PC,$SB2D    ;CONVERT $MISPO
024144 004537 031034      JSR      R5,REPLZ    ;TYPEOUT $MISPO
024150 000004      .WORD    4          ;TYPE 4 DIGITS
024152 104401 073271      TYPE      ,BLNKS1      ;TYPE 1 BLANK
2474 024156 016046 000056  MOV      $TOTAL(R0),-(SP) ;CALCULATE NUMBER OF OTHER ERRORS
2477 024162 166016 000060  SUB      $SOFT(R0),(SP) ;SUBTRACT $SOFT FROM $TOTAL
024166 166016 000062  SUB      $HARD(R0),(SP) ;SUBTRACT $HARD FROM $TOTAL
024172 166016 000064  SUB      $SKI(R0),(SP)  ;SUBTRACT $SKI FROM $TOTAL
024176 166016 000066  SUB      $MISPO(R0),(SP) ;SUBTRACT $MISPO FROM $TOTAL
2478 024202 004737 031524      JSR      PC,$SB2D    ;CONVERT 'OTHER' COUNT
2479 024206 004537 031034      JSR      R5,REPLZ    ;TYPE IT
2480 024212 000004      .WORD    4          ;TYPE 4 DIGITS
2481 024214 012602      MOV      (SP)+,R2    ;:POP STACK INTO R2
2482 024216 000207      RTS      PC
2483
2498
2499

```

```

;ROUTINE TO INCREMENT $SOFT
;
;NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)

```

```

024220 005737 001364      INCSOF: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
024224 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024226 026027 000060 023417      CMP      $SOFT(R0),#9999. ;IS $SOFT ALREADY AT MAXIMUM ?
024234 103002      BHS      1$              ;BR IF IT IS
024236 005260 000060      INC      $SOFT(R0)      ;INCREMENT $SOFT
024242 000207      RTS      PC              ;RETURN
1$:

```

2500

```

;ROUTINE TO INCREMENT $HARD
;
;NOTE: $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)

```

```

024244 005737 001364      INCHRD: TST      BADSEC      ;SEE IF BAD TRK/SEC INDICATOR SET
024250 001006      BNE      1$              ;BR IF IT'S SET, DON'T INCREMENT COUNT
024252 026027 000062 023417      CMP      $HARD(R0),#9999. ;IS $HARD ALREADY AT MAXIMUM ?
024260 103002      BHS      1$              ;BR IF IT IS
024262 005260 000062      INC      $HARD(R0)      ;INCREMENT $HARD
1$:

```

```

024266 000207          1$:   RTS   PC           ;RETURN

2501

;ROUTINE TO INCREMENT $SKI
;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)

024270 005737 001364   INCSKI: TST   BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024274 001006          BNE    1$                ;BR IF IT'S SET, DON'T INCREMENT COUNT
024276 026027 000064 023417   CMP    $SKI(RO),#9999.    ;IS $SKI ALREADY AT MAXIMUM?
024304 103002          BHS    1$                ;BR IF IT IS
024306 005260 000064          INC    $SKI(RO)          ;INCREMENT $SKI
024312 000207          1$:   RTS   PC           ;RETURN

2502

;ROUTINE TO INCREMENT $MISPO
;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)

024314 005737 001364   INCMIS: TST   BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024320 001006          BNE    1$                ;BR IF IT'S SET, DON'T INCREMENT COUNT
024322 026027 000066 023417   CMP    $MISPO(RO),#9999.  ;IS $MISPO ALREADY AT MAXIMUM?
024330 103002          BHS    1$                ;BR IF IT IS
024332 005260 000066          INC    $MISPO(RO)        ;INCREMENT $MISPO
024336 000207          1$:   RTS   PC           ;RETURN

2503

;ROUTINE TO INCREMENT $TOTAL
;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)

024340 005737 001364   INCTOT: TST   BADSEC          ;SEE IF BAD TRK/SEC INDICATOR SET
024344 001006          BNE    1$                ;BR IF IT'S SET, DON'T INCREMENT COUNT
024346 026027 000056 023417   CMP    $TOTAL(RO),#9999.  ;IS $TOTAL ALREADY AT MAXIMUM?
024354 103002          BHS    1$                ;BR IF IT IS
024356 005260 000056          INC    $TOTAL(RO)        ;INCREMENT $TOTAL
024362 000207          1$:   RTS   PC           ;RETURN

2504
2505
2506
;ROUTINE TO TYPE THE TIME
$TIME: TST   CLKFLG          ;CLOCK ON THE SYSTEM?
      BNE    1$                ;BR IF NOT
      TYPE  , $CRLF           ;CR-LF
      MOV   HOUR, -(SP)        ;PUT 'HOURS' ON THE STACK
      JSR  PC, $SB2D           ;CONVERT TO DECIMAL
      JSR  R5, REPLZ           ;TYPE IT
      .WORD 2                  ;TYPE 2 DIGITS
      TYPE  , COLON           ;':'
      MOV   MINUTE, -(SP)      ;PUT 'MINUTES' ON THE STACK
      JSR  PC, $SB2D           ;CONVERT TO DECIMAL
      JSR  R5, REPLZ           ;TYPE IT
      .WORD 2                  ;TYPE 2 DIGITS
      TYPE  , COLON           ;':'
      MOV   SECOND, -(SP)     ;PUT SECONDS ON THE STACK
      JSR  PC, $SB2D           ;CONVERT TO DECIMAL
      JSR  R5, REPLZ           ;TYPE IT

```

```

2523 024456 000002          .WORD 2          ;TYPE 2 DIGITS
2524 024460 000207          1$: RTS PC
2525
2526          .CLOCK HANDLER ROUTINE
2527
2528 024462 005337 001374    CLOCK: DEC SIXTEE          ;INCREMENT THE 1/60 SECOND COUNTER
2529 024466 001035          BNE 1$          ;BR IF A SECOND NOT COUNTED
2530 024470 013737 001314 001374    MOV HZ,SIXTEE      ;RESTORE THE VALUE
2531 024476 005237 001372          INC SECOND        ;COUNT THE SECOND
2532 024502 022737 000074 001372    CMP #60.,SECOND   ;AT MAXIMUM ?
2533 024510 001024          BNE 1$          ;BR IF NOT
2534 024512 005037 001372          CLR SECOND       ;CLEAR THE SECOND'S COUNTER
2535 024516 005237 001474          INC INTRVL+2     ;COUNT THE PERFORMANCE SUMMARY INTERVAL
2536 024522 005237 001370          INC MINUTE       ;COUNT THE MINUTE
2537 024526 022737 000074 001370    CMP #60.,MINUTE   ;AT MAXIMUM ?
2538 024534 001012          BNE 1$          ;BR IF NOT
2539 024536 005037 001370          CLR MINUTE       ;CLEAR THE MINUTE'S COUNTER
2540 024542 005237 001366          INC HOUR         ;COUNT THE HOURS
2541 024546 022737 001747 001366    CMP #999.,HOUR    ;AT MAXIMUM
2542 024554 103002          BHIS 1$         ;BR IF NOT
2543 024556 005037 001366          CLR HOUR        ;CLEAR THE HOURS
2544 024562 012746 000020          1$: MOV #20,-(SP)   ;1 MS ON THE STACK
2545 024566 004737 042310          JSR PC,RMTMR     ;DRIVER TIMER ROUTINE
2546 024572 005737 001472          TST INTRVL       ;DISPLAY THE PERFORMANCE SUMMARY ?
2547 024576 001411          BEQ 2$          ;BR IF NOT
2548 024600 023737 001472 001474    CMP INTRVL,INTRVL+2 ;DISPLAY INTERVAL FINISHED ?
2549 024606 001005          BNE 2$          ;BR IF NOT
2550 024610 012737 177777 001316    MOV #-1,STATIN   ;SET PERFORMANCE SUMMARY DISPLAY FLAG
2551 024616 005037 001474          CLR INTRVL+2     ;CLEAR THE PERFORMANCE INTERVAL COUNTER
2552 024622 000002          2$: RTI
2553
2554          ;COMMAND DECODE ROUTINE
2555          ;CALL:
2556          ;   MOV #-1,CFLAG          ;'CFLAG' IS NORMALLY SET BY THE TTY SERVICE
2557          ;   ;                     ;ROUTINE IN INTERRUPT MODE
2558          ;   JSR PC,KSR
2559          ;   RETURN1
2560          ;   RETURN2
2561          ;   ;                     ;SYSTEM BUSY RETURN
2562          ;   ;                     ;RETURN AFTER KEYBOARD SERVICED
2562 024624 005737 001544    KSR: TST ORDERQ+16 ;ANY OPERATIONS ACTIVE ?
2563 024630 001402          BEQ KSR1
2564 024632 104401 074567          TYPE ,BUSY
2565 024636 104412          KSR1: SAVREG
2566 024640 012737 000200 177776    MOV #PR4,PS      ;'SYSTEM BUSY...'
2567 024646 005037 001362          CLR CFLAG       ;SAVE THE REGISTERS
2568 024652 004737 024364          JSR PC,$TIME    ;SET PRIORITY TO 4
2569 024656 005777 154300          TST @STKB       ;CLEAR THE 'CONTROL C' FLAG
2570 024662 104401 074332          TYPE ,ENTCOM    ;TYPE THE TIME
2571          ;   ;                     ;CLEAR ANY GARBAGE IN THE TTY BUFFER
2572 024666 104411          RDLIN          ;'ENTER COMMAND'
2573 024670 012605          MOV (SP)+,R5    ;READ THE KEYBOARD
2574 024672 005737 001362          TST CFLAG       ;GET ADDRESS OF INPUT STRING
2575 024676 001122          BNE 11$        ;CHECK THE CONTROL C FLAG
2576 024700 005205          INC R5          ;EXIT IF 'CONTROL C' ENTERED
2577 024702 122715 000124          CMPB #'T,(R5)   ;POINT TO SECOND CHARACTER
2578 024706 001462          BEQ 7$         ;EQ TO A 'T' ?
2579 024710 122715 000101          CMPB #'A,(R5)   ;YES
;EQ TO AN 'A'

```

2580	024714	001410		BEQ	2\$:BR IF IT IS
2581	024716	121527	000067	CMPB	(R5),#7				:DRIVE NUMBER GREATER THAN AN ASCII 7 ?
2582	024722	101105		BHI	10\$:BR IF IT IS
2583	024724	121527	000060	CMPB	(R5),#0				:DRIVE NUMBER LESS THAN AN ASCII 0 ?
2584	024730	103502		BLO	10\$:BR IF IT IS
2585	024732	142715	177770	BICB	#^C7,(R5)				:LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'
2586	024736	122765	000124	2\$:	CMPB	#^T,-1(R5)			:EQ TO 'T'
2587	024744	001003		BNE	3\$:BR IF NOT EQ
2588	024746	004737	025574	JSR	PC,NEWASN				:ASSIGN DRIVE FOR TEST
2589	024752	000474		BR	11\$:EXIT
2590	024754	122765	000104	3\$:	CMPB	#^D,-1(R5)			:EQ TO 'D' ?
2591	024762	001003		BNE	4\$:BR IF NOT EQ
2592	024764	004737	025604	JSR	PC,DEASGN				:DEASSIGN DRIVE
2593	024770	000465		BR	11\$:EXIT
2594	024772	122765	000123	4\$:	CMPB	#^S,-1(R5)			:EQ TO 'S'
2595	025000	001003		BNE	5\$:BR IF NOT EQ
2596	025002	004737	025704	JSR	PC,SCMND				:TYPE STATISTICS
2597	025006	000456		BR	11\$:EXIT
2598	025010	122765	000127	5\$:	CMPB	#^W,-1(R5)			:EQ TO 'W'
2599	025016	001007		BNE	6\$:BR IF NOT EQ
2600	025020	032777	000001	154126	BIT	#SW0,@SWR			:IS SWITCH 0 SET ?
2601	025026	001040		BNE	9\$:BR IF SET, CAN'T DO 'W' COMMAND
2602	025030	004737	026132	JSR	PC,DATAPK				:WRITE A DATA PACK
2603	025034	000443		BR	11\$:EXIT
2604	025036	122765	000122	6\$:	CMPB	#^R,-1(R5)			:EQ TO 'R' ?
2605	025044	001034		BNE	10\$:BR IF NOT EQ
2606	025046	004737	026162	JSR	PC,REDAPK				:READ A DATA PACK
2607	025052	000434		BR	11\$:EXIT
2608	025054	122765	000127	7\$:	CMPB	#^W,-1(R5)			:WT COMMAND ?
2609	025062	001025		BNE	10\$:NO
2610	025064	122765	000101	000001	CMPB	#^A,1(R5)			:ALL DRIVES ?
2611	025072	001413		BEQ	8\$:YES
2612	025074	126527	000001	000067	CMPB	1(R5),#7			:GREAT THAN 7
2613	025102	101015		BHI	10\$:YES
2614	025104	126527	000001	000060	CMPB	1(R5),#0			:LESS THAN 0
2615	025112	103411		BLO	10\$:YES
2616	025114	142765	177770	000001	BICB	#^C7,1(R5)			:CHOP OFF THE HIGHER BITS
2617	025122	004737	026144	8\$:	JSR	PC,WATPAK			:ASSIGN DRIVES WITH WT COMMAND
2618	025126	000406		BR	11\$				
2619	025130	104401	074252	9\$:	TYPE	,MSWRO			:TYPE 'CAN'T WRITE WITH SW00 SET'
2620	025134	000644		BR	1\$:TRY AGAIN
2621	025136	104401	074307	10\$:	TYPE	,INVLD			:TYPE 'INVALID COMMAND' MESSAGE
2622	025142	000641		BR	1\$:TRY AGAIN
2623	025144	104413		11\$:	RESREG				:RESTORE R0 - R5
2624	025146	005777	154010		TST	@\$TKB			:CLEAR THE TTY BUFFER
2625	025152	052777	000100	154000	BIS	#BIT06,@\$TKS			:SET TTY INTERRUPT ENABLE
2626	025160	005037	177776		CLR	PS			:SET PRIORITY BACK TO ZERO
2627	025164	000207			RTS	PC			:RETURN
2628									
2629									
2630									:ROUTINE TO PROCESS THE ASSIGN REQUEST ('T', 'R', OR 'W' COMMANDS)
2631	025166	122715	000101	ASSIGN:	CMPB	#^A,(R5)			:ASSIGN ALL DRIVES?
2632	025172	001432			BEQ	ASGN2			:BR IF ALL DRIVES
2633	025174	111504		ASGN1:	MOVB	(R5),R4			:PUT DRIVE # IN R4
2634	025176	012737	073402	030052	MOV	#UNTASN,ASNMSG			: 'DRIVE ASSIGNED' MESSAGE ADDRESS
2635	025204	005737	001452		TST	XXDP			:LOADED FROM THIS DEVICE ?
2636	025210	001412			BEQ	1\$:BR IF NO

```

2637 025212 123704 001452          CMPB   XXDP,R4          ;LOADED FROM THIS DRIVE ?
2638 025216 001007          BNE    1$              ;BR IF NO
2639 025220 146437 035750 001550    BICB   ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
2640 025226 012737 073515 030052    MOV    #LODEV,ASNMSG   ;'DRIVE IS LOAD DEVICE' MESSAGE ADDRESS
2641 025234 000407          BR     2$              ;
2642 025236 136437 035750 001550 1$:   BITB   ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
2643 025244 001003          BNE    2$              ;BR IF IT IS
2644 025246 004737 025356          JSR    PC,ASGN3        ;SEE IF DRIVE ON THE SYSTEM
2645 025252 000207          RTS    PC              ;RETURN
2646 025254 000137 030026          2$:   JMP    ASNERR        ;EXIT ERROR
2647
2648 025260 005004          ASGN2: CLR    R4          ;START WITH DRIVE 0
2649 025262 012737 073402 030052 1$:   MOV    #UNTASN,ASNMSG  ;ERROR MESSAGE
2650 025270 005737 001452          TST    XXDP           ;LOADED FROM THIS DEVICE ?
2651 025274 001412          BEQ    2$              ;BR IF NO
2652 025276 123704 001452          CMPB   XXDP,R4          ;LOADED FROM THIS DRIVE ?
2653 025302 001007          BNE    2$              ;BR IF NO
2654 025304 146437 035750 001550    BICB   ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
2655 025312 012737 073515 030052    MOV    #LODEV,ASNMSG   ;'DRIVE IS LOAD DEVICE' MESSAGE ADDRESS
2656 025320 000413          BR     4$              ;
2657 025322 136437 035750 001550 2$:   BITB   ATABIT(R4),ASNLST ;ALREADY ASSIGNED ?
2658 025330 001007          BNE    4$              ;YES
2659 025332 004737 025356          JSR    PC,ASGN3        ;ASSIGN THE DRIVE
2660 025336 005204          3$:   INC    R4            ;INCREMENT DRIVE #
2661 025340 020427 000007          CMP    R4,#7          ;ALL DRIVE CHECKED ?
2662 025344 003746          BLE    1$              ;NO
2663 025346 000207          RTS    PC              ;YES
2664 025350 004737 030026          4$:   JSR    PC,ASNERR      ;ERROR MESSAGE
2665 025354 000770          BR     3$              ;TO LOOP
2666
2667 025356 136437 035750 001550 ASGN3: BITB   ATABIT(R4),ASNLST ;DRIVE ALREADY ASSIGNED ?
2668 025364 001054          BNE    ASGN4          ;BR IF IT IS
2673 025366 110437 066070          MOVB   R4,GENDPB      ;GET DRIVE NUMBER
2674 025372 006304          ASL    R4            ;MAKE R4 WORD INDEX
2675 025374 016400 002106          MOV    BLKADR(R4),R0  ;PUT BLOCK'S ADDR INTO R0
2676 025400 004737 015652          JSR    PC,RECALO      ;RECALIBRATE DRIVE
2677 025404 006204          ASR    R4            ;MAKE R4 BYTE INDEX
2678 025406 105764 035634          TSTB   DRVSTA(R4)     ;DRIVE AVAILABLE?
2679 025412 001447          BEQ    ASGN7          ;BR IF DRIVE OFFLINE OR NONEXISTENT
2680 025414 100441          BMI    ASGN6          ;BR IF DRIVE UNSAFE
2684 025416 004737 026174          JSR    PC,CLRDPB     ;CLEAR BLOCK FOR DRIVE JUST ASSIGNED
2685 025422 104401 074136          TYPE   ,DRNUM        ;TYPE DRIVE MESSAGE
2686 025425 010446          MOV    R4,-(SP)      ;SAVE R4 FOR TYPEOUT
      025430 104403          TYPOS  ;GO TYPE--OCTAL ASCII
      025432 002          .BYTE 2             ;TYPE 2 DIGIT(S)
      025433 000          .BYTE 0             ;SUPPRESS LEADING ZEROS
2687 025434 104401 001203          TYPE   ,$CRLF        ;CR-LF
2688 025440 006304          ASL    R4            ;MAKE R4 WORD INDEX
2689 025442 004737 026604          JSR    PC,GETID       ;GET DRIVE I.D.
2690 025446 004737 026374          JSR    PC,DRVPRM     ;GET THE DRIVE'S ADDRESS LIMITS
2691 025452 004537 026700          JSR    R5,GETADR     ;RETRIEVE BAD SPOT FILE
2692 025456 000416          BR     2$              ;UNSUCCESSFUL !
2693 025460 004737 027226          JSR    PC,MANTER     ;MANUALLY ENTER BAD SECTOR INFORMATION
2694 025464 016464 002106 001574    MOV    BLKADR(R4),NEWUNT(R4) ;DPB ADDRESS
2695 025472 012760 000001 000070    MOV    #1,$PASSC(R0) ;PRESET PASS COUNT TO 1
2696 025500 005737 001320          TST    PACK          ;WRITE DATA PACK ?
2697 025504 001403          BEQ    2$              ;BR IF NOT

```

```

2698 025506 113760 001320 000026      MOVB   PACK,$PACK(R0)  ;SET READ/WRITE DATA PACK INDICATOR
2699 025514 006204                2$:   ASR   R4          ;MAKE R4 BYTE INDEX
2700 025516 000207                ASGN4: RTS   PC         ;RETURN
2701
2702 025520 012737 073505 030052  ASGN6: MOV   #NOTSAF,ASNMSG ;'UNSAFE' MESSAGE ADDRESS
2703 025526 000137 030026                JMP   ASNERR           ;TO ERROR ROUTINE
2704
2705 025532 105764 035644                ASGN7: TSTB  DRVYP(R4)   ;DRIVE PRESENT?
2706 025536 001405                BEQ   1$              ;BR IF NOT
2707 025540 100010                BPL   2$              ;BR IF DRIVE OFFLINE
2708 025542 012737 073430 030052      MOV   #NOTRM,ASNMSG   ;ADDRESS OF 'NOT RM05/3/2' MSG
2709 025550 000407                BR    3$              ;EXIT
2710 025552 012737 073451 030052  1$:   MOV   #NOTPRS,ASNMSG ;ADDRESS OF 'NOT PRESENT' MSG
2711 025560 000403                BR    3$              ;EXIT
2712 025562 012737 073337 030052  2$:   MOV   #UNTOFF,ASNMSG ;ADDRESS OF 'DRIVE OFFLINE' MESSAGE
2716 025570 000137 030026                3$:   JMP   ASNERR           ;TO ERROR ROUTINE
2717
2718                                ;'T' COMMAND (ROUTINE TO ASSIGN A DRIVE)
2719
2720 025574 005037 001320      NEWASN: CLR   PACK      ;CLEAR 'W' COMMAND INDICATOR
2721 025600 000137 025166                JMP   ASSIGN          ;GO TO THE ASSIGN ROUTINE
2722
2723                                ;'D' COMMAND (ROUTINE TO DEASSIGN A DRIVE)
2724
2725 025604 005004      DEASGN: CLR   R4        ;START WITH DRIVE 0
2726 025606 012703      MOV   #8,R3         ;COUNTER
2727 025612 122715      CMPB  #'A',(R5)     ;DEASSIGN ALL DRIVES ?
2728 025616 001403      BEQ   1$           ;BR IF YES
2729 025620 111504      MOVB  (R5),R4      ;GET DRIVE NUMBER
2730 025622 012703      MOV   #1,R3        ;SET R3 FOR ONE UNIT
2731 025626 136437 035750 001550  1$:   BITB  ATABIT(R4),ASNLST ;DRIVE ASSIGNED ?
2732 025634 001414      BEQ   3$           ;BR IF NOT
2733 025636 146437 035750 001550      BICB  ATABIT(R4),ASNLST ;DELETE THE DRIVE FROM THE ASSIGNED LIST
2734 025644 006304      ASL   R4           ;MAKE ADDR INTO A WORD INDEX
2735 025646 016464 002106 001552      MOV   BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
2736 025654 006204      ASR   R4
2737 025656 005303      2$:   DEC   R3        ;ANY MORE DRIVES ?
2738 025660 001410      BEQ   4$           ;BR IF NOT
2739 025662 005204      INC   R4
2740 025664 000760      BR    1$
2741 025666 012737 073360 030052  3$:   MOV   #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
2742 025674 004737 030026      JSR   PC,ASNERR    ;REPORT IT
2743 025700 000766      BR    2$
2744 025702 000207      4$:   RTS   PC
2745
2746                                ;'S' COMMAND (ROUTINE TO TYPE DRIVE PERFORMANCE SUMMARY)
2747
2748 025704 005004      SCMND: CLR   R4
2749 025706 012703      MOV   #8,R3        ;COUNTER
2750 025712 122715      CMPB  #'A',(R5)     ;ALL STATISTICS ?
2751 025716 001421      BEQ   2$           ;BR IF YES
2752 025720 111504      MOVB  (R5),R4      ;GET DRIVE NUMBER
2753 025722 136437 035750 001550      BITB  ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
2754 025730 001406      BEQ   1$           ;BR IF NOT
2755 025732 006304      ASL   R4           ;MAKE DRIVE ADDR INTO WORD INDEX
2756 025734 016400 002106      MOV   BLKADR(R4),R0 ;ADDR OF BLOCK
2757 025740 004737 023616      JSR   PC,SUMARY    ;TYPE DRIVE STATISTICS SUMMARY

```



```

2758 025744 000471          BR      10$          ;EXIT
2759
2760 025746 012737 073360 030052 1$:  MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
2761 025754 004737 030026          JSR      PC,ASNERR   ;TYPE ERROR MESSAGE
2762 025760 000463          BR      10$          ;EXIT
2763 025762 105737 001550          2$:  TSTB    ASNLST    ;ANY DRIVE ASSIGNED ?
2764 025766 001001          BNE     3$          ;YES
2765 025770 000457          BR      10$          ;RETURN
2766 025772 004737 023530          3$:  JSR      PC,STATPR ;TYPE ALL STATISTICS
2767 025776 105737 001322          TSTB    DATE       ;SEE IF 'DATE' ENTERED
2768 026002 001404          BEQ     4$          ;BR IF NOT
2769 026004 104401 074466          TYPE    ,DATEIS    ;'DATE: '
2770 026010 104401 001322          TYPE    ,DATE      ;THE OPERATOR ENTERED DATE
2771 026014 105737 001334          4$:  TSTB    OPERID    ;SEE IF OPERATOR I.D. ENTERED
2772 026020 001404          BEQ     5$          ;BR IF NOT
2773 026022 104401 074476          TYPE    ,IDIS      ;'OPERATOR I.D.: '
2774 026026 104401 001334          TYPE    ,OPERID    ;THE OPERATOR I.D.
2775 026032 104401 074517          5$:  TYPE    ,HEDLIN   ;HEADER LINE
2776 026036 012737 046316 026104  MOV      #DRIVE0+$DRVID,8$ ;DRIVE I.D. FIELD ADDRESS
2777 026044 136437 035750 001550  6$:  BITB    ATABIT(R4),ASNLST ;SEE IF DRIVE ASSIGNED
2778 026052 001417          BEQ     9$          ;BR IF NOT ASSIGNED
2779 026054 010446          MOV     R4,-(SP)   ;SAVE R4 FOR TYPEOUT
                    ;TYPE DRIVE NUMBER
                    ;GO TYPE--OCTAL ASCII
                    ;TYPE 2 DIGIT(S)
                    ;SUPPRESS LEADING ZEROS
                    ;TYPE 4 BLANKS
026056 104403          TYPOS   ;TYPE 'NONE'
026060 002          .BYTE  2          ;CONTINUE
026061 000          .BYTE  0          ;TYPE THE DRIVE !.D.
2780 026062 104401 073266          TYPE    ,BLNKS4   ;ADDRESS OF DRIVE I.D. FIELD HERE
2781 026066 105777 000012          TSTB    @8$       ;CR-LF
2782 026072 001003          BNE     7$          ;DECREMENT THE COUNTER
2783 026074 104401 074540          TYPE    ,NONE     ;BR IF AT END
2784 026100 000404          BR      9$          ;INCREMENT THE MESSAGE FIELD ADDRESS
2785 026102 104401          7$:  TYPE    !.D.     ;INCREMENT DRIVE ADDRESS
2786 026104 000000          8$:  .WORD  0          ;CONTINUE
2787 026106 104401 001203          TYPE    ,$CRLF    ;CONTINUE
2788 026112 005303          9$:  DEC     R3       ;CONTINUE
2789 026114 003405          BLE     10$        ;CONTINUE
2790 026116 062737 002166 026104  ADD     #SRMEC2+2,8$ ;CONTINUE
2791 026124 005204          INC     R4       ;CONTINUE
2792 026126 000746          BR      6$          ;CONTINUE
2793 026130 000207          10$:  RTS     PC      ;CONTINUE
2794
2795          ;'W' COMMAND (ROUTINE TO WRITE A DATA PACK)
2796
2797 026132 012737 177777 001320  DATAPK: MOV     #-1,PACK ;SET THE 'W' COMMAND INDICATOR
2798 026140 000137 025166          JMP     ASSIGN    ;ASSIGN REQUESTED DRIVE
2799
2800          ;'WT' COMMAND (TO WRITE A PACK AND FOLLOWED BY TEST PACK)
2801
2802 026144 116515 000001          WATPAK: MOVB    1(R5),(R5) ;ADJUST DRIVE NUMBER ADDRESS
2803 026150 012737 177776 001320  MOV     #-2,PACK ;PACK WRITE COMMAND
2804 026156 000137 025166          JMP     ASSIGN    ;JUMP TO ASSIGN ROUTINE
2805
2806          ;'R' COMMAND (ROUTINE TO READ A DATA PACK)
2807
2808 026162 012737 000001 001320  REDAPK: MOV     #1,PACK ;SET THE 'READ' INDICATOR
2809 026170 000137 025166          JMP     ASSIGN    ;ASSIGN THE REQUESTED DRIVE
2810

```

```

2811 ;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED DRIVE
2812 ;CALL:
2813 ;       MOV     #DPB,R0           ;DPB ADDRESS
2814 ;       JSR     PC,CLRDPB
2815 ;       RETURN
2816 ;
2817 ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
2818
2819 CLRDPB:
026174 MOV     R1,-(SP)           ;;PUSH R1 ON STACK
026176 MOV     R3,-(SP)           ;;PUSH R3 ON STACK
026200 MOV     R4,-(SP)           ;;PUSH R4 ON STACK
026202 MOV     R5,-(SP)           ;;PUSH R5 ON STACK
2820 026204 MOV     R0,R4           ;GET THE DPB ADDRESS
2821 026206 ADD     #2,R4           ;ADDRESS OF FIRST LOCN TO BE CLEARED
2822 026212 MOV     #5,R3           ;NUMBER OF LOCNS TO BE CLEARED
2823 026216 1$: CLR     (R4)+       ;CLEAR THE LOCATION
2824 026220 DEC     R3           ;DECREMENT THE COUNTER
2825 026222 BNE     1$           ;BR IF NOT FINISHED
2826 026224 ADD     #2,R4           ;MOVE THE ADDRESS PAST THE 'REG' ADDR
2827 026230 MOV     #SNEXT-$REG,R3 ;NUMBER OF LOCNS TO BE CLEARED
2828 026234 2$: CLR     (R4)+       ;CLEAR
2829 026236 SUB     #2,R3           ;DECREMENT THE LOCN COUNTER
2830 026242 BNE     2$           ;BR IF NOT FINISHED
2831 026244 ADD     #12,R4          ;MOVE PAST ADDRESS LIMITS
2832 026250 MOV     #SRMEC2-MINSEC,R3 ;NUMBER OF LOCNS TO BE CLEARED
2833 026254 3$: CLR     (R4)+       ;CLEAR A LOCATION
2834 026256 SUB     #2,R3           ;DECREMENT THE COUNTER
2835 026262 BNE     3$           ;BR IF NOT DONE
2836 026264 MOV     BECOD,$CODE(R0) ;INITIAL COMMAND CODE
2837 026272 MOV     BECOD,R1           ;GET THE ACTUAL OP CODE
2838 026276 MOV     COMTBL(R1),$COMND(R0) ;OPERATION CODE
2839 026304 MOV     BEGPAT,$PATT(R0) ;PATTERN CODE
2840 026312 ASLB    $PATT(R0)       ;CONVERT CODE TO A TABLE INDEX
2841 026316 MOV     BEGSIZ,$WRDL(R0) ;BEGINNING RECORD SIZE
2842 026324 MOV     BEGSIZ,$WRDM(R0) ;VALUE FOR DATA TRANSFER
2843 026332 NEG     $WRDM(R0)       ;MAKE IT INTO 2'S COMPLEMENT
2844 026336 MOV     #256,$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
2845 026344 BITB   #1,$CODE(R0)   ;HEADER ORDER ?
2846 026352 BEQ     4$           ;BR IF NOT
2847 026354 ADD     #2,$SSEC(R0) ;ADD HEADER SIZE TO SECTOR SIZE
2848 026362 4$: MOV     (SP)+,R5     ;;POP STACK INTO R5
026364 MOV     (SP)+,R4     ;;POP STACK INTO R4
026366 MOV     (SP)+,R3     ;;POP STACK INTO R3
026370 MOV     (SP)+,R1     ;;POP STACK INTO R1
2849 026372 RTS     PC           ;RETURN
2850
2851 ;ROUTINE TO GET ADDRESS LIMITS FROM THE OPERATOR
2852 ;CALL
2853 ;       MOV     #DPB,R0           ;DPB ADDRESS
2854 ;       JSR     PC,DRVPRM        ;CALL ROUTINE
2855 ;
2856 ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
2857
2858 DRVPRM: MOV     R3,-(SP)       ;SAVE R3
2859 026376 MOV     R4,-(SP)       ;SAVE R4

```

```

2860 026400 005737 000042      TST      @#42      ;RUNNING UNDER MONITOR CONTROL
2861 026404 001002              BNE      1$      ;BR IF YES
2862 026406 104401 074377      TYPE     ,ENTLMT  ;'ENTER ADDRESSES'
2863
2864 026412 062760 177777 000122 1$:  ADD     #-1,$+IRST(RO) ;SEE IF FIRST TIME STARTED
2865 026420 103421              BCS      3$      ;BR IF NOT
2866 026422 013760 001444 000106  MOV     CYLIMT,MAXCYL(RO) ;LOAD MAXIMUM CYLINDER
2867 026430 005060 000110  CLR     MINCYL(RO)      ;CLEAR MINIMUM CYLINDER
2868 026434 004737 026546  JSR     PC,GETLMT      ;GET ADDRESS LIMITS
2869 026440 013760 001450 000112  MOV     TRKLMT,MAXTRK(RO) ;LOAD MAXIMUM TRACK
2870 026446 005060 000114  CLR     MINTRK(RO)      ;CLEAR MINIMUM TRACK
2871 026452 013760 001446 000116  MOV     SECLMT,MAXSEC(RO) ;LOAD MAXIMUM SECTOR
2872 026460 005060 000120  CLR     MINSEC(RO)      ;CLEAR MINIMUM SECTOR
2873 026464 016403 075460          3$:  MOV     TABLE(R4),R3    ;PARAMETER TABLE ADDRESS
2874 026470 013763 001450 000016  MOV     TRKLMT,16(R3)    ;LOAD TRACK LIMIT FOR LAST TRACK
2875 026476 013763 001450 000024  MOV     TRKLMT,24(R3)   ;LOAD TRACK LIMIT FOR STARTING TRACK
2876 026504 005737 000042      TST      @#42      ;UNDER MONITOR CONTROL ?
2877 026510 001002              BNE      4$      ;BR IF YES
2878 026512 004737 027702  JSR     PC,PARENT      ;GET THE DRIVE'S PARAMETERS
2879 026516 116060 000120 000010 4$:  MOVVB  MINSEC(RO),$SEC(RO) ;INITIAL SECTOR VALUE
2880 026524 116060 000114 000011  MOVVB  MINTRK(RO),$TRK(RO) ;INITIAL TRACK VALUE
2881 026532 016060 000110 000012  MOV     MINCYL(RO),$CYL(RO) ;INITIAL CYLINDER VALUE
2882 026540 012604              MOV     (SP)+,R4      ;RESTORE R4
2883 026542 012603              MOV     (SP)+,R3      ;RESTORE R3
2884 026544 000207              RTS      PC          ;RETURN
2885
2886                          ;ROUTINE TO GET THE ADDRESS LIMITS FOR THE CURRENT DRIVE TYPE
2887                          ;CALL
2888                          ;
2889                          ;      JSR      PC,GETLMT      ;CALL ROUTINE
2890                          ;
2891                          ;RO = DPB ADDRESS BEFORE CALLING THE ROUTINE
2892 026546  GETLMT:
2893 026546 010146              MOV     R1,-(SP)      ;;PUSH R1 ON STACK
2894 026550 005001              CLR     R1            ;START FRESH
2895 026552 111001              MOVVB  (RO),R1        ;GET DRIVE NUMBER
2896 026554 012737 000022 001450  MOV     #18,TRKLMT    ;ASSUME LAST TRACK FOR AN RM05
2897 026562 122761 000007 035644  CMPB   #7,DRVTYP(R1) ;IS DRIVE AN RM05 ?
2898 026570 001403              BEQ     1$            ;BR IF YES
2899 026572 012737 000004 001450  MOV     #4,TRKLMT    ;GET LAST TRACK FOR AN RM05
2900 026600 026600 012601          1$:  MOV     (SP)+,R1      ;;POP STACK INTO R1
2901 026602 000207              RTS      PC          ;RETURN
2902
2903                          ;ROUTINE TO GET THE DRIVE I.D. FROM THE OPERATOR
2904 026604  GETID:
2905 026606 010546              MOV     R5,-(SP)      ;SAVE R5
2906 026612 005737 000042      TST     42            ;UNDER MONITOR CONTROL ?
2907 026614 001030              BNE     2$            ;BR IF NOT
2908 026620 005037 001362 001362  CLR     CFLAG        ;CLEAR THE 'CONTROL C' FLAG
2909 026624 104411 074353          TYPE     ,ENTDRV     ;'ENTER DRV I.D.:'
2910
2911 026626 104411              RDLIN
2912 026630 012605              MOV     (SP)+,R5      ;GET THE ENTRY ADDRESS
2913 026634 005737 001362      TST     CFLAG        ;'CONTROL C' ENTERED ?
2914 026636 001367              BNE     1$            ;BR IF IT WAS

```

```

2915 026636 121527 000056
2916 026642 001414
2917 026644 112560 002106
2920 026650 112560 002107
      026654 112560 002110
      026660 112560 002111
      026664 112560 002112
      026670 112560 002113
2921 026674 012605
2922 026676 000207
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935 026700
      026700 010146
      026702 010246
      026704 010346
2936 026706 004737 026546
2937 026712 010001
2938 026714 062701 000124
2939 026720 010146
      026722 012746 000771
2940 026726 011602
2941 026730 012721 177777
2942 026734 005302
2943 026736 001374
2944 026740 012602
      026742 012601
2945 026744 111037 066070
2946 026750 012737 001466 066102
2947 026756 113737 001450 066101
2948 026764 112737 000000 066100
2949 026772 012737 177400 066074
2950 027000 112737 000171 066072
2951 027006 012737 000010 001270
2952 027014 012703 076746
2953 027020 004037 036550
2954 027024 066070
2955 027026 000772
2956 027030 005737 066106
2957 027034 001775
2958 027036 100010
2959 027040 062737 000002 066100
2960 027046 123737 001270 066100
2961 027054 103357
2962 027056 000440
2963 027060 005713
2964 027062 00432

```

```

      CMPB    (R5),#' .      ;PERIOD ENTERED ?
      BEQ     2$             ;BR IF YES
      MOVB   (R5)+,$SDRVID(R0) ;STORE THE DRIVE I.D.
      MOVB   (R5)+,$SDRVID+1(R0)
      MOVB   (R5)+,$SDRVID+2(R0)
      MOVB   (R5)+,$SDRVID+3(R0)
      MOVB   (R5)+,$SDRVID+4(R0)
      MOVB   (R5)+,$SDRVID+5(R0)
2$:    MOV     (SP)+,R5       ;RESTORE R5
      RTS    PC              ;RETURN

;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS IN THE DEC 144
;BAD SECTOR FILE (UP TO A MAX OF 126. FOR MFG PORTION AND
;126. FOR USR PORTION OF DEC 144 FILE.)
;CALL
      MOV     #DPB,R0        ;DPB ADDRESS
      JSR    R5,GETADR
      RET1   ERROR RET
      RET2   NORMAL RET

;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     R0,R1          ;DPB ADDRESS
      ADD    #SBDSEC,R1      ;ADDRESS OF BAD SECTOR TABLE
      MOV     R1,-(SP)       ;;PUSH R1 ON STACK
      MOV     #<126.*4>+1,-(SP) ;;PUSH #<126.*4>+1 ON STACK
      MOV     (SP),R2        ;NUMBER OF TOTAL ENTRIES AVAILABLE PLUS TERMINATOR
3$:    MOV     #-1,(R1)+      ;INITIALIZE ALL LOCS TO -1
      DEC    R2              ;DECREMENT WORD
      BNE   1$              ;BRANCH IF NOT DONE
      MOV     (SP)+,R2       ;;POP STACK INTO R2
      MOV     (SP)+,R1       ;;POP STACK INTO R1
      MOVB   (R0),GENDPB     ;DRIVE NUMBER
      MOV     #822,GENDPB+$CYL ;LAST CYLINDER
      MOVB   TRKLMT,GENDPB+$TRK ;GET LAST TRACK
      MOVB   #0,GENDPB+$SEC    ;GET STARTING SECTOR OF MFG FILE
      MOV     #-256,GENDPB+$WRDM ;ONE SECTOR WORD COUNT
      MOVB   #RDDAT,GENDPB+$COMND ;READ DATA COMMAND
      MOV     #8,$CDW2        ;GET LAST SECTOR OF MFG FILE
2$:    MOV     #CYLNDR,R3     ;GET READ BUFFER ADDRESS
      JSR    R0,RM05         ;READ CURRENT SECTOR
      GENDPB
      BR     2$             ;WAIT FOR QUE
3$:    TST    GENDPB+$STATUS  ;READ DONE YET ?
      BEQ    3$             ;BR IF NO
      BPL   4$             ;BR IF NO ERROR, ELSE
      ADD    #2,GENDPB+$SEC  ;INCREMENT NEXT SECTOR TO READ
      CMPB   $CDW2,GENDPB+$SEC ;WERE ALL SECTORS TRIED ?
      BHS   2$             ;BR IF NO
      BR     9$             ;BR IF UNSUCCESSFUL ON RETRIES
4$:    TST    (R3)           ;ARE LSB'S OF SERIAL NUMBER VALID ?
      BMI   7$             ;BR IF NO

```

```

2965 027064 005763 000002      TST      2(R3)          ;ARE MSB'S OF SERIAL NUMBER VALID ?
2966 027070 100427      BMI      7$           ;BR IF NO
2967 027072 062323      ADD      (R3)+,(R3)+  ;IS SERIAL NUMBER ZERO ?
2968 027074 001425      BEQ      7$           ;BR IF YES
2969 027076 005723      TST      (R3)+        ;IS 3RD WORD ALL 0'S ?
2970 027100 001023      BNE      7$           ;BR IF NO
2971 027102 005723      TST      (R3)+        ;IS 4TH WORD ALL 0'S ? (ALIGNMENT PACK ?)
2972 027104 001022      BNE      8$           ;BR IF NO
2973 027106 022713 177777      5$:     CMP      #-1,(R3) ;IS NEXT WORD A TERMINATOR ?
2974 027112 001403      BEQ      6$           ;BR IF YES
2975 027114 012321      MOV      (R3)+,(R1)+  ;STORE BAD CYLINDER ADDRESS
2976 027116 012321      MOV      (R3)+,(R1)+  ;STORE BAD TRK/SEC ADDRESS
2977 027120 000772      BR       5$
2978 027122 123727 066100 000012 6$:     CMPB     GENDPB+$SEC,#10. ;USR BAD FILE DONE YET ?
2979 027130 103031      BHIS     12$          ;BR IF YES
2980 027132 012737 000036 001270      MOV      #30.,$CDW2   ;GET LAST SECTOR OF USR BAD SECTOR FILE
2981 027140 112737 000012 066100      MOVB     #10.,GENDPB+$SEC ;GET STARTING SECTOR OF USR BAD SECTOR FILE
2982 027146 000722      BR       2$
2983 027150 005725      7$:     TST      (R5)+        ;OK TO USE PACK ANYWAY
2984 027152 104401 074774      8$:     TYPE     ,MERR2    ;INVALID FILE STRUCTURE
2985 027156 000402      BR       10$
2986 027160 104401 074716      9$:     TYPE     ,MERR1    ;FAILS TO RETRIEVE BAD SPOT FILE
2987 027164 104401 073331      10$:    TYPE     ,UNMSG    ;ON DRIVE
2988 027170 011046      MOV      (R0),-(SP)   ;:SAVE (R0) FOR TYPEOUT
      027172 104403      TYPOS    ;:GO TYPE--OCTAL ASCII
      027174 002      .BYTE    2           ;:TYPE 2 DIGIT(S)
      027175 000      .BYTE    0           ;:SUPPRESS LEADING ZEROS
2989 027176 104401 001203      TYPE     ,$CRLF      ;CR-LF
2990 027202 012721 177777      11$:    MOV      #-1,(R1)+ ;INITIALIZE ALL LOCS TO -1
2991 027206 005302      DEC      R2           ;DECREMENT WORD CTR
2992 027210 001374      BNE      11$          ;BRANCH IF NOT DONE
2993 027212 000401      BR       13$          ;ERROR RETURN
2994 027214 005725      12$:    TST      (R5)+        ;ADJUST FOR NORMALRETURN
2995 027216      13$:
      027216 012603      MOV      (SP)+,R3     ;:POP STACK INTO R3
      027220 012602      MOV      (SP)+,R2     ;:POP STACK INTO R2
      027222 012601      MOV      (SP)+,R1     ;:POP STACK INTO R1
2996 027224 000205      RTS      R5           ;EXIT
2997
2998 ;ROUTINE TO ENTER BAD SECTOR INFORMATION MANUALLY
2999 ;CALL
3000 ;
3001 ;     MOV      #DPB,R0      ;DPB ADDRESS
3002 ;     JSR      PC,MANTER    ;CALL ROUTINE
3003 ;
3004 ;R0 = DPB ADDRESS BEFORE CALLING THE ROUTINE
3005 MANTER:
      027226 010146      MOV      R1,-(SP)     ;:PUSH R1 ON STACK
      027230 010246      MOV      R2,-(SP)     ;:PUSH R2 ON STACK
      027232 010346      MOV      R3,-(SP)     ;:PUSH R3 ON STACK
      027234 010446      MOV      R4,-(SP)     ;:PUSH R4 ON STACK
3006 027236 105737 001150      TSTB     $AUTOB       ;RUNNING UNDER AUTO MODE ?
3007 027242 001402      BEQ      1$           ;BR IF NO
3008 027244 000137 027670      JMP      21$          ;YES, EXIT
3009 027250 005037 001362      1$:     CLR      CFLAG      ;CLEAR THE CONTROL-C FLAG
3010 027254 104401 074427      TYPE     ,ENTADR      ;MESSAGE TO ENTER...
3011

```

```

3012 027260 012704 000124      MOV      #SBDSEC,R4      ;INDEX VALUE OF TABLE ADDRESS
3013 027264 060004              ADD      R0,R4          ;TABLE STARTING ADDRESS
3014 027266 012701 000374      MOV      #<126.*2>,R1   ;TOTAL BAD SPOTS ALLOWED
3015 027272 022714 177777      2$:    CMP      #-1,(R4)    ;ENTRY IN THE TABLE ?
3016 027276 001407              BEQ      3$             ;BRANCH IF SO
3017 027300 062704 000004      ADD      #4,R4          ;ADJUST THE TABLE ENTRY POINTER
3018 027304 005301              DEC      R1             ;DECREMENT THE BAD SECTOR COUNT
3019 027306 001371              BNE     2$             ;BR IF TO NEXT ENTRIES POSITION
3020 027310 104401 075040      TYPE    ,MSGFULL       ;TYPE 'BAD SPOT TABLE IS FULL'
3021 027314 000565              BR      21$            ;EXIT..
3022
3023 027316 010146              3$:    MOV      R1,-(SP)     ;THE COUNTER AND FIRST
3024 027320 010446              MOV      R4,-(SP)     ;ENTRY POINTER PAIR
3025 027322 012714 177777      4$:    MOV      #-1,(R4)    ;RESET CYLINDER TO -1
3026 027326 104401 075072      TYPE    ,MSGCYL        ;TYPE 'CYLIND'
3027 027332 012746 177777      MOV      #-1,-(SP)    ;SAVE #-1 FOR TYPEOUT
3028 027336 104405              TYPDS   ;GO TYPE--DECIMAL ASCII WITH SIGN
3029 027340 104401 075216      TYPE    ,SLASH         ;TYPE ' / '
3030 027344 104411              RDLIN   ;READ IN THE BAD SPOTS
3031 027346 012601              MOV      (SP)+,R1     ;READ IN TEXT ADDRESS
3032 027350 005737 001362      TST     CFLAG          ;CONTROL-C ENTERED ?
3033 027354 001411              BEQ     7$             ;BRANCH IF NOT
3034 027356 012604              5$:    MOV      (SP)+,R4   ;RETRIEVE THE FIRST ENTRY POINTER
3035 027360 012601              MOV      (SP)+,R1     ;RETRIEVE THE SPOT COUNT
3036 027362 012724 177777      6$:    MOV      #-1,(R4)+  ;RESET THE TABLE
3037 027366 012724 177777      MOV      #-1,(R4)+  ;TO -1
3038 027372 005301              DEC     R1             ;DONE WITH TABLE YET ?
3039 027374 001372              BNE     6$            ;BR IF NO
3040 027376 000724              BR      1$            ;START AGAIN
3041 027400              7$:
3042 027400 013702 001444      MOV      CYLIMT,R2    ;UPPER LIMIT OF INPUT
3043 027404 004537 031366      JSR     R5,CK.DIG     ;CHECK THE DIGIT(S)
3044 027410 027664              20$    ;CARRIAGE RETURN ONLY ENTERED
3045 027412 027664              20$    ;PERIOD ONLY ENTERED
3046 027414 027424              8$     ;ILLEGAL INPUT
3047 027416 027444              10$    ;TERMINATED WITH A CARRIAGE RETURN
3048 027420 027424              8$     ;TERMINATED WITH A '...'
3049 027422 027432              9$     ;TERMINATED WITH A '...'
3050 027424 104401 074546      8$:    TYPE    ,BADENT    ;TYPE BAD ENTRY MESSAGE
3051 027430 000734              BR      4$            ;AND TRY AGAIN.
3052 027432 010214              9$:    MOV      P?,(R4)    ;ENTER CYLINDER ADDRESS
3053 027434 012764 177777 000002  MOV      #-1,2(R4)    ;RESET TRACK AND SECTOR FIELD
3054 027442 000510              BR      20$           ;ENTER CYLINDER ADDRESS
3055 027444 010214              10$:   MOV      R2,(R4)
3056 027446 012764 177777 000002  11$:   MOV      #-1,2(R4)    ;RESET TRACK AND SECTOR FIELDS TO -1
3057 027454 104401 075101      TYPE    ,MSGTRK       ;TYPE 'TRACK'
3058 027460 012746 177777      MOV      #-1,-(SP)   ;SAVE #-1 FOR TYPEOUT
3059 027464 104405              TYPDS   ;GO TYPE--DECIMAL ASCII WITH SIGN
3060 027466 104401 075216      TYPE    ,SLASH         ;TYPE ' / '
3061 027472 104411              RDLIN   ;READ IN THE BAD SPOTS
3062 027474 012601              MOV      (SP)+,R1     ;READ IN TEXT ADDRESS
3063 027476 005737 001362      TST     CFLAG          ;CONTROL-C ENTERED ?
3064 027502 001325              BNE     5$            ;BR IF YES
3065 027504 013702 001450      MOV      TRKLMT,R2    ;UPPER LIMIT OF INPUT
3066 027510 004537 031366      JSR     R5,CK.DIG     ;CHECK THE DIGIT(S)

```

	027514	027550			15\$:CARRIAGE RETURN ONLY ENTERED	
	027516	027664			20\$:PERIOD ONLY ENTERED	
	027520	027530			12\$:ILLEGAL INPUT	
	027522	027544			14\$:TERMINATED WITH A CARRIAGE RETURN	
	027524	027530			12\$:TERMINATED WITH A ''''	
	027526	027536			13\$:TERMINATED WITH A ''''	
3058	027530	104401	074546	12\$:	TYPE	,BADENT	:TYPE BAD ENTRY MESSAGE	
3059	027534	000744			BR	11\$:AND TRY AGAIN.	
3060	027536	110264	000003	13\$:	MOVB	R2,3(R4)	:ENTER TRACK ADDRESS AND	
3061	027542	000450			BR	20\$:EXIT	
3062	027544	110264	000003	14\$:	MOVB	R2,3(R4)	:ENTER TRACK ADDRESS	
3063								
3064	027550	112764	177777	000002	15\$:	MOVB	#-1,2(R4)	:RESET SECTOR TO -1
3065	027556	104401	075110		TYPE	,MSGSEC	:TYPE SECTOR'	
3066	027562	012746	177777		MOV	#-1,-(SP)	::SAVE #-1 FOR TYPEOUT	
	027566	104405			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN	
3067	027570	104401	075216		TYPE	,SLASH	:TYPE ' / '	
3068	027574	104411			RDLIN		:READ IN THE BAD SPOTS	
3069	027576	012601			MOV	(SP)+,R1	:READ IN TEXT ADDRESS	
3070	027600	005737	001362		TST	CFLAG	:CONTROL-C ENTERED ?	
3071	027604	001264			BNE	5\$:BR IF YES	
3072	027606	013702	001446		MOV	SECLMT,R2	:UPPER LIMIT OF INPUT	
	027612	004537	031366		JSR	R5,CK.DIG	:CHECK THE DIGIT(S)	
	027616	027652			19\$:CARRIAGE RETURN ONLY ENTERED	
	027620	027664			20\$:PERIOD ONLY ENTERED	
	027622	027632			16\$:ILLEGAL INPUT	
	027624	027646			18\$:TERMINATED WITH A CARRIAGE RETURN	
	027626	027632			16\$:TERMINATED WITH A ''''	
	027630	027640			17\$:TERMINATED WITH A ''''	
3073	027632	104401	074546	16\$:	TYPE	,BADENT	:TYPE BAD ENTRY MESSAGE	
3074	027636	000744			BR	15\$:AND TRY AGAIN.	
3075	027640	110264	000002	17\$:	MOVB	R2,2(R4)	:ENTER SECTOR ADDRESS AND	
3076	027644	000407			BR	20\$:EXIT	
3077	027646	110264	000002	18\$:	MOVB	R2,2(R4)	:ENTER SECTOR ADDRESS	
3078								
3079	027652	005303			19\$:	DEC	R3	:MORE ENTRY ?
3080	027654	001403			BEQ	20\$:BRANCH IF EXHAUSTED	
3081	027656	062704	000004		ADD	#4,R4	:ADJUST FOR THE NEXT TABLE ENTRY	
3082	027662	000617			BR	4\$:ENTER NEXT SPOT ADDRESS	
3083	027664	062706	000004	20\$:	ADD	#4,SP	:RESTORE STACK	
3084	027670			21\$:				
	027670	012604			MOV	(SP)+,R4	::POP STACK INTO R4	
	027672	012603			MOV	(SP)+,R3	::POP STACK INTO R3	
	027674	012602			MOV	(SP)+,R2	::POP STACK INTO R2	
	027676	012601			MOV	(SP)+,R1	::POP STACK INTO R1	
3085	027700	000207			RTS	PC	:EXIT	
3086								
3087								
3088								
3089								
3090								
3091								
3092	027702	010346						
3093	027704	005037	001362		PARENT:	MOV	R3,-(SP)	:SAVE THE PARAMETER TABLE ADDRESS
3094	027710	012337	027720		1\$:	CLR	CFLAG	:CLEAR THE 'CONTROL C' FLAG
3095	027714	001442			MOV	(R3)+,3\$:ADDRESS OF PARAMETER NAME	
3096	027716	104401			BEQ	9\$:BR IF AT END OF TABLE	
					TYPE		:TYPE THE PARAMETER NAME	

:PARAMETER ENTRY ROUTINE

:CALL

:
: MOV #ADR,R3
: JSR PC,PARENT

:PARAMETER TABLE ADDRESS
:GET THE PARAMETERS

:SAVE THE PARAMETER TABLE ADDRESS
:CLEAR THE 'CONTROL C' FLAG
:ADDRESS OF PARAMETER NAME
:BR IF AT END OF TABLE
:TYPE THE PARAMETER NAME

```

3097 027720 000000      3$:      .WORD      0      ;ADDRESS OF PARAMETER NAME TEXT
3098 027722 012302      MOV      (R3)+,R2    ;MAXIMUM PARAMETER VALUE
3099 027724 012305      MOV      (R3)+,R5    ;ADDRESS OF PARAMETER
3100 027726 011546      MOV      (R5),-(SP)  ;CURRENT VALUE OF PARAMETER
3101 027730 104405      TYPDS    ;TYPE THE CURRENT VALUE OF THE PARAMETER
3102 027732 104401 075216 TYPE      ,SLASH    ;' / '
3103 027736 104411      RDLIN   ;READ THE KEYBOARD
3104 027740 012601      MOV      (SP)+,R1    ;INPUT ASCII STRING ADDRESS
3105 027742 005737 001362 TST      CFLAG      ;'CONTROL C' ENTERED ?
3106 027746 001021      BNE     8$          ;BR IF IT WAS
3107 027750 004537 031366 JSR      R5,CK.DIG   ;CHECK THE DIGIT(S)
      027754 027710      1$      ;CARRIAGE RETURN ONLY ENTERED
      027756 030022      9$      ;PERIOD ONLY ENTERED
      027760 027774      6$      ;ILLEGAL INPUT
      027762 027770      5$      ;TERMINATED WITH A CARRIAGE RETURN
      027764 027774      6$      ;TERMINATED WITH A ''
      027766 030006      7$      ;TERMINATED WITH A ''
3108 027770 010215      5$:      MOV      R2,(R5)  ;MOVE NEW VALUE TO PARAMETER LOCATION
3109 027772 000745      BR      1$          ;GET MORE PARAMETERS
3110 027774 104401 074546 6$:      TYPE      ,BADENT ;'BAD ENTRY'
3111 030000 162703 000006 SUB      #6,R3       ;DECREMENT THE TABLE POINTER
3112 030004 000741      BR      1$          ;TRY AGAIN
3113 030006 010215      7$:      MOV      R2,(R5)  ;NEW VALUE
3114 030010 000404      BR      9$          ;EXIT
3115 030012 005037 001362 8$:      CLR      CFLAG      ;CLEAR THE 'CONTROL C' FLAG
3116 030016 011603      MOV      (SP),R3    ;RELOAD THE PARAMETER TABLE ADDRESS
3117 030020 000733      BR      1$          ;TRY AGAIN
3118 030022 005726      9$:      TST      (SP)+    ;CORRECT THE STACK POINTER
3119 030024 000207      RTS      PC        ;RETURN
3120
3121      ;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE
3122      ;CALL
3123      ;
3124      ;      MOV      #MESADR,ASNMSG ;ERROR MESSAGE ADDRESS
3125      ;      JSR      PC,ASNERR
3126      ;      RETURN
3127 030026 104401 001203 ASNERR: TYPE      ,SCLF   ;CR-LF
3128 030032 104401 073327 TYPE      ,QUES      ;'?'
3129 030036 104401 073331 TYPE      ,UNTMSG    ;TYPE 'DRIVE'
3130 030042 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
      030044 104403      TYPDS    ;TYPE DRIVE NUMBER
      030046      002      .BYTE      2      ;GO TYPE--OCTAL ASCII
      030047      000      .BYTE      0      ;TYPE 2 DIGIT(S)
3131 030050 104401      TYPE      ;SUPPRESS LEADING ZEROS
3132 030052 000000      ASNMSG: .WORD      0      ;TYPE SPECIFIC MESSAGE
3133 030054 000207      RTS      PC        ;MESSAGE ADDRESS
3134
3135      ;DEASSIGN DRIVE IF A FATAL ERROR OCCURS
3136      ;CALL
3137      ;      JSR      PC,DROP
3138      ;      RETURN
3139
3140 030056 005004      DROP:  CLR      R4      ;CLEAR R4 FOR DRIVE NUMBER
3141 030060 111004      MOV      (R0),R4    ;MOVE DRIVE NUMBER TO R4
3142 030062 146437 035750 001550 BIC      ATABIT(R4),ASNLS ;REMOVE DRIVE FROM ASSIGNED LIST
3143 030070 006304      ASL      R4        ;MAKE DRIVE NUMBER INTO A TABLE INDEX

```



```

3144 030072 010064 001552      MOV      R0,DUNIT(R4)      ;PUT DRIVE IN DROP LIST
3145 030076 104401 001203      TYPE    , $CRLF
3146 030102 104401 074011      TYPE    ,DROPNNG          ;TYPE 'FATAL OR EXCESSIVE ERRORS'
3147 030106 104401 074064      TYPE    ,MSGON            ;TYPE 'ON'
3148 030112 104401 073331      TYPE    ,UNTMSG          ;TYPE 'DRIVE'
3149 030116 006204              ASR      R4                ;DRIVE NUMBER
3150 030120 010446              MOV      R4,-(SP)         ;;SAVE R4 FOR TYPEOUT
                                ;;TYPE DRIVE NUMBER
                                ;;GO TYPE--OCTAL ASCII
                                ;;TYPE 2 DIGIT(S)
                                ;;SUPPRESS LEADING ZEROS
                                ;;MORE DRIVES ACTIVE
                                ;;YES
                                ;;ANY MONITOR ?
                                ;;NO
                                ;;CLEAR STACK
                                ;;GIVE CONTROL TO MONITOR
                                1$:
                                RTS      PC

                                ;ROUTINE TO DEASSIGN DRIVE IF ERRORS BECOMES EXCESSIVE
3151 030126 105737 001550      TSTB    ASNLST
3152 030132 001006              BNE     1$
3153 030134 005737 000042      TST     @#42
3154 030140 001403              BEQ     1$
3155 030142 005726              TST     (SP)+
3156 030144 000137 030526      JMP     $GET42
3157 030150 000207              1$:    RTS      PC

                                ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
3158
3159
3160
3161 030152 032777 000020 150774 ABNRML: BIT      #SW04,@SWR      ;SEE IF SWITCH 4 SET
3162 030160 001006              BNE     1$                ;BR IF IT'S SET
3163 030162 023760 001470 000056      CMP     MAXER,$TOTAL(R0)  ;CHECK TOTAL ERROR VALUE
3164 030170 103002              BHI     1$                ;BR IF ERRORS DO NOT EXCEED MAX
3165 030172 000137 030056      JMP     DROP              ;DEASSING THE DRIVE
3166 030176 000207              1$:    RTS      PC
                                ;RETURN

                                ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
3167
3168
3169
3240
3241

                                .SBTTL  END OF PASS ROUTINE

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

                                $EOP:
                                TST     ENDET          ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
                                BEQ     EOP1          ;BR IF SEEKS
                                CMP     $READ+2(R0),ENDCON+2 ;CHECK MSW OF WORDS READ COUNT
                                BHI     EOP2          ;BR IF MSW GREATER THAN LIMIT
                                BLO     EOPX          ;BR IF MSW LESS THAN LIMIT
                                CMP     $READ(R0),ENDCON ;CHECK LSW AGAINST LIMIT
                                BHI     EOP2          ;BR IF EQUAL OR GREATER
                                BR      EOPX          ;EXIT

                                EOP1:
                                CMP     $POSIT+2(R0),ENDSEK+2 ;CHECK MSW OF SEEK COUNT
                                BHI     EOP2          ;BR IF MSW GREATER THAN LIMIT
                                BLO     EOPX          ;EXIT IF MSW LESS THAN LIMIT
                                CMP     $POSIT(R0),ENDSEK ;CHECK LSW OF SEEK COUNT
                                BLO     EOPX          ;EXIT IF LSW LESS THAN LIMIT

                                EOP2:
                                TYPE    , $CRLF      ;CR-LF
                                TYPE    ,ENDPAS      ;END OF PASS FOR THE DRIVE
                                MOV     $PASSC(R0),-(SP) ;;SAVE $PASSC(R0) FOR TYPEOUT

```

```

030270 104405          TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
030272 111037 001346  MOVB      (R0),UNIT      ;;STORE THE DRIVE NUMBER
030276 104401 074064  TYPE      ,MSGON         ;;TYPE 'ON'
030302 104401 073331  TYPE      ,UNTMSG        ;;'DRIVE '
030306 013746 001346  MOV       UNIT,-(SP)     ;;SAVE UNIT FOR TYPEOUT
030312 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
030314      002          .BYTE      2             ;;TYPE 2 DIGIT(S)
030315      000          .BYTE      0             ;;SUPPRESS LEADING ZEROS
030316 104401 001203  TYPE      ,SCRLF         ;;CR-LF

030322 032777 000020 150624  BIT      #SW04,@SWR      ;;TYPE END OF TEST MESSAGE ?
030330 001017          BNE      1$             ;;BR IF NO
030332 026037 000070 001464  CMP      $PASSC(R0),PASCNT ;;SEE IF AT END OF TEST
030340 103413          BLO      1$             ;;BR IF NOT
030342 104401 074071  TYPE      ,ENDTST        ;;TYPE 'END OF TEST'
030346 104401 074107  TYPE      ,MSGFOR        ;;TYPE 'FOR'
030352 104401 073331  TYPE      ,UNTMSG        ;;'DRIVE '
030356 013746 001346  MOV       UNIT,-(SP)     ;;SAVE UNIT FOR TYPEOUT
030362 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
030364      002          .BYTE      2             ;;TYPE 2 DIGIT(S)
030365      000          .BYTE      0             ;;SUPPRESS LEADING ZEROS
030366 000421          BR       3$             ;;DEASSIGN THE DRIVE

030370 004737 023616          1$:      JSR      PC,SUMARY      ;;TYPE THE DRIVE'S STATISTICS SUMMARY
030374 010346          MOV      R3,-(SP)      ;;SAVE R3
030376 010446          MOV      R4,-(SP)      ;;SAVE R4
030400 010004          MOV      R0,R4         ;;DRIVE'S BLOCK ADDRESS
030402 062704 000036  ADD      #$OPERC,R4      ;;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
030406 012703 000010  MOV      #8.,R3         ;;NUMBER OF LOCNS TO BE CLEARED
                                ;;(ERROR COUNTERS NOT CLEARED)

030412 005024          2$:      CLR      (R4)+          ;;CLEAR THE LOCN
030414 005303          DEC      R3             ;;DECREMENT THE LOCATION COUNTER
030416 001375          BNE      2$             ;;BR IF MORE TO GO
030420 012604          MOV      (SP)+,R4        ;;RESTORE R4
030422 012603          MOV      (SP)+,R3        ;;RESTORE R3
030424 005260 000070  INC      $PASSC(R0)      ;;INCREMENT THE PASS COUNT
030430 000422          BR       EOPX         ;;EXIT

030432 005004          3$:      CLR      R4             ;;CLEAR R4 FOR DRIVE NUMBER
030434 111004          MOVB     (R0),R4         ;;MOVE DRIVE NUMBER
030436 146437 035750 001550  BICB     ATABIT(R4),ASNLST ;;DELETE DRIVE FROM ASSIGNED LIST
030444 006304          ASL      R4             ;;MAKE DRIVE NUMBER INTO TABLE INDEX
030446 010064 001552  MOV      R0,DUNIT(R4)    ;;PUT BLOCK ADDRESS INTO DROP LIST
030452 105737 001550  TSTB     ASNLST         ;;ALL DRIVES ARE DESIGNED ?
030456 001007          BNE      EOPX         ;;BRANCH IF NOT
030460 005237 001216  INC      $DEVCT         ;;INCREMENT DEVICE COUNT
030464 005237 001214  INC      $PASS          ;;INCREMENT THE PASS NUMBER FOR APT
030470 042737 100000 001214  BIC      #100000,$PASS   ;;AVOID NEGATIVE NUMBER
030476 000207          EOPX:    RTS          PC      ;;RETURN
030500 005237 001214  INC      $PASS          ;;INCREMENT THE PASS NUMBER
030504 042737 100000 001214  BIC      #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
030512 005327          DEC      (PC)+        ;;LOOP?
030514 000001          $EOPCT: .WORD      1
030516 003013          BGT      $DOAGN        ;;YES
030520 012737          MOV      (PC)+,@(PC)+  ;;RESTORE COUNTER
030522 000001          $ENDCT: .WORD      1
030524 030514          $EOPCT

```

```

030526 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
030532 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
030534 000005 RESET ;;CLEAR THE WORLD
030536 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
030540 000240 NOP ;;SAVE ROOM
030542 000240 NOP ;;FOR
030544 000240 NOP ;;ACT11
030546 $DOAGN:
030546 000137 JMP @ (PC)+ ;;RETURN
030550 030552 $RTNAD: .WORD FISK
3242 030552 005237 001212 FISK: INC $TESTN ;INCREMENT THE TEST NUMBER IN THE MAIL BOX
3243 030556 000137 006260 JMP MAIN1 ;RETURN TO LOOP
3244
3245 ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
3246 ;CALL
3247 ;
3248 ; MOV NUMBER,R5 ;DIVISOR INTO R5
3249 ; JSR PC,GETREM
3250 ; RETURN ;REMAINDER IS IN R5
3251 030562 013746 035114 GETREM: MOV $LONUM,-(SP) ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
3252 030566 013746 035112 MOV $HINUM,-(SP) ;UPPER PART
3253 030572 010546 MOV R5,-(SP) ;PUT THE DIVISOR ONTO THE STACK
3254 030574 004737 030610 JSR PC,LINKDV ;DIVIDE THE RANDOM NUMBERS
3255 030600 012605 MOV (SP)+,R5 ;PUT THE REMAINDER INTO R5
3256 030602 005726 TST (SP)+ ;ADJUST THE STACK POINTER
3257 030604 000240 NOP ;FOR DEBUGGING HALT
3258 030606 000207 RTS PC
3259
3260 ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
3261 ; THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
3262 ; CALLING SEQUENCE TO BE USED
3263
3264 030610 104412 LINKDV: SAVREG ;STORE R0 - R5
3265 030612 016605 000026 MOV 26(SP),R5 ;DIVISOR
3266 030616 005004 CLR R4 ;OTHER DIVISOR WORD
3267 030620 016602 000030 MOV 30(SP),R2 ;UPPER DIVIDEND WORD
3268 030624 016603 000032 MOV 32(SP),R3 ;LOWER DIVIDEND WORD
3269 030630 005000 CLR R0 ;CLEAR OTHER DIVIDEND REGISTERS
3270 030632 005001 CLR R1
3271 030634 004737 030656 JSR PC,M.DPID ;GO TO THE DIVIDE ROUTINE
3272 030640 010166 000030 MOV R1,30(SP) ;REMAINDER ON THE STACK
3273 030644 010366 000032 MOV R3,32(SP) ;QUOTIENT ON THE STACK
3274 030650 104413 RESREG ;RESTORE R0 - R5
3275 030652 012616 MOV (SP)+,(SP) ;MOVE RETURN UP THE STACK
3276 030654 000207 RTS PC
3277
3278 ; DIVISION UTILITY SUBROUTINE
3279 ; R0-R1-R2-R3=DIVIDEND
3280 ; R4-R5=DIVISOR
3281 ; R0-R1=REMAINDER AFTER DIVISION
3282 ; R2-R3=QUOTIENT AFTER DIVISION
3283 ; ENTER WITH JSR PC,M.DPID
3284
3285
3286 030656 012746 000040 M.DPID: MOV #40,-(SP) ;COUNTER FOR DIVISION CYCLES
3287 030662 010446 MOV R4,-(SP) ;HIGH ORDER
3288 030664 010546 MOV R5,-(SP) ;LOW ORDER DIVISOR TO THE STACK

```

```

3289 030666 005466 000002      NEG      2(SP)      ;FORM NEGATIVE
3290 030672 005416              NEG      @SP        ;VERSION OF THE DIVISOR
3291 030674 005666 000002      SBC      2(SP)
3292 030700 061601              ADD      @SP,R1
3293 030702 005500              ADC      R0
3294 030704 066600 000002      ADD      2(SP),R0    ;PERFORM THE INITIAL SUBTRACTION
3295 030710 103445              BCS      M.DP50      ;IF CARRY THEN OVERFLOW HAS OCCURRED
3296 C_0712 005046              CLR      -(SP)      ;THIS IS A LONGER LASTING CARRY BIT
3297 030714 006103      M.DP40: ROL      R3
3298 030716 006102              ROL      R2
3299 030720 006101              ROL      R1
3300 030722 006100              ROL      R0
3301 030724 005716              TST      @SP        ;TEST 'CARRY' INDICATOR
3302 030726 001410              BEQ      M.DP41      ;IF NO 'CARRY' THEN ADD ELSE SUBTRACT
3303 030730 005016              CLR      @SP        ;CLEAR UP FOR NEXT TIME
3304 030732 066601 000002      ADD      2(SP),R1
3305 030736 005500              ADC      R0          ;ADD -(DIVISOR)
3306 030740 005516              ADC      @SP        ;SET 'CARRY'
3307 030742 066600 000004      ADD      4(SP),R0 ;<- I
3308 030746 000404              BR       M.DP42
3309 030750 060501      M.DP41: ADD      R5,R1
3310 030752 005500              ADC      R0          ;ADD +(DIVISOR)
3311 030754 005516              ADC      @SP        ;SET 'CARRY'
3312 030756 060400              ADD      R4,R0      ;<- I
3313 030760 005516      M.DP42: ADC      @SP        ;SET 'CARRY'
3314 030762 005716              TST      @SP        ;TEST THE UPDATE INDICATOR
3315 030764 001401              BEQ      .+4        ;IF ZERO FORGET IT
3316 030766 005203              INC      R3          ;NO CARRY POSSIBLE HERE
3317 030770 005366 000006      DEC      6(SP)      ;DECREMENT COUNTER
3318 030774 003347              BGT      M.DP40      ;BRANCH IF MORE TO DO
3319 030776 006003              ROR      R3
3320 031000 103404              BCS      M.DP44
3321 031002 060501              ADD      R5,R1
3322 031004 005500              ADC      R0
3323 031006 060400              ADD      R4,R0
3324 031010 000241              CLC
3325 031012 006103      M.DP44: ROL      R3
3326 031014 062706 000010      ADD      #10,SP     ;ADJUST STACK BY 4 WORDS
3327 031020 000242              CLV
3328 031022 000207              RTS      PC
3329 031024 062706 000006      M.DP50: ADD      #6,SP
3330 031030 000262              SEV
3331 031032 000207              RTS      PC
3332
3333
3334      ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
3335      ;CALL
3336      ;
3337      ;      MOV      #ADR, -(SP)      ;ADDRESS OF NUMBER (IN ASCII)
3338      ;      JSR      R5,REPLZ
3339      ;      .WORD      N          ;'N' IS NUMBER OF DIGITS TO BE TYPED
3340 031034 010046      REPLZ: MOV      R0, -(SP)      ;SAVE R0
3341 031036 012746 000012      MOV      #10, -(SP)      ;MAXIMUM NUMBER OF DIGITS TO BE TYPED
3342 031042 162516              SUB      (R5)+, (SP)      ;SUBTRACT DIGITS TO FORM INDEX,
3343 031044 016600 000006      MOV      6(SP), R0      ;ADDRESS OF NUMBER TO R0
3344 031050 122710 000060      1$:  CMPB     #'0', (R0)      ;BYTE EQUAL TO ASCII '0' ?
3345 031054 001004              BNE     2$             ;BR IF NOT

```

```

3346 031056 112710 000040      MOVB    #40,(R0)      ;REPLACE THE ZERO WITH A SPACE
3347 031062 005200              INC     R0            ;INCREMENT THE BYTE ADDRESS
3348 031064 000771              BR     1$            ;GO BACK AND LOOK FOR MORE LEADING ZEROS
3349 031066 105710      2$:  TSTB    (R0)      ;SEE IF ZERO BYTE TERMINATOR
3350 031070 001003              BNE    3$            ;BR IF NOT
3351 031072 005300              DEC     R0            ;BACKUP STRING POINTER
3352 031074 112710 000060      MOVB    #'0,(R0)     ;PUT A ZERO BACK IN
3353 031100 016637 000006 031114 3$:  MOV     6(SP),4$     ;PUT ADDRESS IN LOCATION FOR TYPEOUT
3354 031106 062637 031114      ADD     (SP)+,4$    ;BEGINNING OF SIGNIFICANT DIGITS
3355 031112 104401              TYPE                   ;TYPE THE NUMBER
3356 031114 000000      4$:  .WORD  0          ;ADDRESS OF NUMBER
3357 031116 012600              MOV     (SP)+,R0     ;RESTORE R0
3358 031120 012616              MOV     (SP)+,(SP)   ;MOVE RETURN ADDRESS
3359 031122 000205              RTS     R5           ;RETURN
3360
3361              ;TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
3362
3363              ;CALL
3364              ;
3365              ;      MOV     #NUMADR,-(SP) ;FIRST ADDRESS OF ASCIZ STRING
3366              ;      JSR     PC,$SUPRS
3367 031124 010046      $SUPRS: MOV     R0,-(SP)   ;SAVE R0
3368 031126 016600 000004      MOV     4(SP),R0    ;PICKUP THE POINTER
3369 031132 105710      1$:  TSTB    (R0)      ;TERMINATOR ?
3370 031134 001403              BEQ    2$            ;BR IF YES
3371 031136 122720 000060      CMPB   #'0,(R0)+    ;IS THIS AN ASCII '0' ?
3372 031142 001773              BEQ    1$            ;BR IF YES
3373 031144 005300      2$:  DEC     R0          ;BACKUP BY '1'
3374 031146 010037 031154      MOV     R0,3$       ;SAVE FOR TYPING
3375 031152 104414      DISPLY                   ;GO PRINT
3376 031154 000000      3$:  .WORD  0          ;ASCIZ POINTER GOES HERE
3377 031156 012600              MOV     (SP)+,R0     ;RESTORE R0
3378 031160 012616              MOV     (SP)+,(SP)   ;RESTORE THE STACK
3379 031162 000207              RTS     PC           ;RETURN
3380
3381              ;ROUTINE TO TYPE AT PRIORITY 4
3382
3383 031164 013746 177776      TYPRI4: MOV     @MPS,-(SP) ;SAVE THE PRESENT STATUS
3384 031170 012737 000200 177776      MOV     #200,@MPS   ;CHANGE THE PRIORITY TO 4
3385 031176 012537 031206      MOV     (R5)+,1$    ;MESSAGE ADDRESS
3386 031202 004737 032606      JSR     PC,$TYPE    ;TYPE THE MESSAGE
3387 031206 000000      1$:  .WORD  0          ;MESSAGE ADDRESS GOES HERE
3388 031210 000205              RTS     R5           ;RETURN
3389
3390              ;ROUTINE TO TYPE ERRORS
3391              ;CALL
3392              ;
3393              ;      DISPLY                   ;MUST DEFINED IN 'TRAP' TABLE
3394              ;      MESADR                   ;ADDRESS OF MESSAGE
3395              ;      RETURN
3396 031212 032777 020000 147734  $DSPLY: BIT     #BIT13,@SWR ;INHIBIT ERROR TYPEOUT ?
3397 031220 001004              BNE    1$            ;BR IF YES
3398 031222 005037 177776      CLR     @MPS        ;SET PRIORITY TO ZERO
3399 031226 000137 032606      JMP     $TYPE       ;TYPE THE MESSAGE
3400 031232 062716 000002      1$:  ADD     #2,(SP)   ;INCREMENT THE RETURN
3401 031236 000002              RTI
3402

```

```
3403 :THIS ROUTINE IS USED TO CHECK IF AN
3404 :ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
3405 :CALL
3406 :
3407 :       MOV     #ADR,R1       ;ADDRESS OF ASCII CHARACTER
3408 :       JSR     R5,CK.OCT     ;CHECK THE CHARACTER
3409 :       RETURN1                    ;CHARACTER IS NOT BETWEEN 0-7
3410 :       RETURN2                    ;CHARACTER IS IN R2 AS A
3411 :                               ;OCTAL DIGIT
3412 CK.OCT: CMPB   (R1),#'0     ;LESS THAN ZERO?
3413         BLO   1$            ;YES -- BRANCH
3414         CMPB   (R1),#'7     ;GREATER THAN SEVEN?
3415         BHI   1$            ;YES -- BRANCH
3416         MOVB  (R1),R2       ;GET THE CHARACTER
3417         BIC   #'C7,R2       ;STRIP AWAY THE ASCII
3418         TST   (R5)+         ;ADJUST FOR RETURN
3419 1$:     RTS    R5           ;RETURN
3420
3421 :THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
3422 :AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
3423 :CALL
3424 :
3425 :       MOV     #ADR,R1       ;ADDRESS OF ASCII CHARACTER
3426 :       JSR     R5,CK.DEC     ;CHECK THE CHARACTER
3427 :       RETURN1                    ;NOT BETWEEN 0 AND 9
3428 :       RETURN2                    ;BETWEEN 0 AND 9
3429 :                               ;R2 = DIGIT
3430 CK.DEC: CMPB   (R1),#'0     ;LESS THAN ZERO?
3431         BLO   1$            ;YES -- BRANCH
3432         CMPB   (R1),#'9     ;GREATER THAN NINE?
3433         BHI   1$            ;YES -- BRANCH
3434         MOVB  (R1),R2       ;GET THE CHARACTER
3435         BIC   #'0,R2        ;STRIP AWAY THE ASCII
3436         TST   (R5)+         ;ADJUST FOR RETURN
3437 1$:     RTS    R5           ;RETURN
3438
3439 :THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
3440 :DETERMINE WHAT IT IS.
3441 :CALL
3442 :
3443 :       MOV     #ADR,R1       ;ADDRESS OF ASCII CHARACTER
3444 :       JSR     R5,CK.CHR     ;CHECK CHARACTER
3445 :       RETURN ADR1            ;UNKNOWN CHARACTER
3446 :       RETURN ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
3447 :       RETURN ADR3            ;COMMA * (R1)=ADR+1
3448 :       RETURN ADR4            ;PERIOD * (R1)=ADR+1
3449 :       RETURN ADR5            ;DIGIT BETWEEN 0 AND 7.
3450 :       RETURN ADR6            ;DIGIT BETWEEN 8 AND 9.
3451 :                               ;R2 = DIGIT * (R1)=ADR+1
3452 CK.CHR: TSTB   (R1)         ;'CARRIAGE RETURN'?
3453         BEQ   3$            ;YES -- BRANCH
3454         CMPB   (R1),#',      ;'COMMA'?
3455         BEQ   2$            ;YES -- BRANCH
3456         CMPB   (R1),#'.      ;'PERIOD'?
3457         BEQ   1$            ;YES -- BRANCH
3458         JSR   R5,CK.DEC     ;'DIGIT'?
3459         BR    4$            ;NO -- BRANCH
```

```

3460 031342 004537 031240      JSR    R5,CK.OCT      ;OCTAL ?
3461 031346 005725             TST    (R5)+          ;DIGIT BETWEEN 8-9
3462 031350 005725             TST    (R5)+          ;DIGIT BETWEEN 0-7
3463 031352 005725             1$:   TST    (R5)+          ;PERIOD
3464 031354 005725             2$:   TST    (R5)+          ;COMMA
3465 031356 005725             3$:   TST    (R5)+          ;CARRIAGE RETURN
3466 031360 005201             INC    R1             ;MOVE POINTER TO NEXT CHARACTER
3467 031362 011505             4$:   MOV    (R5),R5    ;UNKNOWN CHARACTER
3468 031364 000205             RTS     R5            ;RETURN
3469
3470                               ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
3471                               ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
3472                               ;CALL
3473                               ;
3474                               ;   MOV    #ADR,R1      ;ADDRESS OF ASCII STRING
3475                               ;   MOV    #NUM,R2      ;MAX. MAGNITUDE OF INPUT NUMBER
3476                               ;   JSR    R5,CK.DIG     ;CHECK DIGITS
3477                               ;   RETURN  ADR1        ;'CR' ONLY ENTERED -- R2=0
3478                               ;   RETURN  ADR2        ;'PERIOD' ONLY ENTERED -- R2=0
3479                               ;   RETURN  ADR3        ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
3480                               ;   RETURN  ADR4        ;'CR' -- R2 = NUMBER
3481                               ;   RETURN  ADR5        ;'COMMA' -- R2 = NUMBER
3482                               ;   RETURN  ADR6        ;'PERIOD' -- R2 = NUMBER
3483 031366 010446             CK.DIG: MOV    R4,-(SP)    ;SAVE R4
3484 031370 010346             MOV    R3,-(SP)    ;SAVE R3
3485 031372 010246             MOV    R2,-(SP)    ;SAVE THE MAX. SIZE ON THE STACK
3486 031374 005002             CLR    R2          ;START WITH 0
3487 031376 005003             CLR    R3
3488 031400 005004             CLR    R4
3489 031402 004537 031314     JSR    R5,CK.CHR    ;CHECK ONE CHARACTER
3490 031406 031502             6$:   ;ILLEGAL CHARACTER
3491 031410 031510             9$:   ;CARRIAGE RETURN
3492 031412 031502             6$:   ;
3493 031414 031504             7$:   ;
3494 031416 031422             1$:   ;DIGIT 0-7
3495 031420 031422             1$:   ;DIGIT 8-9
3496 031422 062705 000004     1$:   ADD    #4,R5       ;STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
3497 031426 006303             2$:   ASL    R3          ;INPUT NUMBER *2
3498 031430 010346             MOV    R3,-(SP)    ;SAVE *2
3499 031432 006303             ASL    R3          ;*4
3500 031434 006303             ASL    R3          ;*8
3501 031436 062603             ADD    (SP)+,R3    ;(*2)+(*8) = *10
3502 031440 060203             ADD    R2,R3       ;UPDATE THE INPUT NUMBER
3503 031442 004537 031314     JSR    R5,CK.CHR    ;CHECK ONE CHARACTER
3504 031446 031506             8$:   ;ILLEGAL CHARACTER
3505 031450 031472             5$:   ;CARRIAGE RETURN
3506 031452 031470             4$:   ;
3507 031454 031462             3$:   ;
3508 031456 031426             2$:   ;DIGIT 0-7
3509 031460 031426             2$:   ;DIGIT 8-9
3510 031462 105711             3$:   TSTB   (R1)       ;DOES A 'CR' FOLLOW THE 'PERIOD'
3511 031464 001010             BNE   8$          ;BR IF NOT
3512 031466 005724             TST   (R4)+       ;INCREMENT THE RETURN
3513 031470 005724             4$:   TST   (R4)+       ;INCREMENT THE RETURN
3514 031472 005724             5$:   TST   (R4)+       ;INCREMENT THE RETURN
3515 031474 020316             CMP   R3,(SP)     ;CHECK THE MAGNITUDE OF THE NUMBER
3516 031476 101004             BHI   9$          ;BR IF ENTERED NUMBER TOO LARGE

```

```

3505 031500 000402          BR      8$          ;BYPASS INCREMENT
3506 031502 005725          6$:   TST      (R5)+      ;INCREMENT RETURN PAST INVALID RETURN
3507 031504 005725          7$:   TST      (R5)+      ;INCREMENT RETURN
3508 031506 060405          8$:   ADD      R4,R5      ;SETUP RETURN POINTER
3509 031510 010302          9$:   MOV      R3,R2      ;ENTERED VALUE
3510 031512 005726          TST      (SP)+      ;CLEAN MAX. SIZE OFF OF STACK
3511 031514 012603          MOV      (SP)+,R3      ;RESTORE R3
3512 031516 012604          MOV      (SP)+,R4      ;RESTORE R4
3513 031520 011505          MOV      (R5),R5      ;GET RETURN ADDRESS
3514 031522 000205          RTS      R5           ;RETURN
3515
3516          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINAPY NUMBER TO AN
3517          ;UNSIGNED DECIMAL ASCIZ NUMBER.
3518          ;CALL
3519          ;      MOV      NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
3520          ;      JSR      PC,$SB2D         ;CALL
3521          ;      RETURN                    ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
3522
3523          ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
3524          ;      THE SYSMAC LIBRARY, REV C AND LATER
3525
3526 031524 016637 000002 031550 $SB2D: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
3527 031532 012746 031550          MOV      #1$,-(SP)      ;SET THE POINTER
3528 031536 004737 035212          JSR      PC,$DB2D       ;CALL THE DOUBLE LENGTH CONVERT
3529 031542 012666 000002          MOV      (SP)+,2(SP)    ;PICKUP THE POINTER
3530 031546 000207          RTS      PC             ;RETURN
3531 031550 000000 000000          1$:   .WORD    0,0
3532
3533          ;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
3534          ;UNSIGNED OCTAL ASCIZ NUMBER.
3535          ;CALL
3536          ;      MOV      NUMBER,-(SP)      ;PUT THE NUMBER ON THE STACK
3537          ;      JSR      PC,$SB2D         ;CALL
3538          ;      RETURN                    ;ADDRESS OF THE 1ST ASCIZ CHAR IS ON THE STACK
3539
3540          ;NOTE: THE PROGRAM REQUIRES THIS FORM OF '$SB2D', NOT THE VERSION ON
3541          ;      THE SYSMAC LIBRARY, REV C AND LATER
3542
3543 031554 016637 000002 031600 $SB2D: MOV      2(SP),1$      ;SAVE THE BINARY NUMBER
3544 031562 012746 031600          MOV      #1$,-(SP)      ;SET THE POINTER
3545 031566 004737 035406          JSR      PC,$DB2D       ;CALL THE DOUBLE LENGTH CONVERT
3546 031572 012666 000002          MOV      (SP)+,2(SP)    ;PICKUP THE POINTER
3547 031576 000207          RTS      PC             ;RETURN
3548 031600 000000 000000          1$:   .WORD    0,0
3549
3550          ;KEYBOARD INTERRUPT INITIALIZATION ROUTINE
3551          ;CALL
3552          ;      JSR      PC,$TKINT        ;
3553          ;      RETURN
3554
3555 031604 012737 031634 000060 $TKINT: MOV      #$TKSRV,TKVEC ;SETUP VECTOR
3556 031612 012737 000100 000062          MOV      #100,TKVEC+2 ;PRIORITY TO 4
3557 031620 005777 147336          TST      @TKB          ;CLEAR THE BUFFER
3558 031624 012777 000100 147326          MOV      #BIT06,@TKS   ;SET INTERRUPT ENABLE
3559 031632 000207          RTS      PC             ;RETURN
3560
3561          ;KEYBOARD INTERRUPT SERVICE ROUTINE

```



```
3562 ;CALL
3563 ; ENTER VIA INTERRUPT
3564
3565 031634 104410 $TKSRV: RDCHR ;READ THE KEYBOARD
3566 031636 112637 031764 MOV (SP)+,5$ ;GET THE CHARACTER
3567 031642 023727 031764 000003 CMP 5$,#3 ;'CONTROL C' ?
3568 031650 001012 BNE 1$ ;BR IF NOT
3569 031652 104401 001203 TYPE ,SCRLF ;CR-LF
3570 031656 104401 032256 TYPE ,SCNTLC ;'AC'
3571 031662 012737 177777 001362 MOV #-1,CFLAG ;SET THE 'CONTROL C' FLAG
3572 031670 005077 147264 CLR @STKS ;CLEAR THE TTY INTERRUPT
3573 031674 000432 BR 4$ ;EXIT
3574 031676 023727 001154 000176 1$: CMP SWR,#SWREG ;SOFTWARE SWITCH REGISTER IN USE ?
3575 031704 001024 BNE 3$ ;BR IF NOT
3576 031706 023727 031764 000007 CMP 5$,#7 ;'CONTROL G' ?
3577 031714 001020 BNE 3$ ;BR IF NOT
3578 031716 104401 001203 TYPE ,SCRLF ;CR-LF
3579 031722 104401 034765 TYPE ,SCNTLG ;'AG'
3580 031726 013746 177776 MOV PS,-(SP) ;PUT THE STATUS WORD ON THE STACK
3581 031732 012746 031764 MOV #2$,-(SP) ;RETURN ADDRESS
3582 031736 005077 147216 CLR @STKS ;CLEAR THE TTY INTERRUPT ENABLE
3583 031742 000137 034426 JMP $GTSWR ;GET THE SWITCH REGISTER ENTRY
3584 031746 012777 000100 147204 2$: MOV #100,@STKS ;ENABLE TTY KEYBOARD INTERRUPT
3585 031754 000402 BR 4$ ;EXIT
3586 031756 104401 031764 3$: TYPE ,5$ ;ECHO THE CHARACTER
3587 031762 000002 4$: RTI ;RETURN
3588
3589 031764 000000 5$: .WORD 0 ;ENTERED CHARACTER
3590
3591 ;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3592 ;CALL:
3593 ; RDLIN ;:INPUT A STRING FROM THE TTY
3594 ; RETURN HERE ;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3595 ; ;TERMINATOR WILL BE A BYTE OF ALL 0'S
3596
3597 031766 010346 $RDLIN: MOV R3,-(SP) ;SAVE R3
3598 031770 005046 CLR -(SP) ;CLEAR THE RUBOUT KEY
3599 031772 012703 032244 1$: MOV #STTYIN,R3 ;GET ADDRESS
3600 031776 022703 032256 2$: CMP #STTYIN+10.,R3 ;BUFFER FULL?
3601 032002 101467 BLOS 4$ ;BR IF YES
3602 032004 104410 RDCHR ;GO READ ONE CHARACTER FROM THE TTY
3603 032006 112613 MOV (SP)+,(R3) ;GET CHARACTER
3604 032010 122713 000177 CMPB #177,(R3) ;IS IT A RUBOUT
3605 032014 001022 BNE 5$ ;BR IF NO
3606 032016 005716 TST (SP) ;IS THIS THE FIRST RUBOUT?
3607 032020 001007 BNE 6$ ;BR IF NO
3608 032022 112737 000134 032242 MOVB #' \ ,9$ ;TYPE A BACK SLASH
3609 032030 104401 032242 TYPE ,9$
3610 032034 012716 177777 MOV #-1,(SP) ;SET THE RUBOUT KEY
3611 032040 005303 6$: DEC R3 ;BACKUP BY ONE
3612 032042 020327 032244 CMP R3,#STTYIN ;STACK EMPTY?
3613 032046 103445 BLO 4$ ;BR IF YES
3614 032050 111337 032242 MOV (R3),9$ ;SET TO TYPEOUT THE DELETED CHAR.
3615 032054 104401 032242 TYPE ,9$ ;GO TYPE
3616 032060 000746 BR 2$ ;GO READ ANOTHER CHAR.
3617 032062 005716 5$: TST (SP) ;RUBOUT KEY SET?
3618 032064 001406 BEQ 7$ ;BR IF NO
```

```

3619 032066 112737 000134 032242      MOVB   #' \ ,9$      ;TYPE A BACK SLASH
3620 032074 104401 032242      TYPE   ,9$
3621 032100 005016          CLR    (SP)          ;CLEAR THE RUBOUT KEY
3622 032102 122713 000025      7$:   CMPB   #2 , (R3) ;IS CHARACTER A CTRL U?
3623 032106 001003          BNE   10$           ;BR IF NO
3624 032110 104401 034760      TYPE   , $CNTLU     ;TYPE A CONTROL 'U'
3625 032114 000726          BR    1$           ;GO START OVER
3626 032116 122713 000003      10$:  CMPB   #3 , (R3) ;IS CHARACTER A CTRL C ?
3627 032122 001006          BNE   8$           ;BR IF NOT
3628 032124 012737 177777 001362      MOV    #-1 ,CFLAG  ;SET CNTRL C FLAG
3629 032132 104401 032256      TYPE   , $CNTLC     ;ECHO IT
3630 032136 000427          BR    11$          ;EXIT
3631 032140 122713 000012      8$:   CMPB   #12 , (R3) ;IS CHARACTER A 'LF'?
3632 032144 001011          BNE   3$           ;BRANCH IF NO
3633 032146 105013          CLRB  (R3)         ;CLEAR THE CHARACTER
3634 032150 104401 001203      TYPE   , $CRLF     ;TYPE A 'CR' & 'LF'
3635 032154 104401 032244      TYPE   , $TTYIN    ;TYPE THE INPUT STRING
3636 032160 000706          BR    2$           ;GO PICKUP ANOTHER CHACTER
3637 032162 104401 001202      4$:   TYPE   , $QUES  ;TYPE A '?'
3638 032166 000701          BR    1$           ;CLEAR THE BUFFER AND LOOP
3639 032170 111337 032242      3$:   MOVB   (R3) ,9$ ;ECHO THE CHARACTER
3640 032174 104401 032242      TYPE   ,9$
3641 032200 122723 000015      CMPB   #15 , (R3)+ ;CHECK FOR RETURN
3642 032204 001274          BNE   2$           ;LOOP IF NOT RETURN
3643 032206 105063 177777      CLRB  -1 (R3)      ;CLEAR RETURN (THE 15)
3644 032212 104401 001204      TYPE   , $LF       ;TYPE A LINE FEED
3645 032216 005726          11$:  TST    (SP)+       ;CLEAN RUBOUT KEY FROM THE STACK
3646 032220 012603          MOV    (SP)+ ,R3   ;RESTORE R3
3647 032222 011646          MOV    (SP) , -(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
3648 032224 016666 000004 000002      MOV    4(SP) ,2(SP) ;FIRST ASCII CHARACTER ON IT
3649 032232 012766 032244 000004      MOV    #$TTYIN ,4(SP)
3650 032240 000002          RTI              ;RETURN
3651 032242      000          9$:   .BYTE   0          ;STORAGE FOR ASCII CHAR. TO TYPE
3652 032243      000          .BYTE   0          ;TERMINATOR
3653 032244          $TTYIN: .BLKB  10. ;RESERVE 10 BYTES FOR TTY INPUT
3654 032256      136      103      200 $CNTLC: .ASCIZ / ^C / <CRLF> ;CONTROL 'C'
3655
3656      .EVEN
3657
3658      .SBTTL  MACRO ROUTINES
3659
3668      .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
;*      ERR R  N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

032262
032262 104407
032264 010337 001344
$ERROR:
      CKSWR      ;;TEST FOR CHANGE IN SOFT-SW
      MOV    R3,ATTN ;SAVE THE ATTENTION REGISTER CONTENTS

```

```

032270 010137 001220      MOV      R1,DRIVE      ;DRIVE NUMBER
032274 032777 020000 146652  BIT      #SW13,@SWR    ;INHIBIT PRINTOUTS ?
032302 001004              BNE      .+12          ;BR IF YES
032304 104401 001203      TYPE    ,$CRLF        ;CR-LF
032310 004737 024364      JSR     PC,$TIME      ;TYPE THE TIME
032314 105237 001117      7$: INCB  $ERFLG      ;;SET THE ERROR FLAG
032320 001775              BEQ     7$            ;;DON'T LET THE FLAG GO TO ZERO
032322 013777 001116 146626  MOV     $TSTNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
032330 032777 002000 146616  BIT     ?BIT10,@SWR    ;BELL ON ERROR?
032336 001402              BEQ     1$            ;NO - SKIP
032340 104401 001176      TYPE    ,SBELL        ;RING BELL
032344 005237 001126      1$: INC  $ERTTL       ;COUNT THE NUMBER OF ERRORS
032350 C11637 001132      MOV     (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
032354 162737 000002 001132  SUB     #2,$ERRPC
032362 117737 146544 001130  MOVB   @ $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
032370 032777 020000 146556  BIT     #BIT13,@SWR   ;SKIP TYPEOUT IF SET
032376 001004              BNE     20$          ;SKIP TYPEOUTS
032400 004737 032452      JSR     PC,$ERRTYP   ;GO TO USER ERROR ROUTINE
032404 104401 001203      TYPE    ,CRLF
032410              20$: CMPB   #APTENV,$ENV    ;RUNNING IN APT MODE
032410 122737 000001 001226  BNE     2$            ;NO,SKIP APT ERROR REPORT
032416 001007              MOVB   $ITEMB,21$   ;SET ITEM NUMBER AS ERROR NUMBER
032420 113737 001130 032432  JSR     FC,$ATY4     ;REPORT FATAL ERROR TO APT
032426 004737 033160      21$: .BYTE 0
032432 000              .BYTE 0
032433 000              BR     22$          ;APT ERROR LOOP
032434 000777              2$: TST  @SWR          ;HALT ON ERROR
032436 005777 146512      BPL    3$            ;SKIP IF CONTINUE
032442 100002              HALT   ;HALT ON ERROR!
032444 000000              CKSWR ;TEST FOR CHANGE IN SOFT-SWR
032446 104407              3$: RTI      ;RETURN
032450              ;
032450 000002

```

3669
3670

.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.

```

```

032452              $ERRTYP:
032452 104401 001203      TYPE    ,CRLF        ;'CARRIAGE RETURN' & 'LINE FEED'
032456 010046      MOV     R0,-(SP)    ;SAVE R0
032460 005000      CLR     R0          ;PICKUP THE ITEM INDEX
032462 153700 001130      BISB   @#$ITEMB,R0
032466 001004      BNE     1$            ;IF ITEM NUMBER IS ZERO, JUST
                                ;TYPE THE PC OF THE ERROR
032470 C13746 001132      MOV     $ERRPC,-(SP) ;SAVE $ERRPC FOR TYPEOUT
                                ;ERROR ADDRESS
032474 104402              TYPCC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
032476 000426      BR     6$            ;GET OUT
032500 005300      1$: DEC  R0          ;ADJUST THE INDEX SO THAT IT WILL
032502 006300      ASL   R0          ; WORK FOR THE ERROR TABLE
032504 006300      ASL   R0
032506 006300      ASL   R0
032510 062700 003546      ADD   # $ERRTB,R0   ;FORM TABLE POINTER

```

```
032514 012037 032524      MOV      (R0)+,2$      ;;PICKUP 'ERROR MESSAGE' POINTER
032520 001404              BEQ      3$            ;;SKIP TYPEOUT IF NO POINTER
032522 104401              TYPE                    ;;TYPE THE 'ERROR MESSAGE'
032524 000000      2$:  .WORD      0      ;;'ERROR MESSAGE' POINTER GOES HERE
032526 104401 001203      TYPE      ,SCLF        ;;'CARRIAGE RETURN' & 'LINE FEED'
032532 012037 032542      3$:  MOV      (R0)+,4$      ;;PICKUP 'DATA HEADER' POINTER
032536 001404              BEQ      5$            ;;SKIP TYPEOUT IF 0
032540 104401              TYPE                    ;;TYPE THE 'DATA HEADER'
032542 000000      4$:  .WORD      0      ;;'DATA HEADER' POINTER GOES HERE
032544 104401 001203      TYPE      ,SCLF        ;;'CARRIAGE RETURN' & 'LINE FEED'
032550 011000      5$:  MOV      (R0),R0      ;;PICKUP 'DATA TABLE' POINTER
032552 001004              BNE      7$            ;;GO TYPE THE DATA
032554 012600      6$:  MOV      (SP)+,R0      ;;RESTORE R0
032556 104401 001203      TYPE      ,SCLF        ;;'CARRIAGE RETURN' & 'LINE FEED'
032562 000207              RTS      PC            ;;RETURN
032564
032564 013046      MOV      @ (R0)+,-(SP)  ;;SAVE @ (R0)+ FOR TYPEOUT
032566 104402      TYPOC                    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
032570 005710      TST      (R0)          ;;IS THERE ANOTHER NUMBER?
032572 001770      BEQ      6$            ;;BR IF NO
032574 104401 032602      TYPE      ,8$          ;;TYPE TWO(2) SPACES
032600 000771      BR      7$            ;;LOOP
032602      040      040      000 8$:  .ASCIZ  / /          ;;TWO(2) SPACES
      .EVEN
```

3671
3672

.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
;*
```

```
032606 105737 001173  $TYPE:  TSTB      $TPFLG      ;;IS THERE A TERMINAL?
032612 100002              BPL      1$            ;;BR IF YES
032614 000000              HALT                    ;;HALT HERE IF NO TERMINAL
032616 000430              BR      3$            ;;LEAVE
032620 010046      1$:  MOV      R0,-(SP)      ;;SAVE R0
032622 017600 000002      MOV      @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
032626 122737 000001 001226  CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
032634 001011              BNE      62$          ;;NO,GO CHECK FOR APT CONSOLE
032636 132737 000100 001227  BITB      #APTSPOOL,$ENVM    ;;SPOOL MESSAGE TO APT
032644 001405              BEQ      62$          ;;NO,GO CHECK FOR CONSOLE
032646 010037 032656      MOV      R0,61$        ;;SETUP MESSAGE ADDRESS FOR APT
032652 004737 033150      JSR      PC,$ATY3      ;;SPOOL MESSAGE TO APT
032656 000000      61$:  .WORD      0      ;;MESSAGE ADDRESS
032660 132737 000040 001227 62$:  BITB      #APTCSUP,$ENVM    ;;APT CONSOLE SUPPRESSED
032666 001003              BNE      60$          ;;YES,SKIP TYPE OUT
```

```

032670 112046      2$:   MOVB   (R0)+,-(SP)   ;;PUSH CHARACTER TO BE TYPED ONTO STACK
032672 001005      BNE     4$                ;;BR IF IT ISN'T THE TERMINATOR
032674 005726      TST     (SP)+            ;;IF TERMINATOR POP IT OFF THE STACK
032676 012600      60$:   MOV     (SP)+,R0    ;;RESTORE R0
032700 062716      3$:   ADD     #2,(SP)        ;;ADJUST RETURN PC
032704 000902      RTI                    ;;RETURN
032706 122716      4$:   CMPB   #HT,(SP)       ;;BRANCH IF <HT>
032712 001430      BEQ     8$                ;;BRANCH IF NOT <CRLF>
032714 122716      000200  CMPB   #CRLF,(SP)
032720 001006      BNE     5$                ;;POP <CR><LF> EQUIV
032722 005726      TST     (SP)+            ;;TYPE A CR AND LF
032724 104401      TYPE
032726 001203      $CRLF
032730 105037      033136  CLRB   $CHARCNT        ;;CLEAR CHARACTER COUNT
032734 000755      BR     2$                ;;GET NEXT CHARACTER
032736 004737      033020  JSR   PC,$TYPEC        ;;GO TYPE THIS CHARACTER
032742 123726      001172  6$:   CMPB   $FILLC,(SP)+   ;;IS IT TIME FOR FILLER CHARS.?
032746 001350      BNE     2$                ;;IF NO GO GET NEXT CHAR.
032750 013746      001170  MOV   $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
032754 105366      000001  7$:   =CB   1(SP)          ;;DOES A NULL NEED TO BE TYPED?
032760 002770      BPL    6$                ;;BR IF NO--GO POP THE NULL OFF OF STACK
032762 004737      033020  JSR   PC,$TYPEC        ;;GO TYPE A NULL
032766 105337      033136  DECB  $CHARCNT        ;;DO NOT COUNT AS A COUNT
032772 000770      BR     7$                ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

032774 112716      000040  8$:   MOVB   #' ,(SP)      ;;REPLACE TAB WITH SPACE
033000 004737      033020  9$:   JSR   PC,'TYPEC      ;;TYPE A SPACE
033004 132737      000007  033136  BITB  #7,$CHARCNT    ;;BRANCH IF NOT AT
033012 001372      BNE     9$                ;;TAB STOP
033014 005726      TST     (SP)+            ;;POP SPACE OFF STACK
033016 000724      BR     2$                ;;GET NEXT CHARACTER
033020
033020 105777      146134  $TYPEC: TSTB  @STKS            ;;CHAR IN KYBD BUFFER?
033024 100022      BPL    10$               ;;BR IF NOT
033026 017746      146130  MOV   @STKB,-(SP)       ;;GET CHAR
033032 042716      177600  BIC   #177600,(SP)     ;;STRIP EXTRANEIOUS BITS
033036 122716      000023  CMPB  #$XOFF,(SP)     ;;WAS CHAR XOFF
033042 001012      BNE     102$            ;;BR IF NOT
033044
033044 105777      146110  101$: TSTB  @STKS            ;;WAIT FOR CHAR
033050 100375      BPL    101$             ;;GET CHAR
033052 117716      146104  MOVB  @STKB,(SP)       ;;STRIP IT
033056 042716      177600  BIC   #177600,(SP)     ;;WAS IT XON?
033062 122716      000021  CMPB  #$XON,(SP)     ;;BR IF NOT
033066 001366      BNE     101$
033070
033070 005726      102$: TST   (SP)+            ;;FIX STACK
033072
033072 105777      146066  10$:  TSTB  @STPS            ;;WAIT UNTIL PRINTER IS READY
033076 100375      BPL    10$
033100 116677      000002  146060  MOVB  2(SP),@STPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
033106 122766      000015  000002  CMPB  #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
033114 001003      BNE     1$                ;;BRANCH IF NO
033116 105037      033136  CLRB  $CHARCNT        ;;YES--CLEAR CHARACTER COUNT

```

```

033122 000406          BR      $TYPEX      ;;EXIT
033124 122766 000012 000002 1$:  CMPB  #LF,2(SP)  ;;IS CHARACTER A LINE FEED?
033132 001402          BEQ    $TYPEX      ;;BRANCH IF YES
033134 105227          INCB  (PC)+      ;;COUNT THE CHARACTER
033136 000000          $CHARCNT: .WORD 0  ;;CHARACTER COUNT STORAGE
033140 000207          $TYPEX: RTS   PC

```

3673
3674

.SBTTL APT COMMUNICATIONS ROUTINE

```

;:*****
033142 112737 000001 033406 $ATY1: MOVB  #1,$FFLG      ;;TO REPORT FATAL ERROR
033150 112737 000001 033404 $ATY3: M.VB  #1,$MFLG      ;;TO TYPE A MESSAGE
033156 000403          B^P   $ATYC
033160 112737 000001 033406 $ATY4: MOVB  #1,$FFLG      ;;TO ONLY REPORT FATAL ERROR
033166 010046          $ATYC:
033166 010046          MOV   R0,-(SP)      ;;PUSH R0 ON STACK
033170 010146          MOV   R1,-(SP)      ;;PUSH R1 ON STACK
033172 105737 033404          TSTB $MFLG      ;;SHOULD TYPE A MESSAGE?
033176 001450          BEQ   5$           ;;IF NOT: BR
033200 122737 000001 001226 CMPB  #APTENV,$ENV      ;;OPERATING UNDER APT?
033206 001031          BNE   3$           ;;IF NOT: BR
033210 132737 000100 001227 BITB  #APTSPOOL,$ENVM  ;;SHOULD SPOOL MESSAGES?
033216 001425          BEQ   3$           ;;IF NOT: BR
033220 017600 000004          MOV   @4(SP),R0      ;;GET MESSAGE ADDR.
033224 062766 000002 000004 ADD   #2,4(SP)          ;;BUMP RETURN ADDR.
033232 005737 001206          1$: TST  $MSGTYPE      ;;SEE IF DONE W/ LAST XMISSION?
033236 001375          BNE   1$           ;;IF NOT: WAIT
033240 010037 001222          MOV   R0,$MSGAD      ;;PUT ADDR IN MAILBOX
033244 105720          2$: TSTB (R0)+      ;;FIND END OF MESSAGE
033246 001376          BNE   2$           ;;
033250 163700 001222          SUB   $MSGAD,R0      ;;SUB START OF MESSAGE
033254 006200          ASR   R0           ;;GET MESSAGE LNTH IN WORDS
033256 010037 001224          MOV   R0,$MSGGLT      ;;PUT LENGTH IN MAILBOX
033262 012737 000004 001206 MOV   #4,$MSGTYPE      ;;TELL APT TO TAKE MSG.
033270 000413          BR    5$
033272 017637 000004 033316 3$: MOV   @4(SP),4$      ;;PUT MSG ADDR IN JSR LINKAGE
033300 062766 000002 000004 ADD   #2,4(SP)          ;;BUMP RETURN ADDRESS
033306 013746 177776          MOV   177776,-(SP)    ;;PUSH 177776 ON STACK
033312 004737 032606          JSR  PC,$TYPE      ;;CALL TYPE MACRO
033316 000000          4$: .WORD 0
033320          5$:
033320 105737 033406          10$: TSTB $FFLG      ;;SHOULD REPORT FATAL ERROR?
033324 001416          BEQ   12$          ;;IF NOT: BR
033326 005737 001226          TST  $ENV          ;;RUNNING UNDER APT?
033332 001413          BEQ   12$          ;;IF NOT: BR
033334 005737 001206          11$: TST  $MSGTYPE      ;;FINISHED LAST MESSAGE?
033340 001375          BNE   11$          ;;IF NOT: WAIT
033342 017637 000004 001210 MOV   @4(SP),$FATAL    ;;GET ERROR #
033350 062766 000002 000004 ADD   #2,4(SP)          ;;BUMP RETURN ADDR.
033356 005237 001206          INC  $MSGTYPE      ;;TELL APT TO TAKE ERROR
033362 105037 033406          12$: CLRB $FFLG      ;;CLEAR FATAL FLAG
033366 105037 033405          CLRB $LFLG      ;;CLEAR LOG FLAG
033372 105037 033404          CLRB $MFLG      ;;CLEAR MESSAGE FLAG
033376 012601          MOV   (SP)+,R1      ;;POP STACK INTO R1
033400 012600          MOV   (SP)+,R0      ;;POP STACK INTO R0
033402 000207          RTS   PC          ;;RETURN

```

033404 000
033405 000
033406 000

\$MFLG: .BYTE 0 ;:MESSG. FLAG
\$LFLG: .BYTE 0 ;:LOG FLAG
\$FFLG: .BYTE 0 ;:FATAL FLAG

000200
000001
000100
000040

.EVEN
APTSIZE = 200
APTENV = 001
APTSPOOL = 100
APTCSUP = 040

3675
3676

.SBTTL POWER DOWN AND UP ROUTINES

:POWER DOWN ROUTINE

033410	012737	033554	000024	\$PWRDN: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST UP
033416	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
033424	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
033426	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
033430	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
033432	010346			MOV	R3,-(SP)	::PUSH R3 ON STACK
033434	010446			MOV	R4,-(SP)	::PUSH R4 ON STACK
033436	010546			MOV	R5,-(SP)	::PUSH R5 ON STACK
033440	017746	145510		MOV	@SWR,-(SP)	::PUSH @SWR ON STACK
033444	010637	033560		MOV	SP,\$SAVR6	::SAVE SP
033450	012737	033462	000024	MOV	#\$PWRUP,@#PWRVEC	::SET UP VECTOR
033456	000000			HALT		
033460	000776			BR	.-2	::HANG UP

:POWER UP ROUTINE

033462	012737	033554	000024	\$PWRUP: MOV	#\$ILLUP,@#PWRVEC	::SET FOR FAST DOWN
033470	013706	033560		MOV	\$SAVR6,SP	::GET SP
033474	005037	033560		CLR	\$SAVR6	::WAIT LOOP FOR THE TTY
033500	005237	033560		1\$: INC	\$SAVR6	::WAIT FOR THE INC
033504	001375			BNE	1\$::OF WORD
033506	012677	145442		MOV	(SP)+,@SWR	::POP STACK INTO @SWR
033512	012605			MOV	(SP)+,R5	::POP STACK INTO R5
033514	012604			MOV	(SP)+,R4	::POP STACK INTO R4
033516	012603			MOV	(SP)+,R3	::POP STACK INTO R3
033520	012602			MOV	(SP)+,R2	::POP STACK INTO R2
033522	012601			MOV	(SP)+,R1	::POP STACK INTO R1
033524	012600			MOV	(SP)+,R0	::POP STACK INTO R0
033526	012737	033410	000024	MOV	#\$PWRDN,@#PWRVEC	::SET UP THE POWER DOWN VECTOR
033534	012737	000340	000026	MOV	#340,@#PWRVEC+2	::PRIO:7
033542	104401			TYPE		::REPORT THE POWER FAILURE
033544	033562			\$PWRMG: .WORD	\$POWER	::POWER FAIL MESSAGE COUNTER
033546	012716			MOV	(PC)+,(SP)	::RESTART AT SATPOW
033550	033572			\$PWRAD: .WORD	SATPOW	::RESTART ADDRESS
033552	000002			RTI		
033554	000000			\$ILLUP: HALT		::THE POWER UP SEQUENCE WAS STARTED
033556	000776			BR	.-2	::BEFORE THE POWER DOWN WAS COMPLETE
033560	000000			\$SAVR6: 0		::PUT THE SP HERE
033562	015	012	120	\$PCWER: .ASCIZ	<15><12>'POWER'	
				.EVEN		

3677
3678
3679
3680

:POW UP ROUTINE , WAIT
:FIVE MINUTS, THEN AUTO STARTS AT SETVEC

```

3681 033572 000005          SATPOW: RESET          ;CLEAR THE BUS
3682 033574 005037 177776    CLR @#PS             ;CLEAR PSW
3683 033600 005037 001366    CLR HOUR            ;RESET THE HOUR COUNT
3684 033604 005037 001370    CLR MINUTE          ;RESET THE MINUTS COUNT
3685 033610 005037 001372    CLR SECOND          ;RESET THE SECOND COUNT
3686 033614 005037 001474    CLR INTRVL+2        ;RESET THE INTERVAL COUNT
3687 033620 004737 023326    JSR PC,CKCLK        ;CHECK THE CLOCK
3688 033624 022737 000170 001474 1$:  CMP #120.,INTRVL+2  ;FIVE MINUTES YET ?
3689 033632 101374          BHI 1$              ;WAIT IF NOT
3690 033634 005037 001474    CLR INTRVL+2        ;RESET INTERVAL COUNT
3691 033640 005037 001372    CLR SECOND          ;RESET TIMER
3692 033644 005037 001370    CLR MINUTE          ;
3693 033650 012737 000400 001360  MOV #400,CHGADR      ;FORGE THE AUTO START
3694 033656 012705 001526    MOV #ORDERQ,R5      ;CLEAR UP THE QUEUE AND BUFFER
3695 033662 005025          CLR (R5)+           ;
3696 033664 022705 002106    CMP #BLKADR,R5      ;ALL DONE ?
3697 033670 001374          BNE 2$              ;BRANCH IF NOT
3698 033672 012737 007070 001266  MOV #7070,$CDW1     ;FLAG FROM POWER FAIL
3699 033700 000137 005054    JMP SIZMEM          ;LOOP BACK

```

3700
3701

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

033704 017646 000000          $TYPOS: MOV @ (SP),-(SP)      ;;PICKUP THE MODE
033710 116637 000001 034127    MOVB 1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
033716 112637 034131          MOVB (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
033722 062716 000002          ADD #2,(SP)         ;;ADJUST RETURN ADDRESS
033726 000406          BR $TYPON
033730 112737 000001 034127    $TYPOC: MOVB #1,$OFILL  ;;SET THE ZERO FILL SWITCH
033736 112737 000006 034131    MOVB #6,$OMODE+1    ;;SET FOR SIX(6) DIGITS
033744 112737 000005 034126    $TYPON: MOVB #5,$OCNT  ;;SET THE ITERATION COUNT
033752 010346          MOV R3,-(SP)        ;;SAVE R3
033754 010446          MOV R4,-(SP)        ;;SAVE R4
033756 010546          MOV R5,-(SP)        ;;SAVE R5
033760 113704 034131          MOVB $OMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE

```



```

033764 005404          NEG      R4
033766 062704 000006  ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
033772 110437 034130  MOVVB   R4,$OMODE      ;;SAVE IT FOR USE
033776 113704 034127  MOVVB   $OFILL,R4     ;;GET THE ZERO FILL SWITCH
034002 016605 000012  MOV      12(SP),R5    ;;PICKUP THE INPUT NUMBER
034006 005003          CLR      R3          ;;CLEAR THE OUTPUT WORD
034010 006105          1$:  ROL      R5          ;;ROTATE MSB INTO 'C'
034012 000404          BR       3$          ;;GO DO MSB
034014 006105          2$:  ROL      R5          ;;FORM THIS DIGIT
034016 006105          ROL      R5
034020 006105          ROL      R5
034022 010503          MOV      R5,R3
034024 006103          3$:  ROL      R3          ;;GET LSB OF THIS DIGIT
034026 105337 034130  DECB    $OMODE        ;;TYPE THIS DIGIT?
034032 100016          BPL     7$          ;;BR IF NO
034034 042703 177770  BIC     #177770,R3    ;;GET RID OF JUNK
034040 001002          BNE     4$          ;;TEST FOR 0
034042 005704          TST     R4          ;;SUPPRESS THIS 0?
034044 001403          BEQ     5$          ;;BR IF YES
034046 005204          4$:  INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
034050 052703 000060  BIS     #'0,R3        ;;MAKE THIS DIGIT ASCII
034054 052703 000040  5$:  BIS     #' ,R3        ;;MAKE ASCII IF NOT ALREADY
034060 110337 034124  MOVVB   R3,8$         ;;SAVE FOR TYPING
034064 104401 034124  TYPE    ,8$          ;;GO TYPE THIS DIGIT
034070 105337 034126  7$:  DECB    $OCNT      ;;COUNT BY 1
034074 003347          BGT     2$          ;;BR IF MORE TO DO
034076 002402          BLT     6$          ;;BR IF DONE
034100 005204          INC     R4          ;;INSURE LAST DIGIT ISN'T A BLANK
034102 000744          BR      2$          ;;GO DO THE LAST DIGIT
034104 012605          6$:  MOV     (SP)+,R5    ;;RESTORE R5
034106 012604          MOV     (SP)+,R4    ;;RESTORE R4
034110 012603          MOV     (SP)+,R3    ;;RESTORE R3
034112 016666 000002 000004  MOV     2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
034120 012616          MOV     (SP)+,(SP)
034122 000002          RTI
034124 000          8$:  .BYTE   0          ;;RETURN
034125 000          .BYTE   0          ;;STORAGE FOR ASCII DIGIT
034126 000          $OCNT: .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
034127 000          $OFILL: .BYTE   0          ;;OCTAL DIGIT COUNTER
034130 000000          $OMODE: .WORD   0          ;;ZERO FILL SWITCH
                                ;;NUMBER OF DIGITS TO TYPE

```

3702
3703

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

```

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

```

```

034132          $TYPDS:
034132 010046          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
034134 010146          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
034136 010246          MOV     R2,-(SP)        ;;PUSH R2 ON STACK

```

```

034140 010346          MOV     R3,-(SP)      ;;PUSH R3 ON STACK
034142 010546          MOV     R5,-(SP)      ;;PUSH R5 ON STACK
034144 012746 020200    MOV     #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
034150 016605 000020    MOV     20(SP),R5     ;;GET THE INPUT NUMBER
034154 100004          BPL     1$           ;;BR IF INPUT IS POS.
034156 005405          NEG     R5           ;;MAKE THE BINARY NUMBER POS.
034160 112766 000055 000001  MOVB   #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
034166 005000          CLR     R0           ;;ZERO THE CONSTANTS INDEX
034170 012703 034346    1$:    MOV     #$DBLK,R3    ;;SETUP THE OUTPUT POINTER
034174 112723 000040    MOVB   #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
034200 005002          CLR     R2           ;;CLEAR THE BCD NUMBER
034202 016001 034336    2$:    MOV     $DTBL(R0),R1 ;;GET THE CONSTANT
034206 160105          3$:    SUB     R1,R5        ;;FORM THIS BCD DIGIT
034210 002402          BLT     4$           ;;BR IF DONE
034212 005202          INC     R2           ;;INCREASE THE BCD DIGIT BY 1
034214 000774          BR      3$
034216 060105          4$:    ADD     R1,R5        ;;ADD BACK THE CONSTANT
034220 005702          TST     R2           ;;CHECK IF BCD DIGIT=0
034222 001002          BNE     5$           ;;FALL THROUGH IF 0
034224 105716          TSTB   (SP)         ;;STILL DOING LEADING 0'S?
034226 100407          BMI     7$           ;;BR IF YES
034230 106316          5$:    ASLB   (SP)         ;;MSD?
034232 103003          BCC     6$           ;;BR IF NO
034234 116663 000001 177777  MOVB   1(SP),-1(R3)   ;;YES--SET THE SIGN
034242 052702 000060    6$:    BIS     #'0,R2     ;;MAKE THE BCD DIGIT ASCII
034246 052702 000040    7$:    BIS     #' ,R2     ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
034252 110223          MOVB   R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
034254 005720          TST     (R0)+       ;;JUST INCREMENTING
034256 020027 000010    CMP     R0,#10      ;;CHECK THE TABLE INDEX
034262 002746          BLT     2$           ;;GO DO THE NEXT DIGIT
034264 003002          BGT     8$           ;;GO TO EXIT
034266 010502          MOV     R5,R2       ;;GET THE LSD
034270 000764          BR      6$           ;;GO CHANGE TO ASCII
034272 105726          8$:    TSTB   (SP)+     ;;WAS THE LSD THE FIRST NON-ZERO?
034274 100003          BPL     9$           ;;BR IF NO
034276 116663 177777 177776  MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
034304 105013          9$:    CLRB   (R3)       ;;SET THE TERMINATOR
034306 012605          MOV     (SP)+,R5    ;;POP STACK INTO R5
034310 012603          MOV     (SP)+,R3    ;;POP STACK INTO R3
034312 012602          MOV     (SP)+,R2    ;;POP STACK INTO R2
034314 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
034316 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
034320 104401 034346    TYPE   $DBLK        ;;NOW TYPE THE NUMBER
034324 016666 000002 000004  MOV     2(SP),4(SP)  ;;ADJUST THE STACK
034332 012616          MOV     (SP)+,(SP)
034334 000002          PTI
034336 023420          $DTBL: 1000.       ;;RETURN TO USER
034340 001750          1000.
034342 000144          100.
034344 000012          10.
034346          $DBLK: .BLKW 4
    
```

3704
3705

.SBTTL TTY INPUT ROUTINE

 .ENABL LSB

```

*****
*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 FLAG MODE.
034356 022737 000176 001154 $CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
034364 001074 BNE 15$ ;;BRANCH IF NO
034366 105777 144566 TSTB @STKS ;;CHAR THERE?
034372 100071 BPL 15$ ;;IF NO, DON'T WAIT AROUND
034374 117746 144562 MOVB @STKB,-(SP) ;;SAVE THE CHAR
034400 042716 177600 BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
034404 022726 000007 CMP #7,(SP)+ ;;IS IT A CONTROL G?
034410 001062 BNE 15$ ;;NO, RETURN TO USER
034412 123727 001150 000001 CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
034420 001456 BEQ 15$ ;;BRANCH IF YES

034422 104401 034765 $GTSWR: TYPE , $CNTLG ;;ECHO THE CONTROL-G (^G)
034426 104401 034772 TYPE , $MSWR ;;TYPE CURRENT CONTENTS
034432 013746 000176 MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
034436 104402 TYPOC ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034440 104401 035003 TYPE , $MNEW ;;PROMPT FOR NEW SWR
034444 005046 19$: CLR -(SP) ;;CLEAR COUNTER
034446 005046 CLR -(SP) ;;THE NEW SWR
034450 105777 144504 7$: TSTB @STKS ;;CHAR THERE?
034454 100375 BPL 7$ ;;IF NOT TRY AGAIN

034456 117746 144500 MOVB @STKB,-(SP) ;;PICK UP CHAR
034462 042716 177600 BIC #^C177,(SP) ;;MAKE IT 7-BIT ASCII

034466 021627 000025 9$: CMP (SP),#25 ;;IS IT A CONTROL-U?
034472 001005 BNE 10$ ;;BRANCH IF NOT
034474 104401 034760 TYPE , $CNTLU ;;YES, ECHO CONTROL-U (^U)
034500 062706 000006 20$: ADD #6,SP ;;IGNORE PREVIOUS INPUT
034504 000757 BR 19$ ;;LET'S TRY IT AGAIN

034506 021627 000015 10$: CMP (SP),#15 ;;IS IT A <CR>?
034512 001022 BNE 16$ ;;BRANCH IF NO
034514 005766 000004 TST 4(SP) ;;YES, IS IT THE FIRST CHAR?
034520 001403 BEQ 11$ ;;BRANCH IF YES
034522 016677 000002 144424 MOV 2(SP),@SWR ;;SAVE NEW SWR
034530 062706 000006 11$: ADD #6,SP ;;CLEAR UP STACK
034534 104401 001203 14$: TYPE , $CRLF ;;ECHO <CR> AND <LF>
034540 123727 001151 000001 CMPB $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
034546 001003 BNE 15$ ;;BRANCH IF NOT
034550 012777 000100 144402 MOV #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
034556 000002 15$: RTI ;;RETURN
034560 004737 033020 16$: JSR PC,$TYPEC ;;ECHO CHAR
034564 021627 000060 CMP (SP),#60 ;;CHAR < 0?
034570 002420 BLT 18$ ;;BRANCH IF YES
034572 021627 000067 CMP (SP),#67 ;;CHAR > 7?
034576 003015 BGT 18$ ;;BRANCH IF YES
034600 042726 000060 BIC #60,(SP)+ ;;STRIP-OFF ASCII
034604 005766 000002 TST 2(SP) ;;IS THIS THE FIRST CHAR
034610 001403 BEQ 17$ ;;BRANCH IF YES

```

034612	006316		
034614	006316		
034616	006316		
034620	005266	000002	
034624	056616	177776	
034630	000707		
034632	104401	001202	
034636	000720		

```

ASL      (SP)          ;;NO, SHIFT PRESENT
ASL      (SP)          ;;  CHAR OVER TO MAKE
ASL      (SP)          ;;  ROOM FOR NEW ONE.
17$: INC      2(SP)      ;;KEEP COUNT OF CHAR
BIS      -2(SP),(SP)    ;;SET IN NEW CHAR
BR       7$            ;;GET THE NEXT ONE
18$: TYPE   $QUES      ;;TYPE ?<CR><LF>
BR       20$          ;;SIMULATE CONTROL-U
.DSABL   LSB

```

```

*****
*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
*CALL:

```

```

* RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE    ;;CHARACTER IS ON THE STACK
*               ;;WITH PARITY BIT STRIPPED OFF

```

034640	011646		
034642	016666	000004	000002
034650	105777	144304	
034654	100375		
034656	117766	144300	000004
034664	042766	177600	000004
034672	026627	000004	000023
034700	001013		
034702	105777	144252	
034706	100375		
034710	117746	144246	
034714	042716	177600	
034720	022627	000021	
034724	001366		
034726	000750		
034730	026627	000004	000140
034736	002407		
034740	026627	000004	000175
034746	003003		
034750	042766	000040	000004
034756	000002		
034760	136	125	015
034765	136	107	015
034772	015	012	123
035003	040	040	116

```

$RDCHR: MOV      (SP),-(SP)    ;;PUSH DOWN THE PC
MOV      4(SP),2(SP)         ;;SAVE THE PS
1$: TSTB   @STKS             ;;WAIT FOR
BPL      1$                  ;;A CHARACTER
MOV      @STKB,4(SP)         ;;READ THE TTY
BIC      #^C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
CMP      4(SP),#23           ;;IS IT A CONTROL-S?
BNE      3$                  ;;BRANCH IF NO
2$: TSTB   @STKS             ;;WAIT FOR A CHARACTER
BPL      2$                  ;;LOOP UNTIL 'S THERE
MOV      @STKB,-(SP)         ;;GET CHARACTER
BIC      #^C<177>,(SP)       ;;MAKE IT 7-BIT ASCII
CMP      (SP)+,#21           ;;IS IT A CONTROL-Q?
BNE      2$                  ;;IF NOT DISCARD IT
BR       1$                  ;;YES, RESUME
3$: CMP    4(SP),#140         ;;IS IT UPPER CASE?
BLT      4$                  ;;BRANCH IF YES
CMP      4(SP),#175         ;;IS IT A SPECIAL CHAR?
BGT      4$                  ;;BRANCH IF YES
BIC      #40,4(SP)          ;;MAKE IT UPPER CASE
4$: RTI                      ;;GO BACK TO USER
$CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
$CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
$MSWR:  .ASCIZ  <15><12>/SWR = /
$MNEW:  .ASCIZ  / NEW = /

```

3706
3707

.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

```

035014

\$RAND:

```

035014 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
035016 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
035020 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
035022 013700 035114  MOV     $LONUM,R0     ;;SET R0 WITH LOW
035026 013701 035112  MOV     $SHINUM,R1    ;;SET R1 WITH HIGH
035032 012702 177771  MOV     #-7,R2       ;;SET SHIFT COUNT
035036 006300      1$: ASL     R0         ;;SHIFT R0 LEFT AND
035040 006101      ROL     R1         ;;ROTATE CARRY INTO R1 AND
035042 005202      INC     R2         ;;CHECK FOR DONE
035044 001374      BNE     1$        ;;CONTINUE SHIFT LOOP
035046 063700 035114  ADD     $LONUM,R0     ;;ADD NUMBER TO MAKE X 129
035052 005501      ADC     R1         ;;PROPOGATE CARRY
035054 063701 035112  ADD     $SHINUM,R1    ;;ADD NUMBER TO MAKE X 129
035060 062700 001057  ADD     #1057,R0     ;;ADD LOW CONSTANT
035064 005501      ADC     R1         ;;PROPOGATE CARRY
035066 062701 047401  ADD     #47401,R1    ;;ADD HIGH CONSTANT
035072 010037 035114  MOV     R0,$LONUM    ;;SAVE R0
035076 010137 035112  MOV     R1,$SHINUM   ;;SAVE R1
035102 012602      MOV     (SP)+,R2    ;;POP STACK INTO R2
035104 012601      MOV     (SP)+,R1    ;;POP STACK INTO R1
035106 012600      MOV     (SP)+,R0    ;;POP STACK INTO R0
035110 000207      RTS     PC         ;;RETURN
035112 176543      $SHINUM: .WORD 176543
035114 123456      $LONUM:  .WORD 123456

```

3708
3709

.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

```

```

035116      $SAVREG:
035116 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
035120 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
035122 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
035124 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
035126 010446      MOV     R4,-(SP)      ;;PUSH R4 ON STACK
035130 010546      MOV     R5,-(SP)      ;;PUSH R5 ON STACK
035132 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
035136 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
035142 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PS OF CALL
035146 016646 000022  MOV     22(SP),-(SP)  ;;SAVE PC OF CALL
035152 000002      RTI

```

```

;*RESTORE R0-R5
;*CALL:

```

```

    035154          012666 000022          ;* RESREG
    035154 012666 000022 $RESREG:      MOV (SP)+,22(SP)  ;;RESTORE PC OF CALL
    035160 012666 000022                MOV (SP)+,22(SP)  ;;RESTORE PS OF CALL
    035164 012666 000022                MOV (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
    035170 012666 000022                MOV (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
    035174 012605                MOV (SP)+,R5        ;;POP STACK INTO R5
    035176 012604                MOV (SP)+,R4        ;;POP STACK INTO R4
    035200 012603                MOV (SP)+,R3        ;;POP STACK INTO R3
    035202 012602                MOV (SP)+,R2        ;;POP STACK INTO R2
    035204 012601                MOV (SP)+,R1        ;;POP STACK INTO R1
    035206 012600                MOV (SP)+,R0        ;;POP STACK INTO R0
    035210 000002                RTI
    
```

3710
 3711

.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
    
```

```

    035212 104412          $DB2D: SAVREG  ;;SAVE REGISTERS
    035214 016602 000002  MOV 2(SP),R2  ;;PICKUP THE DATA POINTER
    035220 012700 035372  MOV #SDECLV,R0  ;;GET ADDRESS OF 'SDECLV' STRING
    035224 010066 000002  MOV R0,2(SP)  ;;PUT ADDRESS OF ASCII STRING ON STACK
    035230 012201          MOV (R2)+,R1  ;;PICKUP THE BINARY NUMBER
    035232 012202          MOV (R2)+,R2
    035234 012737 000012 035310  MOV #10,4$  ;;SET UP TO DO 10 CONVERSIONS
    035242 012704 035322  MOV #STNPWR,R4  ;;ADDRESS OF TEN POWER
    035246 012705 035324  MOV #STNPWR+2,R5
    035252 005003          1$: CLR R3  ;;CLEAR PARTIAL
    035254 161401          2$: SUB (R4),R1  ;;SUBTRACT TEN POWER
    035256 005602          SBC R2
    035260 161502          SUB (R5),R2
    035262 002402          BLT 3$  ;;BR IF TEN POWER TOO LARGE
    035264 005203          INC R3  ;;ADD 1 TO PARTIAL
    035266 000772          BR 2$  ;;LOOP
    035270 062401          3$: ADD (R4)+,R1  ;;RESTORE SUBTRACTED VALUE
    035272 005502          ADC R2
    035274 062402          ADD (R4)+,R2
    035276 022525          CMP (R5)+,(R5)+  ;;MOVE TO NEXT TEN POWER
    035300 052703 000060  BIS #0,R3  ;;CHANGE PARTIAL TO ASCII
    035304 110320          MOV R3,(R0)+  ;;SAVE IT
    035306 005327          DEC (PC)+  ;;DONE?
    035310 000000          4$: .WORD 0
    035312 001357          BNE 1$  ;;BR IF NO
    035314 105020          CLRB (R0)+  ;;TERMINATOR
    035316 104413          RESREG  ;;RESTORE REGISTERS
    035320 000207          RTS PC  ;;RETURN
    035322 145000          $TNPWR: 145000  ;;1.0E09
    035324 035632          35632
    
```

035326	160400	160400	::1.0E08
035330	002765	2765	
035332	113200	113200	::1.0E07
035334	000230	230	
035336	041100	041100	::1.0E06
035340	000017	17	
035342	103240	103240	::1.0E05
035344	000001	1	
035346	023420	23420	::1.0E04
035350	000000	0	
035352	001750	1750	::1.0E03
035354	000000	0	
035356	000144	144	::1.0E02
035360	000000	0	
035362	000012	12	::1.0E01
035364	000000	0	
035366	000001	1	::1.0E00
035370	000000	0	
035372			

\$DECVL: .BLKB 12. ::RESERVE STORAGE FOR ASCII STRING

3712
3713

.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      #PNTK, -(SP)      :: POINTER TO LOW WORD OF BINARY NUMBER
*      JSR      PC, @#$DB20      :: CALL THE ROUTINE
*      RETURN
*                                     :: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK

```

035406	104412		\$DB20: SAVREG	::SAVE ALL REGISTERS
035410	016601	000002	MOV 2(SP),R1	::PICKUP THE POINTER TO LOW WORD
035414	012705	035525	MOV #12.,R4	::POINTER TO DATA TABLE
035420	012704	000014	MOV #^C7,R3	::DO ELEVEN CHARACTERS
035424	012703	177770	MOV (R1)+,R0	::MASK
035430	012100		MOV (R1)+,R1	::LOWER WORD
035432	012101		MOV (R1)+,R1	::HIGH WORD
035434	005002		CLR R2	::TERMINATOR
035436	110245		1\$: MOV B R2, -(R5)	::PUT CHARACTER IN DATA TABLE
035440	010002		MOV R0, R2	::GET THIS DIGIT
035442	005304		DEC R4	::COUNT THIS CHARACTER
035444	003007		BGT 3\$::BR IF NOT THE LAST DIGIT
035446	001405		BEQ 2\$::BR IF IT IS THE LAST DIGIT
035450	005205		INC R5	::ALL DIGITS DONE-ADJUST POINTER FOR FIRST
035452	010566	000002	MOV R5, 2(SP)	::ASCII CHAR. & PUT IT ON THE STACK
035456	104413		RESREG	::RESTORE ALL REGISTERS
035460	000207		RTS PC	::RETURN TO USER
035462	006203		2\$: ASR R3	::POSITION THE MASK FOR THE LAST DIGIT
035464	006001		3\$: ROR R1	::POSITION THE BINARY NUMBER FOR
035466	006000		ROR R0	:: THE NEXT OCTAL DIGIT
035470	006001		ROR R1	
035472	006000		ROR R0	
035474	006001		ROR R1	
035476	006000		ROR R0	
035500	040302		BIC R3, R2	::MASK OUT ALL JUNK
035502	062702	000060	ADD #0, R2	::MAKE THIS CHAR. ASCII

3714
3715

035506 000753
035510

BR 1\$
\$OCTVL: .BLKB 14. ;:GO PUT IT IN THE DATA TABLE
;:RESERVE DATA TABLE

.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.

035526 010046
035530 016600 000002
035534 005740
035536 111000
035540 006300
035542 016000 035562
035546 000200

\$TRAP: MOV R0, -(SP) ;:SAVE R0
 MOV 2(SP), R0 ;:GET TRAP ADDRESS
 TST -(R0) ;:BACKUP BY 2
 MOVB (R0), R0 ;:GET RIGHT BYTE OF TRAP
 ASL R0 ;:POSITION FOR INDEXING
 MOV \$TRPAD(R0), R0 ;:INDEX TO TABLE
 RTS R0 ;:GO TO ROUTINE

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

035550 011646
035552 016666 000004 000002
035560 000002

\$TRAP2: MOV (SP), -(SP) ;:MOVE THE PC DOWN
 MOV 4(SP), 2(SP) ;:MOVE THE PSW DOWN
 RTI ;:RESTORE THE PSW

.SBTTL TRAP TABLE

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

: ROUTINE

035562 035550
035564 032606
035566 033730
035570 033704
035572 033744
035574 034132

\$TRPAD: .WORD \$TRAP2
 \$TYPE ;:CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
 \$TYPOC ;:CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
 \$TYPOS ;:CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
 \$TYPON ;:CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
 \$TYPDS ;:CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

035576 034426

\$GTSWR ;:CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING

035600 034356
035602 034640
035604 031766
035606 035116
035610 035154
035612 031212
000032

\$CKSWR ;:CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
 \$RDCHR ;:CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
 \$RDLIN ;:CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE
 \$SAVREG ;:CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE
 \$RESREG ;:CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE
 \$DSPLY ;:CALL=DISPLY TRAP+14(104414) ROUTINE TO TYPE ERROR MESSAGES

3716
3717
3718
3722
3729

\$TERM=.-\$TRPAD


```
1      .SBTTL  SINGLE/DUAL PORT RH/RM DRIVER (REV 6.2) FEBRUARY 1980
2
3      ;NEW DRIVE TYPE ID FOR RM02 *****
4      ;10-AUG-77 *****
5      ;10-MAR-78 THE SC, SC5 CHANGES
6      ;NEW DRIVE TYPE ID FOR RM05 *****
7      ;FEBRUARY 1980 *****
8
9      ;COPYRIGHT (C) 1977
10     ;DIGITAL EQUIPMENT CORP.
11     ;MAYNARD, MA 01754
12     ;AUTHOR(S): JIM LACEY/CHUCK HESS/
13
14     ;*****
15
16     ;STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'
17     ;RMERRS = RMD5
18     ;RMERRS+2 = RMER1
19     ;RMERRS+4 = RMER2
20     ;RMERRS+6 = RMMR2
21
22 035614 000000 000000 000000 RMERRS: .WORD 0,0,0,0
23
24     ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
25     ;DRVACT=0 IF DRIVE IS IDLE
26     ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
27     ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
28
29 035624 000      DRVACT: .BYTE 0          ;DRIVE 0
30 035625 000      .BYTE 0          ;DRIVE 1
31 035626 000      .BYTE 0          ;DRIVE 2
32 035627 000      .BYTE 0          ;DRIVE 3
33 035630 000      .BYTE 0          ;DRIVE 4
34 035631 000      .BYTE 0          ;DRIVE 5
35 035632 000      .BYTE 0          ;DRIVE 6
36 035633 000      .BYTE 0          ;DRIVE 7
37
38     ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
39     ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
40     ;DRVSTA>0 IF DRIVE IS ONLINE
41     ;DRVSTA<0 IF DRIVE IS UNSAFE
42
43 035634 000      DRVSTA: .BYTE 0          ;DRIVE 0
44 035635 000      .BYTE 0          ;DRIVE 1
45 035636 000      .BYTE 0          ;DRIVE 2
46 035637 000      .BYTE 0          ;DRIVE 3
47 035640 000      .BYTE 0          ;DRIVE 4
48 035641 000      .BYTE 0          ;DRIVE 5
49 035642 000      .BYTE 0          ;DRIVE 6
50 035643 000      .BYTE 0          ;DRIVE 7
51
52     ;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
53     ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
54     ;DRV TYP=7 IF DRIVE IS RM05 *****
55     ;DRV TYP=5 IF DRIVE IS RM02 *****
56     ;DRV TYP=4 IF DRIVE IS RM03
57     ;DRV TYP=-1 IF NOT RM05/3/2
```

```

50
51 035644 000
54 035645 000
   035646 000
   035647 000
   035650 000
   035651 000
   035652 000
   035653 000
55
56
57
58
59
60 035654 000
63 035655 000
   035656 000
   035657 000
   035660 000
   035661 000
   035662 000
   035663 000
64
65
66
67
68
69 035664 000
72 035665 000
   035666 000
   035667 000
   035670 000
   035671 000
   035672 000
   035673 000
73
74
75
76
77
78 035674 000000
79
80
81
82
83
84
85
86 035676 000000
87
88
89
90
91
92 035700 000
93
94
    
```

```

DRVTYP: .BYTE 0 ;DRIVE 0
         .BYTE 0 ;DRIVE 1
         .BYTE 0 ;DRIVE 2
         .BYTE 0 ;DRIVE 3
         .BYTE 0 ;DRIVE 4
         .BYTE 0 ;DRIVE 5
         .BYTE 0 ;DRIVE 6
         .BYTE 0 ;DRIVE 7

;TABLE OF DUAL PORT INITIALIZATION INDICATORS
;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
;DPINT<0 IF INITIALIZATION IS IN PROGRESS

DPINT:  .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7

;TABLE OF PENDING DUAL PORT REQUESTS
;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE

DPRQS:  .BYTE 0 ;DRIVE 0
        .BYTE 0 ;DRIVE 1
        .BYTE 0 ;DRIVE 2
        .BYTE 0 ;DRIVE 3
        .BYTE 0 ;DRIVE 4
        .BYTE 0 ;DRIVE 5
        .BYTE 0 ;DRIVE 6
        .BYTE 0 ;DRIVE 7

;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
;'DPB' OF THE I/O OPERATION.

TRNSWT: .WORD 0

;SEARCH WAIT KEYS (SRCHWT=1 WORD)
;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.

SRCHWT: .WORD 0

;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
;ACTDRV=0 IF DRIVER IS INACTIVE
;ACTDRV>0 IF DRIVER IS ACTIVE

ACTDRV: .BYTE 0

;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
    
```

```

95                                     ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
96                                     ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
97
98 035701      000      ACTSTR: .BYTE  0
99
100                                     ;UNLOAD FLAG (ULDFLG=8 BYTES)
101                                     ;ULDFLG=0 IF NO UNLOAD COMMAND
102                                     ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
103                                     ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
104
105 035702      000      ULDFLG: .BYTE  0          ;DRIVE 0
108 035703      000      .BYTE  0          ;DRIVE 1
      035704      000      .BYTE  0          ;DRIVE 2
      035705      000      .BYTE  0          ;DRIVE 3
      035706      000      .BYTE  0          ;DRIVE 4
      035707      000      .BYTE  0          ;DRIVE 5
      035710      000      .BYTE  0          ;DRIVE 6
      035711      000      .BYTE  0          ;DRIVE 7
109
110                                     ;LOOK AHEAD COUNT (LACNT=8 BYTES)
111                                     ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
112
113 035712      000      LACNT:  .BYTE  0          ;DRIVE 0
116 035713      000      .BYTE  0          ;DRIVE 1
      035714      000      .BYTE  0          ;DRIVE 2
      035715      000      .BYTE  0          ;DRIVE 3
      035716      000      .BYTE  0          ;DRIVE 4
      035717      000      .BYTE  0          ;DRIVE 5
      035720      000      .BYTE  0          ;DRIVE 6
      035721      000      .BYTE  0          ;DRIVE 7
117
118                                     ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
119                                     ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
120                                     ;OPERATION IS COMPLETED AS PER (DPB+14).
121                                     ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
122                                     ;(DPB+14), AFTER AN ERROR.
123
124 035722      000000    SAVEFG: .WORD  0
125
126                                     ;SEEK FLAG (SEEKFG=1 WORD)
127                                     ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
128                                     ;FOR A DATA TRANSFER START A SEARCH COMMAND
129                                     ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
130                                     ;DISREGARD THE WINDOW
131
132 035724      177777    SEEKFG: .WORD -1
133
134                                     ;TIMEOUT TABLE (TIMER=8 WORDS)
135                                     ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
136
137 035726      177777    TIMER:  .WORD -1          ;DRIVE 0
140 035730      177777    .WORD -1          ;DRIVE 1
      035732      177777    .WORD -1          ;DRIVE 2
      035734      177777    .WORD -1          ;DRIVE 3
      035736      177777    .WORD -1          ;DRIVE 4
      035740      177777    .WORD -1          ;DRIVE 5
      035742      177777    .WORD -1          ;DRIVE 6
    
```

```

035744 177777          .WORD  -1          ;DRIVE 7
141
142          ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
143          ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
144          ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
145
146 035746 177777      DTUW:  .WORD  -1
147
148          ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
149          ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
150          ;ATTENTION BIT
151
152 035750      001      ATABIT:  .BYTE  1          ;DRIVE 0
153 035751      002          .BYTE  2          ;DRIVE 1
154 035752      004          .BYTE  4          ;DRIVE 2
155 035753      010          .BYTE 10          ;DRIVE 3
156 035754      020          .BYTE 20          ;DRIVE 4
157 035755      040          .BYTE 40          ;DRIVE 5
158 035756      100          .BYTE 100         ;DRIVE 6
159 035757      200          .BYTE 200         ;DRIVE 7
160
161          ;FSRM TO RH11/RH70 'MASSBUS CONTROL BUS PARITY ERRORS'' (MCPE) ALLOWED BEFORE
162          ;CALLING IT FATAL (MCPEMX=1 WORD)
163
164 035760 000003      MCPEMX: .WORD  3
165
166          ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
167          ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
168
169 035762 176700      RMADR:  .WORD 176700
170 035764 000254 000240  RMVEC:  .WORD 254,5*32.
171
172          ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
173 035770 000004      MXLACT: .WORD  4
174
175          ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
176 035772 001000      MXDLTA: .WORD  8.*64.
177
178          ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
179 035774 000200      MNDLTA: .WORD  2*64.
180
181          ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
182 035776 000005      MXWNDW: .WORD  5
183
184          ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
185
186          000000      RMCS1=0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
187          000002      RMWC=2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
188          000004      RMBA=4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
189          000006      RMDA=6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
190          000010      RMCS2=10         ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
191          000012      RMDS=12         ;DRIVE STATUS REGISTER (DRIVE REG 01)
192          000014      RMER1=14        ;ERROR REGISTER #1 (DRIVE REG. 02)
193          000016      RMAS=16         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
194          000020      RMLA=20         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
195          000022      RMDB=22         ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
196          000024      RMMR1=24        ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
    
```

197	000026	RMDT=26	:DRIVE TYPE REGISTER (DRIVE REG. 06)
198	000030	RMSN=30	:SERIAL NUMBER REGISTER (DRIVE REG. 10)
199	000032	RMOF=32	:OFFSET REGISTER (DRIVE REG. 11)
200	000034	RMDC=34	:DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
201	000036	RMHR=36	:DUMMY ADDRESS REGISTER (DRIVE REG. 13)
202	000040	RMMR2=40	:MAINTENANCE REGISTER #2
203	000042	RMER2=42	:ERROR REGISTER #2 (DRIVE REG. 15)
204	000044	RMEC1=44	:ECC POSITION REGISTER (DRIVE REG. 16)
205	000046	RMEC2=46	:ECC PATTERN REGISTER (DRIVE REG. 17)

.SBTTL RH/RM DRIVER INITIALIZATION CODE

:THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
 :AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
 :TO THE PROPER STATE FOR EACH DRIVE.
 :NOTE: THIS ROUTINE CALLS DRVINT

:CALL

JSR PC,RMINIT
 RETURN

:NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED

221	036000	104412		RMINIT: SAVREG	:SAVE R0 - R5
222	036002	013746	177776	MOV PS,-(SP)	:SAVE THE PRESENT PROCESSOR STATUS
223	036006	012737	000240	MOV #<5*32.>,PS	:CHANGE THE PRIORITY TO 5
224	036014	004737	043746	JSR PC,CLRQUE	:CLEAR ALL REQUEST QUEUES
225	036020	012701	035614	MOV #RMERRS,R1	:FIRST ADDRESS TO BE CLEARED
226	036024	012702	035724	MOV #SEEKFG,R2	:LAST ADDRESS TO BE CLEARED
227	036030	005021		1\$: CLR (R1)+	:CLEAR
228	036032	020102		CMP R1,R2	:ARE WE DONE?
229	036034	103775		BLO 1\$:BR IF NO
230	036036	012702	035746	MOV #DTUW,R2	:LAST ADDRESS
231	036042	012721	177777	2\$: MOV #-1,(R1)+	:INITIALIZE
232	036046	020102		CMP R1,R2	:DONE?
233	036050	101774		BLOS 2\$:LOOP IF NO
234	036052	005037	035634	CLR DRVSTA	:SET ALL DRIVES TO OFFLINE
235	036056	005037	035636	CLR DRVSTA+2	
236	036062	005037	035640	CLR DRVSTA+4	
237	036066	005037	035642	CLR DRVSTA+6	
238	036072	013703	035764	MOV RMVEC,R3	:SETUP THE RH/RM VECTOR
239	036076	012723	040634	MOV #ISR,(R3)+	
240	036102	013713	035766	MOV RMVEC+2,(R3)	
241	036106	013704	035762	MOV RMADR,R4	:FIRST ADDRESS OF RH/RM
242	036112	012764	000040	MOV #CLR,RMCS2(R4)	:MASSBUS INIT
243	036120	005001	000010	CLR R1	:START WITH DRIVE 0
244	036122	004037	036212	3\$: JSR R0,DRVINT	:INIT THE DRIVE
245	036126	000401		BR 4\$: 'DVA' NOT SET OR PARITY ERROR
246	036130	000402		BR 5\$:NORMAL RETURN
247	036132	105061	035634	4\$: CLR DRVSTA(R1)	:SET DRIVE STATUS TO OFFLINE
248	036136	005201		5\$: INC R1	:GO TO NEXT DRIVE
249	036140	042701	177770	BIC #^C7,R1	:MASK OUT UNUSED BITS
250	036144	001366		BNE 3\$:BR IF MORE DRIVES TO GO
251	036146	012701	000007	MOV #7,R1	:START WITH DRIVE 7
252	036152	005037	177776	CLR PS	:CLEAR THE PROCESSOR STATUS
253	036156	105761	035654	6\$: TSTB DPINT(R1)	:WAITING FOR DRIVE TO SWITCH PORTS ?

```

254 036162 001405          BEQ      8$          ;BR NOT WAITING
255 036164 004737 043402   JSR      PC,SET.IE    ;SET INTERRUPT
256 036170 105761 035654   7$:     TSTB      DPINT(R1) ;DRIVE SWITCHED PORTS ?
257 036174 001375          BNE      7$          ;BR IF NOT
258 036176 005301          8$:     DEC      R1      ;GO TO THE NEXT DRIVE
259 036200 100366          BPL      6$          ;CHECK NEXT DRIVE
260 036202 012637 177776   MOV      (SP)+,PS     ;RESTORE THE PROCESSOR STATUS
261 036206 104413          RESREG   ;RESTORE R0 - R5
262 036210 000207          RTS      PC          ;BYE-BYE
263
264 ;DRIVE INITIALIZATION ROUTINE
265 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
266 ;AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
267 ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
268 ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
269 ;DRVSTA IS SET TO THE PROPER CONDITION.
270 ;CALL
271 ;
272 ;       MOV      #DRVNUM,R1    ;DRIVE NUMBER TO R1
273 ;       MOV      RMADR,R4     ;UNIBUS ADDRESS OF RH/RM (RMCS1)
274 ;       JSR      R0,DRVINT    ;CALLED BY A JSR
275 ;       RETURN1  ;ERROR OCCURRED (PARITY)
276 ;       RETURN2  ;NORMAL RETURN
277 ;
278 036212 010546          DRVINT: MOV      R5,-(SP)    ;SAVE R5
279 036214 105061 035634   DULP:  CLRB     DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
280 036220 105061 035644   CLRB     DRVSTYP(R1)    ;CLEAR THE DRIVE TYPE INDICATOR
281 036224 105061 035702   CLRB     ULDFLG(R1)    ;CLEAR THE UNLOAD FLAG
282 036230 010164 000010   MOV      R1,RMCS2(R4)  ;SELECT A DRIVE
283 036234 112764 000111 000000  MOVB     #111,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
284 036242 032764 010000 000010  BIT      #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
285 036250 001403          BEQ      1$          ;NO
286 036252 004737 043402   JSR      PC,SET.IE    ;GO SET 'IE' WITHOUT A 'TRE'
287 036256 000520          BR      6$          ;LEAVE THIS ROUTINE
288
289 036260 105061 035634   1$:     CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
290 036264 032764 004000 000000  BIT      #BIT11,RMCS1(R4) ;SFE IF DRIVE AVAILABLE
291 036272 001750          BEQ      DULP        ;BR IF DRIVE NOT AVAILABLE
292 036274 004037 042722   JSR      R0,RD.RM     ;READ THE DRIVE TYPE REG.
293 036300 000026          RMDT
294 036302 036544          8$
295 036304 012605          MOV      (SP)+,R5     ;ERROR RETURN ADDRESS
296 036306 112761 000004 035644  MOVB     #4,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
297 036314 022705 020024   CMP      #20024,R5    ;SET RM03 INDICATOR
298 036320 001431          BEQ      2$          ;SINGLE PORT RM03 ?
299 036322 022705 024024   CMP      #24024,R5    ;BR IF YES
300 036326 001426          BEQ      2$          ;DUAL PORT RM03 ?
301 036330 112761 000005 035644  MOVB     #5,DRVSTYP(R1) ;BR IF YES
302 036336 022705 020025   CMP      #20025,R5    ;SET RM02 INDICATOR
303 036342 001420          BEQ      2$          ;SINGLE PORT RM02 ?
304 036344 022705 024025   CMP      #24025,R5    ;BR IF SO
305 036350 001415          BEQ      2$          ;DUAL PORT RM02 ?
306 036352 112761 000007 035644  MOVB     #7,DRVSTYP(R1) ;BR IF SO
307 036360 022705 020027   CMP      #20027,R5    ;SET RM05 INDICATOR
308 036364 001407          BEQ      2$          ;SINGLE PORT RM05 ?
309 036366 022705 024027   CMP      #24027,R5    ;BR IF YES
310 036372 001404          BEQ      2$          ;DUAL PORT RM05 ?
310 036372 001404          BEQ      2$          ;BR IF YES

```

```

311 036374 112761 177777 035644      MOVB  #-1,DRVSTYP(R1)  ;SET INDICATOR TO 'O' IER'
312 036402 000446                    BR      6$              ;EXIT
313
314 036404 012746 000121                2$:  MOV      #121,-(SP)    ;DO A 'READ-IN PRESET''
315 036410 004037 043076                JSR      RO,WRT.RM
316 036414 000000                    RMCS1
317 036416 036544                    8$
318 036420 012746 010000                MOV      #BIT12,-(SP)   ;SET FMT16=1
319 036424 004037 043076                JSR      RO,WRT.RM
320 036430 000032                    RMOF
321 036432 036544                    8$
322 036434 004037 042722                JSR      RO,RD.RM      ;READ RMDS
323 036440 000012                    RMDS
324 036442 036544                    8$
325 036444 012605                    MOV      (SP)+,R5      ;AND SAVE IT IN R5
326 036446 100015                    BPL      4$            ;BR IF ATA=0
327 036450 116164 035750 000016        MOVB     ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
328 036456 004037 042722                JSR      RO,RD.RM      ;FIND OUT WHY ATA=1
329 036462 000014                    RMER1
330 036464 036544                    8$
331 036466 006126                    ROL      (SP)+         ;IS IT UNSAFE?
332 036470 100004                    BPL      4$            ;BR IF NOT
333 036472 112761 177777 035634        MOVB     #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
334 036500 000407                    BR      6$              ;EXIT
335
336 036502 005105                    4$:  COM      R5          ;CHECK MOL, DPR, DRY, AND VV
337 036504 042705 167077                BIC     #^C<BIT12!BIT08!BIT07!BIT06>,R5
338 036510 001003                    BNE     6$            ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
339 036512 112761 000001 035634        MOVB     #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
340 036520 005720                    6$:  TST      (R0)+       ;STEP OVER THE ERROR RETURN
341 036522 000410                    BR      8$            ;EXIT
342 036524 006301                    7$:  ASL      R1          ;CHANGE INDEX TO ADDRESS WORDS
343 036526 012761 060000 035726        MOV      #60000,TIMER(R1) ;START 2 SEC TIMER
344 036534 006201                    ASR      R1
345 036536 112761 177777 035654        MOVB     #-1,DPINT(R1) ;SET PORT INITIALIZE INIDICATOR
346 036544 012605                    8$:  MOV      (SP)+,R5      ;RESTORE R5
347 036546 000200                    RTS      RO            ;EXIT
348
349                                ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
350                                ;CALL
351                                ;
352                                ;
353                                ;
354                                ;
355                                ;
356                                ;
357                                ;
358                                ;
359 036550 013746 177776                RM05:  MOV      PS,-(SP)   ;SAVE THE CALLING STATUS
360 036554 013737 035766 177776        MOV      RMVEC+2,PS    ;DON'T ALLOW ANY RM INTERRUPTS
361 036562 112737 000001 035700        MOVB     #1,ACTDRV    ;SET 'ACTIVE DRIVER' FLAG
362 036570 104412                    SAVREG ;SAVE R0 - R5
363 036572 011002                    MOV      (R0),R2      ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
364 036574 005062 000016                CLR      16(R2)       ;CLEAR THE STATUS/ERROR INDICATOR
365 036600 111201                    MOVB     (R2),R1      ;PICKUP THE DRIVE NUMBER
366 036602 013704 035762                MOV      RMADR,R4     ;UNIBUS ADDRESS OF RMCS1
367 036606 105761 035634                TSTB     DRVSTA(R1)  ;CHECK DRIVES STATUS

```

```

368 036612 003014          BGT      1$          ;BP IF ONLINE
369 036614 105761 035702   TSTB    ULDFLG(R1)  ;UNLOAD COMMAND IN QUEUE?
370 036620 001036          BNE     3$          ;BR IF YES
371 036622 105761 035654   TSTB    DPINT(R1)   ;TRYING TO INIT THE DRIVE
372 036626 001042          BNE     5$          ;BR IF YES
373 036630 004037 036212   JSR     R0,DRVINT   ;GO INIT. THE DRIVE
374 036634 000434          BR      4$          ;ERROR RETURN
375 036636 105761 035634   TSTB    DRVSTA(R1)  ;IS DRIVE STATUS ONLINE?
376 036642 003445          BLE     6$          ;BR IF NOT
377 036644 105761 035664   1$:    TSTB    DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
378 036650 001031          PNE     5$          ;BR IF YES
379 036652 010164 000010   MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
380 036656 004037 044044   JSR     R0,DRVQUE   ;PUT THIS REQUEST IN QUEUE
381 036662 000460          BR      9$          ;QUEUE IS FULL
382 036664 122762 000103 000002  CMPB    #103,2(R2)   ;IS THIS REQ. FOR AN UNLOAD?
383 036672 001003          BNE     2$          ;BR IF NO
384 036674 112761 177777 035702  MOVB    #-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
385 036702 105761 035624   2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
386 036706 001043          BNE     8$          ;BR IF YES
387 036710 004737 037042   JSR     PC,OPT      ;CALL THE OPTIMIZER
388 036714 000440          BR      8$
389 036716 012762 120000 000016  3$:    MOV     #BIT15!BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
390 036724 000434          BR      8$          ;EXIT
391 036726 004737 040122   4$:    JSR     PC,C17    ;GO HANDLE THE PARITY ERROR
392 036732 000431          BR      8$
393 036734 004037 044044   5$:    JSR     R0,DRVQUE   ;PUT REQUEST IN QUEUE
394 036740 000431          BR      9$          ;QUEUE IS FULL
395 036742 032714 000100   BIT     #BIT06,(R4) ;IE BIT SET ?
396 036746 001023          BNE     8$          ;YES
397 036750 004737 043402   JSR     PC,SET.IE   ;SET THE INTERRUPT
398 036754 000420          BR      8$          ;RETURN
399 036756 105761 035634   6$:    TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
400 036762 002412          BLT     7$          ;BR IF UNSAFE
401 036764 012762 140000 000016  MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
402 036772 105761 035644   TSTB    DRVTP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
403 036776 001007          BNE     8$          ;BR IF OFFLINE
404 037000 012762 100002 000016  MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
405 037006 000403          BR      8$          ;GO TO EXIT
406 037010 012762 110000 000016  7$:    MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
407 037016 104413          8$:    RESREG   ;RESTORE R0 - R5
408 037020 005720          TST     (R0)+       ;SETUP FOR NORMAL RETURN
409 037022 000401          BR      10$         ;FINISH UP, THEN EXIT
410 037024 104413          9$:    RESREG   ;RESTORE R0 - R5
411 037026 005720          10$:   TST     (R0)+       ;CORRECT THE RETURN ADDRESS
412 037030 105037 035700   CLRB   ACTDRV      ;CLEAR 'ACTIVE DRIVER' FLAG
413 037034 012637 177776   MOV     (SP)+,PS    ;RETURN 'PS' TO USER LEVEL
414 037040 000200          RTS     R0          ;RETURN TO CALLER
415
416          ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
417          ;
418          ;CALL
419          ;
420          ;      MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
421          ;      JSR     PC,OPT          ;SETUP A COMMAND
422 037042 104412          OPT:   SAVREG   ;SAVE R0 - R5
423 037044 013746 177776   MOV     PS,-(SP)    ;SAVE PROC. STATUS
424 037050 146137 035750 035676  BICB   ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG

```



```

425 037056 105061 035664      CLRB   DPRQS(R1)      ;RESET THE PORT REQ FLAG ****
426 037062 004737 044120      JSR    PC,GETREQ     ;GET 'DPB' POINTER OF REQUEST
427 037066 005702                TST    R2            ;IS THERE A REQUEST IN QUEUE?
428 037070 001472                BEQ    7$            ;NO--BR TO EXIT
429 037072 010164 000010      MOV    R1, RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
430 037076 012764 000111 000000  MOV    #111, RMCS1(R4) ;CLEAR THE DRIVE
431 037104 032764 004000 000000  BIT    #BIT11, RMCS1(R4) ;DVA SET ?
432 037112 001446                BEQ    5$            ;TO PROT REQUEST ,IF NOT
433 037114 105761 035634      10$:  TSTB   DRVSTA(R1)    ;IS DRIVE ONLINE?
434 037120 003014                BGT    1$            ;YES
435 037122 004737 044142      JSR    PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
436 037126 012762 140000 000016  MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
437 037134 105761 035634      TSTB   DRVSTA(R1)    ;IS DRIVE UNSAFE ?
438 037140 100053                BPL    8$            ;BR TO EXIT IF NOT
439 037142 012762 110000 000016  MOV    #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
440 037150 000447                BR     8$            ;BR TO EXIT
441 037152                1$:
442                :
443                :
444                :
445                :
446                :
447                :
448 037152 122762 000150 000002  MOV    #111,-(SP)     ;LOAD COMMAND ONTO THE STACK
449 037160 002403                JSR    RO,WRT.RM     ;LOAD THE REGISTER
450 037162 004737 037506      RMCS1  ;REGISTER INCREMENT
451 037166 000440                6$
452 037170 005737 035746      2$:  BIT    #BIT11,(R4)   ;DRIVE AVAILABLE ?
453 037174 002012                BEQ    9$            ;BR IF NOT
454 037176 005737 035724      CMPB   #150,2(R2)    ;IS THE REQUEST FOR I/O?
455 037202 100404                BLT    2$            ;YES
456 037204 004037 040460      JSR    PC,C14        ;CALL THE COMMAND INITIATOR
457 037210 000427                BR     8$            ;BR TO EXIT
458 037212 000403                TST   DTUW           ;DATA TRANSFER UNDERWAY?
459 037214 004737 037300      3$:  BGE    4$            ;YES--GO START A SEARCH
460 037220 000423                TST   SEEKFG        ;DO IMPLIED SEEKS?
461 037222 004737 037406      4$:  BMI    3$            ;YES
462 037226 000420                JSR    RO,LA         ;NO--DO LOOK AHEAD
463 037230 112761 177777 035664  5$:  BR     8$            ;RETURN HERE ON A PARITY ERROR
464 037236 010103                BR     4$            ;GO START A SEARCH
465 037240 006303                JSR    PC,C11        ;START A DATA TRANSFER
466 037242 012763 060000 035726  6$:  BR     8$            ;START A SEARCH
467 037250 000402                BR     8$            ;GO TO THE EXIT
468 037252 004737 040122      7$:  MOVB   #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
469 037256 032714 000100      MOV    R1,R3         ;SET UP TO ADDRESS WORDS
470 037262 001002                ASL   R3             ;CONVERT TO WORD INDEX
471 037264 004737 043402      MOV    #60000,TIMER(R3) ;START 10 SEC TIMER
472 037270 012637 177776      8$:  BR     7$            ;EXIT
473 037274 104413                JSR    PC,C17        ;PROCESS THE PARITY ERROR
474 037276 000207                BIT    #BIT06,(R4)   ;SEE IF 'IE' ALREADY SET
475                :
476                :
477                :
478                :
479                :
480                :
481                :
482                :
483                :
484                :
485                :
486                :
487                :
488                :
489                :
490                :
491                :
492                :
493                :
494                :
495                :
496                :
497                :
498                :
499                :
500                :
501                :
502                :
503                :
504                :
505                :
506                :
507                :
508                :
509                :
510                :
511                :
512                :
513                :
514                :
515                :
516                :
517                :
518                :
519                :
520                :
521                :
522                :
523                :
524                :
525                :
526                :
527                :
528                :
529                :
530                :
531                :
532                :
533                :
534                :
535                :
536                :
537                :
538                :
539                :
540                :
541                :
542                :
543                :
544                :
545                :
546                :
547                :
548                :
549                :
550                :
551                :
552                :
553                :
554                :
555                :
556                :
557                :
558                :
559                :
560                :
561                :
562                :
563                :
564                :
565                :
566                :
567                :
568                :
569                :
570                :
571                :
572                :
573                :
574                :
575                :
576                :
577                :
578                :
579                :
580                :
581                :
582                :
583                :
584                :
585                :
586                :
587                :
588                :
589                :
590                :
591                :
592                :
593                :
594                :
595                :
596                :
597                :
598                :
599                :
600                :
601                :
602                :
603                :
604                :
605                :
606                :
607                :
608                :
609                :
610                :
611                :
612                :
613                :
614                :
615                :
616                :
617                :
618                :
619                :
620                :
621                :
622                :
623                :
624                :
625                :
626                :
627                :
628                :
629                :
630                :
631                :
632                :
633                :
634                :
635                :
636                :
637                :
638                :
639                :
640                :
641                :
642                :
643                :
644                :
645                :
646                :
647                :
648                :
649                :
650                :
651                :
652                :
653                :
654                :
655                :
656                :
657                :
658                :
659                :
660                :
661                :
662                :
663                :
664                :
665                :
666                :
667                :
668                :
669                :
670                :
671                :
672                :
673                :
674                :
675                :
676                :
677                :
678                :
679                :
680                :
681                :
682                :
683                :
684                :
685                :
686                :
687                :
688                :
689                :
690                :
691                :
692                :
693                :
694                :
695                :
696                :
697                :
698                :
699                :
700                :
701                :
702                :
703                :
704                :
705                :
706                :
707                :
708                :
709                :
710                :
711                :
712                :
713                :
714                :
715                :
716                :
717                :
718                :
719                :
720                :
721                :
722                :
723                :
724                :
725                :
726                :
727                :
728                :
729                :
730                :
731                :
732                :
733                :
734                :
735                :
736                :
737                :
738                :
739                :
740                :
741                :
742                :
743                :
744                :
745                :
746                :
747                :
748                :
749                :
750                :
751                :
752                :
753                :
754                :
755                :
756                :
757                :
758                :
759                :
760                :
761                :
762                :
763                :
764                :
765                :
766                :
767                :
768                :
769                :
770                :
771                :
772                :
773                :
774                :
775                :
776                :
777                :
778                :
779                :
780                :
781                :
782                :
783                :
784                :
785                :
786                :
787                :
788                :
789                :
790                :
791                :
792                :
793                :
794                :
795                :
796                :
797                :
798                :
799                :
800                :
801                :
802                :
803                :
804                :
805                :
806                :
807                :
808                :
809                :
810                :
811                :
812                :
813                :
814                :
815                :
816                :
817                :
818                :
819                :
820                :
821                :
822                :
823                :
824                :
825                :
826                :
827                :
828                :
829                :
830                :
831                :
832                :
833                :
834                :
835                :
836                :
837                :
838                :
839                :
840                :
841                :
842                :
843                :
844                :
845                :
846                :
847                :
848                :
849                :
850                :
851                :
852                :
853                :
854                :
855                :
856                :
857                :
858                :
859                :
860                :
861                :
862                :
863                :
864                :
865                :
866                :
867                :
868                :
869                :
870                :
871                :
872                :
873                :
874                :
875                :
876                :
877                :
878                :
879                :
880                :
881                :
882                :
883                :
884                :
885                :
886                :
887                :
888                :
889                :
890                :
891                :
892                :
893                :
894                :
895                :
896                :
897                :
898                :
899                :
900                :
901                :
902                :
903                :
904                :
905                :
906                :
907                :
908                :
909                :
910                :
911                :
912                :
913                :
914                :
915                :
916                :
917                :
918                :
919                :
920                :
921                :
922                :
923                :
924                :
925                :
926                :
927                :
928                :
929                :
930                :
931                :
932                :
933                :
934                :
935                :
936                :
937                :
938                :
939                :
940                :
941                :
942                :
943                :
944                :
945                :
946                :
947                :
948                :
949                :
950                :
951                :
952                :
953                :
954                :
955                :
956                :
957                :
958                :
959                :
960                :
961                :
962                :
963                :
964                :
965                :
966                :
967                :
968                :
969                :
970                :
971                :
972                :
973                :
974                :
975                :
976                :
977                :
978                :
979                :
980                :
981                :
982                :
983                :
984                :
985                :
986                :
987                :
988                :
989                :
990                :
991                :
992                :
993                :
994                :
995                :
996                :
997                :
998                :
999                :
1000               :

```


539	037540	000006			RMDA			
540	037542	040122			CI7			
541	037544	000403			BR	2\$:GO LOAD CYLINDER
542	037546	122703	000105	1\$:	CMPB	#105,R3		:IS IT A SEEK COMMAND
543	037552	001007			BNE	3\$:BR IF NO
544	037554	016246	000012	2\$:	MOV	12(R2),-(SP)		:LOAD DESIRED CYLINDER
545	037560	004037	043076		JSR	RO,WRT.RM		
546	037564	000034			RMDC			
547	037566	040122			CI7			
548	037570	000546			BR	CI6		
549	037572	122703	000115	3\$:	CMPB	#115,R3		:IS IT AN 'OFFSET' COMMAND?
550	037576	001013			BNE	4\$:BR IF NO
551	037600	004037	042722		JSR	RO,RD.RM		:MERGE THE OFFSET VALUE INTO RMOF
552	037604	000032			RMOF			:BUT DON'T CHANGE THE UPPER
553	037606	040122			CI7			
554	037610	116216	000001		MOVB	1(R2),(SP)		:BYTE WHEN LOADING THE
555	037614	004037	043076		JSR	RO,WRT.RM		:REGISTER (RMOF)
556	037620	000032			RMOF			
557	037622	040122			CI7			
558	037624	000530			BR	CI6		:GO START THE COMMAND
559	037626	122703	000107	4\$:	CMPB	#107,R3		:IS IT A 'RECALIBRATE' COMMAND?
560	037632	001525			BEQ	CI6		:BR IF YES
561	037634	122703	000117		CMPB	#117,R3		:IS IT A RETURN TO CENTER?
562	037640	001522			BEQ	CI6		:BR IF YES
563	037642	122703	000103		CMPB	#103,R3		:IS IT AN 'UNLOAD' COMMAND?
564	037646	001016			BNE	5\$:BR IF NO
565	037650	112761	000001	035624	MOVB	#1,DRVACT(R1)		:SET THE DRIVE ACTIVE INDICATOR
566	037656	105061	035634		MOV	DRVSTA(R1)		:PUT DRIVE STATUS TO OFFLINE
567	037662	112761	000001	035702	MOVB	#1,ULDFLG(R1)		:SET 'UNLOAD IN PROGRESS' FLAG
568	037670	010346			MOV	R3,-(SP)		:START THE 'UNLOAD' COMMAND
569	037672	004037	043076		JSR	RO,WRT.RM		
570	037676	000000			RMCS1			
571	037700	040122			CI7			
572	037702	000207			RTS	PC		:RETURN TO USER
573	037704	122703	000143	5\$:	CMPB	#145,R3		:IS IT A 'SET FORMAT' COMMAND?
574	037710	001014			BNE	6\$:BR IF NO
575	037712	004037	042722		JSR	RO,RD.RM		:READ THE OFFSET REGISTER
576	037716	000032			RMOF			
577	037720	040122			CI7			
578	037722	116266	000001	000001	MOVB	1(R2),(SP)		:COMBINE 'FMT16','ECI', AND 'HCI'
579	037730	004037	043076		JSR	RO,WRT.RM		:LOAD 'FMT16', 'ECI', AND/OR 'HCI'.
580	037734	000032			RMOF			
581	037736	040122			CI7			
582	037740	000436			BR	12\$		
583	037742	122703	000141	6\$:	CMPB	#141,R3		:IS IT A 'GET REGISTER' COMMAND?
584	037746	001023			BNE	10\$:BR IF NO
585	037750	016203	000006	7\$:	MOV	6(R2),R3		:POINTS TO 1ST ADDRESS OF WHERE
586								:TO PUT THE REGISTER(S)
587	037754	116237	000010	037772	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
588	037762	116205	000011		MOVB	11(R2),R5		:INDEX OF LAST REG. TO MOVE
589	037766	004037	042722	8\$:	JSR	RO,RD.RM		:READ RH/RM REGISTER
590	037772	000000		9\$:	RMCS1			:INDEX OF REG. TO READ
591	037774	040122			CI7			
592	037776	012623			MOV	(SP)+,(R3)+		:GET THE CONTENTS OF RH/RM REG.
593	040000	023705	037772		CMP	9\$,R5		:LAST REG. BEEN READ?
594	040004	001414			BEQ	12\$:GET OUT IF YES
595	040006	062737	000002	037772	ADD	#2,9\$:INCREASE THE INDEX BY 2

```

596 040014 000764          BR      8$          ;LOOP--MORE TO READ
597 040016 122703 000145 10$:  CMPB   #145,R3      ;IS IT A 'SELECT DRIVE' COMMAND?
598 040022 001405          BEQ    12$          ;BR IF YES
599 040024 010346          MOV    R3,-(SP)    ;LOAD THE COMMAND
600 040026 004037 043076  JSR    R0,WRT.RM
601 040032 000000          RMCS1
602 040034 040122          CI7
603 040036 004737 044142 12$:  JSR    PC,POPQUE   ;REMOVE REQ. FROM QUEUE
604 040042 052762 000200 000016  BIS    #BIT07,16(R2) ;SET THE 'DONE' BIT
605 040050 005737 035722  TST    SAVEFG      ;SAVE THE RH/RM REGISTERS?
606 040054 100002          BPL    13$          ;BR IF NO
607 040056 004737 043264  JSR    PC,SVRH70   ;YES--GO SAVE THE REGISTERS
608 040062 000207          RTS    PC          ;RETURN TO USER
609 040064 006301          CI5:  ASL    R1
610 040066 012761 060000 035726  MOV    #60000,TIMER(R1) ;SET A ONE SECOND TIMER
611 040074 006201          ASR    R1
612 040076 112761 000001 035624  MOVB   #1,DRVACT(R1) ;SET THE DRIVE ACTIVE
613 040104 000207          RTS    PC          ;RETURN TO THE USER
614 040106 010346          CI6:  MOV    R3,-(SP)    ;LOAD THE COMMAND
615 040110 004037 043076  JSR    R0,WRT.RM
616 040114 000000          RMCS1
617 040116 040122          CI7
618 040120 000761          BR      CI5
619 040122 032764 010000 000010 CI7:  BIT    #BIT12,RMCS2(R4) ;DRIVE NON-EXISTENT ?
620          BNE    CI8          ;BR IF YES
621 040130 005702          1$:  TST    R2          ;ANYTHING IN QUEUE ?
622          BEQ    CI7B       ;BR IF NOT
623 040132 001001          BNE    2$          ;BR IF QUEUE IS THERE
624 040134 000207          RTS    PC          ;OTHERWISE EXIT
625 040136 012762 104000 000016 2$:  MOV    #BIT15!BIT11,16(R2) ;SET 'PARITY' ERROR INDICATOR
626          JSR    PC,SVRH70   ;GO SAVE THE RH/RM REGISTERS
627 040144 012746 000111 000111 CI7B:  MOV    #111,-(SP)    ;DO A 'DRIVE CLEAR'
628 040150 004037 043076  JSR    R0,WRT.RM
629 040154 000000          RMCS1
630 040156 040222          CI8
631 040160 004737 044024 2$:  JSR    PC,EMPTYQ   ;EMPTY THE QUEUE
632 040164 105061 035664  CLRB   DPRQS(R1)   ;CLEAR THE PORT REQUEST FLAG
633 040170 105061 035702  CLRB   ULDFLG(R1)  ;CLEAR THE UNLOAD IN QUEUE FLAG
634 040174 105061 035624  CLRB   DRVACT(R1)  ;DRIVE IS IDLE
635 040200 020237 035674  CMP    R2,TRNSWT   ;IF THIS DRIVE HAD AN I/O REQUEST
636          CMP    R1,DTUW    ;IF THIS DRIVE HAD AN I/O REQUEST
637 040204 001005          BNE    1$          ;IN PROGRESS CLEAR ALL OF THE FLAGS
638 040206 005037 035674  CLR    TRNSWT
639 040212 012737 177777 035746  MOV    #-1,DTUW
640 040220 000207          1$:  RTS    PC
641 040222 104412          CI8:  SAVREG
642 040224 032764 010000 000010 BIT    #BIT12,RMCS2(R4) ;SAVE R0 - R5
643          BNE    1$          ;IS 'NED' SET ?
644 040232 005001          CLR    R1          ;BR IF YES
645 040234 005003          CLR    R3
646 040236 105761 035624 1$:  TSTB   DRVACT(R1) ;DRIVE ACTIVE?
647          BEQ    5$          ;BR IF NO
648 040242 001003          BNE    22$         ;BR IF IN ACTIVE
649 040244 105761 035664  TSTB   DPRQS(R1)  ;PORT REQUEST
650 040250 001443          BEQ    5$          ;BR IF NOT
651 040252 013702 035674 22$:  MOV    TRNSWT,R2   ;GET THE 'TRANSFER WAIT' QUEUE
652 040256 020137 035746  CMP    R1,DTUW    ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?

```

```

653 040262 001402          BEQ      2$          ;BR IF YES
654 040264 004737 044120   JSR      PC,GETREG  ;GET THE DPB POINTER
655 040270 005702          TST      R2         ;QUEUE ENTRY FOR DRIVE ?
656 040272 001413          BEQ      4$          ;BR IF NOT
657 040274 032764 010000 000010 BIT      #BIT12,RMCS2(R4) ;'NED' SET ?
658 040302 001404          BEQ      3$          ;BR IF NOT
659 040304 012762 100002 000016 MOV      #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
660 040312 000403          BR       4$          ;CONTINUE
661 040314 012762 102000 000016 3$: MOV      #BIT15!BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
662          ;          ;
663 040322 012763 177777 035726 4$: JSR      PC,SVRH7C  ;SAVE RH/RM REGISTERS
664 040330 105061 035624   MOV      #-1,TIMER(R3) ;STOP THE TIMER
665 040334 105061 035664   CLRB    DRVACT(R1)  ;SET 'DRIVE ACTIVE' TO IDLE
666 040340 020137 035746   CLRB    DPRQS(R1)  ;CLEAR PORT REQUEST FLAG
667 040344 001005          CMP      R1,DTUW    ;IS THIS DRIVE SETUP FOR A TRANSFER
668 040346 012737 177777 035746 BNE      5$          ;BR IF NOT
669 040354 005037 035674   MOV      #-1,DTUW  ;RESET THE INDICATOR
670 040360 105061 035702   CLR     TRNSWT     ;CLEAR THE TRANSFER QUEUE
671 040364 032764 010000 000010 5$: CLRB    ULDFLG(R1) ;CLEAR UNLOAD FLAG
672          ;          ;
673 040372 005201          BNF      6$          ;BR IF YES
674 040374 062703 000002   INC     R1         ;MOVE TO THE NEXT DRIVE
675 040400 042701 177770   ADD     #2,R3
676 040404 001314          BIC     #^C7,R1
677 040406 012737 177777 035746 BNE      1$          ;BR IF MORE DRIVES
678 040414 005037 035674   MOV      #-1,DTUW  ;NO DATA TRANSFERS UNDERWAY
679 040420 004737 043746   CLR     TRNSWT     ;CLEAR THE 'TRANSFER WAIT' QUEUE
680 040424 012764 000040 000010 JSR      PC,CLRQUE  ;CLEAR ALL OF THE REQUEST QUEUES
681 040432 000406          MOV      #CLR,RMCS2(R4) ;DO A MASSBUS INIT.
682 040434 004737 044024   BR       7$          ;CONTINUE
683 040440 105061 035634   JSR      PC,EMPTYQ ;CLEAR THE DRIVE'S QUEUE
684 040444 105061 035644   CLRB    DRVSTA(R1) ;SET DRIVE TO OFFLINE
685 040450 004737 043402   CLRB    DRVTP(R1)  ;CLEAR THE DRIVE TYPE INDICATOR
686 040454 104413          JSR      PC,SET.IE ;SET 'IE' WITHOUT 'TRE'
687 040456 000207          RESREG ;RESTORE R0 - R5
688          RTS      PC ;RETURN
689          ;LOOK AHEAD ROUTINE
690          ;CALL
691          ;
692          ;      MOV      #DRVNUM,R1 ;DRIVE NUMBER
693          ;      MOV      #DPB,R2   ;POINT TO DPB
694          ;      JSR      R0,LA    ;GO CHECK THE WINDOW
695          ;      RETURN1 ;ERROR RETURN
696          ;      RETURN2 ;START A SEARCH
697          ;      RETURN3 ;START A DATA TRANSFER
698          ;
699 040460 013704 035762   LA:    MOV      RMADR,R4 ;GET RMCS1'S ADDRESS
700 040464 010164 000010   MOV      R1,RMCS2(R4) ;SELECT DRIVE
701 040470 004037 042722   JSR      R0,RD.RM  ;READ DRIVE STATUS
702 040474 000012          RMD5
703 040476 040626          4$
704 040500 042716 157577   BIC     #^C020200,(SP) ;ERROR RETURN ADDRESS
705 040504 022726 000200   CMP     #200,(SP)+  ;ON CYLINDER ?
706 040510 001044          BNE     3$          ;PIP=0,DRY=1?
707 040512 105261 035712   INCB   LACNT(R1)   ;NO
708 040516 126137 035712 035770 CMPB    LACNT(R1),MXLACT ;INCREMENT THE LOOK AHEAD COUNT
709 040524 003033          BGT     2$          ;EXCEED MAX?
                          ;BR IF YES

```

```

710 040526 116203 000010      MOVB 10(R2),R3      ;GET DESIRED SECTOR ADDRESS AND
711 040532 000303              SWAB R3             ;MULT. BY 64--ALIGN WITH
712 040534 006203              ASR R3             ;LOOK AHEAD REGISTER
713 040536 006203              ASR R3
714 040540 012737 000340 177776  MOV #340,PS        ;PRIORITY LEVEL '7'
715 040546 004037 042722 6$:   JSR RO,RD.RM       ;READ LOOK AHEAD REGISTER
716 040552 000020              RMLA
717 040554 040626              4$
718 040556 021664 000020      CMP (SP),RMLA(R4)  ;CORRECT LA NUMBER ?
719 040562 001407              BEQ 7$             ;YES
720 040564 005726              TST (SP)+         ;NO,CLEAR STACK
721 040566 000415              BR 3$
722 040570 162603              7$: SUB (SP)+,R3     ;CALCULATE THE DELTA
723 040572 002002              BGE 1$
724 040574 062703 004000      ADD #<32.*64.>,R3 ;MAKE THE DELTA POSITIVE
725 040600 023703 035772 1$:   CMP MXDLTA,R3     ;CHECK THE DELTA TO SEE
726 040604 002406              BLT 3$            ;IF IT IS WITHIN THE
727 040606 023703 035774      CMP MNDLTA,R3     ;WINDOW---IF YES, ZERO
728 040612 002003              BGE 3$            ;THE LOOK AHEAD COUNT
729 040614 105061 035712 2$:   CLRB LACNT(R1)   ;AND TAKE THE I/O EXIT
730 040620 005720              TST (R0)+
731 040622 005720 3$:         TST (R0)+         ;ADJUST THE RETURN ADDRESS
732 040624 000402              BR 5$            ;EXIT
733 040626 004737 040122 4$:   JSR PC,C17       ;PROCESS THE ERROR
734 040632 000200 5$:         RTS R0           ;RETURN
735
736 ;INTERRUPT SERVICE ROUTINE
737
738 040634 112737 000001 035700 ISR: MOVB #1,ACTDRV   ;SET 'ACTIVE DRIVER' FLAG
739 040642 104412              SAVREG           ;SAVE R0 - R5
740 040644 013704 035762      MOV RMADR,R4     ;ADDRESS OF RHSCS1
741 040650 013701 035746      MOV DTUW,R1     ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
742 040654 002403              BLT 1$          ;BR IF NO DATA TRANSFER UNDERWAY
743 040656 004737 040700      JSR PC,TD       ;CALL TRANSFER DONE
744 040662 000402              BR 2$          ;EXIT
745 040664 004737 041154 1$:   JSR PC,SC       ;CALL SPECIAL CONDITIONS
746 040670 104413 2$:         RESREG           ;RESTORE R0 - R5
747 040672 105037 035700      CLRB ACTDRV    ;CLEAR 'ACTIVE DRIVER' FLAG
748 040676 000002              RTI           ;RETURN
749
750 ;TRANSFER DONE ROUTINE
751
752 040700 105061 035624 35746 TD:  CLRB DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
753 040704 012737 177777 035746  MOV #-1,DTUW     ;NO DAT. TRANSFERS UNDERWAY
754 040712 006301              ASL R1
755 040714 012761 177777 035726  MOV #-1,TIMER(R1) ;CANCEL TIMEOUT
756 040722 006201              ASR R1
757 040724 013702 035674      MOV TRNSWT,R2   ;GET 'DPB' ADDRESS FROM THE
758 040730 005037 035674      CLR TRNSJT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
759 040734 052762 000200 000016  BIS #BIT0,16(R2) ;SET DONE
760 040742 010164 000010      MOV R1,RMCS2(R4) ;SELECT THE DRIVE
761 040746 004037 042722      JSR RO,RD.RM    ;TRANSFER ERROR(TRE=1)?
762 040752 000000              RMCS1
763 040754 040122              C17
764 040756 006126              ROL (SP)+
765 040760 100421 3$:         BMI 3$          ;BR IF YES
766 040762 005737 035722      TST SAVEFG     ;SAVE THE RH/RM REGISTERS?

```

```

767 040766 100002          BPL      1$          ;BR IF NO
768 040770 004737 043264  JSR      PC,SVRH70    ;YES--SAVE THE REGISTERS
770 040774 004737 041054  1$:     JSR      PC,WC.HK    ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
774 041000 004737 044120  JSR      PC,GETREQ    ;GET DPB POINTER
775 041004 005702          TST      R2          ;ENTRY FOR DRIVE ?
776 041006 001403          BEQ      2$          ;BR IF NOT
777 041010 004737 037042  JSR      PC,OPT      ;CALL OPTIMIZER
778 041014 000457          BR       SC          ;CHECK OTHER DRIVES
779          ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
780 041016 012714 000113  2$:     MOV      #113,(R4) ;RELEASE THE DRIVE
781 041022 000454          BR       SC          ;CHECK FOR OTHER DRIVES
782 041024 052762 100100 000016  3$:     BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
783 041032 004737 044024  JSR      PC,EMPTYQ    ;EMPTY THE 'DRIVE'S WAIT' QUEUE
784 041036 004737 043264  JSR      PC,SVRH70    ;SAVE THE RH/RM REGISTERS
785 041042 012714 040111  MOV      #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
786 041046 012714 000113  MOV      #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
787 041052 000440          BR       SC          ;CHECK FOR OTHER DRIVES
788
789
791 041054 122762 000002 000024  WC.HK:  CMPB     #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
792 041062 001404          BEQ      1$          ;BR IF IT WAS
793 041064 122762 000003 000024  CMPB     #3,$CODE(R2) ;LAST OPERATION WRITE HEADER & DATA ?
794 041072 001027          BNE     2$          ;BR IF NOT
795 041074 004037 044044  1$:     JSR      R0,DRVQUE ;PUT THE OPERATION IN THE QUEUE
796 041100 000424          BR       2$          ;QUEUE IS FULL
797 041102 005062 000016  CLR      16(R2)      ;CLEAR 'DONE' BIT IN DPB
798 041106 116262 002116 000027  MOVB     $RMCS1(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
799 041114 016262 000012 000034  MOV      $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
800 041122 016262 000010 000032  MOV      $SEC(R2),$PREVA(R2)   ;SAVE SECTOR AND TRACK ADDRESSES
801 041130 142762 000002 000024  BICB     #2,$CODE(R2) ;CHANGE WRITE TO CHECK
802 041136 142762 000020 000002  BICB     #20,$COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
803 041144 152762 000010 000002  BISB     #10,$COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
804 041152 000207          2$:     RTS      PC          ;EXIT
806
807          ;SPECIAL CONDITION ROUTINE
808
809 041154 116403 000016  SC:     MOVB     RMAS(R4),R3 ;READ 'RMAS'
810 041160 001012          BNE     2$          ;BR IF ANY 'ATA' BITS SET
811 041162 004037 042722  JSR      R0,RD.RM    ;READ CONTROL AND STATUS REGISTER
812 041166 000000          RMCS1
813 041170 040222          CI8
814          ;
815 041172 106126          ;
816 041174 100403          ROLB     (SP)+      ;IS 'IE'=1?
817 041176 104001          BMI     1$          ;YES, NO DRIVES TO CHECK
818 041200 004737 043402  EMT      1          ;REPORT AN ILLEGAL INTERRUPT
819 041204 000207          JSR      PC,SET.IE  ;SET INTERRUPT ENABLE
820 041206 005046          1$:     RTS      PC          ;RETURN
821 041210 110316          2$:     CLR      -(SP)    ;PROCESS ALL DRIVES THAT HAVE
822 041212 012703 000001  MOVB     R3,(SP)    ;AN 'ATA'=1
823 041216 005001          MOV      #1,R3
824 041220 030316          CLR      R1
825 041222 001005          SC3:    BIT      R3,(SP) ;ATA=1?
826 041224 005201          BNE     SC5         ;YES
827 041226 106303          SC4:    INC      R1     ;MOVE TO THE NEXT DRIVE
828 041230 001373          ASLB     R3
829 041232 005726          BNE     SC3         ;BR IF MORE TO CHECK?
          TST      (SP)+ ;CLEAN OFF THE STACK

```

830	041234	000207			RTS	PC	:RETURN TO USER
831	041236	105761	035654		TSTB	DPINT(R1)	:INITIALIZING THE DRIVE ?
832	041242	001402			BEQ	1\$:BR IF NOT
833	041244	000137	042152		JMP	SC13	:PROCESS THE DRIVE
834	041250	105761	035664		TSTB	DPRQS(R1)	:PORT REQUEST OUTSTANDING ?
835	041254	001402			BEQ	2\$:BR IF NOT
836	041256	000137	042152		JMP	SC13	:START THE OUTSTANDING COMMAND
837	041262	105761	035634		TSTB	DRVSTA(R1)	:CHECK THE DRIVE STATUS
838	041266	003023			BGT	5\$:BR IF ONLINE
839	041270	105761	035702		TSTB	ULDFLG(R1)	:UNLOAD IN PROGRESS?
840	041274	003420			BLE	5\$:BR IF NOT
841	041276	004737	044120		JSR	PC,GETREQ	:GET DPB POINTER
842	041302	004737	043264		JSR	PC,SVRH70	:SAVE THE RH/RM REGISTERS
843	041306	004737	042102		JSR	PC,SC12	:SAVE RMDS, RMER1, RMER2, AND RMMR2
844							:ALSO DO A DRIVE INIT (DRVINT)
845	041312	105761	035634		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
846	041316	003414			BLE	6\$:NO
847	041320	032737	040000	035614	BIT	#BIT14,RMERRS	:WAS THERE AN ERROR?
848	041326	001000			BNE	3\$:BR IF ERROR
849					JMP	SC11	:NO ERROR
850	041330	013705	035616		MOV	RMERRS+2,R5	:YES -- PICKUP RMER1 AND
851	041334	000500			BR	SC6A	:GO PROCESS THE ERROR
852	041336	105761	035624		TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
853	041342	001027			BNE	SC6	:BR IF EITHER
854	041344	004737	042102		JSR	PC,SC12	:SAVE RMDS, RMER1, RMER2, AND RMMR2
855							:ALSO DO A DRVINT
856	041350	105761	035654		TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
857	041354	001323			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
858	041356	105761	035634		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
859	041362	100412			BMI	7\$:BR IF UNSAFE
860	041364	032737	020000	035620	BIT	#BIT13,RMERRS+4	:ADDRESS PLUG CHANGED ?
861	041372	001011			BNE	8\$:BR IF YES
862					MOV	#113,-(SP)	:RELEASE COMMAND
863	041374	012746	000111		MOV	#111,-(SP)	:DRIVE CLEAR
864	041400	004037	043076		JSR	RO,WRT.RM	:WRITE THE COMMAND INTO RMCS1
865	041404	000000			RMCS1		:REGISTER INDEX
866	041406	041742			SC8		:PARITY EXIT ADDRESS
867	041410	011605			MOV	(SP),R5	:PICKUP (RMAS) BEFORE THE ERROR ^ALL
868	041412	104002			EMT	2	:REPORT THE UNEXPECTED ATTENTION
869	041414	000703			BR	SC4	:GO CHECK FOR MORE ATA'S
870	041416						
	041416	104005			EMT	5	:REPORT THE ADDRESS PLUG CHANGE
871	041420	000701			BR	SC4	:CHECK FOR MORE DRIVES
872	041422	006301			ASL	R1	:SETUP TO ADDRESS WORDS
873	041424	012761	177777	035726	MOV	#-1,TIMER(R1)	:STOP THE TIMER
874	041432	006201			ASR	R1	:RESTORE THE DRIVE ADDRESS
875	041434	004737	044120		JSR	PC,GETREQ	:GET THE DPB POINTER FROM THE QUEUE
876	041440	010164	000010		MOV	R1,RMCS2(R4)	:SELECT DRIVE
877	041444	000137	041772		JMP	SC11	:PROCESS THE SEARCH
878	041450	004037	042722		JSR	RO,RD.RM	:READ THE RM'S STATUS REG.
879	041454	000012			RMDS		
880	041456	041742			SC8		
881	041460	011605			MOV	(SP),R5	:AND PUT IT IN R5
882	041462	006126			ROL	(SP)+	:WAS THERE AN ERROR?
883	041464	100407			BMI	1\$:BR IF ERROR
884	041466	105761	035624		TSTB	DRVACT(R1)	:CHECK DRIVE'S STATE
885	041472	003137			BGT	SC11	:BR IF DRIVE ACTIVE WITH ORDER


```

886 041474 052762 100210 000016      BIS      #BIT15!BIT07!BIT03,16(R2)      ;INFORM USER OF ERROR RECOVER COMPLETION
887 041502 000470                      BR      SC7
888 041504 004037 042722      1$:     JSR      R0,RD.RM      ;READ ERROR REGISTER #1
889 041510 000014                      RMER1
890 041512 041742                      SC8
891 041514 012605                      MOV      (SP)+,R5      ;AND SAVE IT IN R5
892 041516 004737 043264      JSR      PC,SVRH70     ;SAVE RH/RM REGISTERS
893 041522 012746 000111      MOV      #111,-(SP)    ;ISSUE A DRIVE CLEAR
894 041526 004037 043076      JSR      R0,WRT.RM
895 041532 000000                      RMCS1
896 041534 041742                      SC8
897 041536 006105                      SC6A:   ROL      R5      ;WAS 'UNSAFE' CONDITION =1?
898 041540 100406                      BMI      1$           ;BR IF YES
899 041542 005702                      TST      R2           ;ANYTHING IN QUEUE ?
900 041544 001447                      BEQ      SC7          ;BR IF NOT
901 041546 052762 100240 000016      BIS      #BIT15!BIT07.BIT05,16(R2)    ;INFORM USER OF ERROR
902 041554 000443                      BR      SC7
903 041556 004037 042722      1$:     JSR      R0,RD.RM      ;READ DRIVE STATUS REG. #1
904 041562 000012                      RMDS
905 041564 041742                      SC8
906 041566 011605                      MOV      (SP),R5      ;SAVE RMDS IN R5
907 041570 006126                      ROL      (SP)+        ;'ERR'=1?
908 041572 100011                      BPL      2$           ;BR IF NO--UNSAFE CLEARED
909 041574 112761 177777 035634      MOVVB   #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
910 041602 004737 043264      JSR      PC,SVRH70     ;SAVE RH/RM REGISTERS
911 041606 052762 110000 000016      BIS      #BIT15!BIT12,16(R2)          ;INFORM USER OF UNSAFE ERROR
912 041614 000423                      BR      SC7
913 041616 032705 010000      2$:     BIT      #BIT12,R5      ;'MOL' = 1 ?
914 041622 001015                      BNE      3$           ;BR IF YES
915 041624 112761 177777 035624      MOVVB   #-1,DRVACT(R1) ;ACTIVE ERROR RECOVER
916 041632 112761 000001 035634      MOVVB   #1,DRVSTA(R1)  ;ONLINE
917 041640 006301                      ASL      R1
918 041642 012761 072460 035726      MOV      #30000.,TIMER(R1) ;START 30 SECOND TIMER
919 041650 006201                      ASR      R1
920 041652 000137 041224                      JMP      SC4
921 041656 052762 100220 000016      3$:     BIS      #BIT15!BIT07!BIT04,16(R2)    ;INFORM USER OF ERROR
922 041664 105061 035624      SC7:   CLRB   DRVACT(R1) ;DRIVE IS IDLE
923                                     ;     JSR      PC,EMPTYQ   ;DUMP THE QUEUE
924 041670 004737 044142                                     ;     JSR      PC,POPQUE   ;REMOVE THE QUEUE
925 041674 105761 035702                                     ;     TSTB   ULDFLG(R1)   ;UNLOAD IN RMOGRESS OR QUEUE?
926 041700 003002                                     ;     BGT      1$         ;BR IF NOT
927 041702 105061 035702                                     ;     CLRB   ULDFLG(R1)  ;CLEAR UNLOAD FLAG
928 041706 116164 035750 000016      1$:     MOVVB   ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
929 041714 105761 035634                                     ;     TSTB   DRVSTA(R1)  ;IS THE DRIVE UNSAFE ?
930 041720 100406                                     ;     BMI      2$         ;BR IF IT IS
931                                     ;     MOV      #113,-(SP) ;RELEASE COMMAND
932 041722 012746 000111                                     ;     MOV      #111,-(SP) ;DRIVE CLEAR COMMAND
933 041726 004037 043076                                     ;     JSR      R0,WRT.RM  ;WRITE THE COMMAND INTO RPCS1
934 041732 000000                                     ;     RMCS1 ;REGISTER INDEX
935 041734 041742                                     ;     SC8 ;PARITY EXIT ADDRESS
936 041736 000137 041224      2$:     JMP      SC4 ;CHECK FOR MORE DRIVES
937 041742 105761 035624      SC8:   TSTB   DRVACT(R1) ;IS DRIVE IDLE?
938 041746 001405                                     ;     BEQ      1$         ;YES
939 041750 004737 044120                                     ;     JSR      PC,GETREQ  ;GET DPB POINTER
940 041754 004737 040122                                     ;     JSR      PC,C17    ;PROCESS THE PARITY ERROR
941 041760 000402                                     ;     BR      2$         ;CONTINUE
942 041762      1$:

```

```

943      ;      JSR      PC,C17      ;PROCESS THE PARITY ERROR
944 041762 004737 040144      JSR      PC,C17B     ;PROCESS THE UNCORRECTABLE PARITY ERROR
945 041766 000137 041224      2$:      JMP      SC4       ;CHECK MORE DRIVES
946 041772 105761 035702      SC11:   TSTB     ULDFLG(R1)  ;'UNLOAD IN PROGRESS'?
947 041776 003402              BLE      1$        ;BR IF NO
948 042000 105061 035702      CLRB     ULDFLG(R1)  ;CLEAR UNLOAD FLAG
949 042004 105061 035624      1$:      CLRB     DRVACT(R1) ;SET DRIVE IDLE
950 042010 136137 035750 035676 BITB     ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
951                               ;AN I/O COMMAND?
952 042016 001012              BNE      2$        ;BR IF YES
953 042020 004737 044142      JSR      PC,POPQUE   ;REMOVE REQUEST FROM QUEUE
954 042024 052762 000200 000016 BIS      #BIT07,16(R2) ;SET 'DONE' BIT
955 042032 005737 035722      TST     SAVEFG      ;SAVE THE REGISTERS?
956 042036 100002              BPL      2$        ;BR IF NO
957 042040 004737 043264      JSR      PC,SVRH70   ;YES--SAVE ALL OF THE RH/RM REG'S
958 042044 116164 035750 000016 2$:      MOVB     ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BI
959 042052 146137 035750 035676 BICB     ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
960 042060 006301              ASL      R1         ;WORD INDEX
961 042062 012761 177777 035726 MOV      #-1,TIMER(R1) ;STOP CLOCK
962 042070 006201              ASR      R1         ;RESTORE R1
963 042072 004737 037042      JSR      PC,OPT      ;START A REQUEST
964 042076 000137 041224      JMP      SC4       ;CHECK FOR MORE DRIVES
965 042102 010164 000010      SC12:   MOV      R1,RMCS2(R4) ;SELECT DRIVE
966 042106 016437 000012 035614 MOV      RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
967 042114 016437 000014 035616 MOV      RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
968 042122 016437 000042 035620 MOV      RMER2(R4),RMERRS+4
969 042130 016437 000040 035622 MOV      RMMR2(R4),RMERRS+6
970 042136 004037 036212      JSR      R0,DRVINT  ;INIT. THE STATE OF THE DRIVE
971 042142 000401              BR      1$        ;TAKE ERROR EXIT
972 042144 000207              RTS      PC        ;RETURN
973 042146 005726      1$:      TST     (SP)+      ;POP PC OFF OF THE STACK
974 042150 000674              BR      SC8       ;PROCESS THE PARITY ERROR
975 042152 006301      SC13:   ASL      R1         ;SETUP TO ADDRESS WORDS
976 042154 012761 177777 035726 MOV      #-1,TIMER(R1) ;STOP THE TIMER
977 042162 006201              ASR      R1         ;
978 042164 010164 000010      MOV      R1,RMCS2(R4) ;SELECT THE DRIVE
979 042170 116164 035750 000016 1$:      MOVB     ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
980 042176 105761 035654      TSTB     DPINT(R1)  ;INITIALIZING THE DRIVE ?
981 042202 001424              BEQ      2$        ;BR IF NOT
982 042204 105061 035654      CLRB     DPINT(R1)  ;CLEAR THE INIT INDICATOR
983 042210 004037 036212      JSR      R0,DRVINT  ;GO INIT THE DRIVE
984 042214 000240              NOP              ;DUMMY PARITY ERROR RETURN
985 042216 105761 035634      TSTB     DRVSTA(R1) ;DRIVE ONLINE ?
986 042222 003014              BGT      2$        ;BR IF YES -- START ORDER
987 042224 005702              TST     R2         ;QUEUE ENTRY FOR THE DRIVE
988 042226 001426              BEQ      3$        ;BR IF NOT
989 042230 004737 044120      JSR      PC,GETREQ   ;GET DPB ADDRESS
990 042234 052762 140000 000016 BIS      #BIT15!BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
991 042242 004737 043264      JSR      PC,SVRH70   ;SAVE THE REGISTERS
992                               ;EMPTY THE REQUEST QUEUE
993 042246 004737 044142      JSR      PC,EMPTYQ   ;REMOVE THE QUEUE
994 042252 000414              BR      3$        ;
995 042254 032764 004000 000000 2$:      BIT      #BIT11,RMCS1(R4) ;DVA SET ?
996 042262 001006              BNE     4$        ;SET THEN CALL OPT
997 042264 006301              ASL      R1         ;
998 042266 012761 060000 035726 MOV      #60000,TIMER(R1)
999 042274 006201              ASR      R1

```

```

1000 042276 000402          BR      3$
1001 042300 004737 037042  4$:   JSR   PC,OPT      ;START THE PENDING REQUEST
1002 042304 000137 041224  3$:   JMP   SC4           ;PROCESS OTHER DRIVES
1003
1004          ;RM TIMER ROUTINE
1005          ;CALL
1006          ;
1007          ;   MOV   #TIME,-(SP)      ;ELAPSED TIME IN MILLISECONDS ON THE STACK
1008          ;   JSR   PC,RMTMR      ;CALL RM05 TIME ROUTINE
1009 042310 005737 035700  RMTMR: TST   ACTDRV      ;CHECK 'ACTDRV & ACTSTR'
1010 042314 001027          BNE   4$             ;IF NON ZERO EXIT
1011 042316 112737 000001 035701  MOVB  #1,ACTSTR      ;SET 'ACTSTR'
1012 042324 104412          SAVREG              ;SAVE R0 - R5
1013 042326 005001          CLR   R1           ;START WITH DRIVE 0
1014 042330 005003          CLR   R3
1015 042332 005763 035726  1$:   TST   TIMER(R3)   ;IS THE TIMER RUNNING?
1016 042336 002406          BLT   2$           ;BR IF NO
1017 042340 166663 000002 035726  SUB   2(SP),TIMER(R3) ;COUNT THE INTERVAL
1018 042346 003002          BGT   2$           ;BR IF NO SOFTWARE TIMEOUT
1019 042350 004737 042400  JSR   PC,STO       ;CALL SOFTWARE TIMEOUT ROUTINE
1020 042354 005201  2$:   INC   R1           ;MOVE TO NEXT DRIVE
1021 042356 005723          TST   (R3)+
1022 042360 022701 000010  CMP   #8.,R1       ;OUT OF DRIVES?
1023 042364 003362          BGT   1$           ;BR IF NO
1024 042366 104413  3$:   RESREG              ;RESTORE R0 - R5
1025 042370 105037 035701  CLRB  ACTSTR       ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1026 042374 012616  4$:   MOV   (SP)+,(SP)   ;ADJUST THE STACK
1027 042376 000207          RTS   PC           ;RETURN
1028
1029          ;SOFTWARE TIMEOUT ROUTINE
1030          ;
1031          ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1032          ;OR GREATER
1033          ;
1034          ;CALL:
1035          ;   STO   #DRVNUM,R1      ;DRIVE NUMBER
1036          ;   JSR   PC,STO         ;CALL
1037          ;   RETURN
1038
1039 042400 010146  STO:  MOV   R1,-(SP)   ;SAVE R1
1040 042402 010246          MOV   R2,-(SP)   ;SAVE R2
1041 042404 010346          MOV   R3,-(SP)   ;SAVE R3
1042 042406 010446          MOV   R4,-(SP)   ;SAVE R4
1043 042410 013704 035762  MOV   RMADR,R4    ;GET ADDRESS OF 'RMCS1'
1044 042414 010164 000010  MOV   R1,RMCS2(R4) ;SELECT THE DRIVE
1045 042420 004037 042722  JSR   R0,RD.RM    ;READ 'DRIVE STATUS REG'
1046 042424 000012          RMDS
1047          ;
1048 042426 042710          STOS
1049 042430 105726          STO9
1050 042432 100436          TSTB  (SP)+
1051 042434 105761 035654  STO1: BMI   STO2      ;IS 'DRY'=1?
1052 042440 001033          TSTB  DPINT(R1)  ;BR IF YES
1053 042442 105761 035664  BNE   STO2        ;TRYING TO INTIALIZE THE DRIVE ?
1054 042446 001030          TSTB  DPRQS(R1)  ;BR IF YES
1055 042450 013702 035674  BNE   STO2        ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1056 042454 020137 035746  MOV   TRNSWT,R2   ;BR IF YES
1057          ;PICKUP TRANSFER WAIT QUEUE
1058          ;TRANSFER UNDERWAY CN THIS DRIVE?

```

```

1057 042460 001404          BEQ      1$          ;BR IF YES
1058 042462 000137 042710    JMP      ST09       ;IF NOT DON'T BOTHER DRIVES
1059 042466 004737 044120    JSR     PC,GETREQ   ;GET DPB ADDRESS
1060 042472 052762 101000 000016 1$:  BIS     #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
1061 042500 004737 043264    JSR     PC,SVRH70   ;SAVE RH/RM REGISTERS
1062          :          MOV     #CLR,RMCS2(R4) ;'INIT' THE MASS BUS
1063 042504 105061 035624    CLR     DRVACT(R1) ;DRIVE IS IDLE
1064 042510 105061 035702    CLR     ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
1065 042514 005037 035674    CLR     TRNSWT      ;CLEAR DPB ADDRESS
1066 042520 012737 177777 035746  MOV     #-1,DTUW    ;CLEAR THE TRANSFER DRIVE #
1067 042526 000470          BR      ST09        ;DON'T BOTHER OTHER DRIVES
1068 042530 116405 000016    ST02:  MOV     RMAS(R4),R5 ;READ ATTENTION REG
1069 042534 136105 035750    BIT     ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
1070 042540 001007          BNE     ST03        ;YES
1071 042542 105761 035654    TST     DPINT(R1)  ;TRYING TO INTIALIZE THE DRIVE ?
1072 042546 001021          BNE     ST06        ;BR IF YES - DRIVE NOT ONLINE
1073 042550 105761 035664    TST     DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1074 042554 001035          BNE     ST07        ;BR IF YES - NO RESPONSE TO REQUEST
1075 042556 000454          BR      ST09        ;OTHER WISE EXIT
1076 042560 105761 035654    ST03:  TST     DPINT(R1) ;INITIALIZING THE DRIVE ?
1077 042564 001003          BNE     1$         ;BR IF INIT PENDING
1078 042566 105761 035664    TST     DPRQS(R1) ;PORT REQUEST PENDING ?
1079 042572 001446          BEQ     ST09        ;BR IF NOT
1080 042574 012763 177777 035726 1$:  MOV     #-1,TIMER(R3) ;STOP THE TIMER
1081 042602 000442          BR      ST09        ;EXIT
1082 042604 004737 040222    ST05:  JSR     PC,C18    ;GO HANDLE THE PARITY ERROR
1083 042610 000437          BR      ST09
1084 042612 105061 035654    ST06:  CLR     DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
1085 042616 105061 035634    CLR     DRVSTA(R1) ;SET UNIT OFFLINE
1086 042622 012763 177777 035726  MOV     #-1,TIMER(R3) ;STOP THE TIMER
1087 042630 004737 044120    JSR     PC,GETREQ   ;GET THE DPB ADDRESS
1088 042634 005702          TST     R2          ;REQUEST IN QUEUE ?
1089 042636 001424          BEQ     ST09        ;BR IF NOT
1090 042640 052762 140000 000016  BIS     #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1091 042646 000414          BR      ST08
1092 042650 012763 177777 035726  ST07:  MOV     #-1,TIMER(R3) ;STOP THE TIMER
1093 042656 105061 035664    CLR     DPRQS(R1)  ;CLEAR PORT REQUEST INDICATOR
1094 042662 004737 044120    JSR     PC,GETREQ   ;GET DPB ADDRESS
1095 042666 005702          TST     R2          ;QUEUE ENTRY FOR DRIVE ?
1096 042670 001407          BEQ     ST09        ;BR IF NONE
1097 042672 012762 100004 000016  MOV     #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1098 042700 004737 044024    ST08:  JSR     PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
1099 042704 004737 043264    JSR     PC,SVRH70   ;SAVE THE REGISTERS
1100 042710 012604    ST09:  MOV     (SP)+,R4    ;RESTORE R4
1101 042712 012603    MOV     (SP)+,R3    ;RESTORE R3
1102 042714 012602    MOV     (SP)+,R2    ;RESTORE R2
1103 042716 012601    MOV     (SP)+,R1    ;RESTORE R1
1104 042720 000207    RTS     PC          ;RETURN
1105
1106          ;ROUTINE TO READ A RH/RM REGISTER
1107          ;
1108          ;CALL
1109          ;
1110          JSR     RO,RD.RM ;GO READ A REGISTER
1111          INDEX ;REG. INDEX FROM BASE
1112          ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
1113          ;
          ;AT THIS ADDRESS
          ;CONTENTS OF REG. IS ON THE STACK

```

```

1114
1115 042722 013737 035760 043064 RD.RM: MOV MCPEMX,RD.RM2 ;MAX. RETRYS ALLOWED
1116 042730 011646 MOV (SP),-(SP) ;SAVE R0 FOR RETURN
1117 042732 013737 035762 042746 MOV RMADR,RD.ADR ;FORM THE DESIRED ADDRESS
1118 042740 062037 042746 ADD (R0)+,RD.ADR ;USING THE BASE AND THE INDEX
1119 042744 013727 RD.RM1: MOV @ (PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RM DRIVE
1120 042746 000000 RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
1121 042750 000000 RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
1122 042752 013766 042750 000002 MOV RD.WRD,2(SP) ;RETURN IT TO THE USER
1123 042760 013746 035762 MOV RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
1124 042764 062716 000010 ADD #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
1125 042770 032736 010000 BIT #BIT12,@(SP, ;CHECK THE 'NED' BIT
1126 042774 001035 BNE RD.RM3 ;BR IF DRIVE NON-EXISTENT
1127 042776 017746 172760 MOV @RMADR,-(SP) ;READ RMCS1
1128 043002 032716 020000 BIT #BIT13,(SP) ;DID MCPE SET?
1129 043006 001002 BNE 1$ ;BR IF YES
1130 043010 022620 CMP (SP)+,(R0)+ ;ADJUST FOR RETURN
1131 043012 000430 BR RD.RM4 ;EXIT
1132 043014 1$:
043014 104003 EMT 3 ;REPORT 'MCPE' ERROR
1133 043016 005737 035746 TST DTUW ;DATA TRANSFER UNDERWAY?
1134 043022 100405 BMI 2$ ;NO
1135 043024 032716 040000 BIT #BIT14,(SP) ;'TRE' = 1 ?
1136 043030 001402 BEQ 2$ ;NO
1137 043032 005726 TST (SP)+ ;YES--CLEAN OFF THE STACK AND
1138 043034 000415 BR RD.RM3 ;TAKE THE FATAL ERROR EXIT
1139 043036 052716 040000 2$: BIS #BIT14,(SP) ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1140 043042 000316 SWAB (SP) ;POSITION BEFORE WRITING
1141 043044 013737 035762 043060 MOV RMADR,3$ ;FORM ADDRESS OF HIGH BYTE
1142 043052 005237 043060 INC 3$
1143 043056 112637 MOVB (SP)+,@(PC)+ ;WRITE THE HIGH BYTE OF RMCS1
1144 043060 000000 3$: .WORD 0 ;ADDRESS STORAGE
1145 043062 005327 DEC (PC)+ ;EXCEEDED MAX. RETRYS
1146 043064 000003 RD.RM2: .WORD 3
1147 043066 002326 BGE RD.RM1 ;BR IF NO
1148 043070 011000 RD.RM3: MOV (R0),R0 ;FATAL ERROR EXIT
1149 043072 012616 MOV (SP)+,(SP)
1150 043074 000200 RD.RM4: RTS R0
1151
1152 ;ROUTINE TO WRITE A REGISTER
1153 ;CALL
1154 ;
1155 ; MOV DATA,-(SP) ;DATA TO BE LOADED ON THE STACK
1156 ; JSR R0,WRT.RM ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1157 ; INDEX ;INDEX OF THE REGISTER TO BE LOADED
1158 ; ERRADR ;ADDRESS TO RETURN TO ON AN ERROR
1159 ; RETURN ;ERROR FREE RETURN
1160
1161 043076 013737 035760 043250 WRT.RM: MOV MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
1162 043104 016637 000002 043164 MOV 2(SP),WRT.WD ;SAVE THE WORD TO WRITE
1163 043112 012616 MOV (SP)+,(SP) ;ADJUST THE STACK
1164 043114 012037 043166 MOV (R0)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
1165 043120 001015 BNE 1$ ;BR IF NOT RMCS1
1166 043122 122737 000150 043164 CMPB #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
1167 043130 002411 BLT 1$ ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1168 043132 004037 042722 JSR R0,RD.RM ;NO---COMBINE A16&A17, & PSEL WITH
1169 043136 000000 RMCS1 ;THE COMMAND BEFORE SENDING IT TO

```

```

1170 043140 043254          WRT.R3          ;THE RH/RM
1171 043142 000316          SWAB          (SP)
1172 043144 042716 177770    BIC          #^C7,(SP)
1173 043150 112637 043165    MOV          (SP)+,WRT.WD+1
1174 043154 063737 035762 043166 1$: ADD          RMADR,WRT.AD          ;FORM THE ADDRESS OF THE DISK REG.
1175 043162 012737          WRT.R1: MOV          (PC)+,@(PC)+          ;LOAD THE DESIRED REG.
1176 043164 000000          WRT.WD: .WORD          0          ;WORD TO WRITE GOES HERE
1177 043166 000000          WRT.AD: .WORD          0          ;ADDRESS IS FORMED HERE
1178 043170 013746 035762    MOV          RMADR,-(SP)          ;PUT THE ADDRESS ON THE STACK
1179 043174 062716 000010    ADD          #RMCS2,(SP)          ;FORM THE ADDRESS OF RMCS2
1180 043200 032736 010000    BIT          #BIT12,@(SP)+          ;CHECK THE 'NED' BIT
1181 043204 001023          BNE          WRT.R3          ;BR IF DRIVE NON-EXISTENT
1182 043206 004037 042722    JSR          RO,RD.RM          ;CHECK FOR PARITY ERROR ON WRITE
1183 043212 000014          RMER1
1184 043214 043254          WRT.R3
1185 043216 032726 000010    BIT          #BIT03,(SP)+
1186 043222 001416          BEQ          WRT.R4          ;BR IF 'PAR=0'
1187 043224 016037 177776 043236 1$: MOV          -2(RO),1$          ;PICKUP THE INDEX
1188 043232 004037 042722          JSR          RO,RD.RM          ;READ THE REG.
1189 043236 000000          .WORD          0          ;REG. INDEX
1190 043240 043254          WRT.R3          ;RETURN TO THIS ADDRESS ON ERROR
1191 043242 104004          EMT          4          ;REPORT THE PARITY ON WRITE ERROR
1192 043244 005726          TST          (SP)+          ;CLEAR OFF THE STACK
1193 043246 005327          DEC          (PC)+          ;DECREMENT THE ERROR COUNT
1194 043250 000003          WRT.R2: .WORD          3          ;RETRY COUNTER
1195 043252 002343          BGE          WRT.R1          ;TRY AGAIN IF NOT FINISHED
1196 043254 011000          WRT.R3: MOV          (RO),RO          ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1197 043256 000401          BR          WRT.R5          ;EXIT
1198 043260 005720          WRT.R4: TST          (RO)+          ;ADJUST FOR ERROR FREE EXIT
1199 043262 000200          WRT.R5: RTS          RO
1200
1201          ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1202          ;
1203          ;CALL
1204          ;
1205          ;      MOV          #DPBNUM,R2          ;DPB POINTER TO R2
1206          ;      JSR          PC,SVRH70          ;SAVE THE DRIVES REG'S
1207 043264 104412          SVRH70: SAVREG
1208 043266 005702          TST          R2          ;SAVE R0 - R5
1209 043270 001442          BEQ          6$          ;QUEUE ENTRY FOR THE DRIVE ?
1210 043272 013704 035762    MOV          RMADR,R4          ;BR IF NONE
1211 043276 111264 000010    MOV          (R2),RMCS2(R4)          ;SELECT DRIVE
1212 043302 016203 000014    MOV          14(R2),R3          ;GET THE ERROR TABLE POINTER
1213 043306 001433          BEQ          6$          ;EXIT IF NO ADDRESS
1214 043310 005037 043344    CLR          3$          ;COUNTER & POINTER
1215 043314 023727 043344 000022 1$: CMP          3$,#RMD8          ;REACHED THE BUFFER REGISTER ?
1216 043322 001006          BNE          2$          ;BR IF NOT
1217 043324 032764 000200 000010  BIT          #BIT07,RMCS2(R4)          ;'OR' SET ?
1218 043332 001002          BNE          2$          ;BR IF SET
1219 043334 005023          CLR          (R3)+          ;STORE RMD8 AS ZEROES
1220 043336 000405          BR          4$          ;CONTINUE
1221 043340 004037 042722    2$: JSR          RO,RD.RM          ;READ THE SELECTED REGISTER
1222 043344 000000          3$: .WORD          0          ;REGISTER INDEX
1223 043346 043372          5$          ;ERROR RETURN ADDRESS
1224 043350 012623          MOV          (SP)+,(R3)+          ;STORE THE REGISTER CONTENTS
1225 043352 023727 043344 000046 4$: CMP          3$,#RMEC2          ;REACHED THE END ?
1226 043360 001406          BEQ          6$          ;BR IF YES

```

```

1227 043362 062737 000002 043344      ADD    #2,3$      ;INCREMENT THE REGISTER INDEX
1228 043370 000751                    BR     1$        ;CONTINUE READING THE REGISTERS
1229 043372 004737 040122      5$:   JSR    PC,C17 ;PROCESS THE UNCORRECTABLE PARITY ERROR
1230 043376 104413                    6$:   RESREG ;RESTORE R0 - R5
1231 043400 000207                    RTS    PC        ;RETURN
1232
1233      ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A 'TRE'
1234      ;CALL
1235      ;:
1236      ;:   MOV    #DRVNUM,R1      ;DRIVE NUMBER TO R1
1237      ;:   JSR    PC,SET.IE     ;SET 'IE'
1238      ;:   RETURN
1239 043402 010446                    SET.IE: MOV   R4,-(SP)      ;SAVE R4
1240 043404 013704 035762          MOV   RMADR,R4      ;PICKUP ADDRESS OF RMCS1
1241 043410 010164 000010          MOV   R1,RMCS2(R4) ;SELECT DRIVE
1242 043414 011446                    MOV   (R4),-(SP)   ;READ RMCS1
1243 043416 052716 040000          BIS   #BIT14,(SP) ;SET THE 'TRE' BIT OF THE WORD READ
1244 043422 000316                    SWAB  (SP)        ;ADJUST FOR DATO
1245 043424 112714 000100          MOVB  #BIT06,(R4) ;SET 'IE'
1246 043430 032764 010000 000010  BIT   #BIT12,RMCS2(R4) ;IS 'NED'=1?
1247 043436 001002                    BNE   1$         ;YES--CLEAR 'TRE'
1248 043440 005726                    TST   (SP)+      ;CLEAN OFF THE STACK
1249 043442 000402                    BR    2$
1250 043444 112664 000001      1$:   MOVB  (SP)+,1(R4) ;CLEAR 'TRE'
1251 043450 012604                    2$:   MOV   (SP)+,R4 ;RESTORE R4
1252 043452 000207                    RTS    PC        ;RETURN TO CALLER
1253
1254      ;QUEUE COUNT
1255 043454      000      QCNT:  .BYTE  0      ;DRIVE 0
1256 043455      000      .BYTE  0      ;DRIVE 1
1257 043456      000      .BYTE  0      ;DRIVE 2
1258 043457      000      .BYTE  0      ;DRIVE 3
1259 043460      000      .BYTE  0      ;DRIVE 4
1260 043461      000      .BYTE  0      ;DRIVE 5
1261 043462      000      .BYTE  0      ;DRIVE 6
1262 043463      000      .BYTE  0      ;DRIVE 7
1263
1264      ;QUEUE INPUT POINTERS
1265
1266 043464 043546      QINPT: .WORD  QDRV0      ;DRIVE 0
1267 043466 043566      .WORD  QDRV1      ;DRIVE 1
1268 043470 043606      .WORD  QDRV2      ;DRIVE 2
1269 043472 043626      .WORD  QDRV3      ;DRIVE 3
1270 043474 043646      .WORD  QDRV4      ;DRIVE 4
1271 043476 043666      .WORD  QDRV5      ;DRIVE 5
1272 043500 043706      .WORD  QDRV6      ;DRIVE 6
1273 043502 043726      .WORD  QDRV7      ;DRIVE 7
1274
1275      ;QUEUE OUTPUT POINTERS
1276
1277 043504 043546      QOUTPT: .WORD  QDRV0      ;DRIVE 0
1278 043506 043566      .WORD  QDRV1      ;DRIVE 1
1279 043510 043606      .WORD  QDRV2      ;DRIVE 2
1280 043512 043626      .WORD  QDRV3      ;DRIVE 3
1281 043514 043646      .WORD  QDRV4      ;DRIVE 4
1282 043516 043666      .WORD  QDRV5      ;DRIVE 5
1283 043520 043706      .WORD  QDRV6      ;DRIVE 6

```

```

1284 043522 043726          .WORD  QDRV7          ;DRIVE 7
1285
1286 043524 043546          QSTART: .WORD  QDRV0          ;DRIVE 0 START ADDRESS
1287 043526 043566          QSTOP:  .WORD  QDRV1          ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1288 043530 043606          .WORD  QDRV2          ;STOP DRIVE 1--START DRIVE 2
1289 043532 043626          .WORD  QDRV3          ;STOP DRIVE 2--START DRIVE 3
1290 043534 043646          .WORD  QDRV4          ;STOP DRIVE 3--START DRIVE 4
1291 043536 043666          .WORD  QDRV5          ;STOP DRIVE 4--START DRIVE 5
1292 043540 043706          .WORD  QDRV6          ;STOP DRIVE 5--START DRIVE 6
1293 043542 043726          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
1294 043544 043746          .WORD  QTERM          ;STOP DRIVE 7
1295
1296          ;DRIVE REQUEST QUEUES
1297
1298 043546          QDRV0:  .BLKW  10
1299 043566          QDRV1:  .BLKW  10
1300 043606          QDRV2:  .BLKW  10
1301 043626          QDRV3:  .BLKW  10
1302 043646          QDRV4:  .BLKW  10
1303 043666          QDRV5:  .BLKW  10
1304 043706          QDRV6:  .BLKW  10
1305 043726          QDRV7:  .BLKW  10
1306          043746          QTERM=.
1307
1308          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1309
1310          ;CALL
1311          ;
1312          JSR      PC,CLRQUE
1313 043746 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
1314 043750 012702 043454          MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
1315 043754 005022          CLR      (R2)+          ;DRIVES 0 & 1
1316 043756 005022          CLR      (R2)+          ;DRIVES 2 & 3
1317 043760 005022          CLR      (R2)+          ;DRIVES 4 & 5
1318 043762 005022          CLR      (R2)+          ;DRIVES 6 & 7
1319 043764 012703 000010          MOV      #8,R3          ;MOVE THE STARTING
1320 043770 012701 043524          MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
1321 043774 012122          1$:  MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
1322 043776 005303          DEC      R3
1323 044000 001375          BNE      1$
1324 044002 012703 000010          MOV      #8,R3          ;MOVE THE STARTING ADDRESS
1325 044006 012701 043524          MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
1326 044012 012122          2$:  MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
1327 044014 005303          DEC      R3
1328 044016 001375          BNE      2$
1329 044020 104413          RESREG
1330 044022 000207          RTS      PC          ;RESTORE R0 - R5
1331
1332          ;EMPTY THE QUEUE SPECIFIED BY R1
1333
1334          ;CALL
1335          ;
1336          MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
1337          JSR      PC,EMPTYQ
1338 044024 105061 043454          EMPTYQ: CLRB      QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
1339 044030 006301          ASL      R1
1340 044032 016161 043464 043504          MOV      QINPT(R1),QOUTPT(R1)          ;SET OUTPUT QUEUE POINTER=INPUT POINTER

```



```

1341 044040 006201          ASR      R1
1342 044042 000207          RTS      PC
1343
1344          ;ROUTINE TO PUT A REQUEST IN QUEUE
1345          ;CALL
1346          ;
1347          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER
1348          ;      MOV      #DPB,R2           ;ADDRESS OF PARAMETER BLOCK
1349          ;      JSR      RO,DRVQUE         ;GO PUT REQUEST IN QUEUE
1350          ;      RETURN1          ;RETURN HERE IF QUEUE IS FULL
1351          ;      RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
1352
1353 044044 122761 000010 043454 DRVQUE: CMPB    #10,QCNT(R1)      ;IS QUEUE FULL?
1354 044052 001421          BEQ     2$          ;BR IF YES-TAKE RETURN1
1355 044054 105261 043454          INCB   QCNT(R1)      ;INCREMENT QUEUE COUNT
1356 044060 006301          ASL    R1
1357 044062 010271 043464          MOV    R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1358 044066 062761 000002 043464          ADD    #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
1359 044074 026161 043464 043526          CMP    QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1360 044102 001003          BNE   1$          ;BR IF NO
1361 044104 016161 043524 043464          MOV    QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1362 044112 006201          1$:   ASR    R1
1363 044114 005720          TST   (R0)+        ;TAKE RETURN 2
1364 044116 000200          2$:   RTS    RO          ;RETURN TO USER
1365
1366          ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1367          ;CALL
1368          ;
1369          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
1370          ;      JSR      PC,GETREQ         ;GO GET THE REQUEST
1371          ;      RETURN          ;R2='DPB' ADDRESS OF THE REQUEST
1372          ;      ;R2=0 IF NO REQUEST IN QUEUE
1373
1374 044120 005002          GETREQ: CLR    R2
1375 044122 105761 043454          TSTB  QCNT(R1)      ;IS THERE ANY REQUEST IN QUEUE?
1376 044126 001404          BEQ   2$          ;NO
1377 044130 006301          1$:   ASL    R1
1378 044132 017102 043504          MOV    @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1379 044136 006201          ASR   R1
1380 044140 000207          2$:   RTS    PC          ;RETURN TO USER
1381
1382          ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1383          ;CALL
1384          ;
1385          ;      MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
1386          ;      JSR      PC,POPQUE         ;CALL TO REMOVE REQUEST
1387          ;      RETURN          ;R2=ADDRESS OF DPB REMOVED
1388
1389 044142 105361 043454          POPQUE: DECB   QCNT(R1)      ;DECREMENT QUEUE COUNT
1390 044146 006301          ASL   R1
1391 044150 017102 043504          MOV    @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1392 044154 005071 043504          CLR    @QOUTPT(R1)    ;REMOVE DPB ADDRESS FROM THE QUEUE
1393 044160 062761 000002 043504          ADD    #2,QOUTPT(R1)  ;UPDATE THE QUEUE POINTER
1394 044166 026161 043504 043526          CMP    QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1395 044174 001003          BNE   1$          ;NO--BR TO EXIT
1396 044176 016161 043524 043504          MOV    QSTART(R1),QOUTPT(R1) ;YES--RESET THE PCINTER
1397 044204 006201          1$:   ASR    R1
  
```

```

1398 044206 000207          RTS      PC          ;RETURN TO USER
1399
1416
1417          .SBTTL  DATA, CONTROL, & STATUS BLOCKS
1418
1419          ;BLOCK LOCATION EQUATE STATEMENTS
1420
1421          000001          $FMT      =          1          ;FMT,HCI,ECI OR OFFSET CODE
1422          000002          $COMND   =          $FMT+1        ;OPERATION CODE
1423          000003          $PSEL    =          $FMT+2        ;PORT SELECT & BITS A16, A17
1424          000004          $WRDM    =          $FMT+3        ;WORD COUNT (2'S COMP)
1425          000006          $BUF     =          $FMT+5        ;BUFFER ADDR OR REGISTER TABLE POINTER
1426          000010          $SEC     =          $FMT+7        ;SECTOR ADDRESS OR 1ST REG ADDR
1427          000011          $TRK    =          $FMT+10       ;TRACK ADDRESS OF LAST REG ADDR
1428          000012          $CYL    =          $FMT+11       ;CYLINDER ADDR
1429          000014          $REG     =          $FMT+13       ;REGISTER STORAGE (IF ERROR)
1430          000016          $STATUS  =          $FMT+15       ;STATUS WORD (SET BY DRIVER)
1431
1432          ;DRIVE'S HISTORY AND CURRENT INDICATOR STORAGE EQUATES
1433
1434          000020          $WRDL    =          $FMT+17        ;WORD COUNT (NOT 2'S COMP)
1435          000022          $SSEC    =          $WRDL+2        ;SECTOR SIZE FOR CURRENT OPERATION
1436          000024          $CODE    =          $WRDL+4        ;PRESENT COMMAND SELECTION CODE
1437          000026          $PACK    =          $WRDL+6        ;WRITE DATA PACK INDICATOR
1438          000027          $PREVO   =          $WRDL+7        ;PREVIOUS COMMAND SELECTION CODE
1439          000030          $PATTC   =          $WRDL+10       ;PATTERN CODE
1440          000032          $PREVA   =          $WRDL+12       ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
1441          000036          $OPERC   =          $WRDL+16       ;OPERATION COUNT
1442          000042          $POSIT   =          $WRDL+22       ;SEEK COUNT
1443          000046          $TRANS   =          $WRDL+26       ;TOTAL BITS XFERED COUNT (R & W)
1444          000052          $READ    =          $WRDL+32       ;TOTAL BITS READ COUNT
1445          000056          $TOTAL   =          $WRDL+36       ;TOTAL ERRORS (ALL TYPES) COUNT
1446          000060          $SOFT   =          $WRDL+40       ;'SOFT' ERROR COUNT
1447          000062          $HARD    =          $WRDL+42       ;'HARD' ERROR COUNT
1448          000064          $SKI     =          $WRDL+44       ;'SKI' OR 'OCYL' ERROR COUNT
1449          000066          $MISPO   =          $WRDL+46       ;PROG DETECTED MISPOSITIONING ERROR S COUNT
1450          000070          $PASSC   =          $WRDL+50       ;PASS COUNTER
1451          000072          $FAIR    =          $WRDL+52       ;OPERATION QUEUE 'FAIRNESS' COUNT
1452
1453          ;INDEX EQUATES TO THE NEXT OPERATION PARAMETERS
1454
1455          000074          $NCODE    =          $WRDL+54        ;NEXT OPERATION CODE
1456          000075          $NPATC   =          $NCODE+1        ;NEXT PATTERN
1457          000076          $NSEC     =          $NCODE+2        ;NEXT SECTOR
1458          000077          $NTRK    =          $NCODE+3        ;NEXT TRACK
1459          000100          $NCYL    =          $NCODE+4        ;NEXT CYLINDER
1460          000102          $NWRDL   =          $NCODE+6        ;NEXT BUFFER SIZE
1461          000104          $NEXT     =          $NCODE+10       ;PARAMETER SELECTION INDICATOR
1462
1463          ;INDEX EQUATES FOR MAXIMUM/MINIMUM ADDRESSES
1464
1465          000106          MAXCYL   =          $NCODE+12       ;MAXIMUM CYLINDER ADDRESS
1466          000110          MINCYL   =          MAXCYL+2        ;MINIMUM CYLINDER ADDRESS
1467          000112          MAXTRK   =          MAXCYL+4        ;MAXIMUM TRACK ADDRESS
1468          000114          MINTRK   =          MAXCYL+6        ;MINIMUM TRACK ADDRESS
1469          000116          MAXSEC   =          MAXCYL+10       ;MAXIMUM SECTOR ADDRESS
1470          000120          MINSEC   =          MAXCYL+12       ;MINIMUM SECTOR ADDRESS
  
```

```

1471      000122      $FIRST =      MAXCYL+14      ;FIRST OPERATION INDICATOR
1472
1473      ;BAD SECTOR/TRACK ADDRESS STORAGE AREA INDEX EQUATE
1474
1475      000124      $BDSEC =      MAXCYL+16      ;BAD SECTOR STORAGE TABLE PLUS TERMINATOR
1476
1477      ;DRIVE ID AREA INDEX EQUATE
1478
1479      002106      $DRVID =      $BDSEC+<126.*8.>+2      ;DRIVE ID
1480
1481      ;RH/RM REGISTER EQUATES
1482
1483      002116      $RMCS1 =      $DRVID+10      ;RM REGISTER STORAGE
1484      002120      $RMWC =      $RMCS1+2
1485      002122      $RMBA =      $RMCS1+4
1486      002124      $RMDA =      $RMCS1+6
1487      002126      $RMCS2 =      $RMCS1+10
1488      002130      $RMDS =      $RMCS1+12
1489      002132      $RMER1 =      $RMCS1+14
1490      002134      $RMAS =      $RMCS1+16
1491      002136      $RMLA =      $RMCS1+20
1492      002140      $RMDB =      $RMCS1+22
1493      002142      $RMMR1 =      $RMCS1+24
1494      002144      $RMDT =      $RMCS1+26
1495      002146      $RMSN =      $RMCS1+30
1496      002150      $RMOF =      $RMCS1+32
1497      002152      $RMDC =      $RMCS1+34
1498      002154      $RMHR =      $RMCS1+36
1499      002156      $RMMR2 =      $RMCS1+40
1500      002160      $RMER2 =      $RMCS1+42
1501      002162      $RMEC1 =      $RMCS1+44
1502      002164      $RMEC2 =      $RMCS1+46
1503
1511      ;BLOCK FOR DRIVE 0
044210      000      000      DRIVE0: .BYTE 0,0      ;DRIVE NUMBER 0
          .BLKW 5
          .WORD .+$RMCS1-$REG
          .BLKB $RMEC2-$REG
044224      046326
          ;BLOCK FOR DRIVE 1
046376      001      000      DRIVE1: .BYTE 1,0      ;DRIVE NUMBER 1
          .BLKW 5
          .WORD .+$RMCS1-$REG
          .BLKB $RMEC2-$REG
046412      050514
          ;BLOCK FOR DRIVE 2
050564      002      000      DRIVE2: .BYTE 2,0      ;DRIVE NUMBER 2
          .BLKW 5
          .WORD .+$RMCS1-$REG
          .BLKB $RMEC2-$REG
050600      052702
          ;BLOCK FOR DRIVE 3
052752      003      000      DRIVE3: .BYTE 3,0      ;DRIVE NUMBER 3
          .BLKW 5
          .WORD .+$RMCS1-$REG
          .BLKB $RMEC2-$REG
052766      055070

```

```

055140 004 000 ;BLOCK FOR DRIVE 4
DRIVE4: .BYTE 4,0 ;DRIVE NUMBER 4
        .BLKW 5
        .WORD .+$RMCS1-$REG
        .BLKB $RMEC2-$REG

057326 005 000 ;BLOCK FOR DRIVE 5
DRIVE5: .BYTE 5,0 ;DRIVE NUMBER 5
        .BLKW 5
        .WORD .+$RMCS1-$REG
        .BLKB $RMEC2-$REG

061514 006 000 ;BLOCK FOR DRIVE 6
DRIVE6: .BYTE 6,0 ;DRIVE NUMBER 6
        .BLKW 5
        .WORD .+$RMCS1-$REG
        .BLKB $RMEC2-$REG

063702 007 000 ;BLOCK FOR DRIVE 7
DRIVE7: .BYTE 7,0 ;DRIVE NUMBER 7
        .BLKW 5
        .WORD .+$RMCS1-$REG
        .BLKB $RMEC2-$REG

```

1512
1513
1514
1515 066070 000
1516 066071 000
1517 066072 000
1518 066073 000
1519 066074 177776
1520 066076 076746
1521 066100 000
1522 066101 000
1523 066102 000000
1524 066104 066110
1525 066106 000000
1526
1527 066110
1528

;GENERAL PURPOSE PARAMETER BLOCK

```

GENDPB: .BYTE 0 ;DRIVER PARAMETER BLOCK, DRIVE #
        .BYTE 0 ;OFFSET VALUE OR FMT16, HCI OR ECI
        .BYTE 0 ;COMMAND CODE
        .BYTE 0 ;PSEL, A16 AND A17
        .WORD -2 ;WORD COUNT (NEG)
        .WORD CYLNDR ;BUFFER ADDRESS
        .BYTE 0 ;SECTOR ADDRESS
        .BYTE 0 ;TRACK ADDRESS
        .WORD 0 ;CYLINDER ADDRESS
        .WORD GENREG ;ADDRESS TO SAVE ALL RH/RM REG'S
        .WORD 0 ;STATUS WORD

GENREG: .BLKW 24 ;REGISTER STORAGE

```

```

1
2
3 066160      122      110      061 EM1:  .ASCIZ @RH11/RH70 INTERRUPT OCCURRED (RMAS = 0)@
4 066230      125      116      105 EM2:  .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
5 066266      115      101      123 EM3:  .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
6 066324      115      101      123 EM4:  .ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
7 066361      101      104      104 EM5:  .ASCIZ /ADDRESS PLUG CHANGE BIT SET/
8 066415      122      110      061 EM6:  .ASCIZ @RH11/RH70 DIDN'T RESPOND TO ADDRESSING@
9 066464      125      116      103 EM10: .ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
10 066527     106      101      124 EM11: .ASCIZ /FATAL MASSBUS PARITY ERROR/
11 066562     120      105      122 EM12: .ASCIZ /PERSISTENT DEVICE UNSAFE/
12 066613     117      120      105 EM13: .ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
13 066665     104      122      111 EM14: .ASCIZ /DRIVE WENT OFFLINE/
14 066710     116      117      040 EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
15 066744     110      105      101 EM20: .ASCIZ /HEADER CRC ERROR/
16 066765     104      101      124 EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
17 067016     127      122      111 EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
18 067071     127      122      111 EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
19 067150     110      105      101 EM24: .ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
20 067216     110      105      101 EM25: .ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
21 067277     106      117      122 EM26: .ASCIZ /FORMAT ERROR ('FER')/
22 067324     110      105      101 EM27: .ASCIZ /HEADER COMPARE ('HCE') ERROR/
23 067361     115      111      123 EM30: .ASCIZ /MISCELLANEOUS DRIVE ERROR/
24 067413     117      120      105 EM31: .ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
25 067456     104      122      111 EM32: .ASCIZ /DRIVE TIMING ('DTE') ERROR/
26 067511     120      101      122 EM33: .ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/
27 067566     127      122      111 EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
28 067630     111      116      126 EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
29 067666     127      122      111 EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
30 067717     104      101      124 EM37: .ASCIZ /DATA CHECK ('DCK') SET DURING WRITE CHECK COMMAND/
31 070001     122      110      061 EM40: .ASCIZ @RH11/RH70 OR UNIBUS TRANSFER ERROR@
32 070044     102      125      123 EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
33 070110     104      101      124 EM42: .ASCIZ /DATA COMPARE ERRORS - NO OTHER ERROR(S) DETECTED/
34 070171     103      101      116 EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
35 070236     105      122      122 EM44: .ASCIZ @ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/RH70@
36 070327     105      103      103 EM45: .ASCIZ /ECC LOGIC FAILURE - POSITION REGISTER VALUE NOT VALID/
37 070415     102      125      123 EM46: .ASCIZ /BUS ADDRESS AND WORD COUNT NOT CONSISTENT/
38 070467     123      105      105 EM50: .ASCIZ /SEEK INCOMPLETE ('SKI') ERROR/
39 070525     120      122      117 EM51: .ASCIZ /PROGRAM DETECTED POSITIONING ERROR/
40 070570     104      122      111 EM60: .ASCIZ /DRIVE UNSAFE ERROR/
41

```

```
1 070613 122 115 101 DH1: .ASCIZ /RMAS/
2 070620 104 122 111 DH2: .ASCIZ /DRIVE RMDS RMER1 RMER2 RMMR2 RMAS/
3 070674 104 122 111 DH3: .ASCIZ /DRIVE REG ADR DATA/
4 070722 104 122 111 DH4: .ASCIZ /DRIVE REG ADR GOOD BAD/
5 070761 044 122 115 DH6: .ASCIZ /$RMADR/
6 070770 104 122 126 DH14: .ASCII /DRV RMCS1 RMCS2 RMDS RMER1 /
7 071034 122 115 115 .ASCIZ /RMMR2 RMER2 RMEC1 RMEC2/<CR><LF>
8 071073 122 115 127 DH15: .ASCII /RMWC RMBA RMDA RMAS RMLA /
9 071143 122 115 104 .ASCIZ /RMDB RMMR1 RMDT/<CR><LF>
10 071172 122 115 123 DH16: .ASCIZ /RMSN RMOF RMDC RMHR STATUS/<CR><LF>
11
12 .EVEN
```

1	071244	001344	000000	DT1:	.WORD	ATTN,0	
2	071250	001220	035614	035616	DT2:	.WORD	DRIVE, RMERRS, RMERRS+2, RMERRS+4, RMERRS+6, ATTN, 0
3	071266	001220	042746	042750	DT3:	.WORD	DRIVE, RD. ADR, RD. WRD, 0
4	071276	001220	043166	043164	DT4:	.WORD	DRIVE, WRT. AD, WRT. WD, RD. WRD, 0
5	071310	001272	000000	DT6:	.WORD	\$RMADR, 0	
6	071314	002116	002126	002130	DT14:	.WORD	\$RMCS1, \$RMCS2, \$RMDS, \$RMER1, \$RMMR2, \$RMER2, \$RMEC1, \$RMEC2, 0
7	071336	002120	002122	002124	DT15:	.WORD	\$RMWC, \$RMBA, \$RMDA, \$RMAS, \$RMLA, \$RMDB, \$RMMR1, \$RMDT, 0
8	071360	002146	002150	002152	DT16:	.WORD	\$RMSN, \$RMOF, \$RMDC, \$RMHR, \$STATUS, 0
9							

1	071374	120	122	105	LIN2C:	.ASCIZ	/PRESENT ORDER = /
2	071415	040	040	120	LIN2P:	.ASCIZ	/ PREVIOUS ORDER = /
3	071441	052	040	105	LIN2S:	.ASCIZ	@* ERROR AT BAD TRACK/SECTOR@
4	071475	105	122	122	LINM3:	.ASCIZ	/ERROR AT C/
5	071510	040	124	000	T:	.ASCIZ	/ T/
6	071513	120	122	105	LINN3:	.ASCIZ	/PRESENT ADDR = C/
7	071534	040	123	000	S:	.ASCIZ	/ S/
8	071537	040	040	040	LINP3:	.ASCIZ	/ PREV ADDR = C/
9	071560	123	124	101	LINS3:	.ASCIZ	/START CYL = /
10	071575	040	040	105	LINEN3:	.ASCIZ	/ END CYL = /
11	071612	040	040	101	LINA3:	.ASCIZ	/ ACTUAL CYL = /
12	071632	040	040	124	LINT3:	.ASCIZ	/ TRK = /
13	071643	040	122	115	LINCA3:	.ASCIZ	/ RMDC = /
14	071654	122	115	104	LINDA3:	.ASCIZ	/RMDA = /
15	071664	122	115	102	LINB3:	.ASCIZ	/RMDA = /
16	071674	040	040	122	LINW3:	.ASCIZ	/ RMWC = /
17	071706	123	124	101	LINST3:	.ASCIZ	/START TRK = /
18	071723	123	124	101	LINSS3:	.ASCIZ	/START SEC = /
19	071740	102	125	106	LINM4:	.ASCIZ	/BUFFER ADDR = /
20	071757	040	040	123	LINS4:	.ASCIZ	/ SIZE = /
21	071771	040	040	101	LINX4:	.ASCIZ	/ ACTUAL NMBR WRDS XFRD = /
22	072024	107	117	117	LIND5:	.ASCIZ	/GOOD DATA = /
23	072041	040	040	102	LINB5:	.ASCIZ	/ BAD DATA = /
24	072057	040	040	127	LINP5:	.ASCIZ	/ WORD POS = /
25	072075	110	105	101	LINS5:	.ASCIZ	/HEADER FROM ERROR SECTOR = /
26	072131	122	115	105	LINEP5:	.ASCIZ	/RMEC1 = /
27	072142	040	122	115	LINEO5:	.ASCIZ	/ RMEC2 = /
28	072154	123	105	103	LINB6:	.ASCIZ	/SECTOR IS ECC CORRECTABLE /
29	072207	123	105	103	LINC6:	.ASCIZ	/SECTOR READ CORRECTLY /
30	072236	103	117	122	LING6:	.ASCIZ	/CORRECTED ON /
31	072254	040	122	105	LINR6:	.ASCIZ	/ RETRIES/
32	072265	125	116	103	LINU06:	.ASCIZ	/UNCORRECTABLE AFTER /
33	072312	040	040	124	LIN7M:	.ASCIZ	/ TOTAL MISPOS ERR = /
34	072340	117	122	104	LIN7O:	.ASCIZ	/ORDERS:/
35	072350	040	124	117	LIN7P:	.ASCIZ	/ TOTAL SEEKS = /
36	072370	040	124	117	LIN7S:	.ASCIZ	/ TOTAL SKI ERR = /
37	072412	040	040	105	LIN7T:	.ASCIZ	/ ERRORS:/
38	072424	040	040	127	LIN7X:	.ASCIZ	/ WRDS XFR:/
39	072440	040	040	127	LIN7R:	.ASCIZ	/ WRDS READ:/
40	072455	105	122	122	LIN8M:	.ASCIZ	/ERROR DURING RETRY/
41	072500	104	101	124	LIN9B:	.ASCIZ	/DATA COMPARISON ERRORS/
42	072527	040	040	040	LIN9H:	.ASCII	/
43	072555	114	117	103		.ASCIZ	/GOOD BAD/<CR><LF>
44	072605	114	117	103	LIN9I:	.ASCIZ	/LOC DATA DATA/<CR><LF>
45	072625	124	117	124	LIN9E:	.ASCIZ	/LOC DATA/<CR><LF>
46	072655	124	110	105	LIN9G:	.ASCIZ	/TOTAL COMPARE ERRORS = /
47	072704	105	122	122	LIN10A:	.ASCIZ	/THE DATA COMPARED OK/<CR><LF>
48	072740	040	111	116	LIN10B:	.ASCIZ	/ERROR BURST BEGINS AT WORD /
49	073001	105	122	122	LIN10C:	.ASCII	/ IN DATA FIELD OF ERROR SECTOR/<CR><LF>
50	073044	105	103	103		.ASCIZ	/ERROR WAS NOT IN THE DATA READ - /<CR><LF>
51	073106	105	103	103	LIN10H:	.ASCII	/ECC CORRECTION CAN'T BE PERFORMED/
52	073136	101	104	104		.ASCIZ	/ECC CORRECTION RESULTS/<CR><LF>
53	073173	103	117	116	LIN11H:	.ASCIZ	/ADDR BAD CORRECTED /<CR><LF>
54	073247	101	104	104		.ASCIZ	/CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<CR><LF>
55	073266	040			BLNKS4:	.ASCII	/
56	073267	040			BLNKS3:	.ASCII	/
57	073270	040			BLNKS2:	.ASCII	/

CZRMJAO RM05/3/2 PERF EXER
ERROR MESSAGES

MACRO V03.01 11-APR-80 14:52:06 PAGE 14-1

B 15

SEQ 0183

58	073271	040	000	BLNK51: .ASCIZ / /
59	073273	122	105	124 LINX5: .ASCIZ /RETRIEVED BY A RDHD COMMAND/
60				

```

1          .SBTTL  TELETYPE MESSAGES
2
3 073327    077    000          QUES:  .ASCIZ  /?/
4 073331    104    122    111  UNTMSG: .ASCIZ  /DRIVE/
5 073337    040    117    106  UNTOFF: .ASCIZ  / OFFLINE/
6 073350    040    117    116  UNTON:  .ASCIZ  / ONLINE/
7 073360    040    116    117  UNTNOT: .ASCIZ  / NOT BEING TESTED/
8 073402    040    101    114  UNTASN: .ASCIZ  / ALREADY BEING TESTED/
9 073430    040    116    117  NOTRM:  .ASCIZ  @ NOT AN RM05/3/2@
10 073451   040    116    117  NOTPRS: .ASCIZ  / NOT PRESENT/
11 073466   040    116    117  NOTAVL: .ASCIZ  / NOT AVAILABLE/
12 073505   040    125    116  NOTSAF: .ASCIZ  / UNSAFE/
13 073515   040    111    123  LODEV:  .ASCIZ  / IS LOAD DEVICE/
14 073535   200    125    116  SYSTAT: .ASCIZ  <CRLF>/UNIT STATUS:/
15 073553   122    115    060  $RM02:  .ASCIZ  /RM02/
16 073560   122    115    060  $RM03:  .ASCIZ  /RM03/
17 073565   122    115    060  $RM05:  .ASCIZ  /RM05/
18 073572   200    012    052  REPHD:  .ASCIZ  <CRLF><LF>/ * * * * * PERFORMANCE REPORT * * * * * /
19 073644   104    122    111  STATHD: .ASCII  /DRIVE PERFORMANCE SUMMARY/<CRLF>
20 073676   104    122    126          .ASCII  /DRV PASS ORDERS  SEEKS  WRDS XFER  WRDS READ /
21 073756   123    117    106          .ASCIZ  /SOFT HARD  SKI MISP OTHER/<CRLF>
22 074011   007    077    106  DROPNG: .ASCIZ  <07>/?FATAL OR EXCESSIVE ERRORS/
23 074045   200    105    116  ENDPAS: .ASCIZ  <CRLF>/END OF PASS #/
24 074064   040    117    116  MSGON:  .ASCIZ  / ON /
25 074071   200    105    116  ENDTST: .ASCIZ  <CRLF>/END OF TEST /
26 074107   106    117    122  MSGFOR: .ASCIZ  /FOR /
27 074114   200    104    122  DEASSG: .ASCIZ  <CRLF>/DRIVE DEASSIGNED/
28 074136   200    052    052  DRNUM:  .ASCIZ  <CRLF>/***** DRIVE #/
29 074162   040    123    124  ASGND:  .ASCIZ  / STARTED/
30 074173   200    007    077  NEDCLK: .ASCIZ  <CRLF><07>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<CRLF>
31 074244   116    000          N:      .ASCIZ  /N/
32 074246   131    000          Y:      .ASCIZ  /Y/
33 074250   056    000          PERIOD: .ASCIZ  /./
34 074252   040    077    103  MSWRO:  .ASCIZ  / ?CAN'T WRITE WITH SWOO SET/<CRLF>
35 074307   040    077    111  INVLD:  .ASCIZ  / ?INVALID COMMAND/<CRLF>
36 074332   200    105    116  ENTCOM: .ASCIZ  <CRLF>/ENTER COMMAND: /
37 074353   200    105    116  ENTDRV: .ASCIZ  <CRLF>/ENTER DRIVE I.D.: /
38 074377   200    105    116  ENTLMT: .ASCIZ  <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
39 074427   200    105    116  ENTADR: .ASCIZ  <CRLF>@ENTER BAD SPOT ADDRESSES: @<CRLF>
40 074464   072    000          COLON:  .ASCIZ  /:/
41 074466   200    104    101  DATEIS: .ASCIZ  <CRLF>/DATE: /
42 074476   200    117    120  IDIS:   .ASCIZ  <CRLF>/OPERATOR I.D.: /
43 074517   200    012    104  HEDLIN: .ASCIZ  <CRLF><LF>/DRV  DRV I.D./<CRLF>
44 074540   116    117    116  NONE:   .ASCIZ  /NONE/<CRLF>
45 074546   040    077    111  BADENT: .ASCIZ  / ?INVALID ENTRY/<CRLF>
46 074567   007    123    131  BUSY:   .ASCIZ  <07>/SYSTEM BUSY.../<CRLF>
47 074610   200    120    122  INTDON: .ASCII  <CRLF>/PROGRAM INITIALIZATION COMPLETE/<CRLF>
48 074651   124    131    120          .ASCIZ  /TYPE A 'CONTROL C' TO ENTER COMMANDS/
49 074716   200    106    101  MERR1:  .ASCIZ  <CRLF>/FAILED TO RETRIEVE BAD SPOT FILE(DEC144) ON /
50 074774   200    111    116  MERR2:  .ASCIZ  <CRLF>/INVALID FILE(DEC144) STRUCTURE ON /
51 075040   040    077    102  MSFULL: .ASCIZ  / ?BAD SPOT TABLE IS FULL/<CRLF>
52 075072   103    131    114  MSGCYL: .ASCIZ  /CYLNDR/
53 075101   124    122    101  MSGTRK: .ASCIZ  /TRACK /
54 075110   123    105    103  MSGSEC: .ASCIZ  /SECTOR/
55
56          .EVEN
57

```

```

1
2
3 075120 075250 020000 001 PARLST: .WORD PAR1,8192.,MAXDL
4 075126 075260 177777 001472 .WORD PAR2,-1,INTRVL
5 075134 075430 177777 001464 .WORD PAR19,-1,PASCNT
6 075142 075270 000017 001514 .WORD PAR3,15.,PATTEN
7 075150 075370 000001 001502 .WORD PAR11,1,WCSEL
8 075156 075400 000007 001504 .WORD PAR14,7,RATIO
9 075164 075420 000001 001512 .WORD PAR16,1,ENDET
10 075172 075360 000001 001500 .WORD PAR10,1,FORMAT
11 : .WORD PAR15,1,AUTOCK
12 075200 075440 000001 001510 .WORD PAR20,1,NOTPRT
13 075206 075450 000001 001516 .WORD PAR21,1,HEADER ;RANDOM ADDRESS FLAG
14 075214 000000 .WORD 0 ;TABLE TERMINATOR
15
16 075216 040 057 040 SLASH: .ASCIZ @ / @
17 075222 200 103 110 ASKPAR: .ASCIZ <CRLF>/CHANGE PARAMETERS ? /
18 075250 123 111 132 PAR1: .ASCIZ /SIZE /
19 075260 115 111 116 PAR2: .ASCIZ /MINUTE /
20 075270 120 101 124 PAR3: .ASCIZ /PATTERN/
21 075300 115 101 130 PAR4: .ASCIZ /MAXCYL /
22 075310 115 111 116 PAR5: .ASCIZ /MINCYL /
23 075320 115 101 130 PAR6: .ASCIZ /MAXTRK /
24 075330 115 111 116 PAR7: .ASCIZ /MINTRK /
25 075340 115 101 130 PAR8: .ASCIZ /MAXSEC /
26 075350 115 111 116 PAR9: .ASCIZ /MINSEC /
27 075360 106 117 122 PAR10: .ASCIZ /FORMAT /
28 075370 122 101 116 PAR11: .ASCIZ /RANDOM /
29 075400 122 101 124 PAR14: .ASCIZ /RATIO /
30 075410 103 110 105 PAR15: .ASCIZ /CHECK /
31 075420 105 116 104 PAR16: .ASCIZ /ENDING /
32 075430 120 101 123 PAR19: .ASCIZ /PASSES /
33 075440 115 105 123 PAR20: .ASCIZ /MESSAGE/
34 075450 110 105 101 PAR21: .ASCIZ /HEADER /
35
36 .EVEN
37
38 ;PARAMETER TABLE POINTERS FOR ADDRESS LIMITS
39 075460 075500 TABLE: .WORD TABLE0 ;PARAMETER TABLE FOR DRIVE 0
42 075462 075546 .WORD TABLE1 ;PARAMETER TABLE FOR DRIVE 1
    075464 075614 .WORD TABLE2 ;PARAMETER TABLE FOR DRIVE 2
    075466 075662 .WORD TABLE3 ;PARAMETER TABLE FOR DRIVE 3
    075470 075730 .WORD TABLE4 ;PARAMETER TABLE FOR DRIVE 4
    075472 075776 .WORD TABLE5 ;PARAMETER TABLE FOR DRIVE 5
    075474 076044 .WORD TABLE6 ;PARAMETER TABLE FOR DRIVE 6
    075476 076112 .WORD TABLE7 ;PARAMETER TABLE FOR DRIVE 7
43
44 ;PARAMETER TABLE FOR ADDRESS LIMITS
54 075500 075310 001465 044320 TABLE0: .WORD PAR5,821.,MINCYL+DRIVE0
    075506 075300 001465 044316 .WORD PAR4,821.,MAXCYL+DRIVE0
    075514 075330 000000 044324 .WORD PAR7,0,MINTRK+DRIVE0
    075522 075320 000000 044322 .WORD PAR6,0,MAXTRK+DRIVE0
    075530 075350 000037 044330 .WORD PAR9,31.,MINSEC+DRIVE0
    075536 075340 000037 044326 .WORD PAR8,31.,MAXSEC+DRIVE0
    075544 000000 .WORD 0 ;TERMINATOR
    075546 075310 001465 046506 TABLE1: .WORD PAR5,821.,MINCYL+DRIVE1

```

075554	075300	001465	046504	.WORD	PAR4,821.,MAXCYL+DRIVE1
075562	075330	000000	046512	.WORD	PAR7,0,MINTRK+DRIVE1
075570	075320	000000	046510	.WORD	PAR6,0,MAXTRK+DRIVE1
075576	075350	000037	046516	.WORD	PAR9,31.,MINSEC+DRIVE1
075604	075340	000037	046514	.WORD	PAR8,31.,MAXSEC+DRIVE1
075612	000000			.WORD	0 ;TERMINATOR
075614	075310	001465	050674	TABLE2: .WORD	PAR5,821.,MINCYL+DRIVE2
075622	075300	001465	050672	.WORD	PAR4,821.,MAXCYL+DRIVE2
075630	075330	000000	050700	.WORD	PAR7,0,MINTRK+DRIVE2
075636	075320	000000	050676	.WORD	PAR6,0,MAXTRK+DRIVE2
075644	075350	000037	050704	.WORD	PAR9,31.,MINSEC+DRIVE2
075652	075340	000037	050702	.WORD	PAR8,31.,MAXSEC+DRIVE2
075660	000000			.WORD	0 ;TERMINATOR
075662	075310	001465	053062	TABLE3: .WORD	PAR5,821.,MINCYL+DRIVE3
075670	075300	001465	053060	.WORD	PAR4,821.,MAXCYL+DRIVE3
075676	075330	000000	053066	.WORD	PAR7,0,MINTRK+DRIVE3
075704	075320	000000	053064	.WORD	PAR6,0,MAXTRK+DRIVE3
075712	075350	000037	053072	.WORD	PAR9,31.,MINSEC+DRIVE3
075720	075340	000037	053070	.WORD	PAR8,31.,MAXSEC+DRIVE3
075726	000000			.WORD	0 ;TERMINATOR
075730	075310	001465	055250	TABLE4: .WORD	PAR5,821.,MINCYL+DRIVE4
075736	075300	001465	055246	.WORD	PAR4,821.,MAXCYL+DRIVE4
075744	075330	000000	055254	.WORD	PAR7,0,MINTRK+DRIVE4
075752	075320	000000	055252	.WORD	PAR6,0,MAXTRK+DRIVE4
075760	075350	000037	055260	.WORD	PAR9,31.,MINSEC+DRIVE4
075766	075340	000037	055256	.WORD	PAR8,31.,MAXSEC+DRIVE4
075774	000000			.WORD	0 ;TERMINATOR
075776	075310	001465	057436	TABLE5: .WORD	PAR5,821.,MINCYL+DRIVE5
076004	075300	001465	057434	.WORD	PAR4,821.,MAXCYL+DRIVE5
076012	075330	000000	057442	.WORD	PAR7,0,MINTRK+DRIVE5
076020	075320	000000	057440	.WORD	PAR6,0,MAXTRK+DRIVE5
076026	075350	000037	057446	.WORD	PAR9,31.,MINSEC+DRIVE5
076034	075340	000037	057444	.WORD	PAR8,31.,MAXSEC+DRIVE5
076042	000000			.WORD	0 ;TERMINATOR
076044	075310	001465	061624	TABLE6: .WORD	PAR5,821.,MINCYL+DRIVE6
076052	075300	001465	061622	.WORD	PAR4,821.,MAXCYL+DRIVE6
076060	075330	000000	061630	.WORD	PAR7,0,MINTRK+DRIVE6
076066	075320	000000	061626	.WORD	PAR6,0,MAXTRK+DRIVE6
076074	075350	000037	061634	.WORD	PAR9,31.,MINSEC+DRIVE6
076102	075340	000037	061632	.WORD	PAR8,31.,MAXSEC+DRIVE6
076110	000000			.WORD	0 ;TERMINATOR
076112	075310	001465	064012	TABLE7: .WORD	PAR5,821.,MINCYL+DRIVE7
076120	075300	001465	064010	.WORD	PAR4,821.,MAXCYL+DRIVE7
076126	075330	000000	064016	.WORD	PAR7,0,MINTRK+DRIVE7
076134	075320	000000	064014	.WORD	PAR6,0,MAXTRK+DRIVE7
076142	075350	000037	064022	.WORD	PAR9,31.,MINSEC+DRIVE7
076150	075340	000037	064020	.WORD	PAR8,31.,MAXSEC+DRIVE7
076156	000000			.WORD	0 ;TERMINATOR

55
56

;ROUTINE TO GET THE DATE AND THE OPERATOR FROM THE OPERATOR

57
58
59
60
61
62
63 076160 104401 076302
64 076164 104411
65 076166 012605
66 076170 112537 001322
69 076174 112537 001323
076200 112537 001324
076204 112537 001325
076210 112537 001326
076214 112537 001327
076220 112537 001330
076224 112537 001331
70 076230 005037 001332
71 076234 104401 076320
72 076240 104411
73 076242 012605
74 076244 112537 001334
77 076250 112537 001335
076254 112537 001336
076260 112537 001337
076264 112537 001340
076270 112537 001341
78 076274 005037 001342
79 076300 000207
80
81 076302 200 105 116
82 076320 105 116 124
83
84
85

```

:CALL:
:      JSR   PC,OPRDAT
:      RETURN
:NOTE: THIS ROUTINE IS ENTERED ONLY AT INITIAL START
OPRDAT: TYPE      ,ENTDAT      ;'ENTER DATE'
        RDLIN     ;READ THE ENTRY
        MOV      (SP)+,R5      ;PUT THE ENTRY ADDRESS INTO R5
        MOV      (R5)+,DATE    ;STORE THE DATE
        MOV      (R5)+,DATE+1  ;STORE THE DATE
        MOV      (R5)+,DATE+2  ;STORE THE DATE
        MOV      (R5)+,DATE+3  ;STORE THE DATE
        MOV      (R5)+,DATE+4  ;STORE THE DATE
        MOV      (R5)+,DATE+5  ;STORE THE DATE
        MOV      (R5)+,DATE+6  ;STORE THE DATE
        MOV      (R5)+,DATE+7  ;STORE THE DATE
        CLR      DATE+8.      ;SET TERMINATOR
        TYPE      ,ENTID      ;'ENTER OPERATOR I.D.'
        RDLIN     ;READ THE ENTRY
        MOV      (SP)+,R5      ;ENTRY ADDRESS
        MOV      (R5)+,OPERID   ;STORE THE I.D.
        MOV      (R5)+,OPERID+1 ;STORE THE I.D.
        MOV      (R5)+,OPERID+2 ;STORE THE I.D.
        MOV      (R5)+,OPERID+3 ;STORE THE I.D.
        MOV      (R5)+,OPERID+4 ;STORE THE I.D.
        MOV      (R5)+,OPERID+5 ;STORE THE I.D.
        CLR      OPERID+6     ;SET TERMINATOR
        RTS      PC           ;RETURN
    
```

```

116 ENTDAT: .ASCIZ <CRLF>/ENTER DATE: /
124 ENTID:  .ASCIZ /ENTER OPERATOR I.D.: /
.EVEN
    
```

.SBTTL ROUTINE TO SIZE MEMORY

```

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

076346 010046
076350 010146
076352 013746 000114
076356 013746 000116
076362 012737 000116 000114
076370 012737 000002 000116
076376 013746 000004
076402 013746 000006
076406 010600

076410 104400
076412 012637 000006
076416 012737 076436 000004
076424 012701 020000
076430 005711

```

$SIZE: MOV      R0,-(SP)      ;;SAVE R0 ON THE STACK
        MOV      R1,-(SP)      ;;SAVE R1 ON THE STACK
        MOV      @#114,-(SP)    ;;SAVE MEMORY ERROR VECTOR PS & PC
        MOV      @#116,-(SP)
        MOV      #116,@#114    ;;IGNORE PARITY ERRORS WHILE SIZING
        MOV      #RTI,@#116
        MOV      @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
        MOV      @#ERRVEC+2,-(SP)
        MOV      SP,R0        ;;SAVE THE STACK POINTER
        ;;SET THE ERRVEC PS TO THE PRESENT PS
        TRAP     ;;PUSH OLD PSW AND PC ON STACK
        MOV      (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
        MOV      #2$,@#ERRVEC    ;;SET FOR TIMEOUT
        MOV      #20000,R1      ;;FIRST ADDRESS
1$:    TST      (R1)          ;;TEST THIS ADDRESS
    
```

```

076432 005721          TST      (R1)+          ;;STEP TO NEXT ADDRESS
076434 000775          BR       1$              ;;TRY ANOTHER
076436 162701 000002 2$: SUB      #2,R1          ;;DROP BACK
076442 010006          MOV      RO,SP          ;;RESTORE THE STACK
076444 012637 000006  MOV      (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
076450 012637 000004  MOV      (SP)+,@#ERRVEC
076454 012637 000116  MOV      (SP)+,@#116      ;;RESTORE MEMORY ERROR VECTOR
076460 012637 000114  MOV      (SP)+,@#114
076464 010137 076476  MOV      R1,$LSTAD       ;;LAST ADDRESS
076470 012601          MOV      (SP)+,R1        ;;RESTORE R1
076472 012600          MOV      (SP)+,R0        ;;RESTORE R0
076474 000207          RTS      PC
076476 000000          $LSTAD: .WORD 0          ;;CONTAINS THE LAST ADDRESS

86
87
88
89
90
91
92
93
94
95
96
97
98 076500 005737 001360 BUSADR: TST      CHGADR          ;INPUT FROM TTY REQUESTED?
99 076504 002036          BGE      3$              ;NO--BRANCH
100 076506 005037 001360 CLR      CHGADR          ;YES--CLEAR THE REQUEST FLAG
101 076512 104401 001203 TYPE     ,SCRLF          ;TYPE A CR-LF
102 076516 012700 001272 1$: MOV      #SRMADR,R0      ;FIRST ADDRESS
103 076522 104401 076620 TYPE     ,MRMCS1         ;'RMCS1='
104 076526 011046          MOV      (R0),-(SP)      ;PRESENT RMCS1 ADDRESS
105 076530 104402          TYPOC          ;TYPE IT
106 076532 104401 073270 TYPE     ,BLNKS2         ;TYPE 2 BLANKS
107 076536 104411          RDLIN          ;GET THE ENTRY
108 076540 012601          MOV      (SP)+,R1        ;ADDRESS OF ASCII TEXT
109 076542 004537 076636 JSR      R5,CK.NUM        ;ENTER AND STORE THE NEW ADDRESS
110 076546 000763          BR       1$              ;ERROR EXIT
111 076550 012700 001274 2$: MOV      #SRMVEC,R0      ;VECTOR ADDRESS
112 076554 104401 076627 TYPE     ,MRMVEC         ;'RMVEC='
113 076560 011046          MOV      (R0),-(SP)      ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
114 076562 104402          TYPOC          ;TYPE IT
115 076564 104401 073270 TYPE     ,BLNKS2         ;TYPE 2 BLANKS
116 076570 104411          RDLIN          ;READ THE ENTRY
117 076572 012601          MOV      (SP)+,R1        ;ASCII TEXT ADDRESS
118 076574 004537 076636 JSR      R5,CK.NUM        ;ENTER AND STORE NEW ADDRESS
119 076600 000763          BR       2$              ;ERROR EXIT
120 076602 012700 001272 3$: MOV      #SRMADR,R0      ;FIRST ADDRESS OF NEW PARAMETERS
121 076606 012701 035762 MOV      #RMADR,R1        ;FIRST ADDRESS OF WHERE TO PUT THEM
122 076612 012021          MOV      (R0)+,(R1)+     ;BUS ADDRESS
123 076614 012021          MOV      (R0)+,(R1)+     ;VECTOR ADDRESS
124 076616 000207          RTS      PC              ;RETURN
125
126 076620          122          115          103 MRMCS1: .ASCIZ @RMCS1=@
127 076627          122          115          126 MRMVEC: .ASCIZ @RMVEC=@
128
129
.SBTTL CK.NUM - CHECK NUMBER (OCTAL)

```

```

130 ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
131 ;AND FORMS AN OCTAL NUMBER IN R2
132 ;CALL:
133 :      MOV      #ADR,R1      ;ADDRESS OF ASCIZ STRING
134 :      JSR      R5,CK.NUM    ;R5 CHANGED
135 :      RET      ;ERROR EXIT
136 :      RET      ;NORMAL EXIT
137
138 076636 010246      CK.NUM: MOV      R2,-(SP)    ;SAVE R2
139 076640 010346      MOV      R3,-(SP)    ;SAVE R3
140 076642 010446      MOV      R4,-(SP)    ;SAVE R4
141 076644 012703 000006  MOV      #6,R3      ;MAX OCTAL DIGITS IN THE NUMBER
142 076650 005002      CLR      R2          ;FINAL OCTAL VALUE
143 076652 112104      1$:  MOVB     (R1)+,R4    ;GET CURRENT POINTED BYTE
144 076654 001424      BEQ      3$          ;BRANCH,IF TERMINATOR DETECTED
145 076656 120427 000060  CMPB     R4,#'0      ;SMALLER THAN ASCII-0 ?
146 076662 103425      BLO      5$          ;YES,ERROR EXIT
147 076664 120427 000067  CMPB     R4,#'7      ;LARGER THAN ASCII-7 ?
148 076670 101022      BHI      5$          ;YES,ERROR EXIT
149 076672 006302      ASL      R2          ;SHIFT LEFT
150 076674 103420      BCS      5$          ;
151 076676 006302      ASL      R2          ;ONE
152 076700 103416      BCS      5$          ;
153 076702 006302      ASL      R2          ;OCTAL DIGIT
154 076704 103414      BCS      5$          ;ERROR IF CARRY BIT SET
155 076706 042704 177770  BIC      #177770,R4  ;CHOP OFF HIGHER BITS
156 076712 060402      ADD      R4,R2      ;APPENDING CURRENT DIGIT TO NUMBER
157 076714 005303      DEC      R3          ;DECREMENT BYTE COUNT
158 076716 001401      BEQ      2$          ;BRANCH,IF LAST BYTE
159 076720 000754      BR       1$          ;LOOPING BACK
160 076722 112104      2$:  MOVB     (R1)+,R4    ;CHECK TERMINATOR
161 076724 001004      BNE      5$          ;ERROR EXIT
162 076726 005702      3$:  TST      R2          ;FINAL VALUE = 0
163 076730 001401      BEQ      4$          ;YES,THEN NOT REPLACE THE ORIGINAL VALUE
164 076732 010210      MOV      R2,(R0)    ;REPLACE THE ORIGINAL VALUE
165 076734 005725      4$:  TST      (R5)+      ;ADJUST FOR NORMAL RETURN
166 076736 012604      5$:  MOV      (SP)+,R4    ;RESTORE R4
167 076740 012603      MOV      (SP)+,R3    ;RESTORE R3
168 076742 012602      MOV      (SP)+,R2    ;RESTORE R2
169 076744 000205      RTS      R5          ;EXIT
170
171 076746      CYLNR: .BLKW 258. ;ONE SECTOR WORD CTR MAX SIZE
172
173      077752      ENDPGM-.
174
175      000200      .END 200

```

ABASE = 176700	ASGN7 025532	BIT8 = 000400	CR = 000015	DROPNG 074011
ABNRML 030152	ASKPAR 075222	BIT9 = 001000	CRLF = 000200	DRQ = 004000
ACDW1 = 000000	ASNERR 030026	BLKADR 002106	CYLIMT 001444	DRVACT 035624
ACDW2 = 000000	ASNLST 001550	BLNKS1 073271	CYLNRD 076746	DRVCLR= 000111
ACK = 000123	ASNMSG 030052	BLNKS2 073270	DATAPK 026132	DRVER 011322
ACPUOP= 000000	ASSIGN 025166	BLNKS3 073267	DATA0 002546	DRVINT 036212
ACTDRV 035700	ASWREG= 000000	BLNKS4 073266	DATA1 002606	DRVPRM 026374
ACTSTR 035701	ATA = 100000	BPTVEC= 000014	DATA10 003246	DRVQUE 044044
ADDW0 = 000000	ATABIT 035750	BUFTBL 001704	DATA11 003306	DRVSTA 035634
ADDW1 = 000000	ATESTN= 000000	BUSADR 076500	DATA12 003346	DRVTyp 035644
ADDW10= 000000	ATIN 001344	BUSY 074567	DATA13 003406	DRY = 000200
ADDW11= 000000	ATO = 000001	CFLAG 001362	DATA14 003446	DSWR = 177570
ADDW12= 000000	AT1 = 000002	CHGADR 001360	DATA15 003506	DTE = 010000
ADDW13= 000000	AT2 = 000004	CHKADR 017656	DATA2 002646	DTEER 012160
ADDW14= 000000	AT3 = 000010	CI1 037300	DATA3 002706	DTUW 035746
ADDW15= 000000	AT4 = 000020	CI3 037406	DATA4 002746	DT00 = 000001
ADDW2 = 000000	AT5 = 000040	CI4 037506	DATA5 003006	DT01 = 000002
ADDW3 = 000000	AT6 = 000100	CI5 040064	DATA6 003046	DT02 = 000004
ADDW4 = 000000	AT7 = 000200	CI6 040106	DATA7 003106	DT03 = 000010
ADDW5 = 000000	AUNIT = 000000	CI7 040122	DATA8 003146	DT04 = 000020
ADDW6 = 000000	AUSWR = 000000	CI7B 040144	DATA9 003206	DT05 = 000040
ADDW7 = 000000	AUTOCK 001506	CI8 040222	DATE 001322	DT06 = 000100
ADDW8 = 000000	AVAIL 001616	CKBUS 013400	DATEIS 074466	DT07 = 000200
ADDW9 = 000000	AVECT1= 000254	CKCLK 023326	DCK = 100000	DT08 = 000400
ADEVCT= 000000	AVECT2= 000000	CKCLK1 023422	DCKER 010154	DT1 071244
ADEVN = 000000	A16 = 000400	CKCLK2 023470	DCKER1 010274	DT14 071314
AENV = 000000	A17 = 001000	CKCLK3 023520	DDISP = 177570	DT15 071336
AENVN = 000000	BADENT 074546	CKERR 013306	DEASGN 025604	DT16 071360
AFATAL = 000000	BADSEC 001364	CKFMT 011354	DEASSG 074114	DT2 071250
AMADR1= 000000	BAI = 000010	CKHCE 011550	DH1 070613	DT3 071266
AMADR2= 000000	BEGCOD 001522	CKSWR = 104407	DH14 070770	DT4 071276
AMADR3= 000000	BEGPAT 001520	CK.CHR 031314	DH15 071073	DT6 071310
AMADR4= 000000	BEGSIZ 001524	CK.DEC 031266	DH16 071172	DULP 036214
AMAMS1= 000000	BIT0 = 000001	CK.DIG 031366	DH2 070620	DUNIT 001552
AMAMS2= 000000	BIT00 = 000001	CK.NUM 076636	DH3 070674	DVA = 004000
AMAMS3= 000000	BIT01 = 000002	CK.OCT 031240	DH4 070722	DVC = 000200
AMAMS4= 000000	BIT02 = 000004	CLKFLG 001312	DH6 070761	ECBADO 001434
AMSGAD= 000000	BIT03 = 000010	CLOCK 024462	DISPLA 001156	ECBAD1 001442
AMSGLG= 000000	BIT04 = 000020	CLR = 000040	DISPLY= 104414	ECBIT 001420
AMSGTY= 000000	BIT05 = 000040	CLRDPB 026174	DISPRE 000174	ECC 014710
AMTYP1= 000000	BIT06 = 000100	CLRQUE 043746	DLT = 100000	ECCX 015446
AMTYP2= 000000	BIT07 = 000200	CMCNT 001412	DONE 007646	ECC1 015306
AMTYP3= 000000	BIT08 = 000400	CMCYL 001414	DPE = 020000	ECC2 015436
AMTYP4= 000000	BIT09 = 001000	CMDAT 013736	DPINT 035654	ECGD 001432
AOE = 001000	BIT1 = 000002	CMHED 013664	DPR = 000400	ECGD1 001440
APASS = 000000	BIT10 = 002000	CMPAR 013464	DPRQS 035664	ECH = 000100
APRIOR= 000000	BIT11 = 004000	CMPARD 013502	DRIVE = 001220	ECI = 004000
APTC SU= 000040	BIT12 = 010000	CMPLMT 001476	DRIVE0 044210	ECMSK0 001424
APTENV= 000001	BIT13 = 020000	CMPRES 020356	DRIVE1 046376	ECMSK1 001426
APTSIZ= 000200	BIT14 = 040000	CMPT 014310	DRIVE2 050564	ECSEC 001422
APTSPO= 000100	BIT15 = 100000	CMPRX 014302	DRIVE3 052752	ECWRD 001430
ASGND 074162	BIT2 = 000004	CMSEC 001416	DRIVE4 055140	ECWRD1 001436
ASGN1 025174	BIT3 = 000010	CMSTR 013602	DRIVE5 057326	EMPTYQ 044024
ASGN2 025260	BIT4 = 000020	CMSTR2 013724	DRIVE6 061514	EMTVEC= 000030
ASGN3 025356	BIT5 = 000040	CMTRK 001417	DRIVE7 063702	EM1 066160
ASGN4 025516	BIT6 = 000100	COLON 074464	DRNUM 074136	EM10 066464
ASGN6 025520	BIT7 = 000200	COMTBL 002126	DROP 030056	EM11 066527

EM12	066562	ERRVEC=	000004	INTDON	074610	LINP5	072057	MDPE	=	000400	
EM13	066613	FACTOR	016310	INTRVL	001472	LINR6	072254	MERR1		074716	
EM14	066665	FAIRNS	001354	INVLD	074307	LINSS3	071723	MERR2		074774	
EM15	066710	FALPAR	007432	IOTVEC=	000020	LINST3	071706	MINCYL=		000110	
EM2	066230	FALPR1	007446	IR	=	000100	LINS3	071560	MINSEC=	000120	
EM20	066744	FER	=	000020	ISR	040634	LINS4	071757	MINTRK=	000114	
EM21	066765	FILBUF	016676	IVC	=	010000	LINS5	072075	MINUTE	001370	
EM22	067016	FISK	030552	KSR	024624	LINT3	071632	MNDLTA		035774	
EM23	067071	FMTER	012354	KSR1	024636	LINU06	072265	MNTBL		002156	
EM24	067150	FMT16 =	010000	LA	040460	LINW3	071674	MOH	=	020000	
EM25	067216	FORMAT	001500	LACNT	035712	LINX4	071771	MOL	=	010000	
EM26	067277	FOWT1	005644	LBC	=	002000	LINX5	073273	MONTR	005614	
EM27	067324	FOWT2	005720	LBT	=	002000	LIN10A	072704	MRMCS1	076620	
EM3	066266	FOWT3	006070	LF	=	000012	LIN10B	072740	MRMVEC	076627	
EM30	067361	FRSTER	001400	LIMIT	001410	LIN10C	073001	MSFULL		075040	
EM31	067413	F0	=	000002	LINA3	071612	LIN10H	073106	MSGCYL	075072	
EM32	067456	F1	=	000004	LINB3	071664	LIN11H	073173	MSGFOR	074107	
EM33	067511	F2	=	000010	LINB5	072041	LIN2C	071374	MSGON	074064	
EM34	067566	F3	=	000020	LINB6	072154	LIN2P	071415	MSGSEC	075110	
EM35	067630	F4	=	000040	LINCA3	071643	LIN2S	071441	MSGTRK	075101	
EM36	067666	GENDPB	066070	LINC6	072207	LIN3.1	022040	MSWRO		074252	
EM37	067717	GENREG	066110	LINDA3	071654	LIN3.3	022146	MXDLTA		035772	
EM4	066324	GETADR	026700	LINDEC	023156	LIN3.4	022200	MXF	=	001000	
EM40	070001	GETBUF	016312	LIND5	072024	LIN6.1	022606	MXLACT		035770	
EM41	070044	GETID	026604	LINEN3	071575	LIN6.2	022630	MXWINDW		035776	
EM42	070110	GETLMT	026546	LINE05	072142	LIN7M	072312	M.DPID		030656	
EM43	070171	GETPAR	020064	LINEP5	072131	LIN7O	072340	M.DP40		030714	
EM44	070236	GETPAT	020022	LINE1	021112	LIN7P	072350	M.DP41		030750	
EM45	070327	GETREG=	000141	LINE2	021166	LIN7R	072440	M.DP42		030760	
EM46	070415	GETREM	030562	LINE2A	021336	LIN7S	072370	M.DP44		031012	
EM5	066361	GETREQ	044120	LINE2B	021354	LIN7T	072412	M.DP50		031024	
EM50	070467	GO	=	000001	LINE3	021574	LIN7X	072424	N	074244	
EM51	070525	GODRIV	017024	LINE3A	021602	LIN8M	072455	NED	=	010000	
EM6	066415	GTSWR =	104406	LINE3B	021610	LIN9B	072500	NEDCLK		074173	
EM60	070570	HCE	=	000200	LINE3C	021622	LIN9E	072625	NEM	=	004000
ENDCMP	014564	HCEER	012432	LINE3D	021632	LIN9G	072655	NEWASN		025574	
ENDCON	001454	HCI	=	002000	LINE3E	021700	LIN9H	072527	NEWUNT		001574
ENDET	001512	HCRC	=	000400	LINE3F	021766	LIN9I	072605	NOMTCH		013034
ENDPAS	074045	HRCRER	011156	LINE4	022244	LKPAR	005232	NONE		074540	
ENDPGM=	077752	HEADER	001516	LINE5	022334	LODEV	073515	NOTAVL		073466	
ENDSEK	001460	HEDLIN	074517	LINE5A	022434	LSC	=	004000	NOTPRS		073451
ENDTST	074071	HOUR	001366	LINE5B	022502	LSTAD	001356	NOTPRT		001510	
ENTADR	074427	HT	=	000011	LINE6	022544	MAIN	006100	NOTRM		073430
ENTCOM	074332	HZ	001314	LINE6A	022556	MAINDA	006126	NOTSAF		073505	
ENTDAT	076302	IAE	=	002000	LINE6B	022564	MAIN1	006260	NSA	=	100000
ENTDRV	074353	IAEER	012274	LINE6C	022572	MAIN2	006400	OFFCOD		002470	
ENTID	076320	IDIS	074476	LINE6D	022600	MAIN3	006500	OFFDIR=		000001	
ENTLMT	074377	IDLE	006676	LINE7	022654	MANter	027226	OFFON =		000001	
ENTPR	005322	IDLE1	006774	LINE7A	022776	MASK	001350	OFFSET=		000115	
EOPX	030476	IE	=	000100	LINE8	023112	MATCH	014632	OFFST		015710
EOP1	030232	ILF	=	000001	LING6	072236	MAXCYL=		000106	OFLIN	007544
EOP2	030254	ILR	=	000002	LINKDV	030610	MAXDL	001466	OFMSG0		002366
ERCTR	001406	INCHRD	024244	LINM3	071475	MAXER	001470	OFMSG1		002421	
ERPRC1	007266	INCMIS	024244	LINM4	071740	MAXSEC=	000116	OFMSG2		002444	
ERPROC	007252	**ICSKI	024270	LINN3	071513	MAXTRK=	000112	OFMTBL		002474	
ERR	=	JNC5OF	024220	LINOCT	023124	MCPE	=	020000	ONES		003450
ERROR	=	JNCTOT	024340	LINP3	071537	MCPEMX		035760	OPE	=	000010

OPERID = 001334	PSW = 177776	RMCS1 = 000000	SEEK = 000105	SW2 = 000004
OPI = 020000	PUNSAF 007350	RMCS2 = 000010	SEEKFG 035724	SW3 = 000010
OPIER 012050	PWRVEC= 000024	RMDA = 000006	SELDRV= 000145	SW4 = 000020
OPIER1 012114	QCNT 043454	RMDB = 000022	SELPAR 017102	SW5 = 000040
OPRDAT 076160	QDRV0 043546	RMDC = 000034	SETFMT= 000143	SW6 = 000100
OPT 037042	QDRV1 043566	RMDS = 000012	SETVEC 005350	SW7 = 000200
OPTBL 002134	QDRV2 043606	RMDT = 000026	SET.IE 043402	SW8 = 000400
OR = 000200	QDRV3 043626	RMEC1 = 000044	SHDTYP 023644	SW9 = 001000
ORDERQ 001526	QDRV4 043646	RMEC2 = 000046	SIXTEE 001374	SYSTAT 073535
PACK 001320	QDRV5 043666	RMERRS 035614	SIZMEM 005054	T 071510
PAR = 000010	QDRV6 043706	RMER1 = 000014	SKI = 040000	TABLE 075460
PARENT 027702	QDRV7 043726	RMER2 = 000042	SKIER 012614	TABLE0 075500
PARER 012202	QINPT 043464	RMHR = 000036	SLASH 075216	TABLE1 075546
PARLST 075120	QOUTPT 043504	RMI.VIT 036000	SPOTCK 020760	TABLE2 075614
PARQ 001662	QSTART 043524	RMLA = 000020	SRCHWT 035676	TABLE3 075662
PAR1 075250	QSTOP 043526	RMMR1 = 000024	STACK = 001100	TABLE4 075730
PAR10 075360	QTERM = 043746	RMMR2 = 000040	START 003626	TABLE5 075776
PAR11 075370	QUES 073327	RMOF = 000032	START1 003636	TABLE6 076044
PAR14 075400	RANCYL 017402	RMR = 000004	START2 003654	TABLE7 076112
PAR15 075410	RANPAT 017636	RMSN = 000030	STATHD 073644	TAB.XY= 001114
PAR16 075420	RANSEC 017242	RMTMR 042310	STATIN 001316	TAP = 040000
PAR19 075430	RANSIZ 017516	RMVEC 035764	STATIS 016154	TBITVE= 000014
PAR2 075260	RANTRK 017316	RMWC = 000002	STATPR 023530	TD 040700
PAR20 075440	RANWRT 017766	RM05 036550	STKLMT= 177774	THEAD 017162
PAR21 075450	RANXIT 017646	RNOP = 000101	STNDAT 002502	TIMER 035726
PAR3 075270	RATIO 001504	RTC = 000117	STO 042400	TKVEC = 000060
PAR4 075300	RDCHR = 1C4410	RTNCTR 015622	STO1 042434	TPVEC = 000064
PAR5 075310	RDDAT = 000171	R6 = %000006	STO2 042530	TRAPVE= 000034
PAR6 075320	RDHD = 000173	R7 = %000007	STO3 042560	TRE = 040000
PAR7 075330	RDLIN = 104411	S 071534	STO5 042604	TRFER 012514
PAR8 075340	RDY = 000200	SATPOW 033572	STO6 042612	TRKLMT 001450
PAR9 075350	RD.ADR 042746	SAVEFG 035722	STO7 042650	TRNSWT 035674
PASCNT 001464	RD.RM 042722	SAVER1 001402	STO8 042700	TRTVEC= 000014
PAT = 000020	RD.RM1 042744	SAVER5 001404	STO9 042710	TST1 003644
PATTEN 001514	RD.RM2 043064	SAVREG= 104412	SUMARY 023616	TYPDS = 104405
PCLOCK 001310	RD.RM3 043070	SC 041154	SVRH70 043264	TYPE = 104401
PERIOD 074250	RD.RM4 043074	SCMND 025704	SWR 001154	TYPOC = 104402
PGE = 002000	RD.WRD 042750	SCOPE = 000004	SWREG 000176	TYPON = 104404
PGM = 001000	READDR 023176	SC04 = 000400	SWTIM 007506	TYPOS = 104403
PIP = 020000	READHD 015734	SC1 = 000100	SW0 = 000001	TYPRI4 031164
PIRQ = 177772	READIN= 000121	SC10 = 001000	SW00 = 000001	UCPAR 007414
PIRQVE= 000240	RECAL = 000107	SC11 041772	SW01 = 000002	ULDFLG 035702
POPQUE 044142	RECALT 015646	SC12 042102	SW02 = 000004	UNIT 001346
POSER 011774	RECALO 015652	SC13 042152	SW03 = 000010	UNS = 040000
PROCES 007162	REDAPK 026162	SC2 = 000200	SW04 = 000020	UNSAF 012754
PRTBAD 015450	REFMT 007040	SC20 = 002000	SW05 = 000040	UNTASN 073402
PRTIM 007610	REFMTX 007160	SC3 041220	SW06 = 000100	UNTMSG 073331
PRO = 000000	RELBUF 016446	SC4 041224	SW07 = 000200	UNTNOT 073360
PR1 = 000040	RELSE = 000113	SC5 041236	SW08 = 000400	UNTOFF 073337
PR2 = 000100	REPHD 073572	SC6 041422	SW09 = 001000	UNTON 073350
PR3 = 000140	REPLZ 031034	SC6A 041536	SW1 = 000002	UPE = 020000
PR4 = 000200	RESREG= 104413	SC7 041664	SW10 = 002000	US1 = 000001
PR5 = 000240	RESVEC= 000010	SC8 041742	SW11 = 004000	US2 = 000002
PR6 = 000300	RETRY 001352	SDETAL 023666	SW12 = 010000	US4 = 070004
PR7 = 000340	RMADR 035762	SEARCH= 000131	SW13 = 020000	VV = 000100
PS = 177776	RMAS = 000016	SECLMT 001446	SW14 = 040000	WAIT 007640
PSEL = 002000	RMBA = 000004	SECOND 001372	SW15 = 100000	WATPAK 026144

WCE = 040000	\$CNTLC 032256	\$ITEMB 001130	\$POSIT= 000042	\$SKI = 000064
WCF = 000040	\$CNTLG 034765	\$LF 001204	\$POWER 033562	\$SOFT = 000060
WCFER 012656	\$CNTLU 034760	\$LFLG 033405	\$PREVA= 000032	\$SSEC = 000022
WCHKX = 000001	\$CODE = 000024	\$LKCSB 001300	\$PREVO= 000027	\$STUP = 177777
WCKD = 000151	\$COMND= 000002	\$LKCSR 001276	\$PSEL = 000003	\$SUPRS 031124
WCKER 010630	\$CPUOP 001234	\$LKS 001304	\$PWRAD 033550	\$SVPC = 000210
WCKHD = 000153	\$CRLF 001203	\$LLVEC 001306	\$PWRDN 033410	\$SWR = 122000
WCSEL 001502	\$CYL = 000012	\$LONUM 035114	\$PWRMG 033544	\$SWREG 001230
WC.HK 041054	\$DBLK 034346	\$LPADR 001122	\$PWRUP 033462	\$STATUS= 000016
WLE = 004000	\$DB2D 035212	\$LPERR 001124	\$QUES 001202	\$TERM = 000032
WLEER 012326	\$DB2O 035406	\$LPVEC 001302	\$RAND 035014	\$TESTN 001212
WRL = 004000	\$DECVL 035372	\$LSTAD 076476	\$RDCHR 034640	\$TIME 024364
WRDAT= 000161	\$DEVCT 001216	\$MADR1 001240	\$RDLIN 031766	\$TKB 001162
WRTHD = 000163	\$DEVM 001264	\$MADR2 001244	\$RDSZ = 000001	\$TKINT 031604
WRTPK 020374	\$DOAGN 030546	\$MADR3 001250	\$READ = 000052	\$TKS 001160
WRTPK1 020424	\$DRVID= 002106	\$MADR4 001254	\$REG = 000014	\$TKCRV 031634
WRTPK2 020656	\$DSPLY 031212	\$MAIL 001206	\$RESRE 035154	\$TMP0 001174
WRTPK3 020702	\$DTBL 034336	\$MAMS1 001236	\$RETRY 016026	\$TN = 000002
WRTPK4 020734	\$ENDAD 030536	\$MAMS2 001242	\$RMADR 001272	\$TNPWR 035322
WRTPK5 020742	\$ENDCT 030522	\$MAMS3 001246	\$RMAS = 002134	\$TOTAL= 000056
WRT.AD 043166	\$ENV 001226	\$MAMS4 001252	\$RMBA = 002122	\$TPB 001166
WRT.RM 043076	\$ENVM 001227	\$MBADR 001102	\$RMCS1= 002116	\$TPFLG 001173
WRT.R1 043162	\$EOP 030200	\$MFLG 033404	\$RMCS2= 002126	\$TPS 001164
WRT.R2 043250	\$EOPCT 030514	\$MISPO= 000066	\$RMDA = 002124	\$TRANS= 000046
WRT.R3 043254	\$ERFLG 001117	\$MNEW 035003	\$RMDB = 002140	\$TRAP 035526
WRT.R4 043260	\$ERMAX 001131	\$MSGAD 001222	\$RMDC = 002152	\$TRAP2 035550
WRT.R5 043262	\$ERROR 032262	\$MSGLG 001224	\$RMDS = 002130	\$TRK = 000011
WRT.WD 043164	\$ERRPC 001132	\$MSGTY 001206	\$RMDT = 002144	\$TRP = 000015
XWCNT 017456	\$ERRTB 003546	\$MSWR 034772	\$RMEC1= 002162	\$TRPAD 035562
XXDP 001452	\$ERRTY 032452	\$MTYP1 001237	\$RMEC2= 002164	\$TSTM 001104
Y 074246	\$ERTTL 001126	\$MTYP2 001243	\$RMER1= 002132	\$TSTM 001116
ZEROS 002546	\$ETABL 001226	\$MTYP3 001247	\$RMER2= 002160	\$TTYIN 032244
ZROIND 001376	\$ETEND 001272	\$MTYP4 001253	\$RMHR = 002154	\$TYPDS 034132
\$APTHD 001100	\$FAIR = 000072	\$NCODE= 000074	\$RMLA = 002136	\$TYPE 032606
\$ATYC 033166	\$FATAL 001210	\$NCYL = 000100	\$RMR1= 002142	\$TYPEC 033020
\$ATY1 033142	\$FFLG 033406	\$NEXT = 000104	\$RMR2= 002156	\$TYPEX 033140
\$ATY3 033150	\$FILLC 001172	\$NPATC= 000075	\$RMOF = 002150	\$TYPOC 033730
\$ATY4 033160	\$FILLS 001171	\$NSEC = 000076	\$RMSN = 002146	\$TYPON 033744
\$AUTOB 001150	\$FIRST= 000122	\$NTRK = 000077	\$RMVEC 001274	\$TYPOS 033704
\$BASE 001262	\$FMT = 000001	\$NULL 001170	\$RMWC = 002120	\$UNIT 001220
\$BDADR 001136	\$GDADR 001134	\$NWRDL= 000102	\$RM02 073553	\$UNITM 001110
\$BDDAT 001142	\$GDDAT 001140	\$NWTST= 000000	\$RM03 073560	\$USWR 001232
\$BDSEC= 000124	\$GET42 030526	\$OCNT 034126	\$RM05 073565	\$VECT1 001256
\$BELL 001176	\$GTSWR 034426	\$OCTVL 035510	\$RTNAD 030550	\$VECT2 001260
\$BUF = 000006	\$HARD = 000062	\$OMODE 034130	\$SAVRE 035116	\$WRDL = 000020
\$CDW1 001266	\$HD = 000000	\$OPERC= 000036	\$SAVR6 033560	\$WRDM = 000004
\$CDW2 001270	\$HIBTS 001100	\$PACK = 000026	\$SB2D 031524	\$XOFF = 000023
\$CHARC 033136	\$HINUM 035112	\$PASS 001214	\$SB20 031554	\$XON = 000021
\$CKSWR 034356	\$ICNT 001120	\$PASSC= 000070	\$SEC = 000010	\$GET4= 000000
\$CMTAG 001114	\$ILLUP 033554	\$PASTM 001106	\$SETUP= 000156	\$OFILL 034127
\$CM3 = 000000	\$INTAG 001151	\$PATTC= 000030	\$SIZE 076346	.\$X = 001100
\$CM4 = 000001				

. ABS. 077752 000
 000000 001
 ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 53264 WORDS (209 PAGES)

M 15

SEQ 0194

CZRMUAO RM05/3/2 PERF EXER MACRO V03.01 11-APR-80 14.52:06 PAGE 16-9 ^{N 15}
SYMBOL TABLE

SEQ 0195

DYNAMIC MEMORY AVAILABLE FOR 69 PAGES
CZRMUA.BIC,CZRMUA/C=CZRMUA.DOC,CZRMUA,SYSMAC/M

SMTYP1 5-04

D 16

SEQ 0198

SMTYP2	5-0#													
SMTYP3	5-0#													
SMTYP4	5-0#													
SNCODE	9-197*	9-?24*	9-?87	9-?90	9-a15	9-a22	9-a70*	9-a98	9-A12	9-A13	10->55#	10->56	10->57	10->58
SNCYL	10->59	10->60	10->61	10->65										
SNEXT	9-193*	9-?28*	9-?29	9-?31*	9-?85*	9-a45	9-A30	10->59#						
SNPATC	9-108	9-203*	9-a28*	9-A38*	9-B15*	9-L27	10->61#							
SNSEC	9-202*	9-a27*	9-A28	10->56#										
SNTRK	9-195*	9-?27*	9-?32*	9-?50*	9-A29	10->57#								
SNUL	9-194*	9-?33*	9-?69*	10->58#										
SNWRDL	5-0#	9-T72	9-T72	9-T72										
SNWTST	9-196*	9-?89*	9-a21*	9-A31	9-A32	10->60#								
SOCNT	8-7#													
SOCTLV	9-U01#	9-U01*	9-U01*											
SOMODE	9-U13	9-U13#												
SOPERC	9-U01	9-U01#	9-U01*	9-U01*	9-U01*	9-U01*								
SPACK	9->98*	9->99*	9-A64	9-A66	9-F15	9-F42	9-H34	9-P41	10->41#					
SPASS	8-240*	9-110	9-A68	9-A87	9-B04	9-B18*	9-J98*	10->37#						
SPASSC	5-0#	8-13*	9-P41	9-P41	9-P41*	9-P41*	9-P41*	9-P41*						
SPASTM	9-H37	9-J95*	9-P41	9-P41	9-P41*	10->50#								
SPATTC	4-275#													
SPOSIT	9->72	9-A28*	9-B14*	9-L39*	9-L40*	10->39#								
SPOWER	9-=40*	9-=41*	9-?02*	9-?03*	9-F47	9-P41	9-P41	10->42#						
SPREVA	9-T76	9-T76#												
SPREVO	9-?00	9-A00*	9-A01*	9-A04*	9-A05*	9-A06*	9-A73*	9-A74*	9-A75*	9-D59	9-D63	9-D67	9-D75	10-799*
SPSEL	10-800*	10->40#												
SPWRAD	9-a96*	9-a97*	9-A17	9-A24	9-A71*	9-C05	10-798*	10->38#						
SPWRDN	10->23#													
SPWRMG	9-T76#													
SPWRUP	8-13	9-T76	9-T76#											
SQUES	9-T76#													
SR2A	9-T76	9-T76#												
SRAND	9-T76	9-T76#												
SRDCHR	5-0#	9-T37	9-T68	9-T68	9-T72	9-T72	9-U05							
SRDDEC	9-U15													
SRDLIN	9-?12	9-a06	9-a83	9-A63	9-U07#									
SRDOCT	9-U05#	9-U15	9-U15											
SRDSZ	?-U15													
SREAD	9-S97#	9-U15	9-U15											
SREG	9-U15													
SRESRE	9-U05	9-U05#												
SRETRY	9-=36*	9-=37*	9-F28	9-P41	9-P41	10->44#								
SRM02	9-L27	10->29#	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11
SRM03	10-?11	10-?11	10-?11	10-?11										
SRM05	9-U09#	9-U15												
SRMADR	9-382	9-467	9-534	9-581	9-622	9-658	9-686	9-756	9-787	9-803	9-<89#	9-=07		
SRMAS	8-185	15-15#												
SRMBA	8-182	15-16#												
SRMCS1	8-188	15-17#												
SRMCS2	6-0#	8-59*	8-60	8-67	13-5	16-102	16-120							
SRMDA	10->90#	13-7												
	9-870	9-:91	9-:22	9-:43	9-:75	9-:95	9-=24	9-C52	9-D02	9-E07	9-E18*	9-E19	10->85#	13-7
	9-211	9-213	9-846	9-a96	9-A71	9-B99	10-798	10->83#	10->84	10->85	10->86	10->87	10->88	10->89
	10->90	10->91	10->92	10->93	10->94	10->95	10->96	10->97	10->98	10->99	10-?00	10-?01	10-?02	10-?11
	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	10-?11	13-6						
	9-319	9-477	9-527	9-573	9-614	9-720	9-735	9-752	9-848	10->87#	13-6			
	9-<33*	9-?27	9-A76	9-D35	9-D90	9-G19	9-G20	10->86#	13-7					

SRMDB 9-E22 10->92# 13-7

F 16

SEQ 0200

STRANS 9--28* 9--29* 9--34* 9--35* 9-F23 10->43*

H 16

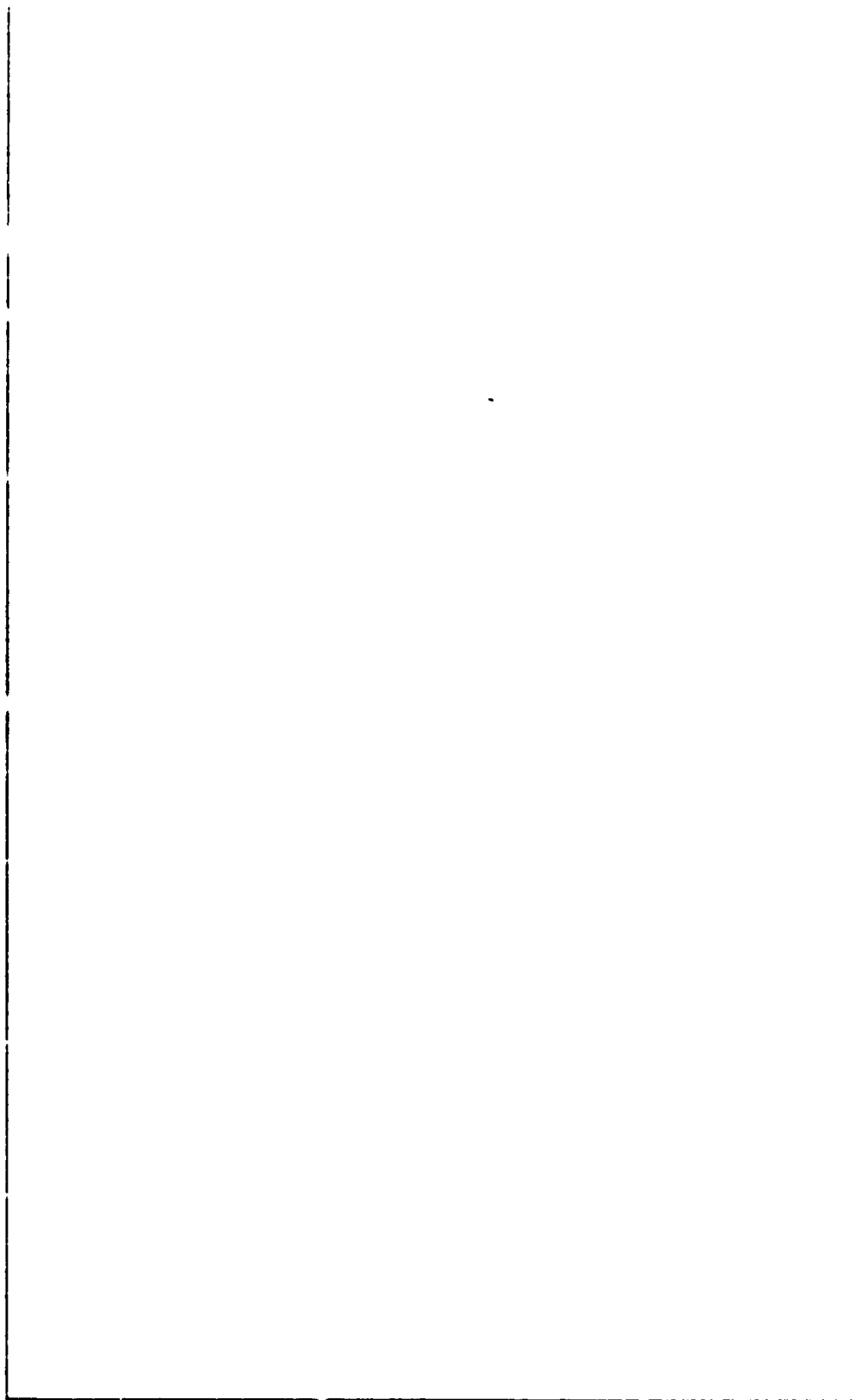
SEQ 0202



ALUNIT 5-0 5-0

L 16

SEQ 0206



CHKADP 9-198 9-292 9-240#

C 1

SEQ 0208

DATA13 6-0 6-0#

E 1

SEQ 0210

DROP 9-256 9-301 9-040# 9-065

G 1

SEQ 0212

EM10

9-262

11-9#

I 1

SEQ 0214

EM11	9-269	11-10#						
EM12	9-250	11-11#						
EM13	9-283	11-12#						
EM14	9-295	11-13#						
EM15	9-306	11-14#						
EM2	7-11	11-4#						
EM20	9-522	11-15#						
EM21	9-392	11-16#						
EM22	9-476	11-17#						
EM23	9-460	11-18#						
EM24	9-569	11-19#						
EM25	9-610	11-20#						
EM26	9-716	11-21#						
EM27	9-731	11-22#						
EM3	7-18	11-5#						
EM30	9-547	11-23#						
EM31	9-651	11-24#						
EM32	9-673	11-25#						
EM33	9-679	11-26#						
EM34	9-779	11-27#						
EM35	9-697	11-28#						
EM36	9-707	11-29#						
EM37	9-479	11-30#						
EM4	7-25	11-6#						
EM40	9-747	11-31#						
EM41	9-873	11-32#						
EM42	9-:21	11-33#						
EM43	9-816	9-821	11-34#					
EM44	9-855	11-35#						
EM45	9-378	11-36#						
EM46	9-C54	11-37#						
EM5	7-32	11-7#						
EM50	9-767	11-38#						
EM51	9-635	11-39#						
EM6	7-39	11-8#						
EM60	9-798	11-40#						
EMPTYQ	10-631	10-682	10-783	10-:98	10-=38#			
EMTVEC	4-81#	8-13*	8-13*					
ENDCMP	9-995	9-:52#						
ENDCON	6-0#	8-210*	8-211*	8-215*	8-216*	9-P41	9-P41	
ENDET	6-0#	9-P41	16-9					
ENDPAS	9-P41	15-23#						
ENDPGM	8-95	8-100	16-173#	:				
ENDSEK	6-0#	9-P41	9-P41					
ENDTST	9-P41	15-25#						
ENTADR	9-N10	15-39#						
ENTCOM	9-I70	15-36#						
ENTDAT	16-63	16-81#						
ENTDRV	9-M09	15-37#						
ENTID	16-71	16-82#						
ENTLMT	9-L62	15-38#						
ENTPR	8-134#							
EOP1	9-P41	9-P41#						
EOP2	9-A89	9-P41	9-P41	9-P41	9-P41#			
EOPX	9-P41	9-P41	9-P41	9-P41	9-P41	9-P41	9-P41#	
ERCTR	6-0#	9-890*	9-922*	9-934	9-947*	9-:54	9-:57	

ERPRC1 9-227 9-232# 9-414 9--14

K 1

SEQ 0216

IDLE1 9-152 9-170#

M 1

SEQ 0218

LINB6 9-E65 9-E78 14-28#

B 2

SEQ 0220

LINT3 9-D88 14-12#

D 2

SEQ 0222

MXLACT 10-173# 10-708

F 2

SEQ 0224

MXWINDW	10-182#									
N	8-128	15-31#								
NED	4-116#									
NEDCLK	9-663	15-30#								
NEM	4-115#									
NEWASN	9-188	9-K20#								
NEWUNT	6-0#	8-234*	9-51	9-66	9-67*	9-J94*				
NOMTCH	9-813#	9-928								
NONE	9-K83	15-44#								
NOTAVL	15-11#									
NOTPRS	8-165	9-K10	15-10#							
NOTPRT	6-0#	9-855	16-12							
NOTRM	8-162	9-K08	15-9#							
NOTSAF	8-171	9-K02	15-12#							
NSA	4-200#									
OFFCOD	6-0#	9-423								
OFFDIR	4-214#									
OFFON	4-138#									
OFFSET	4-248#	9-<50								
OFFST	9-426	9-<49#								
OFLIN	9-241	9-293#								
OFMSG0	6-0	6-0#								
OFMSG1	6-0	6-0#								
OFMSG2	6-0	6-0#								
OFMTBL	6-0#	9-E96								
ONES	6-0	6-0#								
OPE	4-227#									
OPERID	6-0#	9-K71	9-K74	16-74*	16-77*	16-77*	16-77*	16-77*	16-77*	16-78*
OPI	4-165#									
OPIER	9-336	9-648#								
OPIER1	9-657#									
OPRDAT	8-71	16-63#								
OPT	10-387	10-422#	10-777	10-963	10-:01					
OPTBL	6-0#	9-C10	9-C12							
OR	4-111#									
ORDERQ	6-0#	8-73	9-5	9-90	9-101	9-123	9-150	9-162	9-T94	
PACK	6-0#	8-84*	8-254*	9-817*	9-J96	9-J98	9-K20*	9-K97*	9-L03*	9-L08*
PAR	4-155#									
PAR1	16-3	16-18#								
PAR10	16-10	16-27#								
PAR11	16-7	16-28#								
PAR14	16-8	16-29#								
PAR15	16-30#									
PAR16	16-9	16-31#								
PAR19	16-5	16-32#								
PAR2	16-4	16-19#								
PAR20	16-12	16-33#								
PAR21	16-13	16-34#								
PAR3	16-6	16-20#								
PAR4	16-21#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PAR5	16-22#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PAR6	16-23#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PAR7	16-24#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PAR8	16-25#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PAR9	16-26#	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54	16-54
PARENT	8-135	9-L78	9-N92#							

PARER 9-339 9-678#

H 2

SEQ 0226

RD.ADP 10-;17* 10-;18* 10-;20* 13-3

J 2

SEQ 0228

RMSN 10-198#

L 2

SEQ 0230

STATPP 9-179 9-G77# 9-K66

N 2

SEQ 0232

TAP

4-199#

C 3

SEQ 0234

