

RP04

EXERCISER PROGRAM
MD-11-DERPN-B

EP-DERPN-B-DL-A
COPYRIGHT © 1976
FICHE 1 OF 2

NOV 1976
digital
MADE IN USA

This image shows a microfiche card with a grid of frames. Each frame contains a small, high-contrast image of a document page, likely a technical manual or exercise program. The frames are arranged in a regular grid pattern across the card. The text within the frames is too small to be legible, but it appears to be organized into columns and rows, possibly representing a table of contents or a list of exercises. The card is labeled 'RP04' in the top left corner, 'EXERCISER PROGRAM MD-11-DERPN-B' in the top center, 'EP-DERPN-B-DL-A COPYRIGHT © 1976 FICHE 1 OF 2' in the top right, and 'NOV 1976 digital MADE IN USA' in the top right corner.

RP04

MULTI-DRIVE EXERCISER
MD-11-DERPN-B

EP-DERPN-B-DL-A

NOV 1976

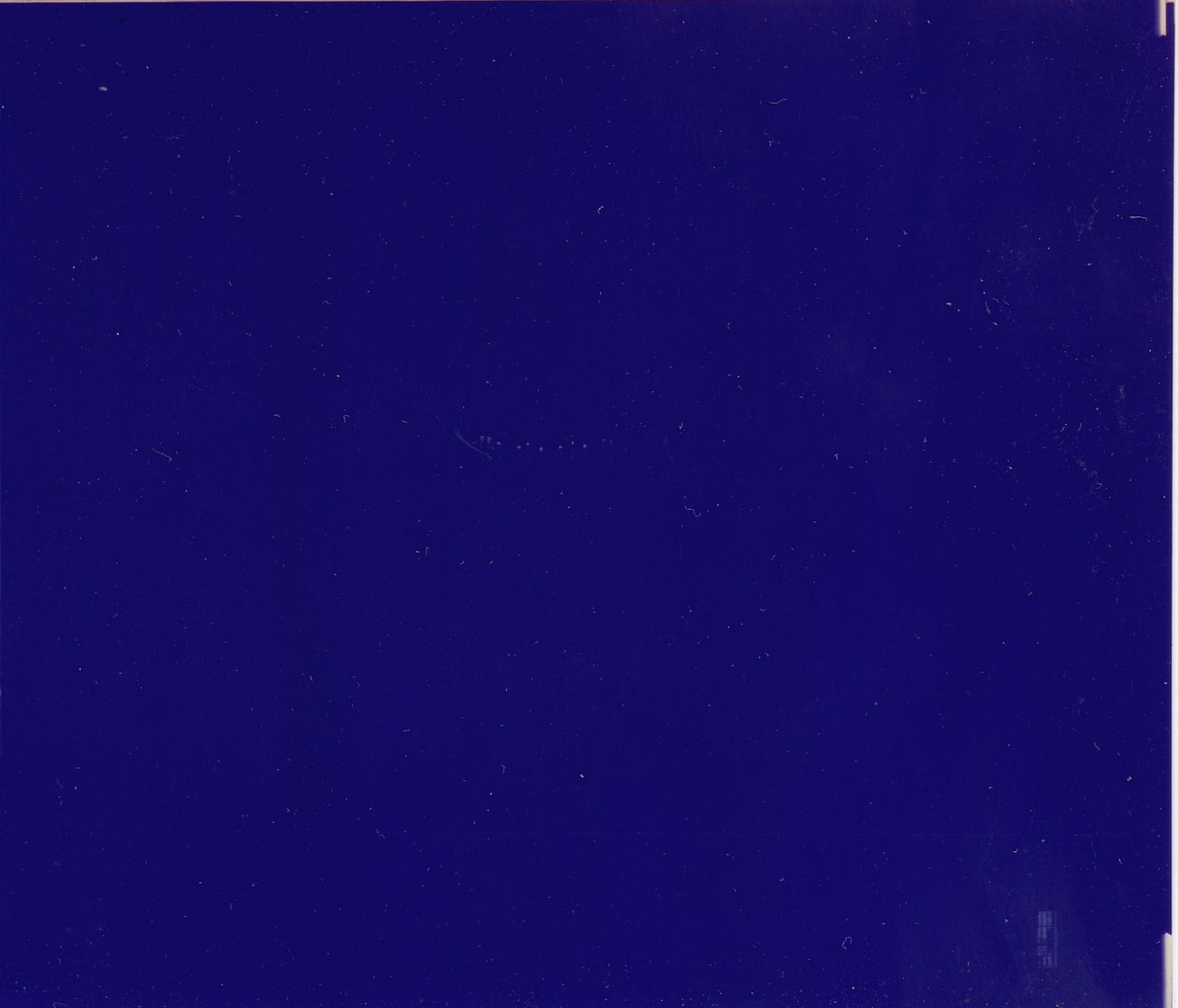
COPYRIGHT © 1976

digital

FICHE 2 OF 2

MADE IN USA

This microfiche card contains a grid of frames. The first column contains frames with vertical text, likely labels for the data. The subsequent columns contain frames with horizontal text, representing data records. The text is too small to be legible in this image.



CONTENTS

1. ABSTRACT
2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 MEDIA
 - 2.3 PRELIMINARY PROGRAMS
3. OPERATING THE PROGRAM
 - 3.1 LOADING THE PROGRAM
 - 3.2 STARTING 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 STARTUP
4. CONTROLLING THE PROGRAM
 - 4.1 DATE & OPERATOR IDENTIFICATION
 - 4.2 PARAMETERS
 - 4.2.1 KEYBOARD ENTRY PARAMETERS
 - 4.2.2 PERIPHERAL ADDRESS PARAMETER LOCATIONS
 - 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 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
 - 6.1 DATA BUFFER COMPARISON
 - 6.2 VERIFICATION OF DATA WRITTEN
 - 6.3 SECTOR REFORMATTING
 - 6.4 BAD TRACK/SECTOR FLAGGING
7. ERROR MESSAGES

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
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

D01

MAINDEC-11-DERPN-S
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 4

100
101
102
103

7.1 ERROR DESCRIPTION LINES
7.2 DETAIL ERROR LINES

104
105
106
107
108
109
110
111
112

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. PROGRAM LISTING

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

1. ABSTRACT

THE RPO4 MULTIDRIVE EXERCISER PROGRAM EXERCISES 1 TO 8 RPO4 DISK DRIVES ATTACHED TO THE SAME RM70. IF 2 OR MORE RPO4 DISK DRIVES ARE BEING EXERCISED, OPERATIONS ON THE DRIVES ARE OVERLAPPED (OTHER DRIVES ARE PERFORMING SEEK/SEARCH OPERATIONS WHILE ONE RPO4 IS PERFORMING A DATA TRANSFER OR WRITE CHECK OPERATION). OPERATIONS AMONG THE RPO4'S ARE OPTIMIZED SO THAT A HIGH SUBSYSTEM DATA TRANSFER RATE OR A HIGH POSITIONING OPERATION RATE IS MAINTAINED.

THE RPO4 MULTIDRIVE EXERCISER PROGRAM WILL EXERCISE RPO4'S CONNECTED AS EITHER SINGLE OR DUAL PORT UNITS. DUAL PORT RPO4'S ARE TESTED BY LOADING AND RUNNING THE PROGRAM FROM BOTH CONTROLLING SYSTEMS. THE PROGRAM WILL EXERCISE A MIXED SYSTEM OF DUAL PORT AND NON DUAL PORT RPO4'S.

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 PROCESSING.

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.

PROGRAM/OPERATOR COMMUNICATIONS ARE THROUGH THE TELETYPE; PROGRAM OPTIONS ARE SELECTED BY SWITCH REGISTER SETTINGS. ERROR ARE NORMALLY REPORTED ON THE TELETYPE; HOWEVER, IF A LINE PRINTER IS AVAILABLE THE PROGRAM WILL USE THE PRINTER FOR ERROR MESSAGE DISPLAY.

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.

2. REQUIREMENTS

2.1 EQUIPMENT

REQUIRED

PDP-11 PROCESSOR

GO1

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 7

169
170
171
172

16K MEMORY
TELETYPE
PROGRAM LOADING DEVICE
KW11-L OR KW11-P CLOCK

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

RH70 WITH 1 RPO4

OPTIONAL

4K TO 12K ADDITIONAL MEMORY
LINE PRINTER
1 TO 7 ADDITIONAL RPO4'S ON THE SAME RH70

2.2 MEDIA

THE RPO4 MULTIDRIVE EXERCISER PROGRAM REQUIRES FORMATTED DISK PACKS WHICH CONTAIN RANDOM OR PATTERNED DATA RECOGNIZED BY THE EXERCISER. DISK PACKS USED BY THE PROGRAM MAY BE GENERATED BY THE RPO4 FORMATTER PROGRAM (MAINDEC-11-DERPL) OR BY THE 'W' COMMAND OF THE RPO4 MULTIDRIVE EXERCISER (SEE SECTION 4.4). THE PACKS MUST BE FORMATTED IN 22 SECTOR (16 BIT) MODE; THE ALTERNATE (20 SECTOR - 18 BIT) MODE IS NOT SUPPORTED.

2.3 PRELIMINARY PROGRAMS

RPO4 DISKLESS CONTROLLER TEST
PART 1 (MAINDEC-11-DERPS)
PART 2 (MAINDEC-11-DERPT)

RPO4 FUNCTIONAL CONTROLLER TEST
PART 1 (MAINDEC-11-DERPU)
PART 2 (MAINDEC-11-DERPV)

RPO4 DUAL CONTROLLER LOGIC TEST (FOR DUAL PORT DRIVE TESTING)
PART 1 (MAINDEC-11-DERPP)
PART 2 (MAINDEC-11-DERPQ)

3. OPERATING THE PROGRAM

3.1 THE PROGRAM MAY BE LOADED WITH THE ABSOLUTE PAPER TAPE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE MEDIA USING THE ASSOCIATED 'XXDP' LOADER. THE PROGRAM CAN BE INSTRUCTED TO PRESERVE EITHER LOADER TYPE.

3.2 THE PROGRAM STARTS AT LOCATION 200(8). PARAMETERS NOT INCLUDED IN THE TELETYPE DIAGLOGUE GROUP MUST BE CHANGED BEFORE THE PROGRAM IS STARTED.

3.3 THE RESTART LOCATION IS 204(8)

3.4 ONCE THE PROGRAM IS LOADED AND STARTED, OPERATIONS ARE INITIATED AND CONTROLLED BY KEYBOARD COMMANDS AND SWITCH REGISTER SWITCH SETTINGS.

3.5 PASS/TEST TERMINATION

3.5.1 PASS TERMINATION

229
230
231
232

END OF PASS MAY BE DETERMINED BY EITHER OF THE FOLLOWING CONDITIONS.
THE END OF PASS CONDITION USED IS DETERMINED BY PARAMETER 'ENDET'.

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
286
287
288

- A. IF PARAMETER 'ENDET' IS 1, END OF PASS OCCURS WHEN THE DRIVE HAS READ 1.875×10^{18} WORDS (3×10^{19} BITS).
- B. IF PARAMETER 'ENDET' IS 0, END OF PASS OCCURS WHEN THE DRIVE HAS PERFORMED 3×10^{16} SEEKS.

3.5.2 TEST TERMINATION

THE TEST FOR A DRIVE IS TERMINATED (SW<04> = 0) WHEN:

- A. THE DRIVE HAS COMPLETED THE NUMBER OF PASSES SPECIFIED IN PARAMETER 'PASCNT'.
- B. THE TOTAL ERRORS ACCUMULATED EXCEED 100.
- C. A FATAL ERROR OCCURS: EM12 OR EM14.

3.6 RUN TIME

THE EXERCISER PROGRAM MAY BE RUN IN TWO MODES. THE MODE IS DETERMINED BY THE VALUE IN PARAMETER 'MAXDL'. IF 'MAXDL' IS ONE SECTOR, THE PROGRAM RUNS IN A SEEK HEAVY MODE; IF 'MAXDL' APPROACHES ONE TRACK IN SIZE (5720 DECIMAL) 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 - APPROXIMATELY 2.5 HRS (TO REACH 1.875×10^{18} WORDS)
- TO
- 8 DRIVES - APPROXIMATELY 11 HRS (FOR ALL DRIVES TO REACH 1.875×10^{18} WORDS)

NOTE: IF SW<01> = 1 (NO SOFTWARE DATA COMPARISONS), THE RUN TIMES ARE THE FOLLOWING VALUES, APPROXIMATELY:

- 1 DRIVE - 1.7 HRS (1.875×10^{18} WORDS READ)
- ADD 1/2 HOUR FOR EACH ADDITIONAL DRIVE TESTED.

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

- 1 DRIVE - APPROXIMATELY 25 HRS (3×10^{16} SEEKS)
- TO
- 8 DRIVES - APPROXIMATELY 40 HRS (3×10^{16} 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.)

UNIBUS VECTOR

K01

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 11

289
290
291
292

UNIT

ADDRESS

ADDRESS

RH70/RP04

176700

254

3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319
3320
3321
3322
3323
3324
3325
3326
3327
3328
3329
3330
3331
3332
3333
3334
3335
3336
3337
3338
3339
3340
3341
3342
3343
3344
3345
3346
3347
3348

TTY PRINTER	177564	NOT USED
TTY KEYBOARD	177560	60
KW11-L	177546	100
KW11-P	172542	104
LINE PRINTER	177514	NOT USED

3.8 DUAL PORT STARTUP

- A. LOAD THE RPO4 MULTIDRIVE EXERCISER PROGRAM (REVISION B OR LATER) INTO BOTH PROCESSORS.
- B. SWITCH THE 'CONTROLLER SELECT' SWITCH TO 'A/B' ON EACH RPO4 WHICH IS TO BE TESTED AS A DUAL PORT DRIVE; CYCLE THE DRIVES UP.
- C. START THE PROGRAM IN EACH PROCESSOR. RUN THE PROGRAM AS THROUGH EACH PROCESSOR WERE RUNNING INDEPENDENTLY OF THE OTHER.

4. CONTROLLING THE PROGRAM

4.1 DATE & OPERATOR IDENTIFICATION

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE PROGRAM WILL ASK FOR A DATE ENTRY AND FOR AN OPERATOR I.D. ENTRY. 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 OPERATION IDENTIFICATION MAY BE ENTERED. BOTH THE DATE AND THE OPERATOR I.D. WILL BE TYPED WHEN THE 'SA' COMMAND IS PERFORMED (SEE SECTION 4.4.3).

4.2 PARAMETERS

WHEN THE PROGRAM IS STARTED FROM LOCATION 200, THE OPERATOR WILL BE ASKED TO ENTER PARAMETERS. THE FOLLOWING MESSAGE WILL BE DISPLAYED:

ENTER PARAMETERS:

THE OPERATOR MUST ENTER A 'Y' TERMINATED BY A CARRIAGE RETURN (CR) IF PARAMETER ENTRIES ARE TO BE MADE OR AN 'N' IF NO ENTRIES WILL BE MADE. THE PROGRAM WILL IDENTIFY THE PARAMETER BY THE NAME GIVEN BELOW, DISPLAY THE CURRENT VALUE OF THE PARAMETER AND THE BASE OF THE PARAMETER (OCTAL OR DECIMAL), AND WAIT FOR THE ENTRY. IF ONLY A CARRIAGE RETURN IS ENTERED, THE PROGRAM WILL USE THE PRESENT VALUE OF THE PARAMETER. THE RUBOUT KEY WILL ALLOW THE OPERATOR TO DELETE AN INCORRECT ENTRY; 'CONTROL U' WILL ALLOW AN ENTIRE ENTRY TO BE DELETED AND REENTERED. THE PROGRAM WILL TYPE A '?' IF AN ALPHABETIC OR INCORRECT NUMBER (FOR THE BASE OF THE

MO1

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 13

349
350
351
352

PARAMETER) IS ENTERED. IF 'CONTROL C' IS ENTERED, THE PROGRAM WILL
USE THE PRESENT VALUES OF THE REMAINING PARAMETERS AS DEFAULT VALUES.

(NOTE: A PARAMETER ENTRY MUST BE TERMINATED BY A 'CARRIAGE

RETURN' TO BE ACCEPTED. IF THE PARAMETER ENTRY IS FOLLOWED BY A 'CONTROL C' INSTEAD OF A 'CARRIAGE RETURN', THE PROGRAM WILL RECOGNIZE ONLY THE 'CONTROL C' AND THE NEW PARAMETER VALUE WILL NOT BE ACCEPTED.)

4.2.1 KEYBOARD ENTRY PARAMETERS

NAME	BASE	DEFAULT VALUE	VALUE RANGE	FUNCTION
MAXDL	10	(SEE NOTE)		CONTROLS THE MAXIMUM BUFFER SIZE USED FOR DATA TRANSFERS
PASCNT	10	1	1 - 999	NUMBER OF PASSES TO END OF TEST.
INTRVL	10	5	0 - 256	DETERMINES THE INTERVAL (IN MINUTES) BETWEEN AUTOMATIC PERFORMANCE SUMMARY TYPEOUTS
CMPLMT	10	3	0 - 'MAXDL'	ERRORS PRINTED OUT IF SW<07>=0
MAXCYL	10	410	0 - 410	THE NUMBER OF DATA COMPARISON
MINCYL	10	0	0 - 410	THE MAXIMUM CYLINDER ADDRESS
MAXTRK	10	18	0 - 18	THE MINIMUM CYLINDER ADDRESS.
MINTRK	10	0	0 - 18	VALUE MUST NOT BE GREATER THAN VALUE IN 'MAXCYL'
MAXSEC	10	21	0 - 21	THE MAXIMUM TRACK ADDRESS
MINSEC	10	0	0 - 21	THE MINIMUM TRACK ADDRESS
BEGCOD	10	5	0 - 5	THE MINIMUM TRACK ADDRESS MUST NOT BE GREATER THAN THE VALUE IN 'MAXTRK'
BEGPAT	10	8	1 - 15	THE MAXIMUM SECTOR ADDRESS
ENDET	8	000001	0 OR 1	THE MINIMUM SECTOR ADDRESS MUST NOT BE GREATER THAN THE VALUE IN 'MAXSEC'
FORMAT	8	000001	0 OR 1	THE INITIAL COMMAND FOR EACH DRIVE EXERCISED.

353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408

409
410
411
412

\$RPADR
\$RPVEC

8
8

176700
254

N/A
N/A

PERFORM WRITE HEADER & DATA
ORDERS
THE RH70 UNIBUS ADDRESS
THE RH70 VECTOR ADDRESS

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

SAVLOD 8 000001 0 OR 1

IF PARAMETER = 0, DO NOT PRESERVE THE LOADER; IF PARAMETER > 0, PRESERVE THE LOADER
CONTROLS THE APPROXIMATE RATIO OF READ TO WRITE ORDERS.

RATIO 8 3 0 - 7

VALUE R/W RATIO

0 15/1
1 7/1
2 6/2
3 5/3
4 4/4
5 3/5
6 2/6
7 1/7

AUTOCK 8 000001 0 OR 1

IF PARAMETER = 1, THE PROGRAM PERFORM WRITE CHECKS AFTER EACH WRITE COMMAND.
IF PARAMETER = 0, THE PROGRAM WILL PERFORM WRITE CHECKS RANDOMLY.

NOTPRT 8 000001 0 OR 1

IF PARAMETER = 1, DO NOT PRINT ERROR MESSAGES FOR DATA ERRORS OCCURING AT LOCATIONS DEFINED BY THE OPERATOR AS BAD PACK LOCATION.
IF PARAMETER = 0, PRINT ERROR MESSAGES ASSOCIATED WITH BAD PACK LOCATIONS.

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 5980 (10) 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 NOT PROVIDE ACCESS TO THESE LOCATIONS.

LOC	TAG	CONTENTS	FUNCTION
---	---	-----	-----
1166	\$LKCSR	172540	ADDRESS OF KW11-P STATUS REGISTER
1170	\$LKCSB	172542	ADDRESS OF KW11-P COUNTER BUFFER
1172	\$LPVEC	104	KW11-P VECTOR ADDRESS
1174	\$LKSL	177546	ADDRESS OF KW11-L STATUS REGISTER

469
470
471
472

1176	\$LLVEC	100
1206	\$LSCS	177514
1210	\$LSDB	177516
1214	HZ	74

KW11-L VECTOR ADDRESS
ADDRESS OF LINE PRINTER STATUS REGISTER
ADDRESS OF LINE PRINTER DATA BUFFER
74 (60 DECIMAL) IF SYSTEM IS 60 HZ;

473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528

1216 LA30 0

62 (50 DECIMAL) IF SYSTEM IS 50 HZ.
0 IF AN LA30 IS ON THE SYSTEM;
177777 IF MOD 33 OR MOD 35 TTY
ON THE SYSTEM

4.3 SWITCH REGISTER SETTINGS

- SW <15> = 1 HALT ON ERROR
- SW <13> = 1 INHIBIT ERROR TIMEOUT
- 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 NORMAL END OF TEST REACHED.
- SW <3> = 1 DISPLAY THE SECTOR IN ERROR (BEFORE RETRY ATTEMPTS) IF 'DCK', 'DTE', OR 'WCF' ERRORS OR AFTER THE 28TH RETRY IF 'UNCORRECTABLE' 'DCK' ERROR. IF DATA COMPARE ERRORS & SW<7> SET, DISPLAY REST OF BUFFER
- SW <1> = 1 INHIBIT DATA COMPARSION AFTER READ ORDERS
- SW <0> = 1 READ ONLY MODE

4.4 KEYBOARD COMMANDS

THE KEYBOARD COMMANDS CONTROL THE ASSIGNMENT/DEASSIGNMENT OF DRIVES, ALLOW THE OPERATOR TO REQUEST DRIVE PERFORMANCE SUMMARIES, AND ALLOW DATA PACKS TO BE WRITTEN

WHEN THE PROGRAM IS STARTED (OR RESTARTED), KEYBOARD COMMANDS ARE NOT RECOGNIZED UNTIL THE PROGRAM HAS INITIALIZED ITSELF. THE OPERATOR MUST WAIT UNTIL THE 'PROGRAM INITIALIZE COMPLETE' MESSAGE IS TYPED BEFORE ATTEMPTING TO USE THE COMMANDS. AFTER INITIALIZATION HAS BEEN COMPLETED, THESE COMMANDS MAY BE USED.

EACH COMMAND, EXCEPT THE 'S' AND THE 'D' COMMANDS, WILL ASK FOR A DRIVE IDENTIFICATION ENTRY AND FOR THE ADDRESSES OF ANY BAD SPOTS ON THE PACK BEING USED ON THE DRIVE WHICH IS BEING ASSIGNED. THE DRIVE IDENTIFICATION ENTRY REQUEST IS MADE BY THE FOLLOWING MESSAGE:

'ENTER I.D. FOR DRV #N:'

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 'CARRIAGE RETURN' ONLY (NULL ENTRY) IN RESPONSE TO THE I.D. REQUEST.

THE PROGRAM WILL THEN ASK FOR THE ADDRESSES OF KNOWN BAD SPOTS ON THE DISK PACK USED:

'BAD TRK/SEC ADRS FOR DRV #N ?'

F02

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 19

529
530
531
532

THE OPERATOR MAY SPECIFY UP TO 8 BAD LOCATIONS FOR THE PACK BEING
USED. THE FORMAT FOR THE BAD SPOT ADDRESS ENTRY IS AS FOLLOWS:

FORMAT 1: C,T,S<CR>

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

- A. LEADING ZEROS ARE NOT REQUIRED. THE ENTRY MUST BE TERMINATED BY A 'CARRIAGE RETURN' (<CR>).
- B. THE PROGRAM WILL INHIBIT DATA ERROR MESSAGES OR WILL IDENTIFY DATA ERRORS WHICH OCCUR AT THE SPECIFIED ADDRESS, DEPENDING ON THE VALUE OF PARAMETER 'NCTPRT'.

FORMAT 2: C,T<CR>

- A. LEADING ZEROS ARE NOT REQUIRED. THE ENTRY MUST BE TERMINATED BY A 'CARRIAGE RETURN' (<CR>).
- B. WHEN THIS FORMAT IS USED, THE ENTIRE TRACK WILL BE CONSIDERED BAD; DATA ERRORS WILL BE HANDLED AS IN 'FORMAT 1', ABOVE.

NOTE: CYLINDER, TRACK, AND SECTOR ENTRIES ARE IN DECIMAL.

THE OPERATOR MAY BYPASS ALL OF THE BAD TRACK/SECTOR ADDRESS ENTRIES BY ENTERING A 'CONTROL C' INSTEAD OF AN ADDRESS. THE ADDRESS ENTRY MAY BE TERMINATED AT ANY POINT BY ENTERING A 'CONTROL C'. ALSO. NOTE THAT AN ENTRY MUST BE TERMINATED BY A 'CARRIAGE RETURN' TO BE ACCEPTED BY THE SYSTEM. THE 'CONTROL C' MUST BE ENTERED IN PLACE OF AN ADDRESS ENTRY AND NOT IN PLACE OF THE 'CARRIAGE RETURN' TERMINATOR FOR AN ENTRY.

FOR BOTH DRIVE I.D. AND BAD TRACK/SECTOR ADDRESS ENTRIES, 'RUBOUT' AND 'CONTROL U' MAY BE USED TO CORRECT MIS-TYPED CHARACTERS AND MIS-TYPED LINES.

4.4.1 'T' COMMAND

USED TO ASSIGN A DRIVE(S) FOR TEST

FORMAT: TN<CR>

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

EXAMPLE: TD<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>

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

EXAMPLE: DD<CR> - DEASSIGN DRIVE 0
DA<CR> - DEASSIGN ALL DRIVES BEING

H02

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 21

589
590
591
592

TESTED.

NOTES: 1. IF THE 'D' COMMAND REFERENCES A
DRIVE NOT ASSIGNED THE PROGRAM

5
11
11

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

WILL TYPEOUT '?DRIVE NOT ASSIGNED'

2. THE DRIVES WILL BE DEASSIGNED AS THEIR OPERATIONS COMPLETE.
3. IF 'DA' IS USED, ONLY DRIVES BEING TESTED WILL BE DEASSIGNED - THE ERROR MESSAGE IN (1) ABOVE WILL NOT BE DISPLAYED.

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 TERMINATED BY A CARRIAGE RETURN <CR>.

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

- NOTES:
1. IF 'SA' IS USED ONLY DRIVES BEING TESTED WILL BE DISPLAYED.
 2. IF PARAMETER 'INTRVL' IS NOT ZERO, THE PROGRAM WILL AUTOMATICALLY DISPLAY A PERFORMANCE SUMMARY FOR EACH DRIVE BEING TESTED AT A RATE DETERMINED BY 'INTRVL'.
 3. 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 RPO4 MULTI-DRIVE EXERCISER PROGRAM.

FORMAT: WN<CR>

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

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

- NOTES:
1. DATA PACKS GENERATED BY THE RPO4 FORMATTER PROGRAM (MD-11-DERPL) OR BY THE RPO4 MECHANICAL & READ/WRITE PROGRAM (MD-11-DERPK), TEST 15, ARE ACCEPTABLE. (PACKS WRITTEN BY TESTS 13, 14 OR 16 OF 'DERPK' CANNOT BE USED AND MUST BE REWRITTEN.)

649
650
651
652

2. THE 'W' COMMAND SHOULD NOT BE USED UNLESS 'MAXDL'
PARAMETER IS APPROXIMATELY 1 TRACK IN SIZE - 5000 (10).
IF THE BUFFER SIZE IS MUCH LESS THAN 1 TRACK, THE

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

TIME REQUIRED TO WRITE A DATA PACK IS TOO GREAT TO BE PRACTICAL.

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 SET DURING THE OPERATION OF THE 'W' COMMAND, THE DRIVE PERFORMING THE 'W' COMMAND WILL IGNORE THE SWITCH.
6. THE DATA WRITTEN IS VERIFIED BY A 'WRITE CHECK DATA' COMMAND.

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

- 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 GENERAL COMMAND INFORMATION

- A. WHEN A COMMAND IS ENTERED, THE PROGRAM WILL TYPE OUT THE TIME
- B. IF THE COMMAND ENTERED IS NOT VALID, THE PROGRAM WILL TYPE 'INVALID COMMAND'.
- C. DRIVES ASSIGNED (WITH THE 'T' COMMAND) OR DEASSIGNED (WITH THE 'D' COMMAND) MAY BE ENTERED IN ANY SEQUENCE.
- D. THE ERROR RESPONSES FROM THE PROGRAM ARE AS FOLLOWS

RESPONSE

COMMAND(S)

709
710
711
712

?UNIT N OFFLINE
?UNIT N NOT ASSIGNED
?UNIT N ALREADY ASSIGNED

T, W, R
D, S, R
T, W, R

?UNIT N NOT PRESENT T, W, R
 ?UNIT N UNSAFE T, W, R
 ?UNIT N NOT AN RPO4 T, W, R

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

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' OR 'OCYL' ERRORS
'MISP'	THE NUMBER OF POSITIONING ERRORS
'OTHER'	THE TOTAL ERRORS OF OTHER TYPES

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

5.2 SOFT/HARD ERROR DEFINITIONS

5.2.1 HARD ERRORS

A. A 'DTE' (DRIVE TIMING ERROR) OR A 'DCK' (DATA CHECK ERROR) WHICH OCCURS DURING A READ DATA OR A READ HEADER & DATA OPERATION AND IS NOT CORRECTED OR BECOMES CORRECTABLE AFTER THE PROGRAM HAS PERFORMED THE COMPLETE RETRY SEQUENCE ON THE BAD SECTOR. THE RETRY SEQUENCE IS 15 RE-READS AT TRACK CENTER AND 2 ATTEMPTS AT EACH OF THE FOLLOWING OFFSETS:

+400 MICRO-INCHES
 -400 MICRO-INCHES
 +800 MICRO-INCHES
 -800 MICRO-INCHES
 +1200 MICRO-INCHES
 -1200 MICRO-INCHES

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.

769
770
771
772

D. 'DTE' ERRORS WHICH ARE CORRECTED OR WHICH BECOME ECC CORRECTABLE
'DCK' ERROR DURING THE RETRY SEQUENCE.

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

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

AFTER EACH WRITE OPERATION, THE DATA WRITTEN IS CHECKED WITH THE APPROPRIATE WRITE CHECK COMMAND - IF PARAMETER 'AUTOCK' IS 1. (IF 'AUTOCK' IS 0, WRITE CHECKS WILL BE PERFORMED RANDOMLY.)

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

6.4 BAD TRACK/SECTOR FLAGGING

SINCE THE RPD4 SUBSYSTEM DOES NOT HAVE AN AUTOMATIC BAD TRACK HANDLING CAPABILITY, THE MULTIDRIVE EXERCISER ALLOWS THE OPERATOR TO IDENTIFY UP TO 8 BAD TRACK/SECTOR LOCATIONS WHEN THE DRIVE IS ASSIGNED FOR TEST. (SEE SECTION 4.1 FOR ADDITIONAL INFORMATION.)

IF ONE OF THE FOLLOWING ERRORS OCCURS AT A LOCATION IDENTIFIED BY THE OPERATOR, 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' & 'HCRC', 'HCE' & 'HCRC', 'HCRC')

OPTIONALLY, THE OPERATOR MAY REQUEST AN ERROR REPORT FOR FLAGGED AREAS. (PARAMETER 'NOTPRT' MUST BE SET TO 0 AT STARTUP.) THE PROGRAM WILL IDENTIFY ERROR MESSAGES ASSOCIATED WITH THE PACK; THESE ERRORS WILL NOT BE ADDED TO THE ERROR TOTALS MAINTAINED BY THE PROGRAM.

0000
0000
0000
0000

7. ERROR MESSAGES

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

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 OCCURED OR A CENTRAL PROCESSOR FAILURE HAS OCCURED.

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 ILLEGAL RH70 INTERRUPT (SC=0 OR RHAS=0)

THE RH70 INTERRUPTED AND EITHER 'SC' FOR NOT SET OR RHAS WAS ZERO AND NO RH70 ERROR WAS INDICATED.

EM2 UNEXPECTED ATTENTION DETECTED

THE INDICATED DRIVE INTERRUPTED BUT THE DRIVE WAS NOT PERFORMING AN OPERATION.

EM3 CONTROL BUS PARITY ERROR DETECTED BY THE RH70

THE RH70 DETECTED A CONTROL BUS PARITY ERROR WHEN READING THE INDICATED REGISTER FROM THE INDICATED DRIVE.

EM4 CONTROL BUS PARITY ERROR DETECTED BY THE RPO4

THE INDICATED RPO4 DETECTED A CONTROL BUS PARITY ERROR WHEN THE RH70 LOADED THE SPECIFIED REGISTER.

EM5 ATTENTION FROM AN OFFLINE DRIVE

THE ATTENTION BIT WAS SET FOR A DRIVE WHICH WAS NOT AVAILABLE. THIS MESSAGE DOES NOT NECESSARILY INDICATE AN ERROR CONDITION. THIS MESSAGE WILL OCCUR IF DRIVES NOT BEING EXERCISED ARE CYCLED UP OR DOWN.

EM10 UNCORRECTABLE MASSBUS PARITY ERROR

THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE THE INDICATED

E03

MAINDEC-11-DERPN-8
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 31

889
890
891
892

REGISTER.

EM11 FATAL MASSBUS PARITY ERROR

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

A CONTROL BUS PARITY ERROR OCCURED WHEN THE RH70 ATTEMPTED TO PROCESS A PREVIOUS, DIFFERENT PARITY ERROR.

EM12 PERSISTENT DEVICE UNSAFE

THE DRIVE BECAME UNSAFE; DRIVE CLEAR TO THE DRIVE DID NOT CLEAR THE UNSAFE CONDITION. THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE DRIVE CANNOT BE EXERCISED UNTIL THE UNSAFE CONDITION HAS BEEN CLEARED.

EM13 OPERATION NOT COMPLETED WITHIN TIME LIMIT

THE DRIVE DID NOT COMPLETE THE OPERATION WITHIN 1 SECOND AFTER THE OPERATION WAS INITIATED.

EM14 UNIT WENT OFFLINE

THE DRIVE WENT OFFLINE DURING THE INDICATED OPERATION. (THE 'MOL' BIT BECAME ZERO.) THE PROGRAM WILL AUTOMATICALLY DEASSIGN THE DRIVE. THE OPERATOR MUST REASSIGN THE DRIVE WITH THE 'T' COMMAND TO RE-INITIATE TESTING.

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 OCCURED 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 OCCURED 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

G03

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 33

949
950
951
952

A WRITE DATA, WRITE CHECK DATA, OR A READ DATA WAS BEING
PERFORMED AND A 'FMT' ERROR OCCURED. THE PROGRAM RE-READ THE
HEADER OF THE ERROR SECTOR AND THE 'HCRC' BIT WAS SET. THE

953 CONTENTS OF THE HEADER ARE DISPLAYED. THE OPERATION WILL
954 BE RETRIED 3 TIMES.
955
956 EM25 HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR
957
958 SIMILAR TO EM24, EXCEPT THAT THE 'HCE' ERROR BIT WAS
959 SET INITIALLY. THE OPERATION WILL BE RETRIED 3 TIMES.
960
961 EM26 FORMAT ERROR ('FER')
962
963 FORMAT ERROR OCCURED. WHEN THE HEADER WAS RE-READ, THE
964 'HCRC' BIT WAS NOT SET. THE CONTENTS OF THE HEADER ARE
965 DISPLAYED. THE OPERATION WILL BE RETRIED 3 TIMES.
966
967 EM27 HEADER COMPARE ('HCE') ERROR
968
969 SIMILAR TO EM26 EXCEPT THAT THE 'HCE' BIT WAS SET INITIALLY.
970 THE OPERATION WILL BE RETRIED 3 TIMES.
971
972 EM30 MISCELLANEOUS DRIVE ERROR
973
974 THIS MESSAGE IS GIVEN FOR THE FOLLOWING ERROR BITS:
975 'IXE', 'AOE', 'RMR', 'ILF', OR 'ILR'
976
977 EM31 OPERATION INCOMPLETE ('OPI') ERROR
978
979 AN OPERATION INCOMPLETE ERROR OCCURED AT THE INDICATED
980 SECTOR.
981
982 EM32 DRIVE TIMING ('DTE') ERROR
983
984 DRIVE TIMING ERROR OCCURED ON THE INDICATED SECTOR. THE
985 OPERATION WILL BE RETRIED 3 TIMES.
986
987 EM33 PARITY ('PAR') ERROR AFTER OPERATION STARTED
988
989 THE 'PAR' BIT WAS SET WHEN THE OPERATION WAS COMPLETED. THE
990 OPERATION WILL BE RETRIED 3 TIMES.
991
992 EM34 WRITE CLOCK FAILURE ('WCF')
993
994 A WRITE CLOCK FAILURE OCCURED DURING THE OPERATION. THE
995 OPERATION WILL BE RETRIED 3 TIMES.
996
997 EM35 INVALID ADDRESS ('IAE') ERROR
998
999 AN INVALID ADDRESS ERROR OCCURED DURING THE OPERATION.
1000
1001 EM36 WRITE LOCK ('WLE') ERROR
1002
1003 A WRITE OPERATION WAS ATTEMPTED BUT THE DRIVE WAS WRITE
1004 LOCKED.
1005
1006 EM40 RH70 OR UNIBUS TRANSFER ERROR
1007
1008 'TRE' IS SET IN THE RH70 CONTROL REGISTER AND NO DRIVE

1009
1010
1011
1012

ERROR HAS OCCURED. THE OPERATION WILL BE RETRIED 3
TIMES IF THE ERROR WAS CAUSED BY 'DLT', 'UPE', 'MXF',
OR 'MDPE'.

1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068

EM41 BUS ADDRESS OR WORD COUNT INCORRECT
NO RH70/RP04 ERROR OCCURED 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 RP04 ERROR DETECTED
NO RP04 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 RH70
THE OPERATION COMPLETED NORMALLY; HOWEVER, THE PROGRAM FOUND EITHER ERROR BITS IN THE RP04 SET OR ERROR BITS IN THE RH70 SET.

EM50 SEEK INCOMPLETE OR OFF CYLINDER ERROR
THE DRIVE SIGNALLED EITHER 'SKI' OR 'OCYL' ERROR BITS.

EM51 PROGRAM DETECTED POSITIONING ERROR
A HEADER COMPARE ERROR OCCURED ('HCE'); HOWEVER, WHEN THE PROGRAM EXAMINED THE HEADER OF THE SECTOR IN ERROR, IT FOUND THAT THE CYLINDER FIELD DID NOT AGREE WITH THE CONTENTS OF 'RHCC' OF THE DRIVE. THE DRIVE WILL BE RECALIBRATED.

EM60 DEVICE UNSAFE
THE INDICATED DRIVE UNSAFE ERROR OCCURED; 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

TT:TT:TT (DESCRIPTION OF ERROR)

'TT:TT:YT' IS THE TIME SINCE THE PROGRAM WAS STARTED. TT:TT:TT IS GIVEN IN HOURS: MINUTES: SECONDS.

LINE 2

'PRESENT ORDER = XXXX PREVIOUS ORDER = YYYY'

K03

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 37

1069
1070
1071
1072

(DISPLAY OF THE RH70/RP04 REGISTERS IN TWO
GROUPS: RHCS1, RHCS2, RHDS1, RHER1, RHER2, RHER3
RHEC1, & RHEC2 FORM THE FIRST GROUP OF REGISTERS;

1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
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

ALL THE OTHER REGISTERS ARE IN THE SECOND GROUP.
IF SW<05> IS SET, ONLY THE REGISTERS IN THE FIRST
GROUP WILL BE DISPLAYED.)

MNEMONICS USED FOR THE DATA TRANSFER ORDERS ARE DEFINED BELOW:

- 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 7)
- RDHD - READ HEADER & DATA (OCTAL 73)

'ERROR OCCURED DURING NON-DATA TRANSFER OPERATION'

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

'* ERROR AT BAD TRACK/SECTOR'

THE ABOVE LINE WILL BE PRINTED IF A DATA ERROR OCCURES 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 RPO4 REGISTERS. THE
CONTENTS OF THIS WORD IDENTIFY HOW THE ERROR WAS PROCESSED BY THE
RPO4 DRIVE HANDLER ROUTINE. THE BITS IN THIS WORD ARE ENCODED
AS FOLLOWS:

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

M03

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 39

1129
1130
1131
1132

4
3

CORRECTABLE UNSAFE CONDITION OCCURED

DRIVE ERROR OCCURED THAT CAUSED AN AUTOMATIC

RECALIBRATE SEQUENCE

1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188

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

RHBA = XXXX RHWC = YYYY

THIS LINE GIVES THE CONTENTS OF THE RH70 BUFFER ADDRESS REGISTER AND THE RH70 WORD COUNT REGISTER. THIS LINE IS NOT PRINTED IF SW<05> IS NOT SET.

LINE 8

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

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

LINE 9

1199
1190
1191
1192

RHDA = XXXX RHCA = YYYY

1193
1194
1195
1196
1197
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

THIS LINE GIVES THE CONTENTS OF THE RPO4 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

RHEC1 = XXXX RHEC2 = YYYY

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

LINE 14

ECC CORRECTABLE WITHOUT OFFSET

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

LINE 15

READ CORRECTLY AT OFFSET X MICRO-INCHES

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

D04

MAINDEC-11-DERPN-S
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 43

1249
1250
1251
1252

LINE 16

ECC CORRECTABLE AT OFFSET X MICRO-INCHES

1253
1254
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

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 COUNT 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 OCCURED.
IF THIS LINE IS PRINTED, THE RH70/RP04 REGISTERS WILL ALSO BE
PRINTED (SEE LINE 2).

LINE 20

DATA COMPARISON ERRORS

A PRINTOUT OF THE DATA COMPARISON ERRORS FOLLOW THIS LINE.

LINE 21

TOTAL COMPARE ERRORS = 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

F04

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 45

1309
1310
1311
1312

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.

1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368LINE 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 'RHEC1' 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.

'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, OCYL 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.

H04

MAINDEC-11-DERPN-8
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 47

1369
1370
1371
1372

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

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
1426
1427
1428

'TOTAL POS ERR' IS THE TOTAL NUMBER OF POSITIONING ERRORS WHICH THE PROGRAM DETECTED FOR THE DRIVE.

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

8. PROGRAM DESCRIPTION

8.1 PROGRAM OPERATION

WHEN THE PROGRAM IS STARTED FROM LOCATION 200(8), ALL TABLES AND PARAMETERS ARE CLEARED OR INITIALIZED. THE PARAMETERS WHICH ARE UNDER OPERATOR TTY ENTRY CONTROL ARE CHECKED FOR VALIDITY AND CONSISTENCY. RH70 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 RPO4, AND IS ONLINE. THE ASSIGNMENT ROUTINE THEN ISSUES A 'READIN PRESET' INSTRUCTION, SETS 'FMT22', 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 (RHLA) OF THE INTERRUPTING DRIVE AND COMPARES THE POSITION OF THE DISK WITH THAT OF THE DESIRED SECTOR.

J04

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 49

1429
1430
1431
1432

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

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
1483
1484
1485
1486
1487
1488

INITIATED. IF A DRIVE IS NOT SELECTED FOR TRANSFER AFTER THREE REVOLUTIONS OF ITS DISK, IT IS GIVEN PRIORITY OVER DRIVES WHICH HAVE NOTE 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 RH70 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
DATA CHECK ERROR - EM21
WRITE CHECK WITH DCK SET - EM22
HEADER CRC ERRORS - EM20
FORMAT ERRORS - EM24, EM26
HEADER COMPARE ERRORS - EM25, EM27
PROGRAM DETECTED POSITIONING ERROR - EM51
SEEK INCOMPLETE OR OFF CYLINDER ERROR - EM50
WRITE CHECK WITHOUT 'DCK' SET - EM23
RH70 OR UNIBUS TRANSFER ERROR - EM40
'OPI' ERROR - EM31
'PAR' ERROR - EM33
'WCF' ERROR - EM34
'IAE' ERROR - EM35
'WLE' ERROR - EM36
MISCELLANEOUS DRIVE ERROR - EM30

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

BUS ADDRESS OR WORD COUNT INCORRECT - EM41
DATA COMPARE ERRORS - NO RPO4 ERROR DETECTED - EM42
CAN'T MATCH DATA READ WITH A PATTERN - EM43
ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH70 - EM44

L04

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 51

1489
1490
1491
1492

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

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
1540
1541
1542
1543
1544
1545
1546
1547
1548

WHEN THE DUAL PORT HANDLER ROUTINE IN THE MULTIDRIVE PROGRAM RECEIVES A REQUEST FOR A DRIVE, THE PROGRAM VERIFIES THAT THE DRIVE IS ONLINE. THE DRIVE IS SELECTED AND 0'S ARE WRITTEN INTO 'RHDS1': IF THE DRIVE IS IN NEUTRAL, THIS WILL SEIZE THE DRIVE. IF THE DRIVE IS SEIZED BY THE OTHER PORT, WRITING INTO 'RHDS1' WILL SET 'PORT REQUEST'. THE PROGRAM CHECKS 'DVA' IN 'RHCS1'. IF THE DRIVE IS AVAILABLE AS INDICATED BY THE 'DVA' BIT, THE COMMAND SEQUENCE WILL BE INITIATED IN THE NORMAL MANNER (SEE SECTION 8.1 ABOVE). IF 'DVA' WAS NOT SET, THE PROGRAM MAKES AN ENTRY FOR THE DRIVE IN AN INTERNAL 'PORT REQUEST PENDING' TABLE AND STARTS A 10 SECOND TIMER FOR THE DRIVE. IF THE DRIVE HAS NOT SWITCHED TO THE REQUESTING SYSTEM WITHIN THE 10 SECOND INTERVAL, THE PROGRAM REPORTS A 'NO RESPONSE TO PORT REQUEST' ERROR. NORMALLY THIS ERROR MESSAGE INDICATES A FAILURE IN THE DUAL PORT CONTROL LOGIC IN THE RPO4 BEING TESTED; HOWEVER, UNDER CERTAIN CONDITIONS (E.G. MASSBUS PARITY ERRORS BEING REPORTED ON THE OTHER SYSTEM ON A MOD33 TTY), THE OTHER PROCESSOR WAS UNABLE TO PROCESS THE DRIVE AFTER IT HAD REQUESTED THE DRIVE. THE OPERATOR MUST BE AWARE OF WHAT THE OTHER SYSTEM IS DOING AT ALL TIMES TO INTERPRET THE PORT RELATED ERROR MESSAGES PROPERLY.

AFTER THE RPO4 HAS COMPLETED AN OPERATION, THE PROGRAM WILL STORE THE REGISTERS AND ISSUE A 'RELEASE' TO THE DRIVE; IF THE OPERATION TERMINATED WITH AN ERROR, THE DRIVE WILL NOT BE RELEASED UNTIL ERROR PROCESSING HAS BEEN COMPLETED. NOTE THAT THIS CAN BE AN EXTENDED INTERVAL IN THE CASE OF A DRIVE UNSAFE WHICH CAUSES THE DRIVE TO CYCLE DOWN BUT WHICH IS CLEARED BY THE 'DRIVE CLEAR'.

SINGLE PORT DRIVES, DRIVES WHICH ARE LOCKED ON PORT, OR 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'.
- B. THE BUFFER SIZE IS RANDOM SELECTED BETWEEN 4 (10) - AND THE VALUE IN 'MAXDL'. THE SIZE SELECTED IS WEIGHTED TO ENSURE THAT AT LEAST 4 WORDS ARE WRITTEN IN THE DATA AREA OF THE LAST SECTOR. 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 KEYWORDS IN THE HEADER (WHEN PERFORMING A WRITE HEADER & DATA ORDER) ARE ZERO FILLED. THE PROGRAM EXPECTS TO FIND THAT THE KEYWORDS ARE ZERO.
- D. THE ORDERS ARE SELECTED RANDOMLY. WRITE CHECK DATA AND WRITE

N04

MAINDEC-11-DERPN-B
DERPNB.P11

MACY11 27(732) 27-SEP-76 15:05 PAGE 53

1549
1550
1551
1552

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

WA

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
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608

A WRITE HEADER & DATA ORDER, THE BUFFER SIZE IS FORCED TO 260 (10); THE PROGRAM WILL NOT PERFORM A MULTI-SECTOR FORMAT WRITE OPERATION.

- E. THE FIRST ORDER PERFORMED AFTER A UNIT IS ASSIGNED WITH A 'T' 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. TO MAINTAIN COMPATIBILITY WITH PACKS WRITTEN BY THE FORMAT PROGRAM (MAINDEC-11-DERPL), THE PROGRAM WILL ACCEPT ALL ZERO'S AND ALL ONE'S PATTERNS; HOWEVER, ALL ZERO'S AND ALL ONE'S PATTERNS ARE NOT WRITTEN BY THE EXERCISER PROGRAM.

PATTERN '8' IS DEFINED AS THE 'WORST CASE' PATTERN.

PAT 1	PAT 2	PAT 3	PAT 4	PAT 5	PAT 6	PAT 7	PAT 8
000001	177776	000000	000000	052525	007417	026455	165555
000003	177774	000000	010421	052525	007417	026455	133333
000007	177770	000000	021042	052525	007417	026455	165555
000017	177760	177777	031463	125252	170360	151322	133333
000037	177740	177777	042104	125252	170360	151322	165555
000077	177700	177777	052525	125252	170360	151322	133333
000177	177600	000000	063146	052525	007417	026455	165555
000377	177400	000000	073567	052525	007417	026455	133333
000777	177000	177777	104210	125252	170360	151322	165555
001777	176000	177777	114631	125252	170360	151322	133333
003777	174000	000000	125252	052525	007417	026455	165555
007777	170000	177777	135673	125252	170360	151322	133333
017777	160000	000000	146314	052525	007417	026455	165555
037777	140000	177777	156735	125252	170360	151322	133333
077777	100000	000000	167356	052525	007417	026455	165555
177777	000000	177777	177777	125252	170360	151322	133333
PAT 9	PAT 10	PAT 11	PAT 12	PAT 13	PAT 14	PAT 15	
000001	177776	172666	077777	153333	000000	177777	
000002	177775	155555	137777	066667	177777	000000	
000004	177773	172666	157777	153333	177777	000000	
000010	177767	155555	167777	066667	177777	000000	
000020	177757	172666	173777	153333	177777	000000	
000040	177737	155555	175777	066667	177777	000000	
000100	177677	172666	176777	153333	177777	000000	
000200	177577	155555	177377	066667	177777	000000	
000400	177377	172666	177577	153333	177777	000000	
001000	176777	155555	177677	066667	177777	000000	
002000	175777	172666	177737	153333	177777	000000	
004000	173777	155555	177757	066667	177777	000000	

1609
1610
1611
1612

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

1613
1614
1615
1616
1617
1618
1619
1620
1621

9. PROGRAM LISTING

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
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667

DOCUMENT

MAINDEC-11-DERPN-B

COPYRIGHT 1975
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

TABLE OF CONTENTS

1668	
1669	
1670	
1671	
1672	
1673	
1674	
1675	
1676	
1677	
1678	
1679	
1680	
1681	
1682	
1683	
1684	
1685	
1686	
1687	
1688	
1689	
1690	
1691	
1692	
1693	
1694	
1695	
1696	
1697	
1698	
1699	
1700	
1701	
1702	
1703	
1704	
1705	
1706	
1707	
1708	
1709	
1710	
1711	
1712	
1713	
1714	
1715	
1716	
13	OPERATIONAL SWITCH SETTINGS
34	BASIC DEFINITIONS
141	RPO4 DRIVER COMMANDS
167	TRAP CATCHER
174	STARTING ADDRESS(ES)
182	COMMON TAGS
246	CONTROL PARAMETERS
291	RPO4 ADDRESS LIMIT VALUES
304	VALUES FOR FIRST OPERATION
326	ERROR POINTER TABLE
393	SETUP AND INITIALIZATION ROUTINE
642	MAIN PROGRAM
2427	ERROR MESSAGE GENERATION ROUTINES
2791	GENERAL SUPPORT SUBROUTINES
3883	TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
3908	TYPE ROUTINE
3956	PRINT ROUTINE
4004	TTY INPUT ROUTINE
4136	MACRO ROUTINES
4140	ERROR HANDLER ROUTINE
4175	ERROR MESSAGE TYPEOUT ROUTINE
4232	BINARY TO OCTAL (ASCII) AND TYPE

1717
1716
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

TABLE OF CONTENTS

4310	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4378	SAVE AND RESTORE R0-R5 ROUTINES
4424	RANDOM NUMBER GENERATOR ROUTINE
4471	ROUTINE TO SIZE MEMORY
4499	DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4562	SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
4580	DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
4620	SINGLE LENGTH BINARY TO OCTAL ASCIZ ROUTINE
4637	TRAP DECODER
4656	TRAP TABLE
4678	RH11/RP04 DRIVER - SINGLE/DUAL PORT VERSION
6095	DATA, CONTROL, & STATUS BLOCKS
6828	TABLES, CONSTANTS, AND VARIABLE LOCATIONS
6996	DATA PATTERNS
7293	PRINTER/TELETYPE MESSAGES
8147	PATCH AREA

1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
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

3 COPYRIGHT (C) 1974,1975
DIGITAL EQUIPMENT CORP.
MAYNARD, MASS. 01754

PROGRAM BY C. HESS

THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
PACKAGE (MAINDEC-11-DZQAC-A2).

13

OPERATIONAL SWITCH SETTINGS

14

SWITCH

USE

15
13
10
7
6
5
5
4
4
3
3
3
3
1
0

HALT ON ERROR
INHIBIT ERROR TYPEOUTS
BELL ON ERROR
DISPLAY ALL DATA COMPARE ERRORS
DON'T CHANGE PARAMETERS (LOOP ON PRESENT
PARTIAL REGISTER DISPLAY IF ERROR
NO ECC CORRECTION RESULTS DISPLAYED IF E
DO NOT CHECK FOR MAXIMUM ERROR COUNTS
DO NOT DROP UNIT WHEN NORMAL END OF TE
DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR
DISPLAY SECTOR IF 'DCK' ERR, UNCORRECTA
28TH RETRY
IF DATA COMPARE ERROR & SW7 SET, DISPL
REMAINDER OF BUFFER
INHIBIT DATA COMPARSION AFTER READ ORDER
READ ONLY MODE

34

BASIC DEFINITIONS

36 INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
47 GENERAL PURPOSE REGISTER DEFINITIONS
59 PRIORITY LEVEL DEFINITIONS
69 "SWITCH REGISTER" SWITCH DEFINITIONS
97 DATA BIT DEFINITIONS (BIT00 TO BIT15)
125 BASIC "CPU" TRAP VECTOR ADDRESSES

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

139 *****

141 *****
RPO4 DRIVER COMMANDS

143 *****

167 *****
TRAP CATCHER

170 ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

174 *****
STARTING ADDRESS(ES)

180 *****

182 *****
COMMON TAGS

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

244 *****

246 *****
CONTROL PARAMETERS

248 *****

289 *****

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
1882
1883
1884
1885
1886

291 *****
RPO4 ADDRESS LIMIT VALUES

293 *****

302 *****

304 *****
VALUES FOR FIRST OPERATION

306 *****

324 *****

326 *****
ERROR POINTER TABLE

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

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

391 *****

393 *****
SETUP AND INITIALIZATION ROUTINE

398 *****

640 *****

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
1939
1940
1941
1942

```

642 *****
MAIN PROGRAM
*****

644 *****

2425 *****

2427 *****
ERROR MESSAGE GENERATION ROUTINES
*****

2429 *****

2789 *****

2791 *****
GENERAL SUPPORT SUBROUTINES
*****

2793 *****

3881 *****

3883 *****
TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
*****

3885 CALL MOV #NUMADR, -(SP) ;FIRST ADDRESS OF ASCIZ
JSR PC, @#SSUPRS

3906 *****

3908 *****
TYPE ROUTINE
*****

3910 ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 B
THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE
NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CH
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
OR
TYPE
MESADR

2) USING A JSR INSTRUCTION
MOV PS, -(SP) ;PUSH PROCESSOR STATUS W
JSR PC, $TYPE ;CALL TYPE ROUTINE

```


1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995

```

MESADDR ;FIRST ADDRESS OF MESSAGE
3954 *****
*****
3956 PRINT ROUTINE
*****
3958 ROUTINE TO PRINT ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0
CALL:
1) USING A TRAP INSTRUCTION
DISPLY ,MESADR ;MESADR IS FIRST ADDRESS OF AN A
OR
PRINT
MESADR
2) USING A JSR INSTRUCTION
MOV PS,-(SP) ;PUSH PROCESSOR STATUS W
JSR PC,$PRINT ;CALL PRINT ROUTINE
MESADDR ;FIRST ADDRESS OF MESSAGE
4002 *****
*****
4004 TTY INPUT ROUTINE
*****
4006 TK INITIALIZE ROUTINE
THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
CALL JSR PC,$TKINT
RETURN
4022 TK SERVICE ROUTINE
THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
4057 *****
THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
CALL:
RDCHR ;INPUT A SINGLE CHARACTER FROM T
RETURN HERE ;CHARACTER IS ON THE STACK
4072 *****
THIS ROUTINE WILL INPUT A STRING FROM THE TTY
CALL:
RDLIN ;INPUT A STRING FROM THE TTY
RETURN HERE ;ADDRESS OF FIRST CHARACTER WILL
;TERMINATOR WILL BE A BYTE OF AL

```

1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051

```

4134 *****
*****
4136 MACRO ROUTINES
*****
4138 *****
*****
4140 ERROR HANDLER ROUTINE
*****
4142 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 SERRTYP ON ERROR
      THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
      SW15=1 HALT ON ERROR
      SW13=1 INHIBIT ERROR TYPEOUTS
      SW10=1 BELL ON ERROR
      CALL          ERROR  N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4173 *****
*****
4175 ERROR MESSAGE TYPEOUT ROUTINE
*****
4177 THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
      ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE"
      AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4230 *****
*****
4232 BINARY TO OCTAL (ASCII) AND TYPE
*****
4234 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIG
      OCTAL (ASCII) NUMBER AND TYPE IT.
      $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS
      CALL:
              MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
              TYPOS          ;;CALL FOR TYPEOUT
              .BYTE  N          ;;N=1 TO 6 FOR NUMBER OF DIGITS
              .BYTE  M          ;;M=1 OR 0
                              ;;1=TYPE LEADING ZEROS
                              ;;0=SUPPRESS LEADING ZER
      $TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE
      $TYPOS OR $TYPOC
      CALL:
              MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
              TYPON          ;;CALL FOR TYPEOUT

```

2052
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
2098
2099

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

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

4308 *****

4310 *****
 CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

4312 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIG
 SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER
 NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE T
 BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS
 REPLACED WITH SPACES.

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

4376 *****

4378 *****
 SAVE AND RESTORE R0-R5 ROUTINES

4380 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

4407 RESTORE R0-R5
CALL: RESREG

4422 *****

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
172
173
174
175
176
177
178
179
180

```

*****
4562 SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
*****

4564 THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
      UNSIGNED DECIMAL ASCIZ NUMBER.
      CALL
            MOV     NUMBER, -(SP)      ;; PUT BINARY NUMBER ON THE STACK
            JSR     PC, J#$$B20      ;; CALL
            RETURN                      ;; ADDRESS OF THE 1ST ASCIZ CHAR.

4578 *****

4580 *****
      DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
      *****

4582 THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
      UNSIGNED OCTAL ASCIZ NUMBER.
      CALL
            MOV     #PNTR, -(SP)      ;; POINTER TO LOW WORD OF BINARY
            JSR     PC, J#$$B20      ;; CALL THE ROUTINE
            RETURN                      ;; THE ADDRESS OF THE FIRST ASCIZ

4618 *****

4620 *****
      SINGLE LENGTH BINARY TO OCTAL ASCIZ ROUTINE
      *****

4622 THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
      UNSIGNED OCTAL ASCIZ NUMBER.
      CALL
            MOV     NUMBER, -(SP)      ;; PUT BINARY NUMBER ON THE STACK
            JSR     PC, J#$$B20      ;; CALL
            RETURN                      ;; ADDRESS OF 1ST ASCIZ CHAR. IS

4635 *****

```

2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230

```
*****  
4637 TRAP DECODER  
*****  
4639 THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTIO  
AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDR  
OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
GO TO THAT ROUTINE.  
  
*****  
4656 TRAP TABLE  
*****  
4658 THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLE  
BY THE "TRAP" INSTRUCTION.  
  
4676 *****  
  
*****  
4678 RH11/RP04 DRIVER - SINGLE/DUAL PORT VERSION  
*****  
  
4681 COPYRIGHT (C) 1974  
DIGITAL EQUIPMENT CORP.  
MAYNARD, MA 01754  
AUTHOR: JIM LACEY/CHUCK HESS  
  
4686 *****  
  
4688 STORAGE FOR RHDS1, RHER1, RHER2, AND RHER3 ON AN ERROR "2" OR "5  
RPERRS = RHDS1  
RPERRS+2 = RHER1  
RPERRS+4 = RHER2  
RPERRS+6 = RHER3  
  
4699 TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)  
DRVACT=0 IMPLIES DRIVE IS IDLE  
DRVACT>0 IMPLIES DRIVE IS ACTIVE WITH A COMMAND  
DRVACT<0 IMPLIES DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATIO  
  
4713 TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)  
DRVSTA=0 IMPLIES DRIVE IS OFFLINE  
DRVSTA>0 IMPLIES DRIVE IS ONLINE  
DRVSTA<0 IMPLIES DRIVE IS UNSAFE OR NONEXISTENT  
  
4727 TABLE OF DRIVE TYPES (DRV TYP=8 WORDS)  
DRV TYP WILL CONTAIN THE DRIVE TYPE OF ALL ONLINE, OFFLINE, AND  
UNSAFE DRIVES. IF A DRIVE IS NONEXISTENT DRV TYP WILL BE ZERO.
```

2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280

- 4740 TABLE OF DUAL PORT INITIALIZATION INDICATORS
DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
DPINT<0 IF INITIALIZATION IS IN PROGRESS
- 4753 TABLE OF PENDING DUAL PORT REQUESTS
DPRQS=0 IMPLIES THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT
DPRQS<0 IMPLIES THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRI
- 4766 TRANSFER WAIT FLAG (TRNSWT=1 WORD)
THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
"DPB" OF THE I/O OPERATION.
- 4772 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 C.
- 4780 RPO4 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
ACTDRV=0 IMPLIES DRIVER IS INACTIVE
ACTDRV>0 IMPLIES DRIVER IS ACTIVE
- 4786 SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
ACTSTR=0 IMPLIES SOFTWARE TIMER ROUTINE IS INACTIVE
ACTSTR>0 IMPLIES SOFTWARE TIMER ROUTINE IS ACTIVE
- 4792 UNLOAD FLAG (ULDFLG=8 BYTES)
ULDFLG=0 IMPLIES NO UNLOAD COMMAND
ULDFLG>0 IMPLIES UNLOAD COMMAND IN PROGRESS
ULDFLG<0 IMPLIES UNLOAD COMMAND IN WAIT QUEUE
- 4806 LOOK AHEAD COUNT (LACNT=8 BYTES)
LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
- 4818 SAVE REGISTERS FLAG (SAVEFG =1 WORD)
SAVEFG <0 IMPLIES SAVE THE RH11/RPO4 REGISTERS WHEN THE
OPERATION IS COMPLETED AS PER (DPB+14).
SAVEFG=0 IMPLIES SAVE THE RH11/RPO4 REGISTERS, AS PER
(DPB+14), AFTER AN ERROR.
- 4826 SEEK FLAG (SEEKFG=1 WORD)
SEEKFG=0 IMPLIES WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
FOR A DATA TRANSFER START A SEARCH COMMAND
SEEKFG<0 IMPLIES DATA TRANSFER WILL DO IMPLIED SEEKS,
DISREGARD THE WINDOW
- 4834 TIMEOUT TABLE (TIMER=8 WORDS)
THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION

2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331

4846 DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
DTUW<0 IMPLIES NO DATA TRANSFER UNDERWAY
DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE

4852 ATTENTION BITS TABLE (ATABIT=8 BYTES)
THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
ATTENTION BIT

4865 RPO4 TO RH11 "MASS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEF
CALLING IT FATAL (MCPEMX=1 WORD)

4870 STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4),
RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5))

4875 MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

4877 MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

4879 MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

4881 MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

4884 DEFINITIONS OF THE RH11/RPO4 ADDRESS INDEXES

4907 RH11/RPO4 DRIVER INIT. CODE
THIS ROUTINE WILL DETERMINE WHICH RPO4 DRIVES ARE
AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
TO THE PROPER STATE FOR EACH DRIVE.
NOTE: THIS ROUTINE CALLS DRVINT
CALL

4915 JSR PC,RPINIT
RETURN

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

4963 DRIVE INIT. ROUTINE
THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
AN RPO4. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,

4968 DRVSTA IS SET TO THE PROPER CONDITION.
CALL

MOV	#DRVNUM,R1	;DRIVE NUMBER TO R1
MOV	RPADR,R4	;UNIBUS ADDRESS OF RH11/RPO4 (RH
JSR	RO,DRVINT	;CALLED BY A JSR
RETURN1		;ERROR OCCURRED (PARITY)
RETURN2		;NORMAL RETURN


```

2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384

```

```

5032 REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
CALL
      JSR      RO,2#RPO4      ;CALL THE RPO4 DRIVER
      PNTADR   ;ADDRESS OF POINTER OF DRIVES PA
      RETURN1  ;RETURN HERE IF QUEUE IS FULL
      RETURN2  ;RETURN HERE IF REQUEST IS IN QU

5096 OPTIMIZER-CALLED FOR A PARTICULAR DRIV
CALL
      MOV      #DRVNUM,R1    ;DRIVE NUMBER TO R1
      JSR      PC,OPT        ;SETUP A COMMAND

5143 COMMAND INITIATOR
CALL
      MOV      #DRVNUM,R1    ;DRIVE NUMBER
      MOV      #DPB,R2      ;ADDRESS OF DPB
      JSR      PC,C1?       ;C1?= C11,C13, OR C14
                          ;WHERE:
                          ;C11=DATA TRANSFER
                          ;C12=SEARCH REQUESTED BY DATA XF
                          ;C14=NOT DATA TRANSFER

5330 LOOK AHEAD ROUTINE
CALL
      MOV      #DRVNUM,R1    ;DRIVE NUMBER
      MOV      #DPB,R2      ;POINT TO DPB
      JSR      RO,LA        ;GO CHECK THE WINDOW
      RETURN1 ;ERROR RETURN
      RETURN2 ;START A SEARCH
      RETURN3 ;START A DATA TRANSFER

5373 INTERRUPT SERVICE ROUTINE
5387 TRANSFER DONE ROUTINE
5421 FORCED WRITE CHECK ROUTINE
5440 SPECIAL CONDITION ROUTINE
5643 RPO4 TIMER ROUTINE
CALL
      MOV      #TIME, -(SP)  ;ELASPED TIME IN MILLISECONDS ON
      JSR      RO,RPTMR     ;CALL RPO4 TIME ROUTINE

5669 SOFTWARE TIMEOUT ROUTINE
5670 CALL:
      STO
      MOV      #DRVNUM,R1    ;DRIVE NUMBER
      JSR      RO,STO       ;CALL--DRVACT MUST BE NONZERO
      RETURN

```

2385
 2386
 2387
 2388
 2389
 2390
 2391
 2392
 2393
 2394
 2395
 2396
 2397
 2398
 2399
 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

```

5764 ROUTINE TO READ A RH11/RP04 REGISTER
CALL
      JSR    RD, RD.RP      ;GO READ A REGISTER
      INDEX  ;REG. INDEX FROM BASE
      ERRADR ;ERROR ADDRESS--PROCESS ERROR ST
              ;AT THIS ADDRESS
      RETURN ;CONTENTS OF REG. IS ON THE STAC

5906 ROUTINE TO WRITE A RH11/RP04 REGISTER
CALL
      MOV    DATA, -(SP)  ;DATA TO BE LOADED ON THE STACK
      JSR    RD, WRT.RP    ;CALL THE ROUTINE TO LOAD(WRITE)
      INDEX  ;INDEX OF THE REGISTER TO BE LOA
      ERRADR ;ADDRESS TO RETURN TO ON AN ERRO
      RETURN ;ERROR FREE RETURN

5850 ROUTINE TO SAVE THE RH11/RP04 REGISTERS AS PER DPB+14
CALL
      MOV    #DPBNUM, R2   ;DPB POINTER TO R2
      JSR    PC, SVRH11    ;SAVE THE DRIVES REG'S

5880 ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
CALL
      MOV    #DRVNUM, R1   ;DRIVE NUMBER TO R1
      JSR    PC, SET.IE    ;SET "IE"
      RETURN

5901 QUEUE COUNT

5955 ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
CALL
      JSR    PC, CLRQUE

5979 EMPTY THE QUEUE SPECIFIED BY R1
CALL
      MOV    DRVNUM, R1    ;DRIVE NUMBER TO R1
      JSR    PC, EMPTYQ

5991 ROUTINE TO PUT A REQUEST IN QUEUE
CALL

5994      MOV    #DRVNUM, R1 ;DRIVE NUMBER
      MOV    #DPB, R2      ;ADDRESS OF PARAMETER BLOCK
      JSR    RD, DRVQUE    ;GO PUT REQUEST IN QUEUE
      RETURN1 ;RETURN HERE IF QUEUE IS FULL
      RETURN2 ;RETURN HERE IF REQUEST IS IN QU

6013 ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
CALL
      MOV    #DRVNUM, R1   ;DRIVE NUMBER TO R1
      JSR    PC, GETREQ    ;GO GET THE REQUEST
      RETURN ;R2="DPB" ADDRESS OF THE REQUEST
              ;R2=0 IF NO REQUEST IN QUEUE
  
```

2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486

6029 ROUTINE TO "POP" THE REQUEST FROM QUEUE
CALL
MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
JSR PC,POPQUE ;CALL TO REMOVE REQUEST
RETURN ;R2=ADDRESS OF DPB REMOVED

6046 ROUTINES TO SAVE R0-R5 AND R1-R5

6048 CALL: SAVR05
JSR R0,SAVR05
RETURN ;R0-R5 IS ON THE STACK

CALL: SAVR15
JSR R0,SAVR15
RETURN ;R1-R5 IS ON THE STACK

UPON RETURN FROM SAVR05 AND SAVR15 THE STACK WILL LOOK LIKE:

+12 R0
+10 R1
+06 R2
+04 R3
+02 R4
TOP R5

6074 ROUTINES TO RESTORE R0-R5 AND R1-R5

CALL: GETR05
JSR R0,GETR05
RETURN ;R0-R5 HAVE BEEN RESTORED

CALL: GETR15
JSR R0,GETR15
RETURN ;R1-R5 HAVE BEEN RESTORED

6093 *****

6095 *****
DATA, CONTROL, & STATUS BLOCKS

6097 *****

6826 *****

2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518

```

6828 *****
      TABLES, CONSTANTS, AND VARIABLE LOCATIONS
      *****

6830 *****

6994 *****

6996 *****
      DATA PATTERNS
      *****

6998 *****

7291 *****

7293 *****
      PRINTER/TELETYPE MESSAGES
      *****

7295 *****

8145 *****

8147 *****
      PATCH AREA
      *****

8149 *****

8153 *****

```

Handwritten marks or signature in the bottom right corner.

2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574

```

!
.TITLE MAINDEC-11-DERPN-B
:*COPYRIGHT (C) 1974,1975
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
*
:*PROGRAM BY C. HESS
*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
*
.SBTTL OPERATIONAL SWITCH SETTINGS
*
*      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           PARTIAL REGISTER DISPLAY IF ERROR
*      5           NO ECC CORRECTION RESULTS DISPLAYED IF ERROR
*      4           DO NOT CHECK FOR MAXIMUM ERROR COUNTS
*      4           DO NOT DROP UNIT WHEN NORMAL END OF TEST REACHED
*      3           DISPLAY ERROR SECTOR IF 'DCK', 'DTE', OR 'WCF' ERROR
*      3           DISPLAY SECTOR IF 'DCK' ERR UNCORRECTABLE AFTER
*      3           28TH RETRY
*      3           IF DATA COMPARE ERROR & SW7 SET, DISPLAY
*      3           REMAINDER OF BUFFER
*      1           INHIBIT DATA COMPARISON AFTER READ ORDERS
*      0           READ ONLY MODE
*
.SBTTL BASIC DEFINITIONS
*
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
*
:*MISCELLANEOUS DEFINITIONS
HT= 11                ;;CODE FOR HORIZONTAL TAB
LF= 12                ;;CODE FOR LINE FEED
CR= 15                ;;CODE FOR CARRIAGE RETURN
CRLF= 200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776           ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774       ;;STACK LIMIT REGISTER
PIRQ= 177772         ;;PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570         ;;HARDWARE SWITCH REGISTER
DDISP= 177570        ;;HARDWARE DISPLAY REGISTER
*
:*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0               ;;GENERAL REGISTER
R1= %1               ;;GENERAL REGISTER
R2= %2               ;;GENERAL REGISTER
R3= %3               ;;GENERAL REGISTER

```

001100

000011
000012
000015
000200
177776

177774
177772
177570
177570

000000
000001
000002
000003

2575	000004	R4=	%4	:: GENERAL REGISTER
2576	000005	R5=	%5	:: GENERAL REGISTER
2577	000006	R6=	%6	:: GENERAL REGISTER
2578	000007	R7=	%7	:: GENERAL REGISTER
2579	000006	SP=	%6	:: STACK POINTER
2580	000007	PC=	%7	:: PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

2583	000000	PR0=	0	:: PRIORITY LEVEL 0
2584	000040	PR1=	40	:: PRIORITY LEVEL 1
2585	000100	PR2=	100	:: PRIORITY LEVEL 2
2586	000140	PR3=	140	:: PRIORITY LEVEL 3
2587	000200	PR4=	200	:: PRIORITY LEVEL 4
2588	000240	PR5=	240	:: PRIORITY LEVEL 5
2589	000300	PR6=	300	:: PRIORITY LEVEL 6
2590	000340	PR7=	340	:: PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

2592	100000	SW15=	100000	
2593	040000	SW14=	40000	
2594	020000	SW13=	20000	
2595	010000	SW12=	10000	
2596	004000	SW11=	4000	
2597	002000	SW10=	2000	
2598	001000	SW09=	1000	
2599	000400	SW08=	400	
2600	000200	SW07=	200	
2601	000100	SW06=	100	
2602	000040	SW05=	40	
2603	000020	SW04=	20	
2604	000010	SW03=	10	
2605	000004	SW02=	4	
2606	000002	SW01=	2	
2607	000001	SW00=	1	
2608		.EQUIV	SW09, SW9	
2609		.EQUIV	SW08, SW8	
2610		.EQUIV	SW07, SW7	
2611		.EQUIV	SW06, SW6	
2612		.EQUIV	SW05, SW5	
2613		.EQUIV	SW04, SW4	
2614		.EQUIV	SW03, SW3	
2615		.EQUIV	SW02, SW2	
2616		.EQUIV	SW01, SW1	
2617		.EQUIV	SW00, SW0	

.*DATA BIT DEFINITIONS (BIT00 TO BIT15)

2620	100000	BIT15=	100000
2621	040000	BIT14=	40000
2622	020000	BIT13=	20000
2623	010000	BIT12=	10000
2624	004000	BIT11=	4000
2625	002000	BIT10=	2000
2626	001000	BIT09=	1000
2627	000400	BIT08=	400
2628	000200	BIT07=	200
2629	000100	BIT06=	100

```

2631      000040      BIT05= 40
2632      000020      BIT04= 20
2633      000010      BIT03= 10
2634      000004      BIT02= 4
2635      000002      BIT01= 2
2636      000001      BIT00= 1
2637      .EQUIV BIT09,BIT9
2638      .EQUIV BIT08,BIT8
2639      .EQUIV BIT07,BIT7
2640      .EQUIV BIT06,BIT6
2641      .EQUIV BIT05,BIT5
2642      .EQUIV BIT04,BIT4
2643      .EQUIV BIT03,BIT3
2644      .EQUIV BIT02,BIT2
2645      .EQUIV BIT01,BIT1
2646      .EQUIV BIT00,BIT0
2647
2648      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
2649      000004      ERRVEC= 4      ;; TIME OUT AND OTHER ERRORS
2650      000010      RESVEC= 10     ;; RESERVED AND ILLEGAL INSTRUCTIONS
2651      000014      TBITVEC=14     ;; "T" BIT
2652      000014      TRTVEC= 14     ;; TRACE TRAP
2653      000014      BPTVEC= 14     ;; BREAKPOINT TRAP (BPT)
2654      000020      IOTVEC= 20     ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
2655      000024      PWRVEC= 24     ;; POWER FAIL
2656      000030      EMTVEC= 30     ;; EMULATOR TRAP (EMT) **ERROR**
2657      000034      TRAPVEC=34     ;; "TRAP" TRAP
2658      000060      TKVEC= 60      ;; TTY KEYBOARD VECTOR
2659      000064      TPVEC= 64      ;; TTY PRINTER VECTOR
2660      000240      PIRGVEC=240    ;; PROGRAM INTERRUPT REQUEST VECTOR
2661
2662      ;;*****
2663
2664      .SBTTL RPO4 DRIVER COMMANDS
2665
2666      ;;*****
2667
2668      000101      RNOP= 101        ;NO OPERATION
2669      000103      UNLOAD= 103     ;UNLOAD
2670      000105      SEEK= 105      ;SEEK
2671      000107      RECAL= 107     ;RECALIBRATE
2672      000111      DRVCLR= 111    ;DRIVE CLEAR
2673      000113      REL= 113       ;RELEASE
2674      000115      OFFSET= 115    ;OFFSET
2675      000117      RTC= 117       ;RETURN TO CENTER LINE
2676      000121      PRESET= 121    ;READ IN PRESET
2677      000123      ACK= 123       ;PACK ACKNOWLEDGE
2678      000131      SEARCH= 131    ;SEARCH
2679      000141      GETREG= 141    ;GET REGISTERS
2680      000143      SETFMT= 143    ;SET FORMAT (& ECI OR HCI)
2681      000145      SELDRV= 145    ;SELECT DRIVE
2682      000151      WCKD= 151      ;WRITE CHECK DATA
2683      000153      WCKHD= 153     ;WRITE CHECK HEADER & DATA
2684      000161      WRTDAT= 161    ;WRITE DATA
2685      000163      WRTHD= 163     ;WRITE HEADER & DATA
2686      000171      RDDAT= 171     ;READ DATA

```


2702
2703
2704
2705
2706
2707
2708 001100
2709 001100
2710 001100 000000
2711 001102 000
2712 001103 000
2713 001104 000000
2714 001106 000000
2715 001110 000000
2716 001112 000000
2717 001114 000
2718 001115 001
2719 001116 000000
2720 001120 000000
2721 001122 000000
2722 001124 000000
2723 001126 000000
2724 001130 000000
2725 001132 000000
2726 001134 000
2727 001135 000
2728 001136 000000
2729 001140 177570
2730 001142 177570
2731 001144 177560
2732 001146 177562
2733 001150 177564
2734 001152 177566
2735 001154 000
2736 001155 002
2737 001156 012
2738 001157 000
2739 001160 177607 000377
2740 001164 077
2741 001165 015
2742 001166 000012
2743
2744 001170 176700
2745 001172 000254
2746 001174 172540
2747 001176 172542
2748 001200 000104
2749 001202 177546
2750 001204 000100
2751 001206 177777
2752 001210 177777
2753 001212 177777
2754 001214 177514
2755 001216 177516
2756 001220 000000
2757 001222 000074

.SBTTL COMMON TAGS

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

 .=1100
\$CMTAG: .WORD 0
\$PASS: .WORD 0
\$STNM: .BYTE 00
\$ERFLG: .BYTE 00
\$ICNT: .WORD 00
\$LPADR: .WORD 00
\$LPERR: .WORD 00
\$ERTTL: .WORD 00
\$ITEMB: .BYTE 0
\$ERMAX: .BYTE 1
\$ERRPC: .WORD 00
\$GDADR: .WORD 00
\$BDADR: .WORD 00
\$GDDAT: .WORD 00
\$BDDAT: .WORD 00
 \$AUTOB: .BYTE 0
 \$INTAG: .BYTE 0
 \$SWR: .WORD DSWR
 \$DISPLAY: .WORD DDISP
\$TKS: 177560
\$TKB: 177562
\$TPS: 177564
\$TPB: 177566
\$NULL: .BYTE 0
\$FILLS: .BYTE 2
\$FILLC: .BYTE 12
\$STPFLG: .BYTE 0
\$BELL: .ASCIZ <207><377><377>
\$QUES: .ASCII /?/
\$CRLF: .ASCII <15>
\$LF: .ASCIZ <12>

\$RPADR: .WORD 176700
\$RPVEC: .WORD 254
\$LKCSR: .WORD 172540
\$LKCSB: .WORD 172542
\$LPVEC: .WORD 104
\$LKS: .WORD 177546
\$LLVEC: .WORD 100
\$PCLOCK: .WORD -1
\$CLKFLG: .WORD -1
\$SPRFLG: .WORD -1
\$LSCS: .WORD 177514
\$LSDB: .WORD 177516
\$PRTER: .WORD 0
HZ: .WORD 74

;; START OF COMMON TAGS
;; CONTAINS PASS COUNT
;; CONTAINS THE TEST NUMBER
;; CONTAINS ERROR FLAG
;; CONTAINS SUBTEST ITERATION COUNT
;; CONTAINS SCOPE LOOP ADDRESS
;; CONTAINS SCOPE RETURN FOR ERRORS
;; CONTAINS TOTAL ERRORS DETECTED
;; CONTAINS ITEM CONTROL BYTE
;; CONTAINS MAX. ERRORS PER TEST
;; CONTAINS PC OF LAST ERROR INSTRUCTION
;; CONTAINS ADDRESS OF 'GOOD' DATA
;; CONTAINS ADDRESS OF 'BAD' DATA
;; CONTAINS 'GOOD' DATA
;; CONTAINS 'BAD' DATA
;; RESERVED--NOT TO BE USED

;; AUTOMATIC MODE INDICATOR
;; INTERRUPT MODE INDICATOR

;; ADDRESS OF SWITCH REGISTER
;; ADDRESS OF DISPLAY REGISTER
;; TTY KBD STATUS
;; TTY KBD BUFFER
;; TTY PRINTER STATUS REG. ADDRESS
;; TTY PRINTER BUFFER REG. ADDRESS
;; CONTAINS NULL CHARACTER FOR FILLS
;; CONTAINS # OF FILLER CHARACTERS REQUIRED
;; INSERT FILL CHARS. AFTER A "LINE FEED"
;; "TERMINAL AVAILABLE" FLAG (BIT(07)=0=YES)
;; CODE FOR BELL
;; QUESTION MARK
;; CARRIAGE RETURN
;; LINE FEED

;; FIRST ADDRESS OF RH11/RP04 REGISTERS
;; RP04 VECTOR ADDRESS
;; ADDR OF KW11-P STATUS REGISTER
;; ADDR OF KW11-P COUNTER BUFFER
;; ADDR OF KW11-P VECTOR
;; ADDR OF KW11-L STATUS REGISTER
;; ADDR OF KW11-L VECTOR
;; '0' IF KW11-P IS ON SYSTEM
;; '0' IF A CLOCK IS AVAILABLE
;; PRINTER AVAILABILITY FLAG
;; PRINTER STATUS WORD ADDRESS
;; PRINTER DATA BUFFER ADDRESS
;; PRINTER ERROR FLAG
;; 74 (8) IF 60 HZ SYSTEM; 62 (8) IF 50 HZ SYSTEM

```

2758 001224 000000 LA30: .WORD 0 ;0 IF LA30, 1'S IF MOD 33 OR MOD 35 TTY ON SYSTEM
2759 001226 001100 STACK1: .WORD STACK ;CONSTANT USED TO CHECK STACK POINTER
2760 001230 000000 STATIN: .WORD 0 ;ERROR RATE DATA DISPLAY INDICATOR
2761 001232 000000 PACK: .WORD 0 ;'W' COMMAND INDICATOR
2762 001234 000000 000000 000000 DATE: .WORD 0,0,0,0,0 ;OPERATOR ENTERED DATE
2763 001242 000000 000000 000000
2764 001246 000000 000000 000000 OPERID: .WORD 0,0,0,0 ;OPERATOR ID
2765 001254 000000
2766 001256 000000 DRIVE: .WORD 0 ;DRIVE # STORAGE: ERRORS 1-5 & 10
2767 001260 000000 ATTN: .WORD 0 ;ATTN REG STORAGE: ERRORS 1-5 & 10
2768 001262 040 060 000 UNIT: .BYTE 40,60,0 ;DRIVE # STORAGE FOR PRINTOUT
2769 001266 .EVEN
2770
2771 ;:*****
2772
2773 .SBTTL CONTROL PARAMETERS
2774
2775 ;:*****
2776
2777 001266 002740 ENDCON: .WORD 002740 ;1.875X1018 WORDS (10) [3X1019 BITS]
2778 001270 005455 .WORD 005455 ;MSW
2779 001272 143300 ENDSEK: .WORD 143300 ;3 X 1016 SEEKS (LSW)
2780 001274 000055 .WORD 55 ;MSW
2781 001276 000001 PASCNT: .WORD 1 ;NUMBER OF PASSES TO END OF TEST
2782 001300 000000 MAXDL: .WORD 0 ;MAXIMUM DATA TRANSFER SIZE IN WORDS
2783 ;(FILLED BY PROGRAM AT STARTUP OR BY OPERATOR
2784 ;DURING PARAMETER ENTRY DIALOG.)
2785 001302 000144 MAXER: .WORD 100. ;MAXIMUM ERRORS - 100(10)
2786 001304 000005 INTRVL: .WORD 5 ;LOW BYTE IS PERFORMANCE TIMEOUT INTERVAL
2787 ;COUNTER. UPPER BYTE IS VALUE.
2788 001306 000004 CMLPMT: .WORD 4 ;NUMBER OF COMPARE ERRORS TYPED OUT
2789 001310 000001 FORMAT: .WORD 1 ;IF EQ 1, ALLOW WRITE HEADER & DATA ORDERS
2790 ;IF EQ 0, DO NOT ALLOW WRITE HEADER & DATA ORDERS
2791 001312 000001 SAVLOD: .WORD 1 ;IF EQ 1, SAVE THE LOADER
2792 ;IF EQ 0, DO NOT SAVE THE LOADER
2793 001314 000003 RATIO: .WORD 3 ;READ/WRITE RATIO [RANGE 0 - 7]
2794 ;0 - 15/1 (READ/WRITE)
2795 ;1 - 7/1
2796 ;2 - 6/2
2797 ;3 - 5/3
2798 ;4 - 4/4
2799 ;5 - 3/5
2800 ;6 - 2/6
2801 ;7 - 1/7
2802 001316 000001 AUTOCK: .WORD 1 ;IF EQ 1, DO AN APPROPRIATE WRITE
2803 ;CHECK AFTER EACH WRITE ORDER.
2804 ;IF EQ 0, SELECT WRITE CHECK ORDERS
2805 ;RANDOMLY.
2806 001320 000001 NOTPRT: .WORD 1 ;IF EQ 1, DO NOT PRINT DATA ERROR MESSAGES
2807 ;ASSOCIATED WITH OPERATOR SPECIFIED
2808 ;BAD PACK AREAS.
2809 ;IF EQ 1, PRINT ERROR MESSAGES RELATING TO
2810 ;THESE AREAS.
2811 001322 000001 ENDET: .WORD 1 ;IF EQ 1, END OF PASS DETERMINED
2812 ;BY THE 'WORDS READ' COUNT.
2813 ;IF EQ 0, END OF PASS DETERMINED

```

;BY THE SEEK COUNT.

2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850

;;*****

.SBTTL RPO4 ADDRESS LIMIT VALUES

;;*****

001324 000632
001326 000000
001330 000022
001332 000000
001334 000025
001336 000000

MAXCYL: .WORD 632 ;MAX CYLINDER [RANGE 0 - 632 (OCTAL)]
MINCYL: .WORD 0 ;MINIMUM CYLINDER [RANGE 0 - (<= MAXCYL)]
MAXTRK: .WORD 22 ;MAX TRACK [RANGE 0 - 22 (OCTAL)]
MINTRK: .WORD 0 ;MIN TRACK [RANGE 0 TO <= MAXTRK]
MAXSEC: .WORD 25 ;MAX SECTOR [RANGE 0 - 25 (OCTAL)]
MINSEC: .WORD 0 ;MIN SECTOR [RANGE 0 TO <= MAXSEC]

;;*****

.SBTTL VALUES FOR FIRST OPERATION

;;*****

001340 000010
001342 000005

BEGPAT: .WORD 10 ;STARTING PATTERN CODE [RANGE 1 - 17 (OCTAL)]
BEGCOD: .WORD 5 ;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')

001344 000404

BEGSIZ: .WORD 404

;STARTING RECORD SIZE [RANGE 4 - MAXMEM]
;NOTE: THE SIZE MUST BE AT LEAST 4 IF
;WRITE DATA OR READ DATA; THE SIZE MUST
;BE AT LEAST 8 IF WRITE HEADER AND
;DATA OR READ HEADER AND DATA.
;IF THE SIZE IS GREATER THAN 1 SECTOR, THE
;SIZE MUST ALLOW FOR OVERLAPPING 4 OR 8
;WORDS INTO THE LAST SECTOR USED.

```

2851 .SBTTL ERROR POINTER TABLE
2852
2853 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
2854 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
2855 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
2856 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
2857 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
2858
2859 ;*      EM      ;;POINTS TO THE ERROR MESSAGE
2860 ;*      DH      ;;POINTS TO THE DATA HEADER
2861 ;*      DT      ;;POINTS TO THE DATA
2862 ;*      DF      ;;POINTS TO THE DATA FORMAT
2863
2864
2865 001346 $ERRTB:
2866 ;ERROR 1
2867
2868 001346 043712 EM1 ;ILLEGAL RH11 INTERRUPT
2869 001350 000000 0
2870 001352 000000 0
2871 001354 000000 0
2872
2873 ;ERROR 2
2874
2875 001356 043762 EM2 ;UNEXPECTED ATTENTION DETECTED
2876 001360 046073 DH2 ;DRV RHAS
2877 001362 046512 DT2 ;DATA POINTER
2878 001364 047352 DF2 ;DRV-DEC, REG=OCTAL
2879
2880 ;ERROR 3
2881
2882 001366 044007 EM3 ;MASSBUS CONTROL BUS PARITY ERROR (SEEN BY THE RH11)
2883 001370 046150 DH3 ;DRV ADDR OF REG CONTENTS
2884 001372 046530 DT3 ;DATA POINTER
2885 001374 047360 DF3 ;ALL OCTAL EXCEPT DRV
2886
2887 ;ERROR 4
2888
2889 001376 044065 EM4 ;CONTROL BUS PARITY ERROR (SEEN BY THE RP04)
2890 001400 046176 DH4 ;DRV REG ADDR WRITTEN READ
2891 001402 046540 DT4 ;DATA POINTER
2892 001404 047363 DF4 ;OCTAL EXCEPT DRV
2893
2894 ;ERROR 5
2895
2896 001406 044143 EM5 ;ATTEN FROM AN OFFLINE OR UNAVAIL DRIVE
2897 001410 046073 DH2 ;DRV RHAS
2898 001412 046512 DT2 ;DATA POINTER
2899 001414 047352 DF2 ;OCTAL
2900
2901 ;ERROR 6
2902
2903 001416 000000 0 ;RESERVED
2904 001420 000000 0
2905 001422 000000 0
2906 001424 000000 0

```

```

2907
2908 ;ERROR 7
2909
2910 001426 000000 0 ;RESERVED
2911 001430 000000 0
2912 001432 000000 0
2913 001434 000000 0
2914
2915
2916 ;*****
2917
2918 .SBTTL SETUP AND INITIALIZATION ROUTINE
2919
2920 ; START ADDRESS = 200
2921 ; RESTART ADDRESS = 204
2922
2923 ;*****
2924
2925 001436 START:
2926 .SBTTL INITIALIZE THE COMMON TAGS
2927 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
2928 001436 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
2929 001442 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
2930 001444 022706 001140 CMP #SWR,R6 ;;DONE?
2931 001450 001374 BNE -6 ;;LOOP BACK IF NO
2932 001452 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
2933 ;;INITIALIZE A FEW VECTORS
2934 001456 012737 026602 000030 MOV #ERROR,#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
2935 001464 012737 000340 000032 MOV #340,#EMTVEC+2 ;;LEVEL 7
2936 001472 012737 030442 000034 MOV #TRAP,#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
2937 001500 012737 000340 000036 MOV #340,#TRAPVEC+2 ;;LEVEL 7
2938 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
2939 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
2940 001506 013746 000004 MOV #ERRVEC,-(SP) ;;SAVE ERROR VECTOR
2941 001512 012737 001546 000004 MOV #64$,#ERRVEC ;;SET UP ERROR VECTOR
2942 001520 012737 177570 001140 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
2943 001526 012737 177570 001142 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
2944 001534 022777 177777 177376 CMP #-1,#SWR ;;TRY TO REFERENCE HARDWARE SWR
2945 001542 001012 BNE 66$ ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
2946 ;;AND THE HARDWARE SWR IS NOT = -1
2947 001544 000403 BR 65$ ;;BRANCH IF NO TIMEOUT
2948 001546 012716 001554 64$: MOV #65$,(SP) ;;SET UP FOR TRAP RETURN
2949 001552 000002 RTI
2950 001554 012737 000176 001140 65$: MOV #SWREG,SWR ;;POINT TO SOFTWARE SWR
2951 001562 012737 000174 001142 MOV #DISPREG,DISPLAY
2952 001570 012637 000004 66$: MOV (SP)+,#ERRVEC ;;RESTORE ERROR VECTOR
2953
2954 001574 104401 052100 1$: TYPE TITLE ;TYPE THE PROGRAM NAME AND MAINDEC NUMBER
2955 001600 012737 001165 001576 MOV #CRLF,1$+2 ;CHANGE 'TITLE' TO CR-LF
2956 001606 012737 000240 000032 MOV #240,#EMTVEC+2 ;CHANGE EMT PRIORITY TO 5
2957 001614 012737 000240 000036 MOV #240,#TRAPVEC+2 ;CHANGE TRAP PRIORITY TO 5
2958 001622 012737 002302 000206 MOV #RSTART,206 ;SETUP RESTART ADDRESS
2959 001630 005037 001230 CLR STATIN ;CLEAR ERROR RATE DATA TYPE INDICATOR
2960 001634 105037 001157 CLR $TPFLG ;SET TTY AVAILABILITY FLAG
2961 001640 012705 041616 MOV #ORDERQ,R5 ;START OF AREA TO CLEAR
2962 001644 005025 CLR (R5)+

```

2963	001646	022705	042124		CMP	#BLKADR,R5	;LOOK FOR END OF CLEAR AREA
2964	001652	001374			BNE	.-6	;BR IF NOT FINISHED
2965	001654	013737	001222	021720	MOV	HZ,SIXTEE	;1/60 TH OR 1/50 TH SECOND COUNTER VALUE
2966	001662	005437	021720		NEG	SIXTEE	;CONVERT TO 2'S COMP
2967	001666	012737	030060	021702	MOV	#30060,HOUR	;ASCII '00' TO HOUR COUNTER
2968	001674	012737	030060	021706	MOV	#30060,MINUTE	;ASCII '00' TO MINUTES COUNTER
2969	001702	012737	030060	021712	MOV	#30060,SECOND	;ASCII '00' TO SECONDS COUNTER
2970	001710	105037	001305		CLRB	INTRVL+1	;CLEAR INTERVAL COUNTER
2971	001714	104401	052427		TYPE	,ENTDAT	; 'ENTER DATE'
2972	001720	104407			RDLIN		;READ THE ENTRY
2973	001722	012605			MOV	(SP)+,R5	;PUT THE ENTRY ADDRESS INTO R5
2974	001724	012537	001234		MOV	(R5)+,DATE	;STORE THE DATE
2975	001730	012537	001236		MOV	(R5)+,DATE+2	;STORE THE DATE
2976	001734	012537	001240		MOV	(R5)+,DATE+4	;STORE THE DATE
2977	001740	012537	001242		MOV	(R5)+,DATE+6	;STORE THE DATE
2978	001744	104401	052446		TYPE	,ENTID	; 'ENTER OPERATOR I.D.'
2979	001750	104407			RDLIN		;READ THE ENTRY
2980	001752	012605			MOV	(SP)+,R5	;ENTRY ADDRESS
2981	001754	012537	001246		MOV	(R5)+,OPERID	;STORE THE I.D.
2982	001760	012537	001250		MOV	(R5)+,OPERID+2	;STORE THE I.D.
2983	001764	012537	001252		MOV	(R5)+,OPERID+4	;STORE THE I.D.

;ROUTINE TO DETERMINE BUFFER AREA SIZE

2987	001770	004737	027740		SIZMEM:	JSR	PC,\$SIZE	;SEE HOW MUCH MEMORY ON SYSTEM
2988	001774	022737	053654	030034		CMP	#ENDPGM,\$LSTAD	;SEE IF ENOUGH MEMORY FOR PROGRAM
2989	002002	103402				BLO	1\$;BR IF MEMORY
2990	002004	000137	020102			JMP	MEMERR	;REPORT NOT ENOUGH MEMORY
2991	002010	012737	000001	042002	1\$:	MOV	#1,BUFTBL	;LOAD NUMBER OF BUFFERS
2992	002016	012737	053654	042004		MOV	#ENDPGM,BUFTBL+2	;STARTING ADDRESS OF BUFFER
2993	002024	013737	030034	042006		MOV	\$LSTAD,BUFTBL+4	;LAST ADDR TO BUFFER ALLOCATION TABLE
2994	002032	162737	053654	042006		SUB	#ENDPGM,BUFTBL+4	;SUBTRACT PROGRAM SPACE
2995	002040	000241				CLC		;CLEAR THE 'C' BIT
2996	002042	006037	042006			ROR	BUFTBL+4	;CONVERT TO WORD COUNT
2997	002046	105737	000041			TSTB	41	;SEE WHO LOADED THE PROGRAM
2998	002052	001004				BNE	2\$;BR IF LOADED BY 'XXDP'
2999	002054	162737	000140	042006		SUB	#96.,BUFTBL+4	;SUBTRACT 'ABS' LOADED SIZE
3000	002062	000403				BR	3\$;CONTINUE WITH BUFFER SPACE SETUP
3001	002064	162737	002002	042006	2\$:	SUB	#1026.,BUFTBL+4	;SUBTRACT 'XXDP' LOADER SIZE
3002	002072	005737	001300		3\$:	TST	MAXDL	;VALUE IN 'MAXDL' ?
3003	002076	001012				BNE	LKPAR	;BR IF VALUE IS
3004	002100	012737	013534	001300		MOV	#5980.,MAXDL	;ASSUME FULL TRACK + 1 SEC MAXIMUM
3005	002106	023737	001300	042006		CMP	MAXDL,BUFTBL+4	;IS THAT TOO LARGE ?
3006	002114	103403				BLO	LKPAR	;BR IF NOT
3007	002116	013737	042006	001300		MOV	BUFTBL+4,MAXDL	;USE MAX AVAIL MEMORY AS MAX BUFFER SIZE

;SEE IF THE OPERATOR WANTS TO CHANGE ANY PARAMETERS

3011	002124	104401	053062		LKPAR:	TYPE	,ASKPAR	;ASK FOR PARAMETERS
3012	002130	104407				RDLIN		;READ THE ENTRY
3013	002132	012605				MOV	(SP)+,R5	;ADDRESS OF ENTRY TO R5
3014	002134	122715	000131			CMPB	#'Y,(R5)	;WAS ENTRY A 'Y' (YES)
3015	002140	001404				BEQ	ENTPAR	;BR IF A 'Y'
3016	002142	122715	000116			CMPB	#'N,(R5)	;WAS ENTRY A 'N' (NO)
3017	002146	001455				BEQ	RSTART	;BR IF AN 'N'
3018	002150	000765				BR	LKPAR	;NOT A VALID RESPONSE - RETRY

```

3019
3020 ;PARAMETER ENTRY ROUTINE
3021
3022 002152 104401 001165 ENTPAR: TYPE $SCLF ;CR-LF
3023 002156 005003 CLR R3 ;CLEAR OCTAL PARAMETER INDICATOR
3024 002160 012704 052736 MOV #PARLST,R4 ;ENTRY PARAMETER LIST TO R4
3025 002164 012437 002202 1$: MOV (R4)+,3$ ;ADDRESS OF PARAMETER NAME
3026 002170 001427 BEQ ENTPR ;BR IF AT END OF TABLE
3027 002172 100002 BPL 2$ ;BR IF NOT DECIMAL-OCTAL SEPARATOR
3028 002174 005203 INC R3 ;SET OCTAL PARAMETER INDICATOR
3029 002176 000772 BR 1$ ;CONTINUE WITH PARAMETER REQUESTS
3030 002200 104401 2$: TYPE ;TYPE THE PARAMETER NAME
3031 002202 000000 3$: .WORD 0 ;ADDRESS OF PARAMETER NAME TEXT
3032 002204 012405 MOV (R4)+,R5 ;ADDRESS OF PARAMETER LOCATION
3033 002206 011546 MOV (R5),-(SP) ;CONTENTS OF PARAMETER LOCATION
3034 002210 005703 TST R3 ;SEE IF OCTAL OR DECIMAL PARAMETER
3035 002212 001007 BNE 4$ ;BR IF OCTAL
3036 002214 104405 TYPDS ;TYPE THE CURRENT VALUE OF THE PARAMETER
3037 002216 104401 053175 TYPE DEC10 ;TYPE DECIMAL INDICATOR
3038 002222 004737 025442 JSR PC,GETDEC ;GET THE DECIMAL ENTRY
3039 002226 012615 MOV (SP)+,(R5) ;MOVE NEW VALUE TO LOCATION
3040 002230 000755 BR 1$ ;GET MORE PARAMETERS
3041 002232 104402 4$: TYPOC ;TYPE THE CURRENT VALUE
3042 002234 104401 053166 TYPE OCT8 ;TYPE OCTAL INDICATOR
3043 002240 004737 025304 JSR PC,GETOCT ;GET THE OCTAL PARAMTER
3044 002244 012615 MOV (SP)+,(R5) ;MOVE NEW VALUE TO PARAMETER LOCATION
3045 002246 000746 BR 1$ ;GET MORE PARAMETERS
3046 002250 005737 001312 ENTPR: TST SAVLOD ;SAVE THE LOADER ?
3047 002254 001012 BNE RSTART ;BR IF LOADER TO BE SAVED
3048 002256 105737 000041 TSTB 41 ;SEE WHO LOADED THE PROGRAM
3049 002262 001004 BNE 1$ ;BR IF 'ACT' OR 'XXDP'
3050 002264 062737 000140 042006 ADD #96.,BUFTBL+4 ;'ABS' LOADER SIZE
3051 002272 000403 BR RSTART ;CONTINUE
3052 002274 062737 002002 042006 1$: ADD #1026.,BUFTBL+4 ;'XXDP' LOADER SIZE
3053
3054 ;RESTART (START FROM LOC 204) ENTERS HERE
3055
3056 002302 012706 001100 RSTART: MOV #STACK,SP ;RESTORE STACK POINTER FOR RESTART
3057 002306 005037 177776 CLR #PS ;CLEAR PROCESSOR STATUS
3058 002312 005037 001232 CLR PACK ;'R' OR 'W' COMMAND INDICATOR
3059 002316 012737 123456 027736 MOV #123456,$LONUM ;INITIALIZE LOW RANDOM NUMBER
3060 002324 012737 176543 027734 MOV #176543,$HINUM ;INITIALIZE HIGH RANDOM NUMBER
3061 002332 113737 001304 001305 MOVB INTRVL,INTRVL+1 ;RESTORE STATISTICS TYPEOUT INTERVAL
3062 002340 042737 170000 001302 BIC #170000,MAXER ;MAKE SURE LIMITS ARE NOT TOO LARGE
3063
3064 ;CHECK THE PARAMETERS FOR VALIDITY
3065
3066 002346 023737 001300 042006 CKADR: CMP MAXDL,BUFTBL+4 ;REQUESTED TRANSFER SIZE TOO LARGE ?
3067 002354 101004 BHI 1$ ;BR IF REQUESTED MAXIMUM TOO LARGE
3068 002356 022737 000004 001300 CMP #4,MAXDL ;SEE IF MAXIMUM TRANSFER SIZE TOO SMALL
3069 002364 101404 BLOS 2$ ;BR IF NOT TOO SMALL
3070 002366 012746 053204 1$: MOV #PAR1,-(SP) ;ADDR OF PARAMETER NAME
3071 002372 000137 002710 JMP BADPAR ;REPORT THE PARAMETER ERROR
3072 002376 023737 001300 001344 2$: CMP MAXDL,BEGSIZ ;IS MAX RECORD LENGTH SMALLER THAN 1 SECTOR ?
3073 002404 103003 BHIS CKPAR ;BR IF NOT
3074 002406 013737 001300 001344 MOV MAXDL,BEGSIZ ;USE MAX RECORD SIZE AS STARTING SIZE

```

3075	002414	022737	000632	001324	CKPAR:	CMP	#410.,MAXCYL	:MAXIMUM CYLINDER ADDR TOO LARGE
3076	002422	103004				BHIS	1\$:BR IF NOT
3077	002424	012746	053234			MOV	#PAR4,-(SP)	:ADDR OF PARAMETER NAME
3078	002430	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3079	002434	023737	001324	001326	1\$:	CMP	MAXCYL,MINCYL	:MAX CYLINDER LARGER THAN MIN CYLINDER
3080	002442	103004				BHIS	2\$:BR IF IT IS
3081	002444	012746	053244			MOV	#PAR5,-(SP)	:ADDR OF PARAMETER NAME
3082	002450	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3083	002454	022737	000022	001330	2\$:	CMP	#18.,MAXTRK	:MAXIMUM TRACK ADDRESS TOO LARGE
3084	002462	103004				BHIS	3\$:BR IF NOT
3085	002464	012746	053254			MOV	#PAR6,-(SP)	:ADDRESS OF PARAMETER NAME
3086	002470	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3087	002474	023737	001330	001332	3\$:	CMP	MAXTRK,MINTRK	:SEE IF MAX TRACK EQ OR LARGER THAN MIN TRACK
3088	002502	103004				BHIS	4\$:BR IF IT IS
3089	002504	012746	053264			MOV	#PAR7,-(SP)	:ADDRESS OF PARAMETER NAME
3090	002510	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3091	002514	022737	000025	001334	4\$:	CMP	#21.,MAXSEC	:SEE IF MAX SECTOR TOO LARGE
3092	002522	103004				BHIS	5\$:BR IF MAX SECTOR OK
3093	002524	012746	053274			MOV	#PAR8,-(SP)	:ADDRESS OF PARAMETER NAME
3094	002530	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3095	002534	023737	001334	001336	5\$:	CMP	MAXSEC,MINSEC	:MINIMUM SECTOR ADDRESS TOO LARGE
3096	002542	103004				BHIS	6\$:BR IF NOT
3097	002544	012746	053304			MOV	#PAR9,-(SP)	:ADDRESS OF PARAMETER NAME
3098	002550	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3099	002554	022737	000007	001314	6\$:	CMP	#7,RATIO	:SEE IF R/W RATIO TOO LARGE
3100	002562	103004				BHIS	7\$:BR IF NOT
3101	002564	012746	053354			MOV	#PAR14,-(SP)	:ADDRESS OF PARAMETER NAME
3102	002570	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3103	002574	022737	000017	001340	7\$:	CMP	#15.,BEGPAT	:SEE IF VALID PATTERN SELECTED
3104	002602	103004				BHIS	8\$:BR IF PATTERN CODE VALID
3105	002604	012746	053414			MOV	#PAR18,-(SP)	:ADDRESS OF PARAMETER NAME
3106	002610	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3107	002614	005737	001342		8\$:	TST	BEGCOD	:IS STARTING CODE OK
3108	002620	001447				BEQ	SETVEC	:BR IF OK
3109	002622	022737	000001	001342		CMP	#1,BEGCOD	:IS STARTING CODE OK ?
3110	002630	001443				BEQ	SETVEC	:BR IF OK
3111	002632	022737	000002	001342		CMP	#2,BEGCOD	:IS STARTING CODE OK ?
3112	002640	001437				BEQ	SETVEC	:BR IF OK
3113	002642	022737	000004	001342		CMP	#4,BEGCOD	:IS STARTING CODE OK ?
3114	002650	001433				BEQ	SETVEC	:BR IF OK
3115	002652	022737	000005	001342		CMP	#5,BEGCOD	:IS STARTING CODE OK ?
3116	002660	001427				BEQ	SETVEC	:BR IF OK
3117	002662	022737	000003	001342		CMP	#3,BEGCOD	:IS STARTING CODE WRITE HEADER & DATA ?
3118	002670	001003				BNE	9\$:BR IF NOT
3119	002672	005737	001310			TST	FORMAT	:FORMAT ORDERS ALLOWED ?
3120	002676	001020				BNE	SETVEC	:BR IF THEY ARE
3121	002700	012746	053404		9\$:	MOV	#PAR17,-(SP)	:ADDRESS OF PARAMETER NAME
3122	002704	000137	002710			JMP	BADPAR	:REPORT PARAMETER ERROR
3123								
3124								
3125								
3126	002710	012637	002722			BADPAR:	MOV (SP)+,1\$:MOVE PARAMETER NAME
3127	002714	104401	053106				TYPE ,NGPAR	: 'BAD PARAMETER'
3128	002720	104401					TYPE	
3129	002722	000000			1\$:	.WORD	0	:PARAMETER NAME
3130	002724	104401	053136			TYPE	,REPAR	: 'REENTER PARAMETERS

5


```

3131 002730 105037 001305          CLRB      INTRVL+1      ;CLEAR PERFORMANCE TYPEOUT INTERVAL COUNTER
3132 002734 000137 002152          JMP       ENTPAR      ;GO THROUGH PARAMETER ENTRY ROUTINE AGAIN
3133
3134          ;DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
3135          ;
3136
3137 002740 004737 020212          SETVEC: JSR      PC,CKPTR      ;SEE IF PRINTER ON SYSTEM
3138 002744 013737 001170 030714    MOV      $RPNADR,RPNADR      ;RPN ADDRESS
3139 002752 013737 001172 030716    MOV      $RPVEC,RPVEC      ;RPN VECTOR ADDRESS
3140 002760 004737 020260          JSR      PC,CKCLK      ;SEE IF CLOCK ON SYSTEM
3141 002764 004737 030732          JSR      PC,RPINIT      ;INITIALIZE THE RPN DRIVER
3142 002770 012737 177777 030654    MOV      #-1,SAVEFG      ;SET THE SAVE REGISTERS FLAG
3143 002776 142737 000007 001263    BICB    #7,UNIT+1      ;CLEAR UNIT NUMBER
3144 003004 005004          CLR      R4      ;DRIVE TABLE POINTER
3145 003006 012703 000010          MOV      #8,R3      ;ITERATION COUNTER
3146 003012 104401 001165          TYPE    ,$CRLF      ;CR-LF
3147 003016 104401 051527          TYPE    ,SYSTAT      ;TYPE STATUS HEADING
3148 003022 104401 001262          1$:     TYPE    ,UNIT      ;TYPE UNIT NUMBER
3149 003026 104401 051362          TYPE    ,LIN4SP      ;SPACES
3150 003032 105764 030556          TSTB    DRVSTA(R4)      ;SEE WHAT DRIVE'S STATUS IS
3151 003036 001405          BEQ     2$      ;BR IF OFFLINE
3152 003040 100410          BMI     3$      ;BR IF UNSAFE OR NONEXISTANT
3153 003042 012737 051405 003144    MOV      #UNTON,8$      ;MESSAGE ADDRESS
3154 003050 000434          BR      7$      ;PRINT IT
3155 003052 012737 051374 003144    2$:     MOV      #UNTOFF,8$      ;OFFLINE MESSAGE ADDRESS
3156 003060 000430          BR      7$      ;PRINT IT
3157 003062 006304          3$:     ASL      R4      ;SETUP TO ADDRESS WORDS
3158 003064 005764 030566          TST     DRVTP(R4)      ;SEE IF UNIT PRESENT
3159 003070 001414          BEQ     4$      ;BR IF NOT PRESENT
3160 003072 022764 020020 030566    CMP     #20020,DRVTP(R4)      ;SEE IF SINLE PORT RPN
3161 003100 001414          BEQ     5$      ;BR IF IT IS
3162 003102 022764 024020 030566    CMP     #24020,DRVTP(R4)      ;SEE IF DUAL PORT RPN
3163 003110 001410          BEQ     5$      ;BR IF IT IS
3164 003112 012737 051465 003144    MOV      #NOTRP,8$      ;UNIT IS NOT AN RPN
3165 003120 000407          BR      6$      ;PRINT IT
3166 003122 012737 051502 003144    4$:     MOV      #NOTPRS,8$      ;UNIT NOT PRESENT
3167 003130 000403          BR      6$      ;PRINT IT
3168 003132 012737 051517 003144    5$:     MOV      #NOTSAF,8$      ;UNIT UNSAFE
3169 003140 006204          6$:     ASR      R4      ;RESTORE R4
3170 003142 104401          7$:     TYPE    ;TYPE THE IDENTIFYING MESSAGE
3171 003144 000000          8$:     .WORD   0      ;MESSAGE ADDRESS HERE
3172 003146 104401 001165          TYPE    , $CRLF      ;CR-LF
3173 003152 005303          DEC     R3      ;DECREMENT THE ITERATION COUNTER
3174 003154 001404          BEQ     9$      ;BR IF END
3175 003156 105237 001263          INCB    UNIT+1      ;INCREMENT THE UNIT NUMBER
3176 003162 005204          INC     R4      ;INCREMENT TABLE POINTER
3177 003164 000716          BR      1$      ;CONTINUE
3178 003166 104401 001165          9$:     TYPE    , $CRLF      ;CR-LF
3179 003172 004737 026066          JSR     PC,$TKINT      ;INITIALIZE THE KEYBOARD INTERRUPT HANDLER
3180 003176 104401 052670          TYPE    ,INTDON      ;TYPE 'INITIALIZE COMPLETE'
3181 003202 000137 003206          JMP     MAIN      ;START THE PROGRAM

```

;;*****

.SBTTL MAIN PROGRAM

3182
3183
3184
3185
3186

```

3187                                     ;:*****
3188                                     ;:*****
3189 003206 023706 001226 MAIN:  CMP      Q#STACK1,SP      ;CHECK STACK POINTER
3190 003212 001401          BEQ      .+4              ;BR IF OK
3191 003214 000000          HALT                    ;*****
3192 003216 005737 001230          TST      STATIN          ;TYPE ERROR RATE STATISTICS ?
3193 003222 001404          BEQ      .+12             ;--> BR IF NOT
3194 003224 005037 001230          CLR      STATIN          ;I CLEAR THE INDICATOR
3195 003230 004737 020450          JSR     PC,STATPR        ;I TYPE THE STATISTICS
3196 003234 012703 000010          MOV     #8.,R3          ;<-- UNIT COUNTER
3197 003240 012705 041650          MOV     #DUNIT,R5       ;ADDRESS OF 'DROP UNIT' TABLE
3198 003244 005715          1$:  TST     (R5)         ;SEE IF ENTRY AT PRESENT POSITION
3199 003246 001005          BNE     3$              ;BR IF THERE IS ONE
3200 003250 062705 000002          2$:  ADD     #2,R5        ;INCREMENT TO NEXT TABLE POSITION
3201 003254 005303          DEC     R3              ;DECREMENT UNIT COUNTER
3202 003256 001372          BNE     1$              ;BR IF MORE TO CHECK
3203 003260 000434          BR      MAIN1           ;FINISHED HERE, GO TRY TO ASSIGN UNITS
3204 003262 012701 041714          3$:  MOV     #AVAIL,R1     ;ADDRESS OF 'AVAILABLE' UNITS TABLE
3205 003266 005711          4$:  TST     (R1)         ;SEE IF AT END OF TABLE
3206 003270 001405          BEQ     5$              ;BR IF AT END: GO CHECK 'WAIT' TABLE
3207 003272 021115          CMP     (R1),(R5)       ;IS UNIT IN 'AVAIL' THE ONE TO BE DROPPED
3208 003274 001414          BEQ     7$              ;BR IF YES
3209 003276 062701 000002          ADD     #2,R1           ;INCREMENT 'AVAIL' TABLE ADDRESS
3210 003302 000771          BR      4$              ;CONTINUE LOOKING
3211 003304 012701 041736          5$:  MOV     #WAIT,R1     ;MOVE THE ADDRESS OF THE BUFFER WAIT TABLE
3212 003310 005711          6$:  TST     (R1)         ;AT THE END OF THE 'WAIT' TABLE ?
3213 003312 001756          BEQ     2$              ;BR IF YES: SEE IF ANY MORE 'DROP' REQUESTS
3214 003314 021115          CMP     (R1),(R5)       ;UNIT IN THE 'WAIT' TABLE ?
3215 003316 001403          BEQ     7$              ;BR IF IT IS
3216 003320 062701 000002          ADD     #2,R1           ;INCREMENT 'WAIT' TABLE ADDRESS
3217 003324 000771          BR      6$              ;CONTINUE LOOK THROUGH THE 'WAIT' TABLE
3218 003326 011100          7$:  MOV     (R1),R0       ;PUT THE UNIT'S BLOCK ADDRESS IN R0
3219 003330 104401 052015          TYPE   DEASSG          ;TYPE 'UNIT DEASSIGNED'
3220 003334 004737 021176          JSR     PC,TYPEST        ;TYPE THE UNIT'S ERROR RATE DATA
3221 003340 005015          CLR     (R5)           ;CLEAR THE 'DROP UNIT' TABLE ENTRY
3222 003342 005011          CLR     (R1)           ;REMOVE THE UNIT FROM THE 'AVAIL' OR 'WAIT' TABLE
3223 003344 004737 015322          JSR     PC,CMPRES        ;COMPRESS THE RESPECTIVE TABLE
3224 003350 000737          BR      2$              ;SEE IF ANY MORE UNITS
3225
3226                                     ;LOOK FOR DRIVES TO BE ASSIGNED
3227
3228 003352 012703 000010 MAIN1: MOV     #8.,R3          ;UNIT COUNT
3229 003356 005002          CLR     R2              ;'AVAIL' INDEX
3230 003360 005004          CLR     R4              ;ASSIGN LIST INDEX
3231 003362 005005          CLR     R5              ;NEW UNIT INDEX
3232 003364 005765 041672          1$:  TST     NEWUNT(R5)    ;NEW UNIT IN THIS POSITION
3233 003370 001006          BNE     3$              ;BR IF THERE IS
3234 003372 062705 000002          2$:  ADD     #2,R5        ;INCREMENT R5
3235 003376 005204          INC     R4              ;INCREMENT ASSIGN INDEX
3236 003400 005303          DEC     R3              ;DECREMENT UNIT COUNT
3237 003402 001370          BNE     1$              ;BR IF MORE UNITS
3238 003404 000506          BR      MAIN2           ;START OPERATIONS FOR THE AVAILABLE UNITS
3239 003406 142737 000007 001263 3$:  BICB   #7,UNIT+1       ;CLEAR PREVIOUS UNIT NUMBER
3240 003414 150437 001263          BISB   R4,UNIT+1       ;PRESENT UNIT NUMBER
3241 003420 104401 051367          TYPE   ,UNTMMSG        ;'UNIT'
3242 003424 104401 001262          TYPE   ,UNIT           ;UNIT NUMBER

```

3243	003430	104401	052065		TYPE	ASGND	;'ASSIGNED'
3244	003434	005762	041714	4\$:	TST	AVAIL(R2)	;AT END OF AVAILABLE TABLE
3245	003440	001403			BEQ	5\$;BR IF YES
3246	003442	062702	000002		ADD	#2,R2	;INCREMENT AVAILABLE TABLE INDEX
3247	003446	000772			BR	4\$;CONTINUE LOOKING FOR END OF TABLE
3248	003450	016562	041672	041714	5\$:	MOV	NEWUNT(R5),AVAIL(R2) ;MOVE ADDR OF UNIL INTO AVAIL LST
3249	003456	005065	041672		CLR	NEWUNT(R5)	;TAKE UNIT OUT OF NEW UNIT TABLE
3250	003462	112764	177777	041640		MOVB	#-1,ASNLST(R4) ;SET UNIT ASSIGNED INDICATOR
3251	003470	016200	041714		MOV	AVAIL(R2),R0	;PUT STARTING ADDRESS OF BLOCK IN R0
3252	003474	113760	001336	000010		MOVB	MINSEC,\$SEC(R0) ;INITIAL SECTOR VALUE
3253	003502	113760	001332	000011		MOVB	MINTRK,\$TRK(R0) ;INITIAL TRACK VALUE
3254	003510	013760	001326	000012		MOV	MINCYL,\$CYL(R0) ;INITIAL CYLINDER VALUE
3255	003516	113760	001342	000030		MOVB	BEGCOD,\$CODE(R0) ;INITIAL COMMAND CODE
3256	003524	013701	001342		MOV	BEGCOD,R1	;GET THE ACTUAL OP CODE
3257	003530	116160	042144	000002		MOVB	COMTBL(R1),\$COMND(R0) ;OPERATION CODE
3258	003536	113760	001340	000034		MOVB	BEGPAT,\$PATT(R0) ;PATTERN CODE
3259	003544	106360	000034		ASLB	\$PATT(R0)	;CONVERT CODE TO A TABLE INDEX
3260	003550	013760	001344	000020		MOV	BEGSIZ,\$WRDL(R0) ;BEGINNING RECORD SIZE
3261	003556	013760	001344	000004		MOV	BEGSIZ,\$WRDM(R0) ;VALUE FOR DATA TRANSFER
3262	003564	005460	000004		NEG	\$WRDM(R0)	;MAKE IT INTO 2'S COMPLEMENT
3263	003570	012760	000400	000026		MOV	#256,\$SSEC(R0) ;INITIAL VALUE OF SECTOR SIZE
3264	003576	132760	000001	000030		BITB	#1,\$CODE(R0) ;HEADER ORDER ?
3265	003604	001403			BEQ	6\$;BR IF NOT
3266	003606	062760	000004	000026		ADD	#4,\$SSEC(R0) ;ADD HEADER SIZE TO SECTOR SIZE
3267	003614	062702	000002		6\$:	ADD	#2,R2 ;INCREMENT AVAILABLE TABLE POINTER
3268	003620	000664			BR	2\$;LOOK FOR MORE UNITS
3269							
3270							;GET PARAMETERS, BUFFER SPACE, AND START ORDERS FOR DRIVES IN
3271							; THE 'AVAILABLE' QUEUE
3272							
3273	003622	005737	041736		MAIN2:	TST	WAIT ;OUTSTANDING BUFFER REQUESTS
3274	003626	001110				BNE	MAIN3 ;BR IF THERE ARE
3275	003630	005737	041714			TST	AVAIL ;ANY UNITS WAITING FOR PARAMETERS
3276	003634	001545				BEQ	IDLE ;BRANCH IF NONE
3277	003636	013700	041714			MOV	AVAIL,R0 ;CONTROL BLOCK ADDR IN R0
3278	003642	005760	000112			TST	\$NEXT(R0) ;PARAMETERS BEEN SELECTED ?
3279	003646	001021				BNE	2\$;BR IF THEY HAVE
3280	003650	105760	000031			TSTB	\$PACK(R0) ;'R' OR 'W' COMMAND FOR THE DRIVE ?
3281	003654	001403				BEQ	1\$;BR IF NOT
3282	003656	004737	015340			JSR	PC,WRTPK ;GET DATA PACK PARAMETERS
3283	003662	000415				BR	3\$;GET THE BUFFER
3284	003664	012701	041760	1\$:	MOV	#PARQ,R1 ;ADDRESS OF THE PARAMETER QUEUE	
3285	003670	020011		7\$:	7\$:	CMP	R0,(R1) ;IS CURRENT UNIT IN THE QUEUE
3286	003672	001403				BEQ	8\$;BR IF IT IS
3287	003674	005721				TST	(R1)+ ;AT END OF THE QUEUE
3288	003676	001403				BEQ	9\$;BR IF AT END
3289	003700	000773				BR	7\$;CONTINUE LOOKING
3290	003702	004737	015322	8\$:	8\$:	JSR	PC,CMPRES ;COMPRESS THE TABLE
3291	003706	004737	014416	9\$:	9\$:	JSR	PC,SELPAR ;SELECT THE PARAMETERS
3292	003712	004737	015124	2\$:	2\$:	JSR	PC,GETPAR ;LOAD NEW PARAMETERS
3293	003716	005046		3\$:	3\$:	CLR	-(SP) ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3294	003720	004737	013504			JSR	PC,GETBUF ;GET BUFFER
3295	003724	012660	000006			MOV	(SP)+,\$BUF(R0) ;MOVE BUFFER ADDR TO DPB
3296	003730	001426				BEQ	5\$;BR IF '0' ADDR (NO BUFFER)
3297	003732	004737	014072			JSR	PC,FILBUF ;FILL THE BUFFER
3298	003736	004737	014340			JSR	PC,GODRIV ;PUT CURRENT DPB IN DRIVER

```

3299 003742 012705 041616      MOV      #ORDERQ,R5      ;ADDRESS OF ORDER QUEUE IN R5
3300 003746 005725              TST      (R5)+          ;END OF QUEUE ?
3301 003750 001376              BNE      -2            ;BR IF NOT
3302 003752 010045              MOV      RO,-(R5)      ;PUT BLOCK ADDRESS INTO QUEUE
3303 003754 105760 000031      TSTB    $PACK(RO)     ;'R' OR 'W' COMMAND FOR DRIVE ?
3304 003760 001005              BNE      10$          ;BR IF EITHER
3305 003762 012705 041760      MOV      #PARQ,R5      ;PUT BLOCK INTO THE PARAMETER QUEUE
3306 003766 005725              4$:     TST      (R5)+          ;FIND THE END OF THE QUEUE
3307 003770 001376              BNE      4$           ;BR IF NOT AT END OF QUEUE
3308 003772 010045              MOV      RO,-(R5)      ;PUT BLOCK ADDRESS INTO THE QUEUE
3309 003774 012701 041714      10$:    MOV      #AVAIL,R1    ;R1 CONTAINS THE TABLE ADDRESS
3310 004000 004737 015322      JSR      PC,CMPRES     ;COMPRESS THE AVAILABILITY TABLE
3311 004004 000706              BR       MAIN2         ;CONTINUE PROCESSING
3312 004006 005005              5$:     CLR      R5          ;R5 USED AS 'WAIT' POINTER
3313 004010 000240              NOP                       ;DEBUGGING AID
3314 004012 005765 041736      TST      WAIT(R5)     ;AT END OF TABLE
3315 004016 001403              BEQ      6$           ;BR IF YES
3316 004020 062705 000002      ADD     #2,R5          ;INCREMENT R5
3317 004024 000770              BR       5$           ;CONTINUE SEARCHING
3318 004026 013765 041714 041736 6$:    MOV      AVAIL,WAIT(R5) ;MOVE ENTRY FROM AVAILABLE TO WAIT
3319 004034 012701 041714      MOV      #AVAIL,R1    ;TABLE TO COMPRESS
3320 004040 004737 015322      JSR      PC,CMPRES     ;COMPRESS THE INDICATED TABLE
3321 004044 000137 004150      JMP      IDLE          ;WAIT FOR A UNIT TO BECOME READY
3322
3323                          ;GET BUFFER ASSIGNMENTS FOR DRIVES INT THE 'BUFFER WAIT' QUEUE
3324
3325 004050 013700 041736      MAIN3:  MOV      WAIT,RO     ;MOVE THE 'WAIT' ENTRY TO RO
3326 004054 005046              CLR      -(SP)         ;MAKE ROOM ON THE STACK FOR THE BUFFER ADDR
3327 004056 004737 013504      JSR      PC,GETBUF     ;TRY TO GET A BUFFER
3328 004062 012660 000006      MOV      (SP)+,$BUF(RO) ;MOVE THE BUFFER ADDR TO THE DPB
3329 004066 001002              BNE      1$           ;BR IF A BUFFER WAS ASSIGNED
3330 004070 000137 004150      JMP      IDLE          ;NO BUFFER AVAILABLE YET
3331 004074 004737 014072      1$:     JSR      PC,FILBUF     ;FILL THE BUFFER
3332 004100 004737 014340      JSR      PC,GODRIV     ;PUT THE ENTRY IN THE DRIVER
3333 004104 012705 041616      MOV      #ORDERQ,R5   ;ADDRESS OF ORDER QUEUE IN R5
3334 004110 005725              TST      (R5)+          ;AT END OF THE QUEUE
3335 004112 001376              BNE      -2            ;BR IF NOT
3336 004114 010045              MOV      RO,-(R5)      ;PUT BLOCK ADDRESS IN QUEUE
3337 004116 105760 000031      TSTB    $PACK(RO)     ;'R' OR 'W' COMMAND FOR DRIVE ?
3338 004122 001005              BNE      3$           ;BR IF YES, DON'T PUT BLOCK INTO 'PARQ'
3339 004124 012705 041760      MOV      #PARQ,R5      ;FIND THE END OF THE PARAMETER QUEUE
3340 004130 005725              2$:     TST      (R5)+          ;OPEN SLOT IN THE QUEUE ?
3341 004132 001376              BNE      2$           ;BR IF NOT
3342 004134 010045              MOV      RO,-(R5)      ;PUT BLOCK ADDRESS INTO THE QUEUE
3343 004136 012701 041736      3$:     MOV      #WAIT,R1     ;ADDRESS OF TABLE TO TO COMPRESS
3344 004142 004737 015322      JSR      PC,CMPRES     ;COMPRESS THE WAIT TABLE
3345 004146 000625              BR       MAIN2         ;LOOK FOR MORE ENTRIES
3346
3347                          ;WAIT FOR AN ORDER TO FINISH
3348
3349 004150 023706 001226      IDLE:   CMP      @#STACK1,SP ;CHECK STACK POINTER
3350 004154 001401              BEQ      .+4          ;BR IF OK
3351 004156 000000              HALT                       ;STACK POINTER NOT 1100 *****
3352 004160 012701 041616      1$:     MOV      #ORDERQ,R1    ;ADDRESS OF THE ORDER QUEUE IN R1
3353 004164 012100              MOV      (R1)+,RO      ;PUT BLOCK ADDRESS INTO RO
3354 004166 001433              BEQ      IDLE1        ;BR IF END OF QUEUE
  
```

3355	004170	005760	000016		TST	\$STATUS(RO)	;SEE IF UNIT FINISHED	
3356	004174	001773			BEQ	1\$;BR IF UNIT NOT FINISHED	
3357	004176	162701	000002		SUB	#2,R1	;CORRECT THE QUEUE POINTER	
3358	004202	010146			MOV	R1,-(SP)	;SAVE THE QUEUE ADDRESS	
3359	004204	004737	013350		JSR	PC,STATIS	;ACCUMULATE STATISTICS FOR UNIT IN RO	
3360	004210	000240			NOP		;DEBUGGING AID	
3361	004212	004737	004460		JSR	PC,PROCES	;PROCESS END OF ORDER	
3362	004216	005037	016416		CLR	PR2B	;CLEAR THE BAD TRK/SEC ERROR INDICATOR	
3363	004222	004737	023704		JSR	PC,ABNRML	;SEE IF ANY UNITS HAVE TOO MANY ERRORS	
3364	004226	004737	023726		JSR	PC,NRML	;SEE IF ANY UNIT HAS XFERED 3X10 ⁹ BITS	
3365	004232	012601			MOV	(SP)+,R1	;RESTORE THE ORDER TABLE INDEX	
3366	004234	012705	041714		MOV	#AVAIL,R5	;FIND THE END OF THE 'AVAILABLE' TABLE	
3367	004240	005725		2\$:	TST	(R5)+	;END OF THE TABLE ?	
3368	004242	001376			BNE	2\$;BR IF NOT AT END OF LIST	
3369	004244	011145			MOV	(R1),-(R5)	;MOV THE BLOCK ADDRESS INTO THE TABLE	
3370	004246	004737	015322		JSR	PC,CMPRES	;COMPRESS THE ORDER QUEUE	
3371	004252	004737	013644		JSR	PC,RELBUF	;RESTORE BUFFER	
3372	004256	005737	041760	IDLE1:	TST	PARQ	;ENTRY IN THE PARAMETER QUEUE ?	
3373	004262	001410			BEQ	1\$;BR IF NOT	
3374	004264	013700	041760		MOV	PARQ,RO	;PUT THE BLOCK ADDRESS INTO RO	
3375	004270	004737	014416		JSR	PC,SELPAR	;GET THE PARAMETERS FOR NEXT OPERATION	
3376	004274	012701	041760		MOV	#PARQ,R1	;SETUP TO COMPRESS THE TABLE	
3377	004300	004737	015322		JSR	PC,CMPRES	;COMPRESS THE PARAMETER QUEUE	
3378	004304	000137	003206	1\$:	JMP	MAIN	;CONTINUE THE LOOP	
3379								
3380							;SETUP TO REFORMAT AN ERROR SECTOR	
3381								
3382	004310	032737	000001	001140	REFMT:	BIT	#SW0,SWR	;READ ONLY SWITCH SET ?
3383	004316	001057			BNE	REFMTX	;BR IF IT IS	
3384	004320	032737	000200	001140	BIT	#SW7,SWR	;SWITCH 7 SET ?	
3385	004326	001053			BNE	REFMTX	;BR IF IT IS	
3386	004330	005737	001310		TST	FORMAT	;WRITE HEADER & DATA ORDERS ALLOWED ?	
3387	004334	001450			BEQ	REFMTX	;BR IF NOT	
3388	004336	016060	000222	000106	MOV	\$RHCC(RO),\$NCYL(RO)	;USE PRESENT CYLINDER	
3389	004344	005046			CLR	-(SP)	;CLEAR STACK	
3390	004346	005046			CLR	-(SP)	;FOR SECTOR & TRACK	
3391	004350	004737	020124		JSR	PC,READDR	;GET CORRECTED SECTOR-TRACK ADDRESSES	
3392	004354	112660	000105		MOVB	(SP)+,\$NTRK(RO)	;TRACK ADDR TO DPB	
3393	004360	112660	000104		MOVB	(SP)+,\$NSEC(RO)	;SECTOR ADDR TO DPB	
3394	004364	012760	000404	000110	MOV	#260,\$NWRDL(RO)	;WORD COUNT FOR FORMAT	
3395	004372	023727	001300	000404	CMP	MAXDL,#260.	;CAN A FULL SECTOR BE WRITTEN ?	
3396	004400	103003			BHIS	1\$;BR IF IT CAN	
3397	004402	013760	001300	000110	MOV	MAXDL,\$NWRDL(RO)	;PUT TRANSFER SIZE INTO THE DPB	
3398	004410	112760	000003	000102	1\$:	MOVB	#3,\$NCODE(RO)	;COMMAND CODE
3399	004416	004737	015076		JSR	PC,GETPAT	;GET A PATTERN	
3400	004422	110560	000103		MOVB	R5,\$NPATC(RO)	;PATTERN CODE TO CONTROL BLOCK	
3401	004426	012760	177777	000112	MOV	#-1,\$NEXT(RO)	;SET PARAMETERS SELECTED INDICATOR	
3402	004434	012701	041760		MOV	#PARQ,R1	;SET UP TO SEE IF BLOCK IN THE PARAMETER QUEUE	
3403	004440	005711		2\$:	TST	(R1)	;SEE IF AT END OF TABLE	
3404	004442	001405			BEQ	REFMTX	;BR IF AT END	
3405	004444	020021			CMP	RO,(R1)+	;SEE IF BLOCK AT PRESENT POSITION	
3406	004446	001374			BNE	2\$;BR IF NOT	
3407	004450	005041			CLR	-(R1)	;CLEAR THE ENTRY	
3408	004452	004737	015322		JSR	PC,CMPRES	;COMPRESS THE TABLE	
3409	004456	000207		REFMTX:	RTS	PC	;RETURN	
3410								

```

3411 ;PROCESS THE ORDER TERMINATION
3412
3413 004460 005760 000016 PROCES: TST STATUS(RO) ;SEE IF DRIVER SIGNALLED AN ERROR
3414 004464 100427 BMI ERPROC ;BR IF ERROR
3415 004466 032760 100000 000164 BIT #BIT15,$RHCS1(RO) ;SEE IF 'SC' SET
3416 004474 001410 BEQ 1$ ;BR IF NOT SET
3417 004476 032760 040000 000164 BIT #BIT14,$RHCS1(RO) ;SEE IF 'TRE' SET
3418 004504 001017 BNE ERPROC ;BR IF SET
3419 004506 032760 040000 000176 BIT #BIT14,$RHDS1(RO) ;SEE IF 'ERR' SET
3420 004514 001013 BNE ERPROC ;BR IF SET
3421 004516 004737 010474 1$: JSR PC,CKERR ;NO ERROR, CHECK ERROR BITS ANYWAY
3422 004522 004737 010574 JSR PC,CKBUS ;NO ERROR, CHECK BUS ADDR & WC
3423 004526 032737 000004 001140 BIT #SW02,SWR ;NORMAL DATA COMPARE ?
3424 004534 001002 BNE 2$ ;BR IF NOT
3425 004536 004737 010660 JSR PC,CMPAR ;NO ERROR, COMPARE DATA
3426 004542 000207 2$: RTS PC ;RETURN
3427
3428 ;ORDER TERMINATED WITH AN ERROR - PROCESS THE ERROR
3429
3430 004544 032760 000200 000016 ERPROC: BIT #BIT07,STATUS(RO) ;DONE BIT SET ?
3431 004552 001402 BEQ ERPC1 ;BR IF ORDER DIDN'T COMPLETE NORMALLY
3432 004554 000137 005202 JMP DONE ;PROCESS ERROR WITH 'DONE' BIT SET
3433
3434 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE NOT' BITS
3435
3436 004560 142737 000007 001263 ERPC1: BICB #7,UNIT+1 ;CLEAR OLD UNIT NUMBER
3437 004566 151037 001263 BISB (RO),UNIT+1 ;SET PRESENT UNIT NUMBER
3438 004572 032760 010000 000016 BIT #BIT12,STATUS(RO) ;SEE IF DRIVE WAS UNSAFE
3439 004600 001025 BNE PUNSAF ;BR IF YES
3440 004602 032760 004000 000016 BIT #BIT11,STATUS(RO) ;PARITY ERROR OCCURRED
3441 004610 001053 BNE UCPAR ;BR IF IT DID
3442 004612 032760 002000 000016 BIT #BIT10,STATUS(RO) ;FATAL PARITY ERROR?
3443 004620 001054 BNE FALPAR ;BR IF THERE IS ONE
3444 004622 032760 001000 000016 BIT #BIT09,STATUS(RO) ;TIMEOUT?
3445 004630 001072 BNE SWTIM ;BR IF YES
3446 004632 032760 040000 000016 BIT #BIT14,STATUS(RO) ;UNIT WENT OFFLINE ?
3447 004640 001107 BNE OFLIN ;BR IF IT DID
3448 004642 032760 000004 000016 BIT #BIT2,STATUS(RO) ;PORT REQUEST TIME OUT ?
3449 004650 001135 BNE PRTIM ;BR IF IT DID
3450 004652 000207 RTS PC ;ERROR. RETURN
3451
3452 ;DRIVE IS PERSISTENTLY UNSAFE
3453
3454 004654 104401 001165 PUNSAF: TYPE ,SCRLF ;CR-LF
3455 004660 104401 044300 TYPE ,EM12 ;'DRIVE UNSAFE' MESSAGE
3456 004664 104401 052040 TYPE ,DRNUM ;DRIVE NUMBER
3457 004670 104401 001262 TYPE ,UNIT ;TYPE THE DRIVE NUMBER
3458 004674 104401 001165 TYPE ,SCRLF ;CR-LF
3459 004700 004737 016002 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3460 004704 104412 044300 DISPLY ,EM12 ;PERSISTENT DEVICE UNSAFE MESSAGE
3461 004710 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3462 004714 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3463 004720 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
3464 004724 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3465 004730 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3466 004734 000137 023622 JMP DROP ;DROP THE UNIT

```

```

3467
3468 ;UNCORRECTABLE MASSBUS PARITY ERROR OCCURED
3469
3470 004740 104401 001165 UCPAR: TYPE ,SCLF ;CR-LF
3471 004744 104401 044202 TYPE ,EM10 ;'UNCORRECTABLE PARITY ERROR' MESSAGE
3472 004750 000404 BR FALPRI ;FINISH PROCESSING THE ERROR
3473
3474 ;'FATAL' MASSBUS PARITY ERROR OCCURED
3475
3476 004752 104401 001165 FALPAR: TYPE ,SCLF ;CR-LF
3477 004756 104401 044245 TYPE ,EM11 ;'FATAL PARITY ERROR' MESSAGE
3478 004762 104401 052040 FALPRI: TYPE ,DRNUM ;DRIVE NUMBER
3479 004766 104401 001262 TYPE ,UNIT ;TYPE THE DRIVE NUMBER
3480 004772 104401 001165 TYPE ,SCLF ;CR-LF
3481 004776 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3482 005002 032737 100000 001140 BIT #SW15,SWR ;HALT ON ERROR ?
3483 005010 001401 BEQ .+4 ;BR IF NOT
3484 005012 000000 HALT ;ERROR HALT
3485 005014 000207 RTS PC
3486
3487 ;SOFTWARE TIMEOUT OCCURED
3488
3489 005016 004737 016002 SWTIM: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3490 005022 104412 044331 DISPLY ,EM13 ;PRINT THE TIME OUT MESSAGE
3491 005026 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3492 005032 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3493 005036 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3494 005042 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3495 005046 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3496 005052 000240 NOP
3497 005054 000240 NOP
3498 005056 000207 RTS PC ;RETURN
3499
3500 ;DRIVE WENT OFFLINE
3501
3502 005060 104401 001165 OFLIN: TYPE ,SCLF ;CR-LF
3503 005064 104401 044403 TYPE ,EM14 ;'DRIVE WENT OFFLINE' MESSAGE
3504 005070 104401 052040 TYPE ,DRNUM ;DRIVE NUMBER
3505 005074 104401 001262 TYPE ,UNIT ;TYPE THE DRIVE NUMBER
3506 005100 104401 001165 TYPE ,SCLF ;CR-LF
3507 005104 004737 016002 JSR PC,LINE1 ;PRINT LINE 1 OF THE ERROR MESSAGE
3508 005110 104412 044403 DISPLY ,EM14 ;PRINT OFFLINE MESSAGE
3509 005114 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF THE ERROR MESSAGE
3510 005120 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF THE ERROR MESSAGE
3511 005124 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF THE ERROR MESSAGE
3512 005130 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3513 005134 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF THE ERROR MESSAGE
3514 005140 000137 023622 JMP DROP ;DROP THE UNIT
3515
3516 ;PORT REQUEST TIMEOUT ERROR
3517
3518 005144 004737 016002 PRTIM: JSR PC,LINE1 ;TYPE LINE 1 OF THE ERROR MESSAGE
3519 005150 104412 044425 DISPLY ,EM15 ;PRINT PORT TIME OUT MESSAGE
3520 005154 004737 016046 JSR PC,LINE2 ;TYPE LINE 2 OF THE ERROR MESSAGE
3521 005160 004737 016420 JSR PC,LINE3 ;TYPE LINE 3 OF THE ERROR MESSAGE
3522 005164 004737 017156 JSR PC,LINE4 ;TYPE LINE 4 OF THE ERROR MESSAGE

```

```

3523 005170 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3524 005174 004737 017622 JSR PC,LINE7 ;TYPE LINE 7 OF THE ERROR MESSAGE
3525 005200 000207 RTS PC ;RETURN
3526
3527 ;PROCESS ORDER COMPLETION WITH 'ERROR' & 'DONE' BITS SET
3528
3529 005202 032760 000030 000016 DONE: BIT #BIT04!BIT03,STATUS(RO) ;UNSAFE OCCURRED
3530 005210 001402 BEQ .+6 ;BR IF NOT
3531 005212 000137 010230 JMP UNSAF ;REPORT UNSAFE
3532 005216 032760 040000 000174 BIT #BIT14,$RHCS2(RO) ;IS 'WCE' SET ?
3533 005224 001402 BEQ .+6 ;BR IF NOT
3534 005226 000137 006146 JMP WCKER ;REPORT 'WCE'
3535 005232 032760 040000 000176 BIT #BIT14,$RHDS1(RO) ;CHECK 'ERR'
3536 005240 001002 BNE .+6 ;BR IF SET
3537 005242 000137 007774 JMP TRFER ;PROCESS 'TRE'
3538 005246 032760 000020 000200 BIT #BIT04,$RHER1(RO) ;'FMT' SET?
3539 005254 001402 BEQ .+6 ;BR IF NOT SET
3540 005256 000137 006624 JMP CKFMT ;CHECK FORMAT ERROR
3541 005262 032760 000200 000200 BIT #BIT07,$RHER1(RO) ;'HCE' SET?
3542 005270 001402 BEQ .+6 ;BR IF NOT SET
3543 005272 000137 007024 JMP CKHCE ;CHECK 'HCE' ERROR
3544 005276 032760 000400 000200 BIT #BIT08,$RHER1(RO) ;'HCRC' SET?
3545 005304 001402 BEQ .+6 ;BR IF NOT
3546 005306 000137 006442 JMP HRCRCR ;PROCESS 'HCRC'
3547 005312 032760 020000 000200 BIT #BIT13,$RHER1(RO) ;'OPI' SET?
3548 005320 001402 BEQ .+6 ;BR IF NOT SET
3549 005322 000137 007330 JMP OPIER ;REPORT 'OPI'
3550 005326 032760 000010 000200 BIT #BIT3,$RHER1(RO) ;'PAR' SET?
3551 005334 001402 BEQ .+6 ;BR IF NOT SET
3552 005336 000137 007462 JMP PARER ;REPORT 'PAR'
3553 005342 032760 000040 000200 BIT #BITS,$RHER1(RO) ;'WCF' SET?
3554 005350 001402 BEQ .+6 ;BR IF NOT SET
3555 005352 000137 010132 JMP WCFER ;REPORT 'WCF'
3556 005356 032760 002000 000200 BIT #BIT10,$RHER1(RO) ;'IAE' SET?
3557 005364 001402 BEQ .+6 ;BR IF NOT SET
3558 005366 000137 007554 JMP IAEER ;REPORT 'IAE'
3559 005372 032760 004000 000200 BIT #BIT11,$RHER1(RO) ;'WLE' SET?
3560 005400 001402 BEQ .+6 ;BR IF NOT SET
3561 005402 000137 007606 JMP WLEER ;REPORT 'WLE'
3562 005406 032760 001000 000200 BIT #BIT9,$RHER1(RO) ;'AOE' SET?
3563 005414 001405 BEQ 1$ ;BR IF NOT SET
3564 005416 032760 002000 000176 BIT #BIT10,$RHDS1(RO) ;'LST' SET?
3565 005424 001401 BEQ 1$ ;BR IF NOT SET
3566 005426 000207 RTS PC ;'AOE' & 'LST' SET, EXIT
3567 005430 032760 010000 000200 1$: BIT #BIT12,$RHER1(RO) ;SEE IF 'DTE' SET
3568 005436 001402 BEQ .+6 ;BR IF NOT
3569 005440 000137 007440 JMP DTEER ;REPORT 'DTE' ERROR
3570 005444 005760 000200 TST $RHER1(RO) ;SEE IF 'DCK' SET
3571 005450 100002 BPL .+6 ;BR IF NOT
3572 005452 000137 005476 JMP DCKER ;PROCESS 'DCK'
3573 005456 032760 140000 000226 BIT #BIT15!BIT14,$RHER3(RO) ;'SKI' OR 'OCYL' SET
3574 005464 001402 BEQ .+6 ;BR IF NOT SET
3575 005466 000137 010074 JMP SKIER ;REPORT ERROR
3576 005472 000137 006572 JMP DRIVER ;REPORT DRIVE ERROR
3577
3578 ;PROCESS DATA ('DCK') CHECK ERROR

```


3579								
3580	005476	004737	015646		DCKER:	JSR	PC, SPOTCK	;SEE IF ERROR AT A BAD SPOT ON THE PACK
3581	005502	000207				RTS	PC	;IT IS, DON'T REPORT IT
3582	005504	004737	016002			JSR	PC, LINE1	;PRINT LINE 1 OF ERROR MESSAGE
3583	005510	104412	044502			DISPLY	EM21	;DATA CHECK ERROR
3584	005514	004737	016046		DCKER1:	JSR	PC, LINE2	;PRINT LINE 2 OF ERROR MESSAGE
3585	005520	004737	016420			JSR	PC, LINE3	;PRINT LINE 3 OF ERROR MESSAGE
3586	005524	004737	017156			JSR	PC, LINE4	;PRINT LINE 4 OF ERROR MESSAGE
3587	005530	004737	012610			JSR	PC, PRIBAD	;SEE IF BAD SECTOR TO BE PRINTED
3588	005534	012760	110100	000024		MOV	#BIT15!BIT12!BIT06, \$MASK(RO)	;LOAD ERROR MASK
3589	005542	032760	010100	000200		BIT	#BIT12!BIT06, \$RHER1(RO)	;CHECK 'DTE' & 'ECH'
3590	005550	001003				BNE	1\$;BR IF SET
3591	005552	004737	017512			JSR	PC, LINE6	;PRINT LINE 6 OF ERROR MESSAGE
3592	005556	000530				BR	8\$;FINISH THE ERROR REPORT
3593	005560	012760	010000	000022	1\$:	MOV	#10000, \$RETRY(RO)	;RETRY COUNT
3594	005566	005001				CLR	R1	;R1 IS OFFSET CODE POINTER
3595	005570	004737	013644			JSR	PC, RELBUF	;RELEASE THE BUFFER
3596	005574	005046				CLR	-(SP)	;MAKE ROOM FOR THE SECTOR ADDRESS
3597	005576	005046				CLR	-(SP)	;MAKE ROOM FOR THE TRACK ADDRESS
3598	005600	004737	020124			JSR	PC, READDR	;GET THE ADDRESS OF THE ERROR SECTOR
3599	005604	112660	000011			MOVB	(SP)+, \$TRK(RO)	;TRACK ADDRESS OF ERROR SECTOR
3600	005610	112660	000010			MOVB	(SP)+, \$SEC(RO)	;SECTOR ADDRESS OF ERROR SECTOR
3601	005614	016060	000222	000012		MOV	\$RHCC(RO), \$CYL(RO)	;PRESENT CYLINDER
3602	005622	026060	000026	000020		CMP	\$SSEC(RO), \$WRDL(RO)	;SEE IF TRANSFER LENGTH LESS THAN 1 SECTOR
3603	005630	103010				BHIS	15\$;BR IF IT IS; USE PRESENT TRANSFER LENGTH
3604	005632	016060	000026	000020		MOV	\$SSEC(RO), \$WRDL(RO)	;CHANGE TRANSFER SIZE TO 1 SECTOR
3605	005640	016060	000020	000004		MOV	\$WRDL(RO), \$WRDM(RO)	;SETUP WORD COUNT FOR OPERATION
3606	005646	005460	000004			NEG	\$WRDM(RO)	;CHANGE COUNT TO 2'S COMP
3607	005652	005046			15\$:	CLR	-(SP)	;SPACE FOR NEW BUFFER ADDRESS
3608	005654	004737	013504			JSR	PC, GETBUF	;GET A BUFFER
3609	005660	012660	000006			MOV	(SP)+, \$BUF(RO)	;NEW BUFFER ADDRESS TO DPB
3610	005664	004737	014340		2\$:	JSR	PC, GODRIV	;RETRY
3611	005670	005760	000016		3\$:	TST	\$STATUS(RO)	;TEST FOR DONE
3612	005674	001775				BEQ	3\$;BR IF NOT DONE
3613	005676	100070				BPL	10\$;BR IF NOT ERROR
3614	005700	032760	000200	000016		BIT	#BIT7, \$STATUS(RO)	;SEE IF ORDER TERMINATED NORMALLY
3615	005706	001511				BEQ	14\$;BR IF NOT
3616	005710	036060	000024	000200		BIT	\$MASK(RO), \$RHER1(RO)	;LOOK AT CURRENT ERROR
3617	005716	001427				BEQ	5\$;BR IF DIFFERENT ERROR
3618	005720	032760	010100	000200		BIT	#BIT12!BIT6, \$RHER1(RO)	; 'ECH' OR 'DTE' STILL SET ?
3619	005726	001440				BEQ	7\$;BR IF NEITHER SET
3620	005730	105260	000022			INCB	\$RETRY(RO)	;INCREMENT RETRY COUNT
3621	005734	105360	000023			DECB	\$RETRY+1(RO)	;DECREMENT RETRY LIMIT
3622	005740	001351				BNE	2\$;BR IF MORE RETRY TO GO
3623	005742	005201				INC	R1	;INCREMENT TABLE INDEX
3624	005744	116137	042232	041527		MOVB	OFFCOD(R1), GENDPB+\$FMT	;OFFSET CODE
3625	005752	001437				BEQ	9\$;BR IF END OF OFFSET TABLE
3626	005754	112760	000002	000023		MOVB	#2, \$RETRY+1(RO)	;NEW RETRY LIMIT
3627	005762	004737	013040			JSR	PC, OFFST	;OFFSET
3628	005766	005737	041544		4\$:	TST	GENDPB+\$STATUS	;SEE IF FINISHED WITH OFFSET
3629	005772	001775				BEQ	4\$;BR IF NOT
3630	005774	100333				BPL	2\$;BR IF OFFSET OK
3631	005776	004737	020070		5\$:	JSR	PC, LINE8	;PRINT LINE 8 OF ERROR MESSAGE
3632	006002	004737	021262		6\$:	JSR	PC, INCHRD	;INCREMENT 'HARD' ERROR COUNT
3633	006006	004737	021356			JSR	PC, INCTOT	;INCREMENT TOTAL ERROR COUNT
3634	006012	004737	017622			JSR	PC, LINE7	;PRINT LINE 7 OF ERROR MESSAGE

3635	006016	004737	012610		JSR	PC,PRTBAD	:PRINT THE BAD SECTOR
3636	006022	004737	004310		JSR	PC,REFMT	:REFORMAT THE ERROR SECTOR
3637	006026	000207			RTS	PC	:RETURN
3638	006030	004737	017552	7\$:	JSR	PC,LINE6B	:PRINT LINE 6B OF ERROR MESSAGE
3639	006034	004737	017450		JSR	PC,LINE5B	:PRINT LINE 5B OF THE ERROR MESSAGE
3640	006040	004737	021236	8\$:	JSR	PC,INCSOF	:INCREMENT 'SOFT' ERROR COUNT
3641	006044	004737	012024		JSR	PC,ECC	:CORRECT THE ERROR USING ECC AND CHECK IT
3642	006050	000407			BR	11\$:COMPARE THE BUFFER
3643	006052	004737	017612	9\$:	JSR	PC,LINE6D	:PRINT LINE 6D OF ERROR MESSAGE
3644	006056	000751			BR	6\$:INCREMENT ERROR COUNT
3645	006060	004737	017524	10\$:	JSR	PC,LINE6A	:PRINT LINE 6A OF ERROR MESSAGE
3646	006064	004737	021236		JSR	PC,INCSOF	:INCREMENT 'SOFT' ERROR COUNT
3647	006070	112737	000001	011407 11\$:	MOVB	#1,FRSTER	:SET 'DCKER' INDICATOR
3648	006076	004737	010710		JSR	PC,CMPARD	:COMPARE THE BUFFER
3649	006102	005737	011414		TST	ERCTR	:SEE IF COMPARE ERROR DETECTED
3650	006106	001006			BNE	13\$:BR IF ONE DID
3651	006110	004737	021356		JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3652	006114	104412	050751		DISPLY	LIN9G	: 'DATA COMPARE OK' MESSAGE
3653	006120	004737	017622	12\$:	JSR	PC,LINE7	:PRINT LINE 7 OF ERROR MESSAGE
3654	006124	004737	004310	13\$:	JSR	PC,REFMT	:REFORMAT THE ERROR SECTOR
3655	006130	000207			RTS	PC	:RETURN
3656	006132	004737	021356	14\$:	JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3657	006136	104412	050537		DISPLY	LIN8M	: 'DIFFERENT ERROR DURING RETRY'
3658	006142	000137	004560		JMP	ERPRC1	:SEE WHICH ERROR
3659							
3660							
3661							
3662	006146	004737	016002		JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
3663	006152	032760	100000	000200	WCKER: BIT	#BIT15,\$RHER1(RO)	:SEE IF 'DCK' SET ALSO
3664	006160	001032			BNE	2\$:BR IF IT IS
3665	006162	104412	044606		DISPLY	EM23	:PRINT WCE & DCK NOT
3666	006166	005060	000024		CLR	\$MASK(RO)	:CLEAR ERROR MASK
3667	006172	004737	016046		JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
3668	006176	004737	016420		JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
3669	006202	004737	017156		JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
3670	006206	004737	017256		JSR	PC,LINE5	:PRINT LINE 5 OF ERROR MESSAGE
3671	006212	004737	021356		JSR	PC,INCTOT	:INCREMENT TOTAL ERROR COUNT
3672	006216	012760	001400	000022	MOV	#1400,\$RETRY(RO)	:RETRY LIMIT
3673	006224	004737	013146		JSR	PC,RETRY	:RETRY THE OPERATION
3674	006230	000403			BR	1\$:RETRY UNSUCCESSFUL
3675	006232	004737	017562		JSR	PC,LINE6C	:PRINT LINE 6C OF ERROR MESSAGE
3676	006236	000465			BR	8\$:FINISH PROCESSING THE ERROR
3677	006240	004737	017612	1\$:	JSR	PC,LINE6D	:PRINT LINE 6D OF ERROR MESSAGE
3678	006244	000471			BR	10\$:FINISH PROCESSING THE ERROR
3679	006246	004737	015646	2\$:	JSR	PC,SPOTCK	:SEE IF ERROR AT BAD SPOT ON THE PACK
3680	006252	000207			RTS	PC	:BR IF IT IS
3681	006254	104412	044533		DISPLY	EM22	:WCK & DCK ERRS
3682	006260	004737	016046		JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
3683	006264	004737	016420		JSR	PC,LINE3	:PRINT LINE 3 OF ERROR MESSAGE
3684	006270	004737	017156		JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
3685	006274	004737	017256		JSR	PC,LINE5	:PRINT LINE 5 OF ERROR MESSAGE
3686	006300	032760	000100	000200	BIT	#BIT06,\$RHER1(RO)	:ECH SET ALSO ?
3687	006306	001441			BEQ	8\$:FINISH PROCESSING THE ERROR
3688	006310	012760	010000	000022	3\$:	MOV	#10000,\$RETRY(RO)
3689	006316	004737	014340	4\$:	JSR	PC,GODRIV	:RETRY THE ORDER
3690	006322	005760	000016	5\$:	TST	\$STATUS(RO)	:ORDER FINISHED ?

```

3691 006326 001775      BEQ      5$      ;BR IF NOT
3692 006330 100405      BMI      6$      ;BR IF ERROR ON ORDER
3693 006332 105260 000022 INCB     $RETRY(RO) ;INCREMENT RETRY COUNT
3694 006336 004737 017562 JSR      PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3695 006342 000430      BR       9$      ;FINISH ERROR PROCESSING
3696 006344 105260 000022 6$: INCB     $RETRY(RO) ;INCREMENT RETRY COUNT
3697 006350 105360 000023 DECB     $RETRY+1(RO) ;DECREMENT RETRY LIMIT
3698 006354 001731      BEQ      1$      ;BR IF AT RETRY LIMIT
3699 006356 032760 100000 000200 BIT      #BIT15,$RHER1(RO) ;'DCK' SET
3700 006364 001407      BEQ      7$      ;BR IF NOT - DIFFERENT ERROR
3701 006366 032760 000100 000200 BIT      #BIT06,$RHER1(RO) ;'ECH' ALSO SET ?
3702 006374 001350      BNE     4$      ;BR IF IT IS; RETRY ORDER
3703 006376 004737 017562 JSR      PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3704 006402 000403      BR       8$      ;FINISH PROCESSING ERROR
3705 006404 004737 020070 7$: JSR      PC,LINE8 ;PRINT LINE 8 - 'DIFFERENT ERROR'
3706 006410 000405      BR       9$      ;FINISH PROCESSING ERROR
3707 006412 004737 021356 8$: JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3708 006416 004737 017622 JSR      PC,LINE7 ;FINISH THE ERROR MESSAGE
3709 006422 000207      RTS      PC      ;RETURN
3710 006424 004737 021356 9$: JSR      PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3711 006430 004737 017622 10$: JSR     PC,LINE7 ;FINISH THE ERROR MESSAGE
3712 006434 004737 004310 JSR     PC,REFMT ;REFORMAT THE SECTOR IN ERROR
3713 006440 000207      RTS      PC      ;RETURN
3714
3715 ;REPORT 'HCRC' ERROR
3716
3717 006442 004737 015646 HRCRCR: JSR     PC,SPOTCK ;SEE IF ERROR AT PACK BAD SPOT
3718 006446 000207      RTS      PC      ;RETURN IF IT IS
3719 006450 004737 016002 JSR     PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3720 006454 104412 044461 DISPLY   EM20 ;REPORT 'HCRC'
3721 006460 004737 016046 JSR     PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3722 006464 004737 016420 JSR     PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3723 006470 004737 017156 JSR     PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3724 006474 032760 040000 000174 BIT      #BIT14,$RHCS2(RO) ;'WCE' ERROR ALSO ?
3725 006502 001402      BEQ     +6 ;BR IF NOT
3726 006504 004737 017256 JSR     PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3727 006510 004737 021236 JSR     PC,INCSOF ;INCREMENT 'SOFT' ERROR COUNT
3728 006514 004737 021356 JSR     PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3729 006520 012760 000400 000024 MOV     #BIT8,$MASK(RO) ;SET ERROR MASK
3730 006526 012760 001400 000022 MOV     #1400,$RETRY(RO) ;RETRY LIMIT
3731 006534 004737 013146 JSR     PC,RETRY ;RETRY ORDER
3732 006540 000405      BR       1$      ;RETRY NOT SUCCESSFUL
3733 006542 004737 017562 JSR     PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3734 006546 004737 017622 JSR     PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3735 006552 000207      RTS      PC      ;RETURN
3736 006554 004737 017612 1$: JSR     PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3737 006560 004737 017622 JSR     PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3738 006564 004737 004310 JSR     PC,REFMT ;REFORMAT THE ERROR SECTOR
3739 006570 000207      RTS      PC      ;RETURN
3740
3741 ;REPORT DRIVE ERROR
3742
3743 006572 004737 016002 DRVER: JSR     PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3744 006576 104412 045076 DISPLY   EM30 ;REPORT DRIVE ERROR
3745 006602 004737 016046 JSR     PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3746 006606 004737 016420 JSR     PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE

```

```

3747 006612 004737 021356      JSR    PC,INCTOT      ;INCREMENT TOTAL ERROR COUNT
3748 006616 004737 017622      JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3749 006622 000207              RTS     PC            ;RETURN
3750
3751                ;PROCESS FORMAT ('FMT') ERROR
3752
3753 006624 032760 000400 000016 CKFMT: BIT    #BIT8,$STATUS(RO) ;'HCRC' SET ON ORIGINAL ERROR ?
3754 006632 001402              BEQ    .+6           ;BR IF NOT SET
3755 006634 000137 006442              JMP    HRCRCER       ;REPORT HCRC ERROR
3756 006640 005046              CLR    -(SP)         ;CLEAR STACK FOR DECREMENTED
3757 006642 005046              CLR    -(SP)         ;SECTOR AND TRACK
3758 006644 004737 020124              JSR    PC,READDR     ;GET CORRECTED TRACK & SECTOR ADDRSSES
3759 006650 004737 013064              JSR    PC,READHD     ;READ HEADER
3760 006654 032737 000400 041562      BIT    #BIT8,GENREG+RHER1 ;'HCRC' SET WHEN HEADER READ?
3761 006662 001002              BNE    .+6           ;BR IF 'HCRC' SET
3762 006664 000137 007634              JMP    FMTER         ;NO ERROR IS 'FMT' ONLY
3763 006670 004737 015646      1$: JSR    PC,SPOTCK     ;SEE IF ERROR AT BAD SPOT ON THE PACK
3764 006674 000207              RTS     PC            ;RETURN IF IT IS
3765 006676 004737 016002              JSR    PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3766 006702 104412 044665              DISPLY .EM24         ;HEADER READ ERROR - FMT BIT DROPPED UP
3767 006706 004737 016046              JSR    PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3768 006712 004737 016420              JSR    PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3769 006716 004737 017156              JSR    PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3770 006722 032760 040000 000174      BIT    #BIT14,$RHCS2(RO) ;'WCE' ERROR ALSO ?
3771 006730 001402              BEQ    .+6           ;BR IF NOT
3772 006732 004737 017256              JSR    PC,LINES      ;DISPLAY WORDS WHICH CAUSED 'WCE'
3773 006736 004737 017356              JSR    PC,LINESA     ;DISPLAY HEADER
3774 006742 004737 021236              JSR    PC,INCSOF     ;INCREMENT SOFT ERROR COUNT
3775 006746 004737 021356              JSR    PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3776 006752 012760 000020 000024      MOV    #BIT4,$MASK(RO) ;SET ERROR MASK
3777 006760 012760 001400 000022      MOV    #1400,$RETRY(RO) ;RETRY LIMIT
3778 006766 004737 013146              JSR    PC,RETRY      ;RETRY THE ORDER
3779 006772 000405              BR     2$            ;RETRY NOT SUCCESSFUL
3780 006774 004737 017562              JSR    PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
3781 007000 004737 017622              JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3782 007004 000207              RTS     PC            ;RETURN
3783 007006 004737 017612      2$: JSR    PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
3784 007012 004737 017622              JSR    PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
3785 007016 004737 004310              JSR    PC,REFMT      ;REFORMAT THE ERROR SECTOR
3786 007022 000207              RTS     PC            ;RETURN
3787
3788                ;PROCESS HEADER COMPARE ('HCE') ERROR
3789
3790 007024 032760 000400 000200 CKHCE: BIT    #BIT8,$RHER1(RO) ;HCRC SET ON ORIGINAL ERROR ?
3791 007032 001402              BEQ    .+6           ;BR IF NOT SET
3792 007034 000137 006442              JMP    HRCRCER       ;REPORT HEADER CRC ERROR
3793 007040 005046      1$: CLR    -(SP)         ;CLEAR STACK
3794 007042 005046              CLR    -(SP)         ;CLEAR STACK
3795 007044 004737 020124              JSR    PC,READDR     ;GET CURRENT SECTOR & TRACK ADDR
3796 007050 004737 013064              JSR    PC,READHD     ;READ HEADER OF CURRENT SECTOR
3797 007054 032737 000400 041562      BIT    #BIT8,GENREG+RHER1 ;'HCRC' SET
3798 007062 001016              BNE    2$            ;BR IF SET
3799 007064 042737 010000 053644      BIC    #BIT12,CYLDER ;CLEAR FORMAT BIT FROM HEADER
3800 007072 026037 000222 053644      CMP    $RHCC(RO),CYLDER ;CORRECT CYLINDER ?
3801 007100 001402              BEQ    .+6           ;BR IF IT IS
3802 007102 000137 007254              JMP    POSER         ;REPORT POSITIONING ERROR

```

```

3803 007106 052737 010000 053644 BIS #BIT12,CYLDER ;RESTORE THE FORMAT BIT
3804 007114 000137 007712 JMP HCEER ;REPORT 'HCE' ERROR
3805 007120 004737 015646 2$: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
3806 007124 000207 RTS PC ;RETURN IF IT IS
3807 007126 004737 016002 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3808 007132 104412 044733 DISPLY EM25 ;HEADER READ ERROR - 'HCE' SET
3809 007136 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3810 007142 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3811 007146 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3812 007152 032760 040000 000174 BIT #BIT14,$RHCS2(RO) ;'WCE' ERROR ALSO ?
3813 007160 001402 BEQ +6 ;BR IF NOT
3814 007162 004737 017256 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3815 007166 004737 017356 JSR PC,LINESA ;PRINT LINE 5 OF ERROR MESSAGE
3816 007172 004737 021236 JSR PC,INCSOF ;INCREMENT SOFT ERROR COUNT
3817 007176 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3818 007202 012760 000200 000024 MOV #BIT7,$MASK(RO) ;SET ERROR MASK
3819 007210 012760 001400 000022 MOV #1400,$RETRY(RO) ;RETRY LIMIT
3820 007216 004737 013146 JSR PC,RETRY ;RETRY THE ORDER
3821 007222 000405 BR 3$ ;RETRY NOT SUCESSFUL
3822 007224 004737 017562 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3823 007230 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3824 007234 000207 RTS PC ;RETURN
3825 007236 004737 017612 3$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3826 007242 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3827 007246 004737 004310 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
3828 007252 000207 RTS PC ;RETURN
3829
3830 ;REPORT POSSIBLE POSITIONING ERROR
3831
3832 007254 004737 013006 POSER: JSR PC,RECALT ;RECALIBRATE
3833 007260 004737 016002 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3834 007264 104412 046012 DISPLY EM51 ;PROGRAM DETECTED POSITIONING ERROR
3835 007270 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3836 007274 004737 016612 JSR PC,LINE3C ;PRINT LINE 3C OF ERROR MESSAGE
3837 007300 052737 010000 053644 BIS #BIT12,CYLDER ;RESTORE THE FORMAT BIT
3838 007306 004737 017356 JSR PC,LINESA ;PRINT LINE 5A OF THE ERROR MESSAGE
3839 007312 004737 021332 JSR PC,INCMIS ;INCREMENT MISPOSITIONING COUNT
3840 007316 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3841 007322 004737 017750 JSR PC,LINE7A ;PRINT LINE 7A OF ERROR MESSAGE
3842 007326 000207 RTS PC ;EXIT
3843
3844 ;REPORT 'OPI' ERROR
3845
3846 007330 004737 015646 OPIER: JSR PC,SPOTCK ;SEE IF ERROR AT BAD SPOT
3847 007334 000207 RTS PC ;RETURN IF IT IS
3848 007336 004737 016002 JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3849 007342 104412 045130 DISPLY EM31 ;'OPI' ERROR
3850 007346 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3851 007352 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3852 007356 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3853 007362 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3854 007366 012760 020000 000024 MOV #BIT13,$MASK(RO) ;ERROR MASK
3855 007374 012760 001400 000022 OPIER1: MOV #1400,$RETRY(RO) ;RETRY LIMIT
3856 007402 004737 013146 JSR PC,RETRY ;RETRY THE ORDER
3857 007406 000405 BR 1$ ;RETRY UNSUCCESSFUL
3858 007410 004737 017562 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE

```

3859	007414	004737	017622		JSR	PC,LINE7		;PRINT LINE 7 OF ERROR MESSAGE
3860	007420	000207			RTS	PC		;EXIT
3861	007422	004737	017612	1\$:	JSR	PC,LINE6D		;PRINT LINE 6D OF ERROR MESSAGE
3862	007426	004737	017622		JSR	PC,LINE7		;PRINT LINE 7 OF ERROR MESSAGE
3863	007432	004737	004310		JSR	PC,REFMT		;REFORMAT THE ERROR SECTOR
3864	007436	000207			RTS	PC		;RETURN
3865								
3866								
3867								
3868	007440	004737	015646					
3869	007444	000207		DTEER:	JSR	PC,SPOTCK		;SEE IF ERROR AT BAD SPOT
3870	007446	004737	016002		RTS	PC		;RETURN IF IT IS
3871	007452	104412	045173		JSR	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
3872	007456	000137	005514		DISPLY	EM32		; 'DTE' ERROR
3873					JMP	DCKER1		;FINISH PROCESSING THE 'DTE' ERROR
3874								
3875								
3876	007462	004737	016002					
3877	007466	104412	045226		PARER:	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
3878	007472	004737	016046		DISPLY	EM33		;REPORT 'PAR'
3879	007476	004737	017002		JSR	PC,LINE2		;PRINT LINE 2 OF ERROR MESSAGE
3880	007502	004737	017156		JSR	PC,LINE3E		;PRINT LINE 3E OF ERROR MESSAGE
3881	007506	004737	021356		JSR	PC,LINE4		;PRINT LINE 4 OF ERROR MESSAGE
3882	007512	012760	000010	000024	JSR	PC,INCTOT		;INCREMENT TOTAL ERROR COUNT
3883	007520	012760	001400	000022	MOV	#BIT03,\$MASK(RO)		;ERROR MASK
3884	007526	004737	013146		MOV	#1400,\$RETRY(RO)		;RETRY LIMIT
3885	007532	000405			JSR	PC,RETRY		;RETRY ORDER
3886	007534	004737	017562		BR	2\$;RETRY UNSUCCESSFUL
3887	007540	004737	017622		JSR	PC,LINE6C		;RETRY SUCCESSFUL
3888	007544	000207		1\$:	JSR	PC,LINE7		;PRINT LINE 7 OF ERROR MESSAGE
3889	007546	004737	017612		RTS	PC		;EXIT
3890	007552	000772		2\$:	JSR	PC,LINE6D		;PRINT LINE 6D OF ERROR MESSAGE
3891					BR	1\$;FINISH ERROR MESSAGE
3892								
3893								
3894	007554	004737	016002					
3895	007560	104412	045345		;REPORT 'IAE' ERROR			
3896	007564	004737	016046		IAEER:	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
3897	007570	004737	017104		DISPLY	EM35		;REPORT 'IAE'
3898	007574	004737	021356		JSR	PC,LINE2		;PRINT LINE 2 OF ERROR MESSAGE
3899	007600	004737	017622		JSR	PC,LINE3F		;PRINT LINE 3F OF ERROR MESSAGE
3900	007604	000207			JSR	PC,INCTOT		;INCREMENT TOTAL ERROR COUNT
3901					JSR	PC,LINE7		;PRINT LINE 7 OF ERROR MESSAGE
3902					RTS	PC		;RETURN
3903								
3904	007606	004737	016002					
3905	007612	104412	045403		;REPORT 'WLE' ERROR			
3906	007616	004737	016046		WLEER:	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
3907	007622	004737	021356		DISPLY	EM36		;REPORT 'WLE'
3908	007626	004737	017622		JSR	PC,LINE2		;PRINT LINE 2 OF ERROR MESSAGE
3909	007632	000207			JSR	PC,INCTOT		;INCREMENT TOTAL ERROR COUNT
3910					JSR	PC,LINE7		;PRINT LINE 7 OF ERROR MESSAGE
3911					RTS	PC		;RETURN
3912								
3913	007634	004737	016002					
3914	007640	104412	045014		;REPORT FORMAT ERROR			
					FMTER:	PC,LINE1		;PRINT LINE 1 OF ERROR MESSAGE
					DISPLY	EM26		;FORMAT ERROR

```

3915 007644 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3916 007650 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3917 007654 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3918 007660 032760 040000 000174 BIT #BIT14,$RHCS2(RO) ;'WCE' ERROR ALSO ?
3919 007666 001402 BEQ +6 ;BR IF NOT
3920 007670 004737 017256 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3921 007674 004737 017356 JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
3922 007700 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3923 007704 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3924 007710 000207 RTS PC
3925
3926 ;REPORT HEADER COMPARE ERROR
3927
3928 007712 004737 016002 HCEER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3929 007716 104412 045041 DISPLY EM27 ;HEADER COMPARE ERROR
3930 007722 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3931 007726 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3932 007732 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3933 007736 032760 040000 000174 BIT #BIT14,$RHCS2(RO) ;'WCE' ERROR ALSO ?
3934 007744 001402 BEQ +6 ;BR IF NOT
3935 007746 004737 017256 JSR PC,LINES ;DISPLAY WORDS WHICH CAUSED 'WCE'
3936 007752 004737 017356 JSR PC,LINE5A ;PRINT LINE 5A OF ERROR MESSAGE
3937 007756 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3938 007762 004737 017622 JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3939 007766 004737 004310 JSR PC,REFMT ;REFORMAT THE ERROR SECTOR
3940 007772 000207 RTS PC ;RETURN
3941
3942 ;PROCESS CONTROL/INTERFACE TRANSFER ERROR
3943
3944 007774 004737 016002 TRFER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3945 010000 104412 045434 DISPLY EM40 ;RH11 OR UNIBUS TRANSFER ERROR
3946 010004 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3947 010010 004737 016420 JSR PC,LINE3 ;PRINT LINE 3 OF ERROR MESSAGE
3948 010014 004737 017156 JSR PC,LINE4 ;PRINT LINE 4 OF ERROR MESSAGE
3949 010020 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3950 010024 032760 121400 000174 BIT #BIT15:BIT13:BIT9:BIT8,$RHCS2(RO) ;'DLT','UPE','MXF','MDPE' SET ?
3951 010032 001415 BEQ 2$ ;BR IF NONE SET
3952 010034 012760 001400 000022 MOV #1400,$RETRY(RO) ;RETRY LIMIT
3953 010042 005060 000024 CLR $MASK(RO) ;CLEAR ERROR MASK
3954 010046 004737 013146 JSR PC,RETRY ;RETRY THE OPERATION
3955 010052 000403 BR 1$ ;RETURN HERE IF RETRY UNSUCCESSFUL
3956 010054 004737 017562 JSR PC,LINE6C ;PRINT LINE 6C OF ERROR MESSAGE
3957 010060 000402 BR 2$ ;FINISH THE ERROR REPORT
3958 010062 004737 017612 1$: JSR PC,LINE6D ;PRINT LINE 6D OF ERROR MESSAGE
3959 010066 004737 017622 2$: JSR PC,LINE7 ;PRINT LINE 7 OF ERROR MESSAGE
3960 010072 000207 RTS PC
3961
3962 ;PROCESS 'SKI' OR 'OCYL' ERRORS
3963
3964 010074 004737 016002 SKIER: JSR PC,LINE1 ;PRINT LINE 1 OF ERROR MESSAGE
3965 010100 104412 045744 DISPLY EM50 ;'SKI' OR 'OCYL' ERROR
3966 010104 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
3967 010110 004737 016600 JSR PC,LINE3B ;PRINT LINE 3B OF ERROR MESSAGE
3968 010114 004737 021356 JSR PC,INCTOT ;INCREMENT TOTAL ERROR COUNT
3969 010120 004737 021306 JSR PC,INCSKI ;INCREMENT 'SKI' OR 'OCYL' ERROR COUNT
3970 010124 004737 017750 JSR PC,LINE7A ;PRINT LINE 7A OF ERROR MESSAGE

```

```

3971 010130 000207          RTS      PC
3972
3973          ;REPORT WRITE CLOCK FAILURE ('WCF')
3974
3975 010132 004737 016002    WCFER: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3976 010136 104412 045303    DISPLY   ,EM34          ;REPORT WRITE CLOCK FAILURE
3977 010142 004737 016046    JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3978 010146 004737 016426    JSR      PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
3979 010152 004737 017156    JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
3980 010156 004737 021356    JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3981 010162 004737 012610    JSR      PC,PRTBAD     ;SEE IF BAD SECTOR TO BE PRINTED
3982 010166 012760 001400    MOV      #1400,$RETRY(RO) ;RETRY COUNT
3983 010174 012760 000040    MOV      #BIT05,$MASK(RO) ;ERROR MASK
3984 010202 004737 013146    JSR      PC,RETRY      ;RETRY THE ORDER
3985 010206 000405          BR       2$            ;RETURN HERE IF RETRY UNSUCCESSFUL
3986 010210 004737 017562    JSR      PC,LINE6C     ;PRINT LINE 6C OF ERROR MESSAGE
3987 010214 004737 017622    1$:     JSR      PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
3988 010220 000207          RTS      PC
3989 010222 004737 017612    2$:     JSR      PC,LINE6D     ;PRINT LINE 6D OF ERROR MESSAGE
3990 010226 000772          BR       1$
3991
3992          ;PROCESS DRIVE UNSAFE ERROR
3993
3994 010230 004737 016002    UNSAF: JSR      PC,LINE1      ;PRINT LINE 1 OF ERROR MESSAGE
3995 010234 104412 046055    DISPLY   ,EM60          ;REPORT DRIVE UNSAFE
3996 010240 004737 016046    JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
3997 010244 004737 016420    JSR      PC,LINE3      ;PRINT LINE 3 OF ERROR MESSAGE
3998 010250 004737 021356    JSR      PC,INCTOT     ;INCREMENT TOTAL ERROR COUNT
3999 010254 004737 017622    JSR      PC,LINE7      ;PRINT LINE 7 OF ERROR MESSAGE
4000 010260 000207          RTS      PC
4001
4002          ;REPORT AN 'UNKNOWN' DATA PATTERN
4003
4004 010262 105737 011407    NOMTCH: TSTB     FRSTER          ;FIRST ERROR IN THE SECTOR ?
4005 010266 001013          BNE     1$            ;BR IF NOT OR IF PROCESSING 'DCKER'
4006 010270 004737 016002    JSR      PC,LINE1      ;TYPE LINE 1 OF ERROR MESSAGE
4007 010274 104412 045613    DISPLY   ,EM43          ;'CAN'T MATCH DATA WITH PATTERN'
4008 010300 004737 016046    JSR      PC,LINE2      ;PRINT LINE 2 OF ERROR MESSAGE
4009 010304 004737 016426    JSR      PC,LINE3A     ;PRINT LINE 3A OF ERROR MESSAGE
4010 010310 004737 017156    JSR      PC,LINE4      ;PRINT LINE 4 OF ERROR MESSAGE
4011 010314 000404          BR       2$            ;CONTINUE PROCESSING ERROR
4012 010316 104412 045613    1$:     DISPLY   ,EM43          ;'CAN'T MATCH DATA WITH PATTERN'
4013 010322 104412 001165    DISPLY   ,$CRLF         ;CR-LF
4014 010326 104412 050701    2$:     DISPLY   ,LIN9I       ;HEADER FOR DATA PRINTOUT
4015 010332 010146          MOV      R1,-(SP)      ;ADDRESS OF WORD 1
4016 010334 004737 024740    JSR      PC,PRT0CT     ;PRINT IT
4017 010340 104412 051364    DISPLY   ,LINSF         ;SPACES
4018 010344 012146          MOV      (R1)+,-(SP)   ;WORD 1
4019 010346 004737 024740    JSR      PC,PRT0CT     ;PRINT IT
4020 010352 104412 001165    DISPLY   ,$CRLF         ;CR-LF
4021 010356 010146          MOV      R1,-(SP)      ;ADDRESS OF WORD 2
4022 010360 004737 024740    JSR      PC,PRT0CT     ;PRINT IT
4023 010364 104412 051364    DISPLY   ,LINSF         ;SPACES
4024 010370 012146          MOV      (R1)+,-(SP)   ;WORD 2
4025 010372 004737 024740    JSR      PC,PRT0CT     ;PRINT IT
4026 010376 104412 001165    DISPLY   ,$CRLF         ;CR-LF

```



```

4027 010402 010146          MOV    R1,-(SP)          ;ADDRESS OF WORD 3
4028 010404 004737 024740  JSR    PC,PRTCT        ;PRINT IT
4029 010410 104412 051364  DISPLY ,LINSF          ;SPACES
4030 010414 012146          MOV    (R1)+,-(SP)     ;WORD 3
4031 010416 004737 024740  JSR    PC,PRTCT        ;PRINT IT
4032 010422 104412 001165  DISPLY ,$CRLF          ;CR-LF
4033 010426 010146          MOV    R1,-(SP)          ;ADDRESS OF WORD 4
4034 010430 004737 024740  JSR    PC,PRTCT        ;PRINT IT
4035 010434 104412 051364  DISPLY ,LINSF          ;SPACES
4036 010440 012146          MOV    (R1)+,-(SP)     ;WORD 4
4037 010442 004737 024740  JSR    PC,PRTCT        ;PRINT IT
4038 010446 104412 001165  DISPLY , $CRLF          ;CR-LF
4039 010452 062701 000770  ADD    #770,R1         ;INCREMENT BUFFER POINTER
4040 010456 060237 011414  ADD    R2,ERCTR        ;ADD SECTOR (OR LESS) COUNT TO THE
4041                                     ;COMPARISON ERROR COUNT
4042 010462 005002          CLR    R2              ;CLEAR 'WORDS TO COMPARE' COUNT IN R2
4043 010464 112737 177777 011407  MOVB   #-1,FRSTER     ;SET 'NOT FIRST ERROR' INDICATOR
4044 010472 000207          RTS    PC              ;RETURN
4045
4046                                     ;CHECK ERROR BITS IN THE RH11 & RPO4 REGISTERS
4047
4048 010474 032760 060000 000164  CKERR: BIT    #60000,$RHCS1(RO) ;SEE IF 'TRE' OR 'MCPE' SET
4049 010502 001015          BNE    1$             ;BR IF EITHER SET
4050 010504 032760 177400 000174  BIT    #177400,$RHCS2(RO) ;SEE IF ERROR BITS IN CS2 SET
4051 010512 001011          BNE    1$             ;BR IF ANY SET
4052 010514 005760 000200  TST    $RHER1(RO)     ;ANY BITS SET IN ER1
4053 010520 001006          BNE    1$             ;BR IF ANY SET
4054 010522 005760 000224  TST    $RHER2(RO)     ;ANY BITS SET IN ER2 ?
4055 010526 001003          BNE    1$             ;BR IF ANY SET
4056 010530 005760 000226  TST    $RHER3(RO)     ;ANY BITS SET IN ER3 ?
4057 010534 001416          BEQ    2$             ;BR IF NONE SET
4058 010536 004737 016002  1$:   JSR    PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
4059 010542 104412 045660  DISPLY  EM44          ;ERROR BITS SET, BUT 'SC' OR 'TRE' NOT SET
4060 010546 004737 016046  JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
4061 010552 004737 016420  JSR    PC,LINE3     ;PRINT LINE 3 OF ERROR MESSAGE
4062 010556 004737 017156  JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
4063 010562 004737 021356  JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
4064 010566 004737 017622  JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE
4065 010572 000207  2$:   RTS    PC              ;RETURN
4066
4067                                     ;CHECK BUS ADDRESS REGISTER & WORD COUNT REGISTER
4068
4069 010574 005760 000166  CKBUS: TST    $RHWC(RO) ;CHECK WORD COUNT
4070 010600 001010          BNE    1$             ;BR IF NOT ZERO
4071 010602 016046 000020  MOV    $WRDL(RO),-(SP) ;WORD LENGTH
4072 010606 006316          ASL    (SP)           ;CHANGE INTO BYTE COUNT
4073 010610 066016 000006  ADD    $BUF(RO),(SP)  ;ADD THE STARTING LOCATION
4074 010614 022660 000170  CMP    (SP)+,$RABA(RO) ;BUFFER ADDRESS PROPER ?
4075 010620 001416          BEQ    2$             ;BR IF OK
4076 010622 004737 016002  1$:   JSR    PC,LINE1     ;PRINT LINE 1 OF ERROR MESSAGE
4077 010626 104412 045472  DISPLY  EM41          ;BUS ADDRESS OR WORD COUNT INCORRECT
4078 010632 004737 016046  JSR    PC,LINE2     ;PRINT LINE 2 OF ERROR MESSAGE
4079 010636 004737 016734  JSR    PC,LINE3D    ;PRINT LINE 3D OF ERROR MESSAGE
4080 010642 004737 017156  JSR    PC,LINE4     ;PRINT LINE 4 OF ERROR MESSAGE
4081 010646 004737 021356  JSR    PC,INCTOT    ;INCREMENT TOTAL ERROR COUNT
4082 010652 004737 017622  JSR    PC,LINE7     ;PRINT LINE 7 OF ERROR MESSAGE

```

4083	010656	000207			2\$:	RTS	PC	
4084								
4085								;COMPARE THE BUFFER
4086								
4087	010660	132760	000004	000030	CMPAR:	BITB	#BIT02,\$CODE(R0)	;SEE IF READ ORDER
4088	010666	001001				BNE	.+4	;BR IF IT IS
4089	010670	000207				RTS	PC	;RETURN
4090	010672	032737	000002	001140		BIT	#SW01,SWR	;IS SWITCH 1 SET?
4091	010700	001401				BEQ	.+4	;BR IF NOT
4092	010702	000207				RTS	PC	;YES, DON'T COMPARE
4093	010704	105037	011407			CLRB	FRSTER	;CLEAR 'FIRST ERROR' INDICATOR
4094	010710	005037	011414		CMPARD:	CLR	ERCTR	;CLEAR THE ERROR COUNTER
4095	010714	016001	000006			MOV	\$BUF(R0),R1	;BUFFER ADDRESS
4096	010720	016037	000020	011420		MOV	\$WRDL(R0),CMCNT	;WORD COUNT TO WORKING LOCATION
4097	010726	066037	000166	011420		ADD	\$RHC(R0),CMCNT	;CALCULATE ACTUAL WORDS TRANSFERED
4098	010734	016037	000012	011422		MOV	\$CYL(R0),CMCYL	;CYLINDER ADDRESS WORKING LOCATION
4099	010742	052737	010000	011422		BIS	#BIT12,CMCYL	;SET FORMAT BIT
4100	010750	016037	000010	011424		MOV	\$SEC(R0),CMSEC	;SECTOR & TRACK ADDRESSES TO WORKING LOCNS
4101	010756	013737	001306	011416		MOV	CMPLMT,LIMIT	;DISPLAY LIMIT
4102	010764	005237	011416			INC	LIMIT	;CONVERT PARAMETER INTO LIMIT VALUE
4103	010770	112737	177777	011406	CMSTR:	MOVB	#-1,ZROIND	;CLEAR THE 'ZERO'S' INDICATOR
4104	010776	005037	011410			CLR	SAVER1	;CLEAR THE R1 SAVE WORD
4105	011002	005037	011412			CLR	SAVER5	;CLEAR THE R5 SAVE WORD
4106	011006	023760	011420	000026		CMP	CMCNT,\$SSEC(R0)	;IS BUFFER SIZE GREATER THAN ONE SECTOR ?
4107	011014	101003				BHI	.+10	;--> BR IF IT IS
4108	011016	013702	011420			MOV	CMCNT,R2	; I LESS THAN, USE REMAINING BUFFER
4109	011022	000402				BR	.+6	;--I-->
4110	011024	016002	000026			MOV	\$SSEC(R0),R2	<-- I COMPARE SECTOR
4111	011030	166037	000026	011420		SUB	\$SSEC(R0),CMCNT	<---- DECREMENT WORD COUNT
4112	011036	126027	000030	000005		CMPB	\$CODE(R0),#5	;READ HEADER & DATA?
4113	011044	001036				BNE	CMDAT	;BR IF NOT
4114	011046	023721	011422		CMHED:	CMP	CMCYL,(R1)+	;CHECK CYLINDER
4115	011052	001402				BEQ	.+6	;BR IF COMPARE OK
4116	011054	004737	011130			JSR	PC,1\$;REPORT ERROR
4117	011060	023721	011424			CMP	CMSEC,(R1)+	;COMPARE SECTOR & TRACK
4118	011064	001402				BEQ	.+6	;BR IF EQ
4119	011066	004737	011130			JSR	PC,1\$;REPORT ERROR
4120	011072	005721				TST	(R1)+	;1ST KEY WORD ZERO?
4121	011074	001402				BEQ	.+6	;BR IF IT IS
4122	011076	004737	011130			JSR	PC,1\$;REPORT ERROR
4123	011102	005721				TST	(R1)+	;CHECK 2ND KEY WORD
4124	011104	001402				BEQ	.+6	;BR IF ZERO
4125	011106	004737	011130			JSR	PC,1\$;REPORT THE ERROR
4126	011112	162702	000004			SUB	#4,R2	;SUBTRACT HEADER LENGTH FROM SIZE
4127	011116	003530				BLE	CMPRX	;BR IF FINISHED
4128	011120	022702	000004			CMP	#4,R2	;SEE IF AT LEAST 4 MORE WORDS TO CHECK
4129	011124	101125				BHI	CMPRX	;BR IF NOT
4130	011126	000405				BR	CMDAT	;COMPARE THE DATA PORTION
4131	011130	005237	011414		1\$:	INC	ERCTR	;INCREMENT THE ERROR COUNT
4132	011134	004737	011426			JSR	PC,CMPT	;REPORT THE COMPARISON ERROR
4133	011140	000207				RTS	PC	;CHECK THE REST OF THE HEADER
4134	011142	004737	011746		CMDAT:	JSR	PC,MATCH	;FIND THE PATTERN
4135	011146	000403				BR	2\$;FOUND A PATTERN
4136	011150	004737	010262			JSR	PC,NOMTCH	;RETURN HERE IF NO MATCH WITH PATTERN MADE
4137	011154	000456				BR	8\$;BYPASS COMPARE ROUTINE
4138	011156	011405			2\$:	MOV	(R4),R5	;ADDRESS OF PATTERN ADDRESS IN R4

4139	011160	012703	000020		MOV	#20,R3	:R3 IS PATTERN POS COUNTER
4140	011164	022125		3\$:	CMP	(R1)+,(R5)+	:COMPARE BUFFER WITH PATTERN
4141	011166	001016			BNE	5\$:BR IF NOT EQUAL
4142	011170	005737	011414		TST	ERCTR	:ERRORS DETECTED ?
4143	011174	001406			BEQ	4\$:BR IF NO ERRORS
4144	011176	032737	000010	001140	BIT	#SW3,SWR	:SWITCH 3 SET ?
4145	011204	001402			BEQ	4\$:BR IF NOT SET
4146	011206	004737	011426		JSR	PC,CMPRT	:DISPLAY THE WORD
4147	011212	005302		4\$:	DEC	R2	:DECREMENT SIZE COUNT
4148	011214	001436			BEQ	8\$:BR WHEN AT END
4149	011216	005303			DEC	R3	:DECREMENT PATT POS COUNT
4150	011220	001361			BNE	3\$:BR IF NOT AT END OF PATT
4151	011222	000755			BR	2\$:RESTART THE PATTERN
4152	011224	005761	177776		TST	-2(R1)	:IS MISCOMPARED CHARACTER=0
4153	011230	001410			BEQ	6\$:BR IF YES
4154	011232	112737	177777	011406	MOVB	#-1,ZROIND	:SET NON-ZERO MISCOMPARED IND
4155	011240	005237	011414		INC	ERCTR	:INCREMENT THE ERROR COUNTER
4156	011244	004737	011426		JSR	PC,CMPRT	:REPORT ERROR
4157	011250	000760			BR	4\$:CONTINUE COMPARE
4158	011252	105737	011407	6\$:	TSTB	FRSTER	:FIRST ERROR?
4159	011256	100407			BMI	7\$:BR IF NOT
4160	011260	105037	011406		CLRB	ZROIND	:SET THE ZERO INDICATOR
4161	011264	010137	011410		MOV	R1,SAVER1	:SAVE CURRENT R1
4162	011270	010537	011412		MOV	R5,SAVER5	:SAVE CURRENT R5
4163	011274	000746			BR	4\$:CONTINUE COMPARE
4164	011276	105737	011406	7\$:	TSTB	ZROIND	:ANY MISCOMPARISONS NOT ZEROS ?
4165	011302	001743			BEQ	4\$:BR IF NONE-ALL ERRORS=ZERO
4166	011304	004737	011426		JSR	PC,CMPRT	:REPORT ERROR
4167	011310	000740			BR	4\$:CONTINUE COMPARING
4168	011312	005737	011420	8\$:	TST	CMCNT	:AT END OF BUFFER
4169	011316	003430			BLE	CMPRX	:BR IF AT END
4170	011320	126027	000030	000005	CMPB	\$CODE(RO),#5	:SEE IF READ HEADER & DATA
4171	011326	001220			BNE	CMSTR	:BR IF NOT
4172	011330	105237	011424		INCB	CMSEC	:INCREMENT SECTOR
4173	011334	123727	011424	000026	CMPB	CMSEC,#22.	:SECTOR GREATER THAN MAX ?
4174	011342	103612			BLO	CMSTR	:BR IF NOT GREATER THAN MAX
4175	011344	105037	011424		CLRB	CMSEC	:CLEAR SECTOR ADDRESS
4176	011350	105237	011425		INCB	CMTRK	:INCREMENT TRACK
4177	011354	123727	011425	000023	CMPB	CMTRK,#19.	:TRACK GREATER THAN MAX ?
4178	011362	103602			BLO	CMSTR	:BR IF NOT GREATER
4179	011364	105037	011425		CLRB	CMTRK	:RESET TRACK ADDRESS
4180	011370	005237	011422		INC	CMCYL	:INCREMENT CYLINDER ADDRESS
4181	011374	000137	010770		JMP	CMSTR	:CONTINUE WITH COMPARE
4182	011400	004737	011702	CMPRX:	JSR	PC,ENDCMP	:PRINT LAST LINE IF ERRORS
4183	011404	000207			RTS	PC	
4184							
4185	011406	377		ZROIND:	.BYTE	-1	:ZERO INDICATOR
4186	011407	000		FRSTER:	.BYTE	0	:FIRST ERROR INDICATOR
4187							:IF > 0, PROCESSING 'DCKER'
4188							:IF < 0, MISCOMPARSION FOUND
4189	011410	000000		SAVER1:	.WORD	0	:SAVE R1 HERE
4190	011412	000000		SAVER5:	.WORD	0	:SAVE R5 HERE
4191	011414	000000		ERCTR:	.WORD	0	:NUMBER OF ERRORS
4192	011416	000000		LIMIT:	.WORD	0	:DISPLAY LIMIT
4193	011420	000000		CMCNT:	.WORD	0	:WORD COUNT
4194	011422	000000		CMCYL:	.WORD	0	:CYLINDER ADDRESS

4195	011424	000		CMSEC: .BYTE 0		:SECTOR ADDRESS
4196	011425	000		CMTRK: .BYTE 0		:TRACK ADDRESS
4197						
4198						
4199						
4200	011426	005737	011410	CMPT: TST	SAVER1	:PRINT SAVED VALUES ?
4201	011432	001010		BNE	2\$:BR IF NOT
4202	011434	105737	011407	TSTB	FRSTER	:FIRST ERROR?
4203	011440	100402		BMI	1\$:BR IF NOT
4204	011442	004737	011522	JSR	PC,4\$:PRINT INITIAL MESSAGE INFO
4205	011446	004737	011604	1\$: JSR	PC,8\$:PRINT REMAINDER OF MESSAGE
4206	011452	000422		BR	3\$:EXIT
4207	011454			2\$:		
4208	011454	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
4209	011456	010546		MOV	R5,-(SP)	::PUSH R5 ON STACK
4210	011460	013701	011410	MOV	SAVER1,R1	:DISPLAY SAVED R1
4211	011464	013705	011412	MOV	SAVER5,R5	:DISPLAY SAVED R5
4212	011470	004737	011522	JSR	PC,4\$:PRINT INITIAL MESSAGE INFO
4213	011474	004737	011604	JSR	PC,8\$:PRINT SAVED VALUES
4214	011500	005037	011410	CLR	SAVER1	:CLEAR SAVED REGISTER INDICATORS
4215	011504	005037	011412	CLR	SAVER5	:CLEAR THE OTHER ONE
4216	011510	012605		MOV	(SP)+,R5	::POP STACK INTO R5
4217	011512	012601		MOV	(SP)+,R1	::POP STACK INTO R1
4218	011514	004737	011604	JSR	PC,8\$:PRINT REMAINDER OF MESSAGE
4219	011520	000207		3\$: RTS	PC	:RETURN
4220	011522	105737	011407	4\$: TSTB	FRSTER	:FIRST ERROR ?
4221	011526	100425		BMI	7\$:BR IF NOT
4222	011530	001013		BNE	5\$:BR IF FIRST ERROR AND PROCESSING 'DCK' ERROR
4223	011532	004737	016002	JSR	PC,LINE1	:PRINT LINE 1 OF ERROR MESSAGE
4224	011536	104412	045536	DISPLY	EM42	:DATA COMPARE ERROR
4225	011542	004737	016046	JSR	PC,LINE2	:PRINT LINE 2 OF ERROR MESSAGE
4226	011546	004737	016426	JSR	PC,LINE3A	:PRINT LINE 3A OF ERROR MESSAGE
4227	011552	004737	017156	JSR	PC,LINE4	:PRINT LINE 4 OF ERROR MESSAGE
4228	011556	000404		BR	6\$	
4229	011560	104412	050574	5\$: DISPLY	,LIN9B	:HEADER MESSAGE OF PROCESSING 'DCK' ERROR
4230	011564	104412	001165	DISPLY	,\$CRLF	
4231	011570	104412	050623	6\$: DISPLY	,LIN9H	:DISPLAY HEADER
4232	011574	112737	177777 011407	MOVB	#-1,FRSTER	:SET FIRST ERROR INDICATOR
4233	011602	000207		7\$: RTS	PC	:RETURN
4234	011604	005737	011416	8\$: TST	LIMIT	:TYPEOUT LIMIT REACHED ?
4235	011610	001403		BEQ	9\$:BR IF IT HAS
4236	011612	005337	011416	DEC	LIMIT	:DECREMENT LIMIT COUNTER
4237	011616	001005		BNE	10\$:BR IF NOT AT LIMIT
4238	011620	032737	000200 001140	9\$: BIT	#SW07,SWR	:PRINT ALL DATA COMPARE ERRORS ?
4239	011626	001001		BNE	10\$:BR IF YES
4240	011630	000207		RTS	PC	:RETURN
4241	011632	010146		10\$: MOV	R1,-(SP)	:BUFFER ADDRESS
4242	011634	162716	000002	SUB	#2,(SP)	:ADJUST ADDRESS
4243	011640	004737	024740	JSR	PC,PRT0CT	:PRINT IT
4244	011644	104412	051364	DISPLY	,LINSF	:2 SPACES
4245	011650	016546	177776	MOV	-2(R5),-(SP)	:GOOD DATA
4246	011654	004737	024740	JSR	PC,PRT0CT	:PRINT IT
4247	011660	104412	051364	DISPLY	,LINSF	:2 SPACES
4248	011664	016146	177776	MOV	-2(R1),-(SP)	:BAD DATA
4249	011670	004737	024740	JSR	PC,PRT0CT	:PRINT IT
4250	011674	104412	001165	DISPLY	,\$CRLF	:CR-LF

```

4251 011700 000207          RTS      PC          ;RETURN
4252
4253          ;LAST LINE OF COMPARE ERROR REPORTING
4254
4255 011702 005737 011414  ENDCMP: TST      ERCTR          ;SEE IF ANY ERRORS
4256 011706 001416          BEQ      1$          ;BR IF NOT
4257 011710 104412 050721  DISPLY  LIN9E          ;'NUMBER OF ERRORS='
4258 011714 013746 011414  MOV      ERCTR, -(SP) ;NUMBER OF ERRORS
4259 011720 004737 030232  JSR      PC, $$B2D    ;CONVERT IT
4260 011724 004737 025576  JSR      PC, $$SUPRS  ;PRINT IT
4261 011730 104412 001165  DISPLY  $CRLF          ;CR-LF
4262 011734 004737 021356  JSR      PC, INCTOT   ;INCREMENT TOTAL ERROR COUNT
4263 011740 004737 017622  JSR      PC, LINE7    ;PRINT LINE 7 OF ERROR MESSAGE
4264 011744 000207 1$:      RTS      PC
4265
4266
4267          ;FIND THE CORRECT PATTERN - RETURN WITH ADDRESS OF PATTERN IN R4
4268          ;
4269
4270 011746 010146          MATCH: MOV      R1, -(SP)    ;SAVE R1 ON THE STACK
4271 011750 012704 000044  MOV      #4, R4       ;PATTERN TABLE INDEX
4272 011754 011601 1$:      MOV      (SP), R1    ;RELOAD R1
4273 011756 162704 000002  SUB      #2, R4       ;DECREMENT INDEX
4274 011762 016405 042646  MOV      STNDAT(R4), R5 ;ADDRESS OF PATTERN ADDRESS
4275 011766 001411          BEQ      3$          ;BR IF ALL PATTERNS CHECKED AND NO MATCH
4276
4277 011770 012703 000004          ;FOUND
4278 011774 022125 2$:      MOV      #4, R3       ;NUMBER OF LOCATIONS TO CHECK
4279 011776 001366          CMP      (R1)+, (R5)+ ;COMPARE THE BUFFER AGAINST THE PATTERN
4280 012000 005303          BNE      1$          ;BR IF NOT EQUAL, TRY NEXT PATTERN
4281 012002 001374          DEC      R3          ;FINISHED CHECKING?
4282 012004 062704 042646  BNE      2$          ;BR IF NOT FINISHED
4283 012010 000403          ADD      #STNDAT, R4  ;MAKE PATTERN ADDRESS ABSOLUTE
4284 012012 062766 000002 000002 BR      4$          ;EXIT
4285 012020 012601 3$:      ADD      #2, 2(SP)    ;INCREMENT RETURN ADDRESS
4286 012022 000207 4$:      MOV      (SP)+, R1   ;RESTORE R1
4287
4288          ;USE ECC TO CORRECT THE DATA ERROR
4289
4290 012024 016037 000170 012566 ECC:  MOV      $RHBA(RO), ECSEC ;ADDRESS OF LAST LOCN XFERED
4291 012032 016046 000166  MOV      $RHC(RO), -(SP) ;ACT WORDS XFERED (2'S COMP)
4292 012036 066016 000020  ADD      $WRDL(RO), (SP) ;ADD WORDS REQUESTED
4293 012042 005046          CLR      -(SP)       ;CLEAR NEXT STACK LOCN
4294 012044 016046 000026  MOV      $$SSEC(RO), -(SP) ;SECTOR SIZE
4295 012050 004737 024224  JSR      PC, LINKDV   ;DIVIDE WORDS XFERED BY SECTOR SIZE
4296 012054 005716          TST      (SP)        ;PARTIAL SECTOR XFERED ?
4297 012056 001413          BEQ      1$          ;BR IF NOT
4298 012060 006316          ASL      (SP)        ;CONVERT INTO NUMBER OF BYTES
4299 012062 161637 012566  SUB      (SP), ECSEC  ;SUBTRACT SECTOR RESIDUE
4300 012066 126027 000030 000005 CMPB     $CODE(RO), #5 ;WAS OP READ HEAD & DATA
4301 012074 001007          BNE      2$          ;BR IF NOT
4302 012076 062737 000010 012566 ADD      #8., ECSEC   ;ADD HEADER SIZE (IN BYTES) BACK IN
4303 012104 000403          BR      2$          ;GO ADJUST THE STACK POINTER
4304 012106 162737 001000 012566 1$:      SUB      #1000, ECSEC ;SUBTRACT SECTOR DATA FIELD SIZE (IN BYTES)
4305 012114 062706 000004 2$:      ADD      #4, SP     ;ADJUST THE STACK POINTER
4306 012120 016037 000230 012564  MOV      $RHEC1(RO), ECBIT ;ECC POSITION COUNT

```

4307	012126	005337	012564		DEC	ECBIT	:ADJUST THE POSITION COUNT	
4308	012132	013737	012564	012574	MOV	ECBIT,ECWRD	:LOAD THE WORD COUNT LOCATION	
4309	012140	042737	177760	012564	BIC	#17,ECBIT	:SAVE THE BIT OFFSET COUNT	
4310	012146	042737	000017	012574	BIC	#17,ECWRD	:CLEAR THE BIT OFFSET	
4311	012154	006237	012574		ASR	ECWRD	:CHANGE TO BYTE COUNT	
4312	012160	006237	012574		ASR	ECWRD	:CHANGE TO BYTE COUNT	
4313	012164	006237	012574		ASR	ECWRD	:CHANGE TO BYTE COUNT	
4314	012170	104412	051000		DISPLY	.LIN10A	: 'ERROR BURST BEGINS AT '	
4315	012174	013746	012574		MOV	ECWRD,-(SP)	:PUT THE WORD COUNT ON THE STACK	
4316	012200	006216			ASR	(SP)	:CONVERT TO WORD COUNT FOR MESSAGE	
4317	012202	004737	030232		JSR	PC,\$\$SB2D	:CONVERT THE WORD COUNT	
4318	012206	004737	025576		JSR	PC,\$\$SUPRS	:PRINT IT	
4319	012212	104412	051034		DISPLY	.LIN10B	: ' IN DATA FIELD OF ERROR SECTOR'	
4320	012216	063737	012566	012574	ADD	ECSEC,ECWRD	:FIND THE BEGINNING OF THE ERROR BURST	
4321	012224	026037	000170	012574	CMP	\$RHBA(RO),ECWRD	:SEE IF BURST WAS IN DATA READ	
4322	012232	101002			BHI	.+6	:BR IF IN DATA READ	
4323	012234	000137	012552		JMP	ECC2	:NOT IN DATA READ - REPORT IT	
4324	012240	016037	000232	012570	MOV	\$RHEC2(RO),ECMSK0	:GET THE ERROR MASK	
4325	012246	005037	012572		CLR	ECMSK1	:CLEAR THE UPPER MASK WORD	
4326	012252	005737	012564	3\$:	TST	ECBIT	:BIT OFFSET EQUAL ZERO	
4327	012256	001407			BEQ	4\$:BR IF IT IS	
4328	012260	005337	012564		DEC	ECBIT	:DECREMENT THE BIT OFFSET COUNT	
4329	012264	006337	012570		ASL	ECMSK0	:SHIFT THE ERROR MASK	
4330	012270	006137	012572		ROL	ECMSK1	:SHIFT THE LOWER INTO THE UPPER	
4331	012274	000766			BR	3\$:CONTINUE THE SHIFT	
4332	012276	017737	000272	012600	4\$:	MOV	DECWRD,ECBADO	:SAVE THE INCORRECT WORD
4333	012304	005037	012602		CLR	ECWRD1	:CLEAR SECOND INCORRECT WORD ADDRESS	
4334	012310	013746	012570		MOV	ECMSK0,-(SP)	:PUT LOWER MASK ON STACK	
4335	012314	047716	000254		BIC	DECWRD,(SP)	:CLEAR ERRONEOUS ONE BITS FROM MASK	
4336	012320	043777	012570	000246	BIC	ECMSK0,DECWRD	:CLEAR ERRONEOUS ONE BITS FROM BAD WORD	
4337	012326	052677	000242		BIS	(SP)+,DECWRD	:SET DROPPED BITS	
4338	012332	005737	012572		TST	ECMSK1	:DOES BURST GO INTO NEXT WORD ?	
4339	012336	001431			BEQ	ECC1	:BR IF BURST ONLY IN ONE WORD	
4340	012340	013737	012574	012602	MOV	ECWRD,ECWRD1	:DUPLICATE ADDRESS	
4341	012346	062737	000002	012602	ADD	#2,ECWRD1	:INCREMENT ERROR ADDRESS	
4342	012354	026037	000170	012602	CMP	\$RHBA(RO),ECWRD1	:IS NEXT WORD IN THE BUFFER	
4343	012362	101003			BHI	5\$:BR IF IT IS	
4344	012364	005037	012602		CLR	ECWRD1	:CLEAR 2ND WORD ADDRESS	
4345	012370	000414			BR	ECC1	:PRINT WORD CORRECTED	
4346	012372	017737	000204	012606	5\$:	MOV	DECWRD1,ECBAD1	:SAVE THE SECOND BAD WORD
4347	012400	013746	012572		MOV	ECMSK1,-(SP)	:PUT THE UPPER MASK ON THE STACK	
4348	012404	047716	000172		BIC	DECWRD1,(SP)	:CLEAR ERRONEOUS ONE BITS FROM UPPER MASK	
4349	012410	043777	012572	000164	BIC	ECMSK1,DECWRD1	:CLEAR ERRONEOUS ONE BITS FROM DATA WORD	
4350	012416	052677	000160		BIS	(SP)+,DECWRD1	:SET DROPPED BITS	
4351	012422	104412	051202		ECC1:	DISPLY	.LIN10H	
4352	012426	013746	012574		MOV	ECWRD,-(SP)	:PUT ECWRD ON THE STACK	
4353	012432	004737	024740		JSR	PC,PRTOCT	:PRINT ECWRD	
4354	012436	104412	051364		DISPLY	.LINS	:SPACES	
4355	012442	013746	012600		MOV	ECBADO,-(SP)	:PUT ECBADO ON THE STACK	
4356	012446	004737	024740		JSR	PC,PRTOCT	:PRINT ECBADO	
4357	012452	104412	051364		DISPLY	.LINS	:SPACES	
4358	012456	017746	000112		MOV	DECWRD,-(SP)	:PUT DECWRD ON THE STACK	
4359	012462	004737	024740		JSR	PC,PRTOCT	:PRINT DECWRD	
4360	012466	104412	051364		DISPLY	.LINS	:SPACES	
4361	012472	005737	012602		TST	ECWRD1	:PRINT THE NEXT WORD ?	
4362	012476	001427			BEQ	ECCX	:BR IF NOT	

4363	012500	104412	001165		DISPLY	\$CRLF		:CR-LF
4364	012504	013746	012602		MOV	ECWRD1,-(SP)		:PUT ECWRD1 ON THE STACK
4365	012510	004737	024740		JSR	PC,PRTOCT		:PRINT ECWRD1
4366	012514	104412	051364		DISPLY	LINSP		:SPACES
4367	012520	013746	012606		MOV	ECBAD1,-(SP)		:PUT ECBAD1 ON THE STACK
4368	012524	004737	024740		JSR	PC,PRTOCT		:PRINT ECBAD1
4369	012530	104412	051364		DISPLY	LINSP		:SPACES
4370	012534	017746	000042		MOV	DECWRD1,-(SP)		:PUT DECWRD1 ON THE STACK
4371	012540	004737	024740		JSR	PC,PRTOCT		:PRINT DECWRD1
4372	012544	104412	051364		DISPLY	LINSP		:SPACES
4373	012550	000402			BR	ECCX		:EXIT
4374	012552	104412	051075	ECC2:	DISPLY	LIN10C		:ERROR BURST WAS NOT TRANSFERED TO MEMORY
4375	012556	104412	001165	ECCX:	DISPLY	\$CRLF		:CR-LF
4376	012562	000207			RTS	PC		:RETURN
4377								
4378	012564	000000		ECBIT:	.WORD	0		:ERROR BURST BIT OFFSET
4379	012566	000000		ECSEC:	.WORD	0		:ERROR BURST WORD OFFSET (RELATIVE TO SECTOR)
4380	012570	000000		ECMSK0:	.WORD	0		:CORRECTION MASK FOR FIRST ERROR WORD
4381	012572	000000		ECMSK1:	.WORD	0		:CORRECTION MASK FOR SECOND ERROR WORD
4382	012574	000000		ECWRD:	.WORD	0		:LOCATION OF FIRST ERROR WORD
4383	012576	000000		ECGD:	.WORD	0		:GOOD DATA, FIRST WORD
4384	012600	000000		ECBADO:	.WORD	0		:BAD DATA, FIRST WORD
4385	012602	000000		ECWRD1:	.WORD	0		:LOCATION OF SECOND ERROR WORD
4386	012604	000000		ECGD1:	.WORD	0		:GOOD DATA, SECOND WORD
4387	012606	000000		ECBAD1:	.WORD	0		:BAD DATA, ECOND WORD
4388								
4389								:ROUTINE TO DISPLAY THE SECTOR WHICH GAVE THE HARD ERROR
4390								
4391	012610	032737	000010	001140	PRTBAD:	BIT	#SW3,SWR	:PRINT THE BAD SECTOR ?
4392	012616	001460			BEQ	6\$:BR IF NOT
4393	012620	016001	000170		MOV	\$RHBA(R0),R1		:PUT THE END ADDRESS INTO R1
4394	012624	016046	000020		MOV	\$WRDL(R0),-(SP)		:FIND THE BEGINNING OF THE SECTOR
4395	012630	066016	000166		ADD	\$RHWC(R0),(SP)		:SUBTRACT THE WORDS NOT TRANSFERED
4396	012634	005046			CLR	-(SP)		:MAKE THE UPPER DIVIDEND 0
4397	012636	016046	000026		MOV	\$SSEC(R0),-(SP)		:DIVDE THE WORDS TRANSFERED BY THE SECTOR SIZE
4398	012642	004737	024224		JSR	PC,LINKDV		:DIVIDE
4399	012646	005716			TST	(SP)		:REMAINDER = 0 ?
4400	012650	001403			BEQ	1\$:BR IF IT IS - COMPLETE SECTOR TRANSFERED
4401	012652	006316			ASL	(SP)		:CONVERT THE RESIDUAL SECTOR SIZE INTO BYTE COUNT
4402	012654	161601			SUB	(SP),R1		:SUBTRACT IT FROM THE END ADDRESS
4403	012656	000410			BR	2\$:FINISH THE SIZING
4404	012660	162701	001000	1\$:	SUB	#1000,R1		:SUBTRACT FULL SECTOR SIZE FROM END ADDR
4405	012664	126027	000030	000005	CMPB	\$CODE(R0),#5		:WAS OPERATION READ HEADER & DATA ?
4406	012672	001002			BNE	2\$:BR IF NOT
4407	012674	162701	000010		SUB	#10,R1		:SUBTRACK HEADER SIZE FROM ADDR
4408	012700	062706	000004	2\$:	ADD	#4,SP		:RESTORE THE STACK POINTER
4409	012704	104412	051267	3\$:	DISPLY	LIN11H ;PRINT THE HEADER		
4410	012710	012702	000007		MOV	#7,R2		:R2 CONTAINS THE WORDS/LINE COUNT
4411	012714	010146			MOV	R1,-(SP)		:PUT THE ADDRESS ON THE STACK
4412	012716	004737	024740		JSR	PC,PRTOCT		:PRINT IT
4413	012722	020160	000170	4\$:	CMP	R1,\$RHBA(R0)		:PRINTED ALL THE SECTOR ?
4414	012726	001412			BEQ	5\$:BR IF ALL PRINTED
4415	012730	104412	051364		DISPLY	LINSP		:SPACES
4416	012734	012146			MOV	(R1)+,-(SP)		:PUT THE DATA ON THE STACK
4417	012736	004737	024740		JSR	PC,PRTOCT		:PRINT IT
4418	012742	005302			DEC	R2		:DECREMENT THE HORIZONTAL COUNT

```

4419 012744 001366          BNE      4$          ;BR IF NOT AT THE END OF THE LINE
4420 012746 104412 001165  DISPLY  $CRLF        ;CR-LF
4421 012752 000756          BR       3$          ;RESTORE THE WORDS/LINE COUNT
4422 012754 104412 001165  5$:    DISPLY  $CRLF        ;PRINT WHAT REMAINS IN THE BUFFER
4423 012760 000207          6$:    RTS      PC          ;RETURN
4424
4425 ;ROUTINE TO DO AN RTC - UNIT SELECTED IN RO
4426
4427 012762 111037 041526  RTNCTR: MOVB   (RO),GENDPB ;MOVE THE UNIT # TO THE GENERAL DPB
4428 012766 112737 000117 041530  MOVB   #RTC,GENDPB+$COMND ;COMMAND CODE
4429 012774 004037 031442  JSR    RO,RPO4          ;DRIVER ENTRANCE
4430 013000 041526          GENDPB ;DPB ADDRESS FOR ORDER
4431 013002 000000          HALT   ;DRIVER DIDN'T ACCEPT ORDER
4432 013004 000207          RTS    PC          ;RETURN
4433
4434 ;ROUTINE TO DO A RECALIBRATE - UNIT SELECTED IN RO
4435 ;ENTER AT 'RECALO' IF UNIT NUMBER ALREADY LOADED
4436 ;INTO 'GENDPB'
4437
4438 013006 111037 041526  RECAL: MOVB   (RO),GENDPB ;MOVE THE UNIT # TO THE GENERAL DPB
4439 013012 112737 000107 041530  RECALO: MOVB  #RECAL,GENDPB+$COMND ;RECALIBRATE COMMAND
4440 013020 004037 031442  JSR    RO,RPO4          ;DRIVER ENTRANCE
4441 013024 041526          GENDPB ;DPB ADDRESS FOR ORDER
4442 013026 000000          HALT   ;DRIVER DIDN'T ACCEPT THE ORDER
4443 013030 005737 041544  1$:    TST    GENDPB+$STATUS ;SEE IF FINISHED
4444 013034 001775          BEQ    1$          ;BR IF NOT FINISHED
4445 013036 000207          RTS    PC          ;RETURN
4446
4447 ;OFFSET THE UNIT IN RO (OFFSET CODE PRELOADED)
4448
4449 013040 111037 041526  OFFST: MOVB   (RO),GENDPB ;UNIT # TO GENERAL DPB
4450 013044 112737 000115 041530  MOVB   #OFFSET,GENDPB+$COMND ;COMMAND
4451 013052 004037 031442  JSR    RO,RPO4          ;DRIVER ENTRANCE
4452 013056 041526          GENDPB ;DPB ADDRESS FOR ORDER
4453 013060 000000          HALT   ;DRIVER DIDN'T ACCEPT ORDER
4454 013062 000207          RTS    PC
4455
4456 ;UTILITY READ HEADER ROUTINE
4457 ; ENTER WITH TRACK AND SECTOR ADDRS ON THE STACK
4458
4459 013064 116637 000002 041537  READHD: MOVB   2(SP),GENDPB+$TRK ;TRACK ADDRESS
4460 013072 116637 000004 041536  MOVB   4(SP),GENDPB+$SEC ;SECTOR ADDRESS
4461 013100 111037 041526          MOVB   (RO),GENDPB ;DRIVE NUMBER
4462 013104 016037 000222 041540  MOV    $RHCC(RO),GENDPB+$CYL ;CYLINDER ADDRESS
4463 013112 112737 000173 041530  MOVB   #RDHD,GENDPB+$COMND ;COMMAND
4464 013120 004037 031442  JSR    RO,RPO4          ;DRIVER ENTRANCE
4465 013124 041526          GENDPB ;DPB ADDRESS FOR ORDER
4466 013126 000000          HALT   ;DRIVER DIDN'T ACCEPT COMMAND
4467 013130 005737 041544  1$:    TST    GENDPB+$STATUS ;FINISHED?
4468 013134 001775          BEQ    1$          ;BR IF NOT
4469 013136 012666 000002  MOV    (SP)+,2(SP) ;ADJUST STACK FOR RETURN
4470 013142 005726          TST    (SP)+
4471 013144 000207          RTS    PC          ;RETURN
4472
4473 ;RETRY THE PRESENT OPERATION
4474 ; RETURN IF RETRY UNSUCCESSFUL

```



```

4475 ; RETURN+2 IF RETRY SUCESSFUL
4476 ; RETURN TO MAIN PROGRAM IF DIFFERENT ERROR OCCURS
4477
4478 013146 004737 014340 RETRY: JSR PC,GODRIV ;RE-START ORDER
4479 013152 005760 000016 1$: TST $STATUS(RO) ;ORDER FINISHED?
4480 013156 001775 BEQ 1$ ;BR IF NOT
4481 013160 100405 BMI 2$ ;BR IF ERROR
4482 013162 105260 000022 INCB $RETRY(RO) ;INCREMENT RETRY COUNT
4483 013166 062716 000002 ADD #2,(SP) ;INCREMENT RETURN
4484 013172 000424 BR 5$ ;GO TO EXIT
4485 013174 032760 000200 000016 2$: BIT #BIT7,$STATUS(RO) ;DID ORDER TERMINATE NORMALLY ?
4486 013202 001427 BEQ 7$ ;BR IF NOT
4487 013204 005760 000024 TST $MASK(RO) ;IS ERROR MASK 0 ?
4488 013210 001004 BNE 3$ ;BR IF NOT
4489 013212 005760 000200 TST $RHER1(RO) ;MAKE SURE THAT THE DRIVE ERROR REG IS CLEAR
4490 013216 001013 BNE 6$ ;BR IF NOT
4491 013220 000404 BR 4$ ;CONTINUE RETRY
4492 013222 036060 000024 000200 3$: BIT $MASK(RO),$RHER1(RO) ;SAME ERROR?
4493 013230 001406 BEQ 6$ ;BR IF NOT
4494 013232 105260 000022 4$: INCB $RETRY(RO) ;INCREMENT RETRY COUNT
4495 013236 105360 000023 DECB $RETRY+1(RO) ;DECREMENT RETRY LIMIT
4496 013242 001341 BNE RETRY ;BR IF NOT DONE
4497 013244 000207 5$: RTS PC ;RETURN
4498 013246 004737 020070 6$: JSR PC,LIN6B ;REPORT DIFFERENT ERROR
4499 013252 004737 017622 JSR PC,LIN7 ;PRINT LINE 7
4500 013256 005726 TST (SP)+ ;ADJUST STACK POINTER FOR DIRECT RETURN
4501 013260 000207 RTS PC ;RETURN
4502 013262 104412 050537 7$: DISPLY ,LIN6M ;'DIFFERENT ERROR DURING RETRY'
4503 013266 000137 004560 JMP ERPRC1 ;REPORT THE ERROR

```

;ROUTINE TO CLEAR THE DPB FOR THE ASSIGNED UNIT

```

4504
4505
4506
4507 013272 CLRDPB:
4508 013272 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
4509 013274 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
4510 013276 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
4511 013300 010004 MOV RO,R4 ;GET THE DPB ADDRESS
4512 013302 062704 000002 ADD #2,R4 ;ADDRESS OF FIRST LOCN TO BE CLEARED
4513 013306 012703 000005 MOV #5,R3 ;NUMBER OF LOCNS TO BE CLEARED
4514 013312 005024 CLR (R4)+ ;CLEAR THE LOCN
4515 013314 005303 DEC R3 ;DECREMENT THE LOCN COUNTER
4516 013316 001375 BNE .-4 ;BR IF NOT FINISHED
4517 013320 062704 000002 ADD #2,R4 ;MOVE THE ADDRESS PAST THE 'REG' ADDR
4518 013324 012703 000220 MOV #$RHEC2+2-$REG,R3 ;NUMBER OF LOCNS TO BE CLEARED
4519 013330 005024 CLR (R4)+ ;CLEAR
4520 013332 162703 000002 SUB #2,R3 ;DECREMENT THE LOCN COUNTER
4521 013336 001374 BNE .-6 ;BR IF NOT FINISHED
4522 013340 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
4523 013342 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
4524 013344 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
4525 013346 000207 RTS PC

```

;ROUTINE TO UPDATE THE GENERAL STATISTICS FOR THE DRIVE IN RO

```

4526
4527
4528
4529 013350 016037 000170 013502 STATIS: MOV $RHA(RO),FACTOR ;STORE THE FINAL BUFFER ADDRESS
4530 013356 166037 000006 013502 SUB $BUF(RO),FACTOR ;SUBTRACT THE INITIAL ADDRESS

```

```

4531 013364 001434 BEQ 2$ ;BR IF NO DATA TRANSFER
4532 013366 006237 013502 ASR FACTOR ;CONVERT TO A WORD COUNT
4533 013372 063760 013502 000052 ADD FACTOR,$TRANS(R0) ;UPDATE WORD COUNT
4534 013400 005560 000054 ADC $TRANS+2(R0)
4535 013404 132760 000002 000030 BITB #BIT01,$CODE(R0) ;SEE IF ORDER READ OR WRITE
4536 013412 001021 BNE 2$ ;BRANCH IF ORDER WRITE
4537 013414 005737 001316 TST AUTOCK ;AUTO WRITE CHECKS BEING PERFORMED
4538 013420 001411 BEQ 1$ ;BR IF NOT
4539 013422 126027 000030 000001 CMPB $CODE(R0),#1 ;PRESENT OPERATION AN AUTOMATIC WRITE CHECK ?
4540 013430 101005 BHI 1$ ;BR IF NOT
4541 013432 066060 000020 000052 ADD $WRDL(R0),$TRANS(R0) ;ADD WORDS WRITTEN
4542 013440 005560 000054 ADC $TRANS+2(R0) ;ADD A CARRY
4543 013444 063760 013502 000056 1$: ADD FACTOR,$READ(R0) ;UPDATE THE READ WORD COUNT
4544 013452 005560 000060 ADC $READ+2(R0)
4545 013456 026060 000012 000220 2$: CMP $CYL(R0),$RHCA(R0) ;DID MID-TRANSFER SEEK OCCUR
4546 013464 001405 BEQ 3$ ;BR IF NOT
4547 013466 062760 000001 000046 ADD #1,$POSIT(R0) ;INCREMENT SEEK COUNT
4548 013474 005560 000050 ADC $POSIT+2(R0) ;ADD CARRY TO UPPER WORD
4549 013500 000207 3$: RTS PC
4550
4551 013502 000000 FACTOR: .WORD 0 ;USED FOR WORDS TRANSFERED
4552
4553 ;ROUTINE TO GET A BUFFER
4554
4555 013504 010146 GETBUF: MOV R1,-(SP) ;SAVE R1
4556 013506 022737 000002 042002 CMP #2,BUFTBL ;SEE HOW MANY BUFFER BLOCKS ARE IN TABLE
4557 013514 103001 BHIS .+4 ;BR IF NOT MORE THAN 2
4558 013516 000240 NOP ;CHANGE TO HALT FOR DEBUGGING
4559 013520 013702 042002 MOV BUFTBL,R2 ;NUMBER OF SEPARATE BUFFERS
4560 013524 001413 BEQ 5$ ;BR IF NONE AVAILABLE
4561 013526 012701 042004 MOV #BUFTBL+2,R1 ;FIRST ADDRESS OF ALLOCATION TABLE
4562 013532 026061 000020 000002 1$: CMP $WRDL(R0),2(R1) ;SEE IF THERE IS A BLOCK LARGE ENOUGH
4563 013540 101407 BLOS 3$ ;BRANCH IF IT IS
4564 013542 005302 DEC R2 ;DECREMENT TABLE COUNT
4565 013544 001403 BEQ 5$ ;BR IF THROUGH TABLE
4566 013546 062701 000004 ADD #4,R1 ;INCREMENT TABLE POINTER
4567 013552 000767 BR 1$ ;CONTINUE LOOKING
4568 013554 012601 5$: MOV (SP)+,R1 ;RESTORE R1
4569 013556 000207 RTS PC ;RETURN
4570 013560 011166 000004 3$: MOV (R1),4(SP) ;BUFFER ADDRESS TO STACK
4571 013564 166061 000020 000002 SUB $WRDL(R0),2(R1) ;ADJUST BUFFER SIZE
4572 013572 001407 BEQ 4$ ;BR IF DIFFERENCE IS ZERO
4573 013574 006360 000020 ASL $WRDL(R0) ;CONVERT # WORDS TO BYTES
4574 013600 066011 000020 ADD $WRDL(R0),(R1) ;MAKE NEW STARTING ADDRESS
4575 013604 006260 000020 ASR $WRDL(R0) ;RETURN # BYTES TO WORDS
4576 013610 000761 BR 5$ ;RETURN
4577 013612 005337 042002 4$: DEC BUFTBL ;DECREMENT ENTRIES COUNT
4578 013616 001756 BEQ 5$ ;BR IF ALLOCATION TABLE EMPTY
4579 013620 005302 DEC R2 ;DECREMENT TABLE COUNT
4580 013622 001754 BEQ 5$ ;BR IF ITEM WERE LAST ENTRY
4581 013624 010103 MOV R1,R3 ;MOVE TABLE POINTER
4582 013626 062703 000004 ADD #4,R3 ;POINT TO NEXT ENTRY
4583 013632 012321 6$: MOV (R3)+,(R1)+ ;MOVE ITEMS
4584 013634 012321 MOV (R3)+,(R1)+
4585 013636 005302 DEC R2 ;DECREMENT TABLE COUNT
4586 013640 001374 BNE 6$ ;CONTINUE IF NOT AT END OF TABLE

```

4587	013642	000744		BK	5\$;RETURN
4588							
4589							
4590							;ROUTINE TO PUT BUFFER BACK IN TABLE
4591							
4592	013644	010146		RELBUF:	MOV	R1,-(SP)	;SAVE R1
4593	013646	012701	042004		MOV	#BUFTBL+2,R1	;BEGINNING OF TABLE
4594	013652	013702	042002		MOV	BUFTBL,R2	;ENTRY COUNT
4595	013656	001424			BEQ	2\$;BR IF EMPTY TABLE
4596	013660	016003	000020		MOV	\$WRDL(R0),R3	;TRIAL ADDRESS
4597	013664	006303			ASL	R3	;CHANGE TO BYTE COUNT
4598	013666	066003	000006		ADD	\$BUF(R0),R3	;ADDRESS OF HIGHER ADJACENT BLOCK
4599	013672	021103		1\$:	CMP	(R1),R3	;UPPER ADJACENT BLOCK
4600	013674	001432			BEQ	4\$;BR IF YES
4601	013676	062701	000004		ADD	#4,R1	;INCREMENT POINTER
4602	013702	005302			DEC	R2	;DECREMENT ENTRY COUNT
4603	013704	001372			BNE	1\$;CONTINUE SEARCHING
4604	013706	016011	000006		MOV	\$BUF(R0),(R1)	;PUT THE BUFFER BLOCK INTO THE TABLE
4605	013712	016061	000020	000002	MOV	\$WRDL(R0),2(R1)	;BLOCK SIZE
4606	013720	005237	042002		INC	BUFTBL	;INCREMENT ENTRY COUNT
4607	013724	005202			INC	R2	;INCREMENT R2 FOR USE LATER
4608	013726	000422			BR	5\$;SEE IF A LOWER ADJACENT BLOCK IS IN THE TABLE
4609	013730	016021	000006		2\$:	MOV	\$BUF(R0),(R1)+
4610	013734	016021	000020		MOV	\$WRDL(R0),(R1)+	;BLOCK ADDRESS TO TABLE
4611	013740	005237	042002		INC	BUFTBL	;SIZE TO TABLE
4612	013744	022737	000002	042002	3\$:	CMP	#2,BUFTBL
4613	013752	103001			BHIS	.+4	;INCREMENT ENTRY COUNT
4614	013754	000240			NOP		;SEE HOW MANY BLOCKS ARE IN TABLE
4615	013756	012601			MOV	(SP)+,R1	;BR IF NOT MORE THAN 2
4616	013760	000207			RTS	PC	;DEBUGGING AID
4617	013762	016011	000006		4\$:	MOV	\$BUF(R0),(R1)
4618	013766	066061	000020	000002	ADD	\$WRDL(R0),2(R1)	;RESTORE R1
4619	013774	010246			5\$:	MOV	R2,-(SP)
4620	013776	013702	042002		MOV	BUFTBL,R2	;RETURN
4621	014002	012705	042004		6\$:	MOV	#BUFTBL+2,R5
4622	014006	016504	000002		MOV	2(R5),R4	;RELEASED BUFFER IS LOWER ADJACENT
4623	014012	006304			ASL	R4	;INCREMENTED SIZE
4624	014014	061504			ADD	(R5),R4	;SAVE R2
4625	014016	020411			CMP	R4,(R1)	;ENTRY COUNT
4626	014020	001406			BEQ	8\$;BEGINNING OF TABLE
4627	014022	062705	000004		ADD	#4,R5	;BLOCK SIZE (IN WORDS)
4628	014026	005302			DEC	R2	;CHANGE TO BYTE COUNT
4629	014030	001366			BNE	6\$;ADD BLOCK BEGINNING ADDRESS
4630	014032	005726			TST	(SP)+	;R1 STILL POINTS TO INSERTED ENTRY
4631	014034	000743			BR	3\$;LOWER ADJACENT IN TABLE
4632	014036	012602			8\$:	MOV	(SP)+,R2
4633	014040	066165	000002	000002	ADD	2(R1),2(R5)	;INCREMENT POINTER
4634	014046	005337	042002		DEC	BUFTBL	;DECREMENT ENTRY COUNT
4635	014052	010105			MOV	R1,R5	;GET READY TO COMPRESS
4636	014054	062705	000004		ADD	#4,R5	;INCREMENT TO NEXT ENTRY
4637	014060	012521			9\$:	MOV	(R5)+,(R1)+
4638	014062	012521			MOV	(R5)+,(R1)+	;COMPRESS TABLE
4639	014064	005302			DEC	R2	;MOVE SIZE FIELD DOWN
4640	014066	001374			BNE	9\$;DECREMENT ENTRY COUNT
4641	014070	000725			BR	3\$;BR IF NOT FINISHED
4642							;FINISHED

```

4643
4644 ;FILL THE ASSIGNED BUFFER (IF WRITE OR WRITE CHECK ORDER)
4645
4646 014072 026027 000006 053654 FILBUF: CMP $BUF(RO),#ENDPGM ;MAKE SURE BUFFER IS ABOVE PROGRAM
4647 014100 103002 BHIS .+6 ;BR IF IT IS
4648 014102 000000 HALT ;BUFFER ADDRESS WITHIN PROGRAM, ILLEGAL
4649 014104 000776 BR .-2 ;INTERLOCK THE HALT
4650 014106 023760 001300 000020 CMP MAXDL,$WRDL(RO) ;SEE IF WORD LENGTH DOESN'T EXCEED MAXIMUM
4651 014114 103002 BHIS .+6 ;BR IF IT DOESN'T
4652 014116 000000 HALT ;WORD LENGTH FOR OPERATION GREATER THAN AVAIL MEM
4653 014120 000776 BR .-2 ;INTERLOCK THE HALT
4654 014122 132760 000004 000030 BITB #BIT02,$CODE(RO) ;SEE IF READ ORDER
4655 014130 001401 BEQ 1$ ;BR IF WRITE
4656 014132 000207 RTS PC ;RETURN
4657 014134 016001 000006 1$: MOV $BUF(RO),R1 ;BUFFER ADDRESS
4658 014140 016002 000020 MOV $WRDL(RO),R2 ;POSITIVE WORD COUNT
4659 014144 132760 000001 000030 BITB #BIT00,$CODE(RO) ;SEE IF WRITE HEADER TYPE ORDER
4660 014152 001413 BEQ 2$ ;BR IF NOT
4661 014154 016011 000012 MOV $CYL(RO),(R1) ;CYLINDER ADDRESS
4662 014160 052721 010000 BIS #BIT12,(R1)+ ;SET FMT22 BIT
4663 014164 016021 000010 MOV $SEC(RO),(R1)+ ;MOVE SECTOR & TRACK
4664 014170 005021 CLR (R1)+ ;CLEAR FIRST KEY WORD
4665 014172 005021 CLR (R1)+ ;CLEAR THE SECOND
4666 014174 162702 000004 SUB #4,R2 ;ADJUST THE WORD COUNT
4667 014200 003422 BLE 4$ ;BR IF END OF PATTERN
4668 014202 005004 2$: CLR R4 ;CLEAR R4
4669 014204 116004 000034 MOVB $PATT(RO),R4 ;RELATIVE PATTERN ADDRESS
4670 014210 001417 BEQ 5$ ;BR IF RANDOM DATA
4671 014212 016405 042646 MOV STNDAT(R4),R5 ;PATTERN ADDRESS
4672 014216 012703 000020 MOV #20,R3 ;PATTERN COUNT
4673 014222 012521 3$: MOV (R5)+,(R1)+ ;MOVE THE PATTERN INTO THE BUFFER
4674 014224 005302 DEC R2 ;DECREMENT THE WORD COUNT
4675 014226 001407 BEQ 4$ ;BR IF DONE (WORD COUNT = 0)
4676 014230 005303 DEC R3 ;DECREMENT THE PATTERN COUNT
4677 014232 001373 BNE 3$ ;BR IF MORE PATTERN
4678 014234 012703 000020 MOV #20,R3 ;RESTORE PATTERN COUNT
4679 014240 016405 042646 MOV STNDAT(R4),R5 ;RESTORE THE ADDRESS
4680 014244 000766 BR 3$ ;CONTINUE DISTRIBUTING THE PATTERN
4681 014246 000207 4$: RTS PC ;RETURN
4682 014250 132760 000002 000030 5$: BITB #BIT01,$CODE(RO) ;WRITE CHECK ORDER ?
4683 014256 001007 BNE 6$ ;BR IF NOT
4684 014260 016037 000076 027736 MOV $RSV(RO),$LONUM ;RESTORE THE RANDOM NUMBERS
4685 014266 016037 000100 027734 MOV $RSV+2(RO),$SHNUM ;TO RECREATE THE WRITE CHECK BUFFER
4686 014274 000410 BR 7$ ;FILL THE WRITE CHECK BUFFER
4687 014276 004737 027636 6$: JSR PC,$RAND ;GET TWO RANDOM NUMBERS
4688 014302 013760 027736 000076 MOV $LONUM,$RSV(RO) ;SAVE THE RANDOM NUMBERS
4689 014310 013760 027734 000100 MOV $SHNUM,$RSV+2(RO)
4690 014316 013721 027736 7$: MOV $LONUM,(R1)+ ;PUT FIRST WORD IN BUFFER
4691 014322 005302 DEC R2
4692 014324 001750 BEQ 4$ ;BR IF DONE
4693 014326 013721 027734 MOV $SHNUM,(R1)+ ;PUT SECOND WORD IN BUFFER
4694 014332 005302 DEC R2 ;DEC WORD COUNT
4695 014334 001744 BEQ 4$ ;BR IF DONE
4696 014336 000757 BR 6$
4697
4698 ;START THE ORDER FOR THE DPB IN RO

```

```

4699
4700 014340 010046          GODRIV: MOV      RD,-(SP)      ;SAVE RD
4701 014342 010037 014352  MOV      RD,1$          ;CURRENT DPB ADDRESS
4702 014346 004037 031442  JSR      RD,RPO4
4703 014352 000000          1$:      0              ;DPB ADDR GOES HERE
4704 014354 000000          HALT     ;DRIVER REJECTED REQUEST
4705 014356 012600          MOV      (SP)+,RD      ;RESTORE RD
4706 014360 062760 000001 000042  ADD      #1,$OPERC(RD) ;INCREMENT THE OPERATION COUNT
4707 014366 005560 000044  ADC      $OPERC+2(RD)
4708 014372 026060 000040 000012  CMP      $PREVA+2(RD),$CYL(RD) ;DID ORDER REQUIRE A CYLINDER CHANGE
4709 014400 001405          BEQ      2$           ;BR IF NOT
4710 014402 062760 000001 000046  ADD      #1,$POSIT(RD) ;INCREMENT SEEK COUNT
4711 014410 005560 000050  ADC      $POSIT+2(RD) ;ADD ANY CARRY
4712 014414 000207          2$:      RTS      PC
4713
4714          ;GENERATE PARAMETERS FOR THE OPERATION
4715
4716 014416 004737 027636          SELPAR: JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
4717 014422 032737 000001 001140  BIT      #SW0,SWR      ;SEE IF SW0 SET
4718 014430 001012          BNE      2$           ;BR IF SET - READ ONLY
4719 014432 012705 000010          1$:      MOV      #10,R5      ;READ/WRITE SELECTION DIVISOR
4720 014436 004737 024176  JSR      PC,GETREM     ;GET SELECTION VALUE
4721 014442 020537 001314  CMP      R5,RATIO     ;DETERMINE IF READ OR WRITE
4722 014446 003003          BGT      2$           ;BR IF READ
4723 014450 004737 014772  JSR      PC,RANWRT     ;SELECT A WRITE ORDER
4724 014454 000406          BR       3$           ;CONTINUE WITH THE SELECTION
4725 014456 013705 027736          2$:      MOV      $LONUM,R5   ;SELECT READ OPERATION CODE
4726 014462 042705 177776  BIC      #1C1,R5      ;MASK OUT ALL BUT BIT 0
4727 014466 062705 000004  ADD      #4,R5        ;TABLE OFFSET FOR READ CODE
4728 014472 110560 000102          3$:      MOVB     R5,$NCODE(RD) ;ORDER SELECTION CODE TO CONTROL BLOCK
4729
4730          ;GENERATE A RANDOM SECTOR ADDRESS BETWEEN VALUES 'MINSEC' & 'MAXSEC'
4731
4732 014476 013705 001334          MOV      MAXSEC,R5    ;GET MAXIMUM SECTOR ADDRESS
4733 014502 163705 001336          SUB      MINSEC,R5    ;SUBTRACT MINIMUM SECTOR ADDRESS
4734 014506 001403          BEQ      4$           ;BR IF MIN & MAX THE SAME
4735 014510 005205          INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4736 014512 004737 024176  JSR      PC,GETREM     ;GET THE RANDOM AUGMENT
4737 014516 063705 001336          4$:      ADD      MINSEC,R5 ;CONVERT TO ADDRESS
4738 014522 110560 000104          MOVB     R5,$NSEC(RD) ;STORE SECTOR ADDRESS IN DPB
4739
4740          ;GENERATE A RANDOM TRACK ADDRESS BETWEEN VALUES 'MINTRK' & 'MAXTRK'
4741
4742 014526 013705 001330          MOV      MAXTRK,R5   ;GET MAXIMUM TRACK ADDRESS
4743 014532 163705 001332          SUB      MINTRK,R5   ;SUBTRACT MINIMUM TRACK ADDRESS
4744 014536 001403          BEQ      5$           ;BR IF MIN & MAX THE SAME
4745 014540 005205          INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4746 014542 004737 024176  JSR      PC,GETREM     ;GET THE RANDOM TRACK AUGMENT
4747 014546 063705 001332          5$:      ADD      MINTRK,R5 ;CONVERT AUGMENT TO AN ADDRESS
4748 014552 110560 000105          MOVB     R5,$NTRK(RD) ;STORE TRACK ADDRESS IN DPB
4749
4750          ;GENERATE A RANDOM CYLINDER ADDRESS BETWEEN VALUES 'MINCYL' & 'MAXCYL'
4751
4752 014556 013705 001324          MOV      MAXCYL,R5   ;GET MAXIMUM CYLINDER ADDRESS
4753 014562 163705 001326          SUB      MINCYL,R5   ;SUBTRACT MINIMUM CYLINDER ADDRESS
4754 014566 001403          BEQ      6$           ;BR IF MIN & MAX THE SAME

```

```

4755 014570 005205          INC      R5          ;INCREMENT DIFFERENCE TO USE AS DIVISOR
4756 014572 004737 024176    JSR      PC,GETREM   ;GET THE RANDOM AUGMENT
4757 014576 063705 001326    6$:     ADD      MINCYL,R5 ;CONVERT AUGMENT TO AN ADDRESS
4758 014602 010560 000106    MOV      R5,$NCYL(RO);STORE CYLINDER ADDRESS IN DPB
4759 014606 122760 000003 000102  CMPB    #3,$NCODE(RO);WRITE HEADER & DATA ?
4760 014614 001444          BEQ      10$         ;BR IF IT IS
4761
4762                               ;GENERATE A RANDOM BUFFER LENGTH BETWEEN 4 & THE VALUE IN 'MAXDL'
4763
4764 014616 013705 001300    7$:     MOV      MAXDL,R5   ;GET BUFFER SIZE
4765 014622 005205          INC      R5          ;INCREMENT THE MAXIMUM SIZE
4766 014624 004737 024176    JSR      PC,GETREM   ;DIVIDE BY MAX VALUE
4767 014630 005705          TST      R5          ;IS THE REMAINDER 0 ?
4768 014632 001003          BNE      8$         ;NOT 0, CONTINUE
4769 014634 004737 027636    JSR      PC,$RAND    ;CYCLE THE RANDOM NUMBER GENERATOR
4770 014640 000766          BR       7$         ;TRY AGAIN
4771 014642 010560 000110    8$:     MOV      R5,$NWRDL(RO);WORD LENGTH TO CONTROL BLOCK
4772 014646 010546          MOV      R5,-(SP)   ;NEW WORD LENGTH ON STACK FOR CHECK
4773 014650 005046          CLR      -(SP)      ;MAKE UPPER DIVIDEND ZERO
4774 014652 012746 000400    MOV      #256,-(SP) ;SECTOR SIZE IS THE DIVISOR
4775 014656 132760 000001 000102  BITB    #1,$NCODE(RO);SEE IF NEXT ORDER IS A HEADER ORDER
4776 014664 001402          BEQ      .+6        ;BR IF NOT
4777 014666 062716 000004    ADD      #4,(SP)    ;ADD HEADER SIZE TO SECTOR SIZE
4778 014672 004737 024224    JSR      PC,LINKDV  ;DIVIDE BUFFER SIZE BY SECTOR SIZE
4779 014676 012616          MOV      (SP)+,(SP) ;MOV REMAINDER UP THE STACK
4780 014700 022627 000004    CMP      (SP)+,#4   ;SEE IF REMAINDER LESS THAN 4
4781 014704 103003          BHIS    9$         ;BR IF NOT
4782 014706 004737 027636    JSR      PC,$RAND    ;CYCLE THE RANDOM NUMBER GENERATOR
4783 014712 000741          BR       7$         ;TRY AGAIN
4784 014714 122760 000002 000102  9$:     CMPB    #2,$NCODE(RO);SEE IF WRITE DATA
4785 014722 001017          BNE      12$        ;BR IF NOT WRITE DATA
4786 014724 000412          BR       11$        ;GET PATTERN
4787
4788                               ;SETUP BUFFER SIZE FOR A WRITE HEADER AND DATA ORDER
4789
4790 014726 012760 000404 000110 10$:     MOV      #260,$NWRDL(RO);CHANGE WORD LENGTH TO 260 FOR WRTHD ORDER
4791 014734 023727 001300 000404    CMP      MAXDL,#260;CAN A FULL SECTOR BE WRITTEN ?
4792 014742 103003          BHIS    11$        ;BR IF IT CAN
4793 014744 013760 001300 000110    MOV      MAXDL,$NWRDL(RO);CHANGE TRANSFER SIZE
4794 014752 004737 015076    11$:     JSR      PC,GETPAT  ;GET PATTERN CODE
4795 014756 110560 000103          MOVB    R5,$NPATC(RO);MOVE PATTERN CODE TO CONTROL BLOCK
4796 014762 012760 177777 000112 12$:     MOV      #-1,$NEXT(RO);SET PARAMETERS SELECTED INDICATOR
4797 014770 000207          RTS      PC         ;RETURN
4798
4799                               ;ROUTINE TO SELECT A WRITE (OR WRITE CHECK) OPERATION
4800
4801 014772 012705 000004          RANWRT: MOV      #4,R5   ;WRITE OPERATION SELECTION DIVISOR
4802 014776 004737 024176          JSR      PC,GETREM  ;GET SELECTION CODE
4803 015002 005737 001316          TST      AUTOCK    ;ARE WRITE CHECK ORDERS TO BE SELECTED
4804                               ;RANDOMLY ?
4805 015006 001403          BEQ      1$         ;BR IF THEY ARE
4806 015010 152705 000002          BISB    #2,R5      ;SET CODE TO EXCLUDE WRITE CHECK ORDERS
4807 015014 000420          BR       3$         ;COMPLETE SELECTION
4808 015016 020527 000001          1$:     CMP      R5,#1   ;WRITE CHECK SELECTED ?
4809 015022 101015          BHI     3$         ;BR IF NOT
4810 015024 132760 000002 000030  BITB    #2,$CODE(RO);PREVIOUS WRITE OPERATION ?

```

```

4811 015032 001407          BEQ      2$          ;BR IF PREVIOUS WAS READ OR WRITE CHECK
4812 015034 116060 000030 000102      MOVB    $CODE(RO), $NCODE(RO) ;MOVE CODE TO 'NEXT CODE'
4813 015042 142760 000002 000102      BICB    #2, $NCODE(RO) ;CHANGE WRITE TO WRITE CHECK
4814 015050 000411          BR       5$          ;EXIT
4815 015052 052705 000002          2$:     BIS      #2, R5 ;CHANGE WRITE CHECK TO WRITE
4816 015056 005737 001310          3$:     TST     FORMAT ;WRITE HEADER ORDERS ALLOWED ?
4817 015062 001002          BNE     4$          ;BR IF THEY ARE
4818 015064 042705 000001          BIC     #1, R5 ;ALTER POSSIBLE WRITE HEADER
4819 015070 110560 000102          4$:     MOVB    R5, $NCODE(RO) ;SETUP 'NEXT' CODE
4820 015074 000207          5$:     RTS     PC ;RETURN
4821
4822          ;ROUTINE TO SELECT A PATTERN
4823
4824 015076 012705 000020          GETPAT: MOV    #20, R5 ;SELECT PATTERN
4825 015102 004737 024176          JSR    PC, GETREM ;GET CODE
4826 015106 005705          TST    R5 ;WAS PATTERN ZERO SELECTED ?
4827 015110 001003          BNE    1$          ;BR IF NOT ZERO
4828 015112 004737 027636          JSR    PC, $RAND ;CYCLE THE RANDOM NUMBER GENERATOR
4829 015116 000767          BR     GETPAT ;TRY AGAIN
4830 015120 006305          1$:     ASL    R5 ;MAKE CODE INTO TABLE INDEX
4831 015122 000207          RTS     PC
4832
4833          ;ROUTINE TO GET THE PREVIOUSLY SELECTED PARAMETER VALUES
4834
4835 015124 010546          GETPAR: MOV    R5, -(SP) ;SAVE R5
4836 015126 116060 000030 000032      MOVB    $CODE(RO), $PREVO(RO) ;SAVE CURRENT PARAMETERS
4837 015134 032760 000006 000102      BIT     #6, $NCODE(RO) ;SEE IF NEXT OPERATION IS READ OR WRITE
4838 015142 001007          BNE    1$          ;BR IF EITHER
4839 015144 016060 000012 000040          MOV    $CYL(RO), $PREVA+2(RO) ;SAVE STARTING CYLINDER
4840 015152 016060 000010 000036          MOV    $SEC(RO), $PREVA(RO) ;SAVE STARTING SECTOR AND TRACK
4841 015160 000405          BR     2$
4842 015162 004737 015624          1$:     JSR    PC, PREVST ;SAVE CURRENT SECTOR & TRACK
4843 015166 016060 000222 000040          MOV    $RACC(RO), $PREVA+2(RO) ;CURRENT CYLINDER
4844 015174 032737 000100 001140          2$:     BIT     #SW06, SWR ;SWITCH 6 SET ?
4845 015202 001043          BNE    3$          ;BR IF SET
4846 015204 116060 000102 000030          MOVB    $NCODE(RO), $CODE(RO) ;LOGICAL CODE FOR OPERATION
4847 015212 116005 000102          MOVB    $NCODE(RO), R5 ;LOAD R5 FOR USE AS TABLE INDEX
4848 015216 116560 042144 000002          MOVB    COMBL(R5), $COMND(RO) ;RPO4 COMMAND CODE
4849 015224 116060 000103 000034          MOVB    $NPATC(RO), $PATTC(RO) ;PATTERN CODE
4850 015232 016060 000104 000010          MOV    $NSEC(RO), $SEC(RO) ;TRACK AND SECTOR ADDRESSES
4851 015240 016060 000106 000012          MOV    $NCYL(RO), $CYL(RO) ;CYLINDER ADDRESS
4852 015246 016060 000110 000020          MOV    $NWARDL(RO), $WRDL(RO) ;BUFFER SIZE
4853 015254 016060 000110 000004          MOV    $NWARDL(RO), $WRDM(RO) ;WORD COUNT FOR THE RH11
4854 015262 005460 000004          NEG    $WRDM(RO) ;COMPLEMENT IT
4855 015266 012760 000400 000026          MOV    #256, $SSEC(RO) ;INITIAL VALUE OF SECTOR SIZE
4856 015274 032760 000001 000030          BIT     #1, $CODE(RO) ;HEADER OPERATION ?
4857 015302 001403          BEQ    3$          ;BR IF NOT
4858 015304 062760 000004 000026          ADD    #4, $SSEC(RO) ;ADD HEADER SIZE
4859 015312 005060 000112          3$:     CLR    $NEXT(RO) ;RESET 'PARAMETERS LOADED' INDICATOR
4860 015316 012605          MOV    (SP)+, R5 ;RESTORE R5
4861 015320 000207          RTS     PC ;RETURN
4862
4863          ;ROUTINE TO COMPRESS THE TABLE IN R1
4864
4865 015322 016111 000002          CMPRES: MOV    2(R1), (R1) ;COMPRESS THE TABLE IN R1
4866 015326 001403          BEQ    1$          ;BR WHEN ZERO FOUND

```

```

4867 015330 062701 000002          ADD      #2,R1          ;INCREMENT R1
4868 015334 000772          BR       CMPRES       ;CONTINUE COMPRESSING TABLE
4869 015336 000207          1$:     RTS          PC          ;RETURN
4870
4871          ;ROUTINE TO SETUP PARAMETERS FOR A SEQUENTIAL READ OR WRITE OF THE DISK
4872
4873 015340 004737 027636          WRTPK:  JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
4874 015344 005760 000044          TST     $OPERC+2(R0)     ;SEE IF FIRST OPERATION
4875 015350 001007          BNE     WRTPK1         ;BR IF UPPER WORD OF COUNTER NOT ZERO
4876 015352 005760 000042          TST     $OPERC(R0)      ;LOWER WORD ZERO ?
4877 015356 001004          BNE     WRTPK1         ;BR IF NOT 1ST OPERATION
4878 015360 105760 000031          TSTB   $PACK(R0)       ;SEE WHICH - 'R' OR 'W'
4879 015364 100477          SMI     WRTPK3         ;BR IF 'W'
4880 015366 000464          BR      WRTPK2         ;'R' OPERATION
4881 015370 116060 000030 000032          WRTPK1: MOV     $CODE(R0),$PREV(R0) ;SAVE CURRENT PARAMETERS
4882 015376 004737 015624          JSR     PC,PREVST      ;SAVE CURRENT SECTOR & TRACK
4883 015402 016060 000222 000040          MOV     $RACC(R0),$PREVA+2(R0) ;CURRENT CYLINDER
4884 015410 016060 000172 000010          MOV     $RHDA(R0),$SEC(R0) ;NEW SECTOR & TRACK ADDRESS
4885 015416 016060 000220 000012          MOV     $RHCA(R0),$CYL(R0) ;NEW CYLINDER ADDRESS
4886 015424 026037 000012 001324          CMP     $CYL(R0),MAXCYL ;SEE IF AT END
4887 015432 103427          BLO     2$            ;BR IF LESS THAN 'MAXCYL'
4888 015434 101004          BHI     1$            ;BR IF GREATER THAN 'MAXCYL'
4889 015436 126037 000011 001330          CMP     $TRK(R0),MAXTRK ;SEE IF AT MAX TRACK
4890 015444 101422          BLOS   2$            ;BR IF NOT GREATER
4891 015446 113760 001332 000011          1$:     MOV     $MINTRK,$TRK(R0) ;RESET TRACK ADDRESS
4892 015454 113760 001336 000010          MOV     $MINSEC,$SEC(R0) ;RESET SECTOR ADDRESS
4893 015462 013760 001326 000012          MOV     $MINCYL,$CYL(R0) ;RESET CYLINDER ADDRESS
4894 015470 004737 024002          JSR     PC,NRML2      ;DROP THE UNIT (NORMAL TERMINATION)
4895 015474 032737 000020 001140          BIT     #SW04,$SWR     ;IS SWITCH 4 SET ?
4896 015502 001003          BNE     2$            ;BR IF SET
4897 015504 005726          TST     (SP)+         ;INCREMENT THE STACK POINTER
4898 015506 000137 003206          JMP     MAIN          ;RETURN DIRECTLY TO 'MAIN'
4899 015512 013760 001300 000020          2$:     MOV     $MAXDL,$WRDL(R0) ;BUFFER SIZE IS MAXIMUM
4900 015520 013760 001300 000004          MOV     $MAXDL,$WRDM(R0) ;WORD COUNT
4901 015526 005460 000004          NEG     $WRDM(R0)     ;CHANGE WORD COUNT TO 2'S COMPLEMENT
4902 015532 105760 000031          TSTB   $PACK(R0)     ;READ OR WRITE ?
4903 015536 100412          BMI     WRTPK3         ;BR IF WRITE
4904 015540 012760 000404 000026          WRTPK2: MOV     #260,$SSEC(R0) ;SECTOR SIZE FOR READ
4905 015546 112760 000005 000030          MOV     #5,$CODE(R0) ;CODE FOR READ HEADER & DATA
4906 015554 112760 000173 000002          MOV     #RDHD,$COMND(R0) ;DRIVE CODE FOR OPERATION
4907 015562 000415          BR      WRTPK4         ;SET UP FOR EXIT
4908 015564 012760 000400 000026          WRTPK3: MOV     #256,$SSEC(R0) ;SECTOR SIZE
4909 015572 112760 000002 000030          MOV     #2,$CODE(R0) ;CODE FOR WRTDAT
4910 015600 112760 000161 000002          MOV     #WRTDAT,$COMND(R0) ;OP CODE
4911 015606 004737 015076          JSR     PC,GETPAT     ;GET PATTERN CODE
4912 015612 110560 000034          MOV     $R5,$PATT(R0) ;PATTERN CODE
4913 015616 005060 000112          WRTPK4: CLR     $NEXT(R0) ;CLEAR 'PARAMETER SELECTED' INDICATOR
4914 015622 000207          RTS          PC          ;RETURN
4915
4916          ;DECREMENT AND SAVE CURRENT SECTOR & TRACK ADDRESS
4917
4918 015624 005046          PREVST: CLR     -(SP) ;MAKE ROOM ON THE STACK
4919 015626 005046          CLR     -(SP) ;MAKE ROOM ON THE STACK
4920 015630 004737 020124          JSR     PC,READDR    ;DECREMENT SECTOR-TRACK ADDRESS
4921 015634 112660 000037          MOV     (SP)+,$PREVA+1(R0) ;SAVE CURRENT TRACK
4922 015640 112660 000036          MOV     (SP)+,$PREVA(R0) ;SAVE CURRENT SECTOR

```



```

4923 015644 000207          RTS      PC          ;RETURN
4924
4925
4926          ;ROUTINE TO DETERMINE OF ERROR IS AT A LOCATION ON THE PACK DEFINED
4927          ;IN THE BAD TRACK/SECTOR TABLE FOR THE DRIVE.
4928          ;RETURN IF ERROR IS AT AN ADDRESS WHICH IS IN THE TABLE
4929          ;RETURN+2 IS THE ERROR ADDRESS IS NOT CONTAINED IN THE TABLE
4930          ;OR IF THE PARAMETER 'NOTPRT' IS 0
4931
4932          SPOTCK:
4933 015646 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
4934 015650 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
4935 015652 012701 000114    MOV      #SBDSEC,R1    ;;INCREMENT FOR BAD SECTOR TABLE
4936 015656 060001          ADD      RD,R1         ;;ADD THE BLOCK'S STARTING ADDRESS
4937 015660 012702 000010    MOV      #8,R2         ;;BAD SECTOR TABLE SIZE COUNT
4938 015664 021160 000222    1$:     CMP      (R1),SRHCC(RD) ;;IS CYLINDER IN THE TABLE ?
4939 015670 001021          BNE      4$           ;;BR IF NOT
4940 015672 005046          CLR      -(SP)        ;;MAKE ROOM ON THE STACK
4941 015674 005046          CLR      -(SP)        ;;MAKE ROOM ON THE STACK
4942 015676 004737 020124    JSR      PC,READDR     ;;DECREMENT THE SECTOR/TRACK ADDRESS
4943 015702 122661 000003    CMPB    (SP)+,3(R1)    ;;COMPARE THE TRACK ADDRESS
4944 015706 001011          BNE      3$           ;;BR IF IT IS NOT EQUAL
4945 015710 105761 000002    TSTB    2(R1)         ;;IS A SECTOR ADDRESS IN THE TABLE ?
4946 015714 100002          BPL      2$           ;;BR IF ONE IS
4947 015716 005726          TST      (SP)+        ;;INCREMENT THE STACK POINTER
4948 015720 000414          BR       5$           ;;DISPLAY THE MESSAGE
4949 015722 122661 000002    2$:     CMPB    (SP)+,2(R1) ;;COMPARE THE SECTOR ADDRESS
4950 015726 001002          BNE      4$           ;;BR IF NOT EQUAL
4951 015730 000410          BR       5$           ;;CHECK 'NOTPRT'
4952 015732 005726          3$:     TST      (SP)+        ;;INCREMENT THE STACK POINTER
4953 015734 062701 000004    4$:     ADD      #4,R1     ;;GO TO THE NEXT LOCATION IN THE TABLE
4954 015740 005711          TST      (R1)         ;;PAST THE TABLE ENTRIES ?
4955 015742 100411          BMI      6$           ;;BR IF PAST
4956 015744 005302          DEC      R2           ;;DECREMENT THE MAXIMUM ENTRY COUNT
4957 015746 001346          BNE      1$           ;;BR IF MORE TO CHECK
4958 015750 000406          BR       6$           ;;END, EXIT
4959 015752 005737 001320    5$:     TST      NOTPRT     ;;PRINT THE ERROR ANYWAY ?
4960 015756 001006          BNE      7$           ;;BR IF NOT
4961 015760 012737 177777 016416    MOV      #-1,PRT2B    ;;SET THE INDICATOR FOR THE IDENTIFICATION LINE
4962 015766 062766 000002 000004    6$:     ADD      #2,4(SP)  ;;INCREMENT THE RETURN
4963 015774          7$:
4964 015774 012602          MOV      (SP)+,R2     ;;POP STACK INTO R2
4965 015776 012601          MOV      (SP)+,R1     ;;POP STACK INTO R1
4966 016000 000207          RTS      PC          ;RETURN
4967
4968          ;*****
4969
4970          .SBTTL  ERROR MESSAGE GENERATION ROUTINES
4971
4972          ;*****
4973
4974          ;PRINT LINE 1 OF ERROR MESSAGE
4975
4976 016002 032737 002000 001140    LINE1:  BIT      #SW10,SWR  ;;SWITCH 10 SET ?
4977 016010 001402          BEQ     .+6          ;;BR IF NOT
4978 016012 104401 001160          TYPE    ,SBELL      ;;RING THE BELL

```

```

4979 016016 104412 001165      DISPLY  ,SCLF      ;CR-LF
4980 016022 005737 001210      TST     CLKFLG    ;CLOCK ON THE SYSTEM ?
4981 016026 001006      BNE     1$        ;BR IF NONE IS
4982 016030 104412 021702      DISPLY  ,HOUR      ;TYPE THE HOURS
4983 016034 104412 021706      DISPLY  ,MINUTE    ;TYPE THE MINUTES
4984 016040 104412 021712      DISPLY  ,SECOND   ;TYPE THE SECONDS
4985 016044 000207      RTS     PC        ;RETURN & TYPE DESCRIPTION
4986
4987      ;PRINT LINE 2 OF ERROR MESSAGE
4988
4989 016046 104412 001165      LINE2: DISPLY  ,SCLF      ;CR-LF
4990 016052 104412 047370      DISPLY  ,LIN2C    ;'PRESENT ORDER = '
4991 016056 005037 016166      CLR     2$        ;CLEAR LOCN FOR ORDER CODE
4992 016062 116037 000030 016166      MOVB   $CODE(RO),2$ ;ORDER CODE
4993 016070 004737 016142      JSR    PC,1$      ;PRINT THE MNEMONIC OF THE ORDER
4994 016074 104412 047411      DISPLY  ,LIN2P    ;'PREVIOUS ORDER = '
4995 016100 005037 016166      CLR     2$        ;CLEAR FOR PREVIOUS ORDER CODE
4996 016104 116037 000032 016166      MOVB   $PREVO(RO),2$ ;PREVIOUS ORDER CODE
4997 016112 004737 016142      JSR    PC,1$      ;PRINT THE MNEMONIC
4998 016116 032760 000040 000016      BIT    #BITS,$STATUS(RO) ;DID ERROR OCCUR DURING NON-DATA TRANSFER
4999 016124 001422      BEQ    LINE2A    ;BR IF NOT
5000 016126 104412 001165      DISPLY  ,SCLF      ;CR-LF
5001 016132 104412 047435      DISPLY  ,LIN2M    ;'ERROR OCCURED DURING NON-DATA TRANSFER'
5002 016136 000137 016172      JMP    LINE2A    ;PRINT THE REST OF THE ERROR LINE
5003 016142 006337 016166      1$:    ASL     2$        ;CONVERT CODE INTO A TABLE INDEX
5004 016146 006337 016166      ASL     2$        ;CONVERT CODE INTO A TABLE INDEX
5005 016152 006337 016166      ASL     2$        ;CONVERT CODE INTO A TABLE INDEX
5006 016156 062737 042152 016166      ADD    #OPMNEM,2$ ;ADD THE TABLE BASE ADDRESS
5007 016164 104412      DISPLY  ,          ;PRINT THE MNEMONIC
5008 016166 000000      2$:    .WORD   0
5009 016170 000207      RTS     PC        ;RETURN
5010 016172 005737 016416      LINE2A: TST     PRT2B   ;PRINT THE BAD SECTOR LINE ?
5011 016176 001404      BEQ    LINE2B    ;BR IF NOT
5012 016200 104412 001165      DISPLY  ,SCLF      ;CR-LF
5013 016204 104412 047516      DISPLY  ,LIN2S    ;ERROR ADDRESS DEFINED AS BAD AREA
5014 016210 010546      LINE2B: MOV     R5,-(SP)  ;SAVE R5
5015 016212 005005      CLR     R5        ;CLEAR R5 FOR DRIVE ADDRESS
5016 016214 111005      MOVB   (RO),R5   ;MOVE DRIVE# FROM CURRENT BLOCK
5017 016216 142737 000007 001263      BICB   #7,UNIT+1 ;CLEAR UNIT NUMBER
5018 016224 150537 001263      BISB   R5,UNIT+1 ;LOAD UNIT NUMBER
5019 016230 006305      ASL     R5        ;MAKE DRIVE NUMBER INTO A TABLE INDEX
5020 016232 016537 042566 016410      MOV     DT14.T(R5),$LINAD ;ADDRESS OF LOCATIONS TO PRINT
5021 016240 016537 042606 016412      MOV     DT15.T(R5),$LINAD+2 ;OPTIONAL REGISTERS
5022 016246 016537 042626 016414      MOV     DT16.T(R5),$LINAD+4 ;MORE OPTIONAL REGISTERS
5023 016254 104412 001165      DISPLY  ,SCLF      ;CR-LF
5024 016260 104412 046235      DISPLY  ,DH14     ;STANDARD RPO4 REGISTER HEADER
5025 016264 104412 001262      DISPLY  ,UNIT     ;PRINT THE UNIT ADDRESS
5026 016270 104412 051364      DISPLY  ,LINSF    ;SPACES
5027 016274 004737 016356      JSR    PC,2$      ;PRINT THE REGISTERS
5028 016300 032737 000040 001140      BIT    #SW05,SWR  ;PRINT THE OPTIONAL REGISTERS ?
5029 016306 001021      BNE     1$        ;BR IF NOT
5030 016310 104412 046341      DISPLY  ,DH15     ;
5031 016314 013737 016412 016410      MOV     $LINAD+2,$LINAD ;SHIFT ADDRESSES
5032 016322 013737 016414 016412      MOV     $LINAD+4,$LINAD+2 ;FINISH SHIFT
5033 016330 004737 016356      JSR    PC,2$      ;PRINT THEM
5034 016334 104412 046440      DISPLY  ,DH16

```

```

5035 016340 013737 016412 016410      MOV      $LINAD+2,$LINAD      ;FINISH SHIFTING THE ADDRESSES
5036 016346 004737 016356              JSR      PC,2$              ;PRINT THE REGISTERS
5037 016352 012605              1$:     MOV      (SP)+,R5      ;RESTORE R5
5038 016354 000207              RTS      PC
5039 016356 013705 016410      2$:     MOV      $LINAD,R5      ;ADDRESS OF REGISTERS
5040 016362 005715              3$:     TST      (R5)          ;AT END OF LINE ?
5041 016364 001406              BEQ      4$                ;BR IF YES
5042 016366 013546              MOV      2(R5)+,-(SP)      ;PUT THE CONTENTS ON THE STACK
5043 016370 004737 024740      JSR      PC,PRT0CT        ;PRINT THE NUMBER
5044 016374 104412 051364      DISPLY  ,LINSF           ;PRINT 2 SPACES
5045 016400 000770              BR       3$                ;CONTINUE PRINTING
5046 016402 104412 001165      4$:     DISPLY  ,%CRLF        ;CR-LF
5047 016406 000207              RTS      PC
5048
5049 016410 000000 000000 000000  $LINAD: .WORD 0,C,0        ;ADDRESS OF REGISTERS
5050 016416 000000              PRT2B: .WORD 0            ;BAD SECTOR LINE INDICATOR
5051
5052              ;PRINT LINES 3 & 3A OF ERROR MESSAGE
5053
5054 016420 104412 047552      LINE3:  DISPLY  ,LINM3        ;LINE 3 ENTRANCE
5055 016424 000402              BR       LIN3.1           ;FINISH PRINTOUT
5056 016426 104412 047570      LINE3A: DISPLY  ,LINN3        ;LINE 3A ENTRANCE
5057 016432 016046 000222      LIN3.1: MOV      $RHCC(RO),-(SP) ;PUT CYLINDER ADDR ON STACK
5058 016436 004737 030232      JSR      PC,$$SB2D        ;CONVERT TO DECIMAL
5059 016442 004737 025576      JSR      PC,$$SUPRS        ;PRINT CYL ADDR
5060 016446 104412 047565      DISPLY  ,T                ;PRINT ' T '
5061 016452 005046              CLR      -(SP)            ;CLEAR STACK FOR SECTOR ADDR
5062 016454 005046              CLR      -(SP)            ;CLEAR STACK FOR TRACK ADDR
5063 016456 004737 020124      JSR      PC,READDR        ;SET DECREMENTED ADDRESS
5064 016462 004737 030232      JSR      PC,$$SB2D        ;NOW CONVERT TRACK
5065 016466 004737 025576      JSR      PC,$$SUPRS        ;PRINT TRACK VALUE
5066 016472 004737 030232      JSR      PC,$$SB2D        ;CONVERT SECTOR
5067 016476 104412 047611      DISPLY  ,S                ;PRINT ' S '
5068 016502 004737 025576      JSR      PC,$$SUPRS        ;PRINT SECTOR
5069 016506 104412 047614      DISPLY  ,LINP3           ;PRINT 'PREV ADDR'
5070 016512 016046 000040      MOV      $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
5071 016516 004737 030232      JSR      PC,$$SB2D        ;CONVERT CYLINDER
5072 016522 004737 025576      JSR      PC,$$SUPRS        ;PRINT CYLINDER
5073 016526 104412 047565      DISPLY  ,T                ;PRINT ' T '
5074 016532 005046              CLR      -(SP)            ;MAKE ROOM ON THE STACK
5075 016534 116016 000037      MOVB    $PREVA+1(RO),(SP) ;PREVIOUS TRACK ADDRESS
5076 016540 004737 030232      JSR      PC,$$SB2D        ;CONVERT TRACK
5077 016544 004737 025576      JSR      PC,$$SUPRS        ;PRINT TRACK
5078 016550 104412 047611      DISPLY  ,S                ;PRINT ' S '
5079 016554 005046              CLR      -(SP)            ;MAKE ROOM ON THE STACK
5080 016556 116016 000036      MOVB    $PREVA(RO),(SP)   ;PREVIOUS SECTOR ADDRESS
5081 016562 004737 030232      JSR      PC,$$SB2D        ;CONVERT SECTOR
5082 016566 004737 025576      JSR      PC,$$SUPRS        ;PRINT SECTOR
5083 016572 104412 001165      DISPLY  ,%CRLF
5084 016576 000207              RTS      PC
5085
5086              ;PRINT LINES 3B & 3C OF ERROR MESSAGE
5087
5088 016600 004737 016622      LINE3B: JSR      PC,LIN3.3    ;LINE 3B ENTRANCE
5089 016604 104412 001165      DISPLY  ,%CRLF
5090 016610 000207              RTS      PC

```

```

5091 016612 004737 016622      LINE3C: JSR      PC,LIN3.3      ;LINE 3C ENTRANCE
5092 016616 000137 016654      JMP      LIN3.4      ;FINISH MESSAGE
5093
5094 016622 104412 047635      LIN3.3: DISPLY   LINS3      ;LINE '3B & 3C' ENTRANCE
5095 016626 016046 000040      MOV     $PREVA+2(RO),-(SP) ;PREVIOUS CYLINDER
5096 016632 004737 016714      JSR     PC,LIN3.2      ;CONVERT & TYPE
5097 016636 104412 047652      DISPLY  LINEN3      ;PRINT 'END CYL'
5098 016642 016046 000222      MOV     $RHCC(RO),-(SP) ;PRESENT CYLINDER
5099 016646 004737 016714      JSR     PC,LIN3.2      ;CONVERT & TYPE
5100 016652 000207
5101
5102 016654 104412 047667      LIN3.4: DISPLY   LINA3      ;PRINT 'ACTUAL'
5103 016660 013746 053644      MOV     CYLDER,-(SP)    ;ACTUAL CYLINDER
5104 016664 004737 016714      JSR     PC,LIN3.2      ;CONVERT & TYPE
5105 016670 104412 047707      DISPLY  LINT3      ;PRINT TRACK
5106 016674 005046      CLR     -(SP)          ;CLEAR STACK WORD
5107 016676 116016 000173      MOVB   $RHDA+1(RO),(SP) ;PUT TRACK ON STACK
5108 016702 004737 016714      JSR     PC,LIN3.2      ;CONVERT & TYPE
5109 016706 104412 001165      DISPLY  $CRLF
5110 016712 000207      RTS     PC
5111
5112 016714 016646 000002      LIN3.2: MOV     2(SP),-(SP) ;SET UP STACK FOR CONVERT
5113 016720 004737 030232      JSR     PC,$SB2D      ;CONVERT
5114 016724 004737 025576      JSR     PC,$SUPRS     ;TYPE
5115 016730 012616      MOV     (SP)+,(SP)    ;RESTORE STACK POINTER
5116 016732 000207      RTS     PC
5117
5118      ;PRINT LINE 3D OF ERROR MESSAGE
5119
5120 016734 032737 000040 001140 LINE3D: BIT     #SW05,SWR ;SWITCH 5 SET ?
5121 016742 001416      BEQ     1$           ;BR IF IT IS
5122 016744 104412 047741      DISPLY  LINB3      ;'RHBA = '
5123 016750 016046 000170      MOV     $RHBA(RO),-(SP) ;BUFFER ADDR REG CONTENTS
5124 016754 004737 017236      JSR     PC,LIN4.1    ;CONVERT & PRINT IT
5125 016760 104412 047751      DISPLY  LINW3      ;'RHWC = '
5126 016764 016046 000166      MOV     $RHWC(RO),-(SP) ;WORD COUNT REGISTER CONTENTS
5127 016770 004737 017236      JSR     PC,LIN4.1    ;CONVERT IT & PRINT IT
5128 016774 104412 001165      DISPLY  $CRLF
5129 017000 000207      1$:     RTS     PC
5130
5131      ;PRINT LINE 3E OF ERROR MESSAGE
5132
5133 017002 104412 047635      LINE3E: DISPLY   LINS3      ;'START CYL = '
5134 017006 016046 000012      MOV     $CYL(RO),-(SP) ;MOVE CYL TO STACK
5135 017012 004737 030232      JSR     PC,$SB2D      ;CONVERT IT
5136 017016 004737 025576      JSR     PC,$SUPRS     ;PRINT IT
5137 017022 104412 051364      DISPLY  LINS3      ;SPACES
5138 017026 104412 047763      DISPLY  LINST3     ;'START TRK = '
5139 017032 005046      CLR     -(SP)        ;CLEAR STACK
5140 017034 116016 000011      MOVB   $TRK(RO),(SP) ;TRACK TO STACK
5141 017040 004737 030232      JSR     PC,$SB2D      ;CONVERT IT
5142 017044 004737 025576      JSR     PC,$SUPRS     ;PRINT IT
5143 017050 104412 051364      DISPLY  LINS3      ;SPACES
5144 017054 104412 050000      DISPLY  LINS3      ;'START SEC = '
5145 017060 005046      CLR     -(SP)        ;CLEAR STACK
5146 017062 116016 000010      MOVB   $SEC(RO),(SP) ;SECTOR ADDR TO STACK

```

```

5147 017066 004737 030232      JSR      PC,$SB2D      ;CONVERT IT
5148 017072 004737 025576      JSR      PC,$SUPRS     ;PRINT IT
5149 017076 104412 001165      DISPLY   $CRLF
5150 017102 000207      RTS      PC
5151
5152                                ;PRINT LINE 3F OF ERROR MESSAGE
5153
5154 017104 032737 000040 001140 LINE3F: BIT      #SWS,SWR      ;SWITCH 5 SET ?
5155 017112 001420      BEQ      1$           ;BR IF NOT
5156 017114 104412 047731      DISPLY   LINDA3      ;'RHDA = '
5157 017120 016046 000172      MOV      $RHDA(RO),-(SP) ;DESIRED ADDRESS ON STACK
5158 017124 004737 024740      JSR      PC,PRTOCT   ;PRINT IT
5159 017130 104412 051364      DISPLY   LINS4       ;SPACES
5160 017134 104412 047720      DISPLY   LINCA3      ;'RHCA = '
5161 017140 016046 000220      MOV      $RHCA(RO),-(SP) ;DESIRED CYLINDER ADDRESS
5162 017144 004737 024740      JSR      PC,PRTOCT   ;PRINT IT
5163 017150 104412 001165      DISPLY   $CRLF
5164 017154 000207      RTS      PC
5165
5166                                ;PRINT LINE 4 OF ERROR MESSAGE
5167
5168 017156 104412 050015      LINE4: DISPLY  LINM4      ;'PRINT BUFFER'
5169 017162 016046 000006      MOV      $BUF(RO),-(SP) ;BUFFER ADDR ON STACK
5170 017166 004737 017236      JSR      PC,LIN4.1    ;CONVERT TO OCTAL & PRINT
5171 017172 104412 050034      DISPLY   LINS4       ;PRINT 'SIZE'
5172 017176 016046 000020      MOV      $WRDL(RO),-(SP) ;BUFFER SIZE
5173 017202 004737 016714      JSR      PC,LIN3.2    ;CONVERT TO DEC & PRINT
5174 017206 104412 050046      DISPLY   LINX4       ;'ACTUAL NMBR WRDS XFRD = '
5175 017212 016046 000170      MOV      $RHBA(RO),-(SP) ;VALUE IN BUFFER ADDR REGISTER
5176 017216 166016 000006      SUB      $BUF(RO),(SP) ;SUBTRACT STARTING ADDRESS
5177 017222 006216      ASR      (SP)         ;CONVERT INTO A WORD COUNT
5178 017224 004737 016714      JSR      PC,LIN3.2    ;CONVERT TO DEC & PRINT
5179 017230 104412 001165      DISPLY   $CRLF
5180 017234 000207      RTS      PC
5181
5182 017236 016646 000002      LIN4.1: MOV     2(SP),-(SP) ;PUT VALUE IN PROPER LOCN ON STACK
5183 017242 004737 030406      JSR      PC,$SB2D    ;CONVERT TO OCTAL
5184 017246 004737 025576      JSR      PC,$SUPRS   ;PRINT IT
5185 017252 012616      MOV      (SP)+,(SP)  ;RESTORE STACK POINTER
5186 017254 000207      RTS      PC
5187
5188                                ;PRINT LINE 5 OF ERROR MESSAGE
5189
5190 017256 104412 050101 000170 LINES: DISPLY  LIND5      ;PRINT 'GOOD DATA'
5191 017262 162760 000002      SUB      #2,$RHBA(RO) ;BACK THE ADDRESS UP
5192 017270 017046 000170      MOV      2,$RHBA(RO),-(SP) ;'GOOD' DATA - AT THE RHBA LOCATION
5193 017274 004737 024740      JSR      PC,PRTOCT   ;PRINT IT
5194 017300 104412 050116      DISPLY   LINB5       ;PRINT 'BAD DATA'
5195 017304 016046 000206      MOV      $RHDB(RO),-(SP) ;GET BAD CHAR FROM BUFFER
5196 017310 004737 024740      JSR      PC,PRTOCT   ;PRINT IT
5197 017314 016046 000166      MOV      $RAWC(RO),-(SP) ;WORD LENGTH ON STACK
5198 017320 066016 000020      ADD      $WRDL(RO),(SP) ;MAKE INTO A POSITIVE NUMBER
5199 017324 005046      CLR      -(SP)        ;UPPER DIVIDEND TO ZERO
5200 017326 016046 000026      MOV      $SSEC(RO),-(SP) ;SECTOR SIZE ON THE STACK
5201 017332 004737 024224      JSR      PC,LINKDV    ;DIVIDE WORDS XFERED BY SECTOR SIZE
5202 017336 012616      MOV      (SP)+,(SP)  ;MOVE REMAINDER UP THE STACK

```

```

5203 017340 104412 050134      DISPLY  LINP5      ;PRINT 'SECT POS'
5204 017344 004737 016714      JSR     PC,LIN3.2 ;CONVERT & PRINT IT
5205 017350 104412 001165      DISPLY  $CRLF
5206 017354 000207      RTS     PC
5207
5208 ;PRINT LINE 5A OF THE ERROR MESSAGE
5209
5210 LINE5A: DISPLY  LIN5S      ;'HEADER CONTENTS OF ERROR SECTOR'
5211      MOV     CYLDER,-(SP) ;HEADER POSITION TO STACK
5212      JSR     PC,PRTOCT    ;PRINT HEADER POSITION+
5213      DISPLY  LINSP      ;SPACES
5214      MOV     CYLDER+2,-(SP) ;HEADER POSITION+2 TO STACK
5215      JSR     PC,PRTOCT    ;PRINT HEADER POSITION++2
5216      DISPLY  LINSP      ;SPACES
5217      MOV     CYLDER+4,-(SP) ;HEADER POSITION+4 TO STACK
5218      JSR     PC,PRTOCT    ;PRINT HEADER POSITION++4
5219      DISPLY  LINSP      ;SPACES
5220      MOV     CYLDER+6,-(SP) ;HEADER POSITION+6 TO STACK
5221      JSR     PC,PRTOCT    ;PRINT HEADER POSITION++6
5222      DISPLY  LINSP      ;SPACES
5223      DISPLY  $CRLF
5224      RTS     PC
5225
5226 ;PRINT LINE 5B OF ERROR MESSAGE
5227
5228 LINE5B: DISPLY  LINEP5      ;'RHEC1 = '
5229      MOV     $RHEC1(R0),-(SP) ;ECC POSITION REGISTER ON STACK
5230      JSR     PC,PRTOCT    ;PRINT IT
5231      DISPLY  LINSP      ;SPACES
5232      DISPLY  LINEO5      ;' RHEC2 = '
5233      MOV     $RHEC2(R0),-(SP) ;ECC PATTERN REGISTER ON STACK
5234      JSR     PC,PRTOCT    ;PRINT IT
5235      DISPLY  $CRLF
5236      RTS     PC          ;RETURN
5237
5238 ;PRINT LINE 6 OF ERROR MESSAGE
5239
5240 LINE6:  DISPLY  LINB6      ;ECC CORRECTABLE
5241      DISPLY  $CRLF
5242      RTS     PC
5243
5244 ;PRINT LINE 6A OF THE ERROR MESSAGE
5245
5246 LINE6A: DISPLY  LINC6      ;PRINT 'READ CORRECTLY AT OFFSET N'
5247 LIN6.1: ASL   R1          ;DOUBLE THE OFFSET TABLE INDEX
5248      MOV     OFMTBL(R1),1$ ;ADDRESS OF OFFSET POSITION MESSAGE
5249      DISPLY  D            ;OFFSET VALUE
5250 1$:     .WORD 0
5251      DISPLY  $CRLF
5252      RTS     PC
5253
5254 ;PRINT LINE 6B OF THE ERROR MESSAGE
5255
5256 LINE6B: DISPLY  LINB6      ;PRINT 'SECTOR IS ECC CORRECTABLE '
5257      JMP    LIN6.1
5258

```

```

5259 ;PRINT LINE 6C OF THE ERROR MESSAGE
5260
5261 017562 104412 050313 LINE6C: DISPLY ,L6ING6 ;'CORRECTED ON NTH RETRY'
5262 017566 005046 LIN6.2: CLR -(SP) ;CLEAR STACK
5263 017570 116016 000022 MOVB $RETRY(RO),(SP) ;RETRY COUNT
5264 017574 004737 016714 JSR PC,LIN3.2 ;CONVERT & PRINT IT
5265 017600 104412 050331 DISPLY ,LINR6 ;'RETRY'
5266 017604 104412 001165 DISPLY ,$CRLF
5267 017610 000207 RTS PC
5268
5269 ;PRINT LINE 6D OF THE ERROR MESSAGE
5270
5271 017612 104412 050342 LINE6D: DISPLY ,LINU06 ;'UNCORRECTABLE AFTER N RETRIES'
5272 017616 000137 017566 JMP LIN6.2 ;FINISH
5273
5274 ;PRINT LINE 7 OF THE ERROR MESSAGE
5275
5276 017622 104412 050415 LINE7: DISPLY ,LIN70 ;PRINT ORDER COUNT
5277 017626 012746 000042 MOV $OPERC,-(SP) ;TO STACK
5278 017632 060016 030036 ADD RO,(SP) ;ADD THE BASE ADDRESS
5279 017634 004737 030036 JSR PC,$DB2D ;CONVERT IT
5280 017640 004737 025576 JSR PC,$SUPRS ;PRINT IT
5281 017644 104412 050474 DISPLY ,LIN7T ;TOTAL ERRORS
5282 017650 016046 000062 MOV $TOTAL(RO),-(SP) ;TO STACK
5283 017654 004737 016714 JSR PC,LIN3.2 ;CONVERT & PRINT
5284 017660 104412 050506 DISPLY ,LIN7X ;PRINT 'WRDS XFR'
5285 017664 012746 000052 MOV $STRANS,-(SP) ;ADDRESS OF LOW WORD ON STACK
5286 017670 060016 030036 ADD RO,(SP)
5287 017672 004737 030036 JSR PC,$DB2D ;CONVERT
5288 017676 004737 025576 JSR PC,$SUPRS ;PRINT
5289 017702 104412 050522 DISPLY ,LIN7R ;'BITS READ'
5290 017706 012746 000056 MOV $SREAD,-(SP) ;LOW WORD ADDRESS
5291 017712 060016 030036 ADD RO,(SP)
5292 017714 004737 030036 JSR PC,$DB2D ;CONVERT
5293 017720 004737 025576 JSR PC,$SUPRS ;PRINT
5294 017724 104412 001165 DISPLY , $CRLF
5295 017730 104412 001165 DISPLY , $CRLF
5296 017734 032737 100000 001140 BIT $SW15,SWR ;SEE IF 'HALT ON ERROR' - SWITCH 15
5297 017742 001401 BEQ .+4 ;BR IF NOT
5298 017744 000000 HALT ;SWITCH 15 HALT
5299 017746 000207 RTS PC
5300
5301 ;PRINT LINE 7A OF ERROR MESSAGE
5302
5303 017750 104412 050415 LINE7A: DISPLY ,LIN70 ;'ORDERS = '
5304 017754 012746 000042 MOV $OPERC,-(SP) ;ORDER COUNT INCREMENT
5305 017760 060016 030036 ADD RO,(SP) ;ADD BASE ADDRESS
5306 017762 004737 030036 JSR PC,$DB2D ;CONVERT THE COUNT
5307 017766 004737 025576 JSR PC,$SUPRS ;PRINT IT
5308 017772 104412 050425 DISPLY ,LIN7P ;'TOTAL SEEKS = '
5309 017776 012746 000046 MOV $SPOSIT,-(SP) ;TOTAL SEEKS
5310 020002 060016 030036 ADD RO,(SP) ;DEVICE TABLE ADDRESS
5311 020004 004737 030036 JSR PC,$DB2D ;CONVERT THE SEEK COUNT
5312 020010 004737 025576 JSR PC,$SUPRS ;PRINT IT
5313 020014 104412 050367 DISPLY ,LIN7M ;' TOTAL MISPOS ERR = '
5314 020020 016046 000072 MOV $MISPO(RO),-(SP) ;TOTAL ERRORS

```

```

5315 020024 004737 016714 JSR PC,LIN3.2 ;CONVERT & PRINT IT
5316 020030 104412 050445 DISPLY LIN7S ;' TOTAL SKI,OCYL ERR = '
5317 020034 016046 000070 MOV $SKI(RO),-(SP) ;CONVERT & PRINT IT
5318 020040 004737 016714 JSR PC,LIN3.2 ;CONVERT AND PRINT IT
5319 020044 104412 001165 DISPLY , $CRLF
5320 020050 104412 001165 DISPLY , $CRLF
5321 020054 032737 100000 001140 BIT #SW15,SWR ;SEE IF HALT ON ERROR - SWITCH 15 SET
5322 020062 001401 BEQ .+4 ;BR IF NOT
5323 020064 000000 HALT ;SWITCH 13 HALT
5324 020066 000207 RTS PC

```

;PRINT LINE 8 OF THE ERROR MESSAGE

```

5328 020070 104412 050537 LINE8: DISPLY ,LIN8M
5329 020074 004737 016046 JSR PC,LINE2 ;PRINT LINE 2 OF ERROR MESSAGE
5330 020100 000207 RTS PC

```

;*****

.SBTTL GENERAL SUPPORT SUBROUTINES

;*****

;ROUTINE TO INDICATE MEMORY SIZE ERROR AT STARTUP

```

5340 020102 104401 001165 MEMERR: TYPE , $CRLF ;CR-LF
5341 020106 104401 052230 TYPE ,NOTNUF ;TYPE 'NOT ENOUGH MEMORY'
5342 020112 104401 001165 TYPE , $CRLF ;CR-LF
5343 020116 000000 HALT
5344 020120 000137 001436 JMP START ;TRY IT AGAIN

```

;DECREMENT THE SECTOR-TRACK ADDRESS

```

5348 020124 005066 000004 READDR: CLR 4(SP) ;CLEAR STACK FOR SECTOR
5349 020130 116066 000172 000004 MOV $RHDA(RO),4(SP) ;INCREMENTED SECTOR ON STACK
5350 020136 105366 000004 DECB 4(SP) ;DECREMENT THE SECTOR ADDRESS
5351 020142 100017 000004 BPL 1$ ;BR IF SECTOR GREATER THAN 0
5352 020144 012766 000025 000004 MOV #25,4(SP) ;JAM SECTOR ADDRESS TO 21(10)
5353 020152 005066 000002 CLR 2(SP) ;CLEAR STACK FOR TRACK
5354 020156 116066 000173 000002 MOV $RHDA+1(RO),2(SP) ;TRACK ADDRESS
5355 020164 105366 000002 DECB 2(SP) ;DECREMENT TRACK ADDRESS
5356 020170 100007 000002 BPL 2$ ;BR IF IT DIDN'T GO NEG
5357 020172 012766 000022 000002 MOV #22,2(SP) ;RESET TRACK TO 18(10)
5358 020200 000403 BR 2$
5359 020202 116066 000173 000002 1$: MOV $RHDA+1(RO),2(SP) ;TRACK ADDRESS
5360 020210 000207 2$: RTS PC ;RETURN

```

;ROUTINE TO CHECK FOR A PRINTER

```

5364 020212 012737 177777 001212 CKPTR: MOV #-1,$PRFLG ;CLEAR PRINTER AVAILABILITY FLAG
5365 020220 012737 020244 000004 MOV #CKPTR1,#ERRVEC ;RETURN ADDR IF NO PRINTER
5366 020226 005037 000006 CLR #ERRVEC+2 ;NEW PSW
5367 020232 005777 160756 TST #SLSCS ;READ PRINTER STATUS REG
5368 020236 005037 001212 CLR $PRFLG ;SET THE PRINTER AVAILABILITY FLAG
5369 020242 000402 BR CKPTR2 ;FINISHED
5370 020244 062706 000004 CKPTR1: ADD #4,SP ;RESTORE THE STACK POINTER

```


K10

MAINDEC-11-DERPN-8 MACY11 27(732) 27-SEP-76 15:05 PAGE 128
 DERPNB.P11 GENERAL SUPPORT SUBROUTINES

```

5371 020250 012737 000006 000004 CKPTR2: MOV      #6, @#ERRVEC      ;RESET ERROR VECTOR TO HALT
5372 020256 000207          RTS          PC
5373
5374          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
5375
5376 020260 012737 177777 001210 CKCLK:  MOV      #-1, CLKFLG      ;CLEAR CLOCK AVAILABILITY FLAG
5377 020266 012737 177777 001206          MOV      #-1, PCLOCK      ;CLEAR KW11-P CLOCK AVAILABILITY FLAG
5378 020274 012737 020354 000004          MOV      #CKCLK1, @#ERRVEC  ;SET UP VECTOR FOR CLOCK CHECK
5379 020302 005037 000006          CLR      @#ERRVEC+2        ;NEW PSW
5380 020306 005777 160662          TST     @SLKCSR           ;CHECK FOR KW11-P
5381 020312 005037 001210          CLR      CLKFLG          ;SET CLOCK AVAILABILITY FLAG
5382 020316 005037 001206          CLR      PCLOCK         ;SET KW11-P CLOCK FLAG
5383 020322 013701 001200          MOV      $LPVEC, R1      ;KW11-P VECTOR ADDRESS
5384 020326 012721 021432          MOV      #CLOCK, (R1)+   ;SET UP KW11-P VECTOR
5385 020332 012711 000300          MOV      #300, (R1)      ;PSW - PRI 6
5386 020336 012777 177777 160632          MOV      #-1, @SLKCSB    ;LOAD COUNTER BUFFER WITH 1'S
5387 020344 012777 000135 160622          MOV      #135, @SLKCSR   ;SET CLOCK - CNT UP, 16MS, CONT INT
5388 020352 000432          BR      CKCLK3
5389 020354 062706 000004          CKCLK1: ADD     #4, SP      ;RESTORE THE STACK POINTER
5390 020360 012737 020422 000004          MOV      #CKCLK2, @#ERRVEC ;CHANGE ERROR VECTOR TO CHECK FOR KW11-L
5391 020366 005777 160610          TST     @SLKS           ;LOOK FOR KW11-L
5392 020372 005037 001210          CLR      CLKFLG          ;SET CLOCK FLAG
5393 020376 013701 001204          MOV      $LLVEC, R1      ;KW11-L VECTOR ADDRESS
5394 020402 012721 021432          MOV      #CLOCK, (R1)+   ;SET UP KW11-L VECTOR
5395 020406 012711 000300          MOV      #300, (R1)      ;PSW - PRI 6
5396 020412 012777 000100 160562          MOV      #100, @SLKS     ;SET KW11-L INTERRUPT
5397 020420 000407          BR      CKCLK3
5398 020422 062706 000004          CKCLK2: ADD     #4, SP      ;RESTORE THE STACK POINTER
5399 020426 104401 052255          TYPE    , NEDCLK        ;'P OR L CLOCK MUST BE ON SYSTEM'
5400 020432 000000          HALT
5401 020434 000137 001436          JMP     START            ;HALT
5402 020440 012737 000006 000004 CKCLK3: MOV      #6, @#ERRVEC  ;TRY AGAIN
5403 020446 000207          RTS          PC          ;RESTORE THE ERROR VECTOR
5404
5405          ;ROUTINE TO DISPLAY STATISTICS FOR ALL DRIVES ASSIGNED
5406
5407          STATPR:
5408 020450          MOV      R0, -(SP)      ;; PUSH R0 ON STACK
5409 020452          MOV      R2, -(SP)      ;; PUSH R2 ON STACK
5410 020454          MOV      R3, -(SP)      ;; PUSH R3 ON STACK
5411 020456          MOV      R4, -(SP)      ;; PUSH R4 ON STACK
5412 020460          CLR      R4            ;R4 CONTAINS THE UNIT POINTER
5413 020462          MOV      #8, R3        ;R3 CONTAINS THE UNIT COUNTER
5414 020466          MOV      #BLKO, R0      ;ADDR OF FIRST COUNTER
5415 020472          1$: TSTB     ASNLST(R4)   ;SEE IF DRIVE ASSIGNED
5416 020476          BNE     2$            ;BR IF IT IS
5417 020500          INC     R4            ;INCREMENT THE UNIT POINTER
5418 020502          ADD     #SRHEC2+4, R0  ;INCREMENT THE BLOCK ADDRESS
5419 020506          DEC     R3            ;FINISHED ?
5420 020510          BNE     1$            ;BR IF NOT
5421 020512          BR     6$            ;YES, EXIT
5422 020514          2$: JSR     PC, SHDTYP  ;TYPE THE HEADING
5423 020520          3$: TSTB     ASNLST(R4)   ;IS THE UNIT ASSIGNED
5424 020524          BNE     4$            ;BR IF NOT
5425 020526          ADD     #SRHEC2+4, R0  ;ADD TABLE SIZE TO ADDRESS
5426 020532          BR     5$
  
```

```

5427 020534 010002          4$:  MOV      R0,R2          ;PUT BLOCK ADDRESS IN R2
5428 020536 062702 000042  ADD      #SOPERC,R2      ;FIRST COUNTER TO DISPLAY
5429 020542 004737 020644  JSR      PC,SDETAL      ;TYPE THE STATISTICS
5430 020546 062700 000236  ADD      #SRHEC2+4,R0   ;INCREMENT BLOCK ADDRESS
5431 020552 005204          5$:  INC      R4              ;INCREMENT UNIT NUMBER
5432 020554 005303          DEC      R3              ;DECREMENT UNIT COUNTER
5433 020556 001360          SNE      3$              ;BR IF NOT FINISHED
5434 020560          6$:
5435 020560 012604          MOV      (SP)+,R4        ;POP STACK INTO R4
5436 020562 012603          MOV      (SP)+,R3        ;POP STACK INTO R3
5437 020564 012602          MOV      (SP)+,R2        ;POP STACK INTO R2
5438 020566 012600          MOV      (SP)+,R0        ;POP STACK INTO R0
5439 020570 000207          RTS      PC
5440
5441          ;TYPE THE HEADER FOR THE DRIVE PERFORMANCE SUMMARY TYPEOUT
5442
5443 020572 004737 021402  SHDTYP: JSR      PC,$TIME   ;TYPE THE TIME OF DAY
5444 020576 004537 025134  JSR      R5,TYPRI4      ;TYPE AT PRIORITY 4
5445 020602 001165          $CRLF          ;CR-LF
5446 020604 004537 025134  JSR      R5,TYPRI4      ;TYPE THE HEADER
5447 020610 051547          STATHD        ;HEADER
5448 020612 005737 001224  TST      LA30           ;LA30 ON THE SYSTEM ?
5449 020616 001406          BEQ      1$           ;BR IF ONE IS
5450 020620 004537 025134  JSR      R5,TYPRI4      ;SPLIT THE HEADER LINE
5451 020624 001165          $CRLF          ;CR-LF
5452 020626 004537 025134  JSR      R5,TYPRI4      ;TYPE SPACES
5453 020632 051362          LIN4SP        ;4 SPACES
5454 020634 004537 025134  1$:  JSR      R5,TYPRI4      ;FINISH THE HEADER LINE
5455 020640 051662          STAHD1        ;REST OF THE HEADER LINE
5456 020642 000207          RTS      PC
5457
5458          ;TYPE THE ERROR RATE DATA LINE
5459
5460 020644 142737 000007 001263  SDETAL: BICB      #7,UNIT+1   ;CLEAR THE UNIT NUMBER
5461 020652 150437 001263  BISB      R4,UNIT+1     ;SET THE UNIT NUMBER
5462 020656 104401 001262  TYPE      ,UNIT         ;TYPE IT
5463 020662 104401 051364  TYPE      ,LINSF        ;SPACES
5464 020666 016046 000074  MOV      $PASSC(R0),-(SP) ;PUT THE PASS COUNT ON THE STACK
5465 020672 004737 030232  JSR      PC,$SB2D       ;CONVERT IT
5466 020676 004737 024636  JSR      PC,$RPZR5      ;TYPE IT
5467 020702 104401 051364  TYPE      ,LINSF        ;SPACES
5468 020706 010246  MOV      R2,-(SP)       ;PUT $OPERC ON THE STACK
5469 020710 004737 030036  JSR      PC,$DB2D       ;CONVERT IT
5470 020714 004737 024616  JSR      PC,$RPZR8      ;TYPE $OPERC
5471 020720 104401 051364  TYPE      ,LINSF        ;SPACES
5472 020724 062702 000004  ADD      #4,R2          ;INCREMENT R2
5473 020730 010246  MOV      R2,-(SP)       ;PUT $POSIT ON THE STACK
5474 020732 004737 030036  JSR      PC,$DB2D       ;CONVERT IT
5475 020736 004737 024616  JSR      PC,$RPZR8      ;TYPE $POSIT
5476 020742 104401 051364  TYPE      ,LINSF        ;SPACES
5477 020746 062702 000004  ADD      #4,R2          ;INCREMENT R2
5478 020752 010246  MOV      R2,-(SP)       ;PUT $STRANS ON THE STACK
5479 020754 004737 030036  JSR      PC,$DB2D       ;CONVERT $STRANS
5480 020760 004737 024646  JSR      PC,$RPZR0      ;TYPE IT
5481 020764 104401 051364  TYPE      ,LINSF        ;SPACES
5482 020770 062702 000004  ADD      #4,R2          ;INCREMENT R2

```

```

5483 020774 010246      MOV      R2,-(SP)      ;PUT $READ ON THE STACK
5484 020776 004737 030036  JSR      PC,$DB2D     ;CONVERT $READ
5485 021002 004737 024646  JSR      PC,$RPZR0    ;TYPE IT
5486 021006 104401 051365  TYPE     ,LINSPO      ;1 SPACE
5487 021012 062702 000004  ADD      #4,R2        ;INCREMENT R2
5488 021016 005737 001224  TST      LA30         ;LA30 ON THE SYSTEM ?
5489 021022 001406  BEQ      1$          ;BR IF ONE IS
5490 021024 104401 001165  TYPE     , $CRLF      ;SPLIT THE LINE
5491 021030 104401 051362  TYPE     ,LINSPO      ;4 SPACES
5492 021034 104401 051364  TYPE     ,LINSPO      ;2 SPACES
5493 021040 062702 000002  ADD      #2,R2        ;INCREMENT R2 AGAIN
5494 021044 012246 1$:      MOV      (R2)+,-(SP)  ;PUT $SOFT ON THE STACK
5495 021046 004737 030232  JSR      PC,$SB2D     ;CONVERT $SOFT
5496 021052 004737 024626  JSR      PC,$RPZR4    ;TYPEOUT $SOFT
5497 021056 104401 051365  TYPE     ,LINSPO      ;SPACES
5498 021062 012246  MOV      (R2)+,-(SP)  ;PUT $SHRD ON THE STACK
5499 021064 004737 030232  JSR      PC,$SB2D     ;CONVERT $SHRD
5500 021070 004737 024626  JSR      PC,$RPZR4    ;TYPEOUT $SHRD
5501 021074 104401 051365  TYPE     ,LINSPO      ;SPACES
5502 021100 012246  MOV      (R2)+,-(SP)  ;PUT $SKI ON THE STACK
5503 021102 004737 030232  JSR      PC,$SB2D     ;CONVERT $SKI
5504 021106 004737 024626  JSR      PC,$RPZR4    ;TYPEOUT $SKI
5505 021112 104401 051365  TYPE     ,LINSPO      ;SPACES
5506 021116 012246  MOV      (R2)+,-(SP)  ;PUT $MISPO ON THE STACK
5507 021120 004737 030232  JSR      PC,$SB2D     ;CONVERT $MISPO
5508 021124 004737 024626  JSR      PC,$RPZR4    ;TYPEOUT $MISPO
5509 021130 104401 051365  TYPE     ,LINSPO      ;SPACES
5510 021134 016046 000062  MOV      $TOTAL(R0),-(SP) ;CALCULATE NUMBER OF OTHER ERRORS
5511 021140 166016 000064  SUB      $SOFT(R0),(SP) ;SUBTRACT $SOFT FROM $TOTAL
5512 021144 166016 000066  SUB      $SHRD(R0),(SP) ;SUBTRACT $SHRD FROM $TOTAL
5513 021150 166016 000070  SUB      $SKI(R0),(SP)  ;SUBTRACT $SKI FROM $TOTAL
5514 021154 166016 000072  SUB      $MISPO(R0),(SP);SUBTRACT $MISPO FROM $TOTAL
5515 021160 004737 030232  JSR      PC,$SB2D     ;CONVERT 'OTHER' COUNT
5516 021164 004737 024626  JSR      PC,$RPZR4    ;TYPE IT
5517 021170 104401 001165  TYPE     , $CRLF
5518 021174 000207  RTS      PC

```

;ROUTINE TO TYPE STATISTICS FOR THE DRIVE IN R0

```

5520
5521
5522 021176 010046      TYPEST:  MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5523 021176 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
5524 021200 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
5525 021202 004737 020572  JSR      PC,$HDTYP     ;TYPE THE HEADING
5526 021204 005004  CLR      R4            ;CLEAR R4 FOR DRIVE NUMBER
5527 021210 111004  MOVB     (R0),R4       ;DRIVE NUMBER
5528 021212 010002  MOV      R0,R2        ;PUT BLOCK ADDRESS IN R2
5529 021214 062702 000042  ADD      # $OPERC,R2   ;ADDRESS OF '$OPERC' IN PRESENT CLOCK
5530 021216 004737 020644  JSR      PC,$DETAL     ;TYPE THE STATISTICS
5531 021222 012604  MOV      (SP)+,R4      ;;POP STACK INTO R4
5532 021226 012602  MOV      (SP)+,R2      ;;POP STACK INTO R2
5533 021230 012600  MOV      (SP)+,R0      ;;POP STACK INTO R0
5534 021232 000207  RTS      PC            ;RETURN
5535
5536
5537
5538

```

;ROUTINE TO INCREMENT \$SOFT

```

5539
5540 ;NOTE: $SOFT WILL NOT BE INCREMENTED BEYOND 9999 (10)
5541
5542 021236 005737 016416 INCSOF: TST PRT2B ;SEE IF BAD TRK/SEC INDICATOR SET
5543 021242 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
5544 021244 026027 000064 023417 CMP $SOFT(RO),#9999. ;IS $SOFT ALREADY AT MAXIMUM ?
5545 021252 103002 BHIS 1$ ;BR IF IT IS
5546 021254 005260 000064 INC $SOFT(RO) ;INCREMENT $SOFT
5547 021260 000207 1$: RTS PC ;RETURN
5548
5549
5550 ;ROUTINE TO INCREMENT $HARD
5551
5552 ;NOTE: $HARD WILL NOT BE INCREMENTED BEYOND 9999 (10)
5553
5554 021262 005737 016416 INCHRD: TST PRT2B ;SEE IF BAD TRK/SEC INDICATOR SET
5555 021266 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
5556 021270 026027 000066 023417 CMP $HARD(RO),#9999. ;IS $HARD ALREADY AT MAXIMUM ?
5557 021276 103002 BHIS 1$ ;BR IF IT IS
5558 021300 005260 000066 INC $HARD(RO) ;INCREMENT $HARD
5559 021304 000207 1$: RTS PC ;RETURN
5560
5561
5562 ;ROUTINE TO INCREMENT $SKI
5563
5564 ;NOTE: $SKI WILL NOT BE INCREMENTED BEYOND 9999 (10)
5565
5566 021306 005737 016416 INCSKI: TST PRT2B ;SEE IF BAD TRK/SEC INDICATOR SET
5567 021312 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
5568 021314 026027 000070 023417 CMP $SKI(RO),#9999. ;IS $SKI ALREADY AT MAXIMUM ?
5569 021322 103002 BHIS 1$ ;BR IF IT IS
5570 021324 005260 000070 INC $SKI(RO) ;INCREMENT $SKI
5571 021330 000207 1$: RTS PC ;RETURN
5572
5573
5574 ;ROUTINE TO INCREMENT $MISPO
5575
5576 ;NOTE: $MISPO WILL NOT BE INCREMENTED BEYOND 9999 (10)
5577
5578 021332 005737 016416 INCMIS: TST PRT2B ;SEE IF BAD TRK/SEC INDICATOR SET
5579 021336 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
5580 021340 026027 000072 023417 CMP $MISPO(RO),#9999. ;IS $MISPO ALREADY AT MAXIMUM ?
5581 021346 103002 BHIS 1$ ;BR IF IT IS
5582 021350 005260 000072 INC $MISPO(RO) ;INCREMENT $MISPO
5583 021354 000207 1$: RTS PC ;RETURN
5584
5585
5586 ;ROUTINE TO INCREMENT $TOTAL
5587
5588 ;NOTE: $TOTAL WILL NOT BE INCREMENTED BEYOND 9999 (10)
5589
5590 021356 005737 016416 INCTOT: TST PRT2B ;SEE IF BAD TRK/SEC INDICATOR SET
5591 021362 001006 BNE 1$ ;BR IF IT'S SET, DON'T INCREMENT COUNT
5592 021364 026027 000062 023417 CMP $TOTAL(RO),#9999. ;IS $TOTAL ALREADY AT MAXIMUM ?
5593 021372 103002 BHIS 1$ ;BR IF IT IS
5594 021374 005260 000062 INC $TOTAL(RO) ;INCREMENT $TOTAL

```

```

5595 021400 000207 1$: RTS PC ;RETURN
5596
5597
5598 ;ROUTINE TO TYPE THE TIME
5599
5600 021402 005737 001210 $TIME: TST CLKFLG ;CLOCK ON THE SYSTEM ?
5601 021406 001010 BNE 1$ ;BR IF NOT
5602 021410 104401 001165 TYPE ,SCALF
5603 021414 104401 021702 TYPE ,HOUR ;TYPE THE HOURS
5604 021420 104401 021706 TYPE ,MINUTE ;TYPE THE MINUTES
5605 021424 104401 021712 TYPE ,SECOND ;TYPE THE SECONDS
5606 021430 000207 1$: RTS PC
5607
5608 ;CLOCK HANDLER ROUTINE
5609
5610 021432 010046 CLOCK: MOV RO,-(SP) ;SAVE RO
5611 021434 005237 021720 INC SIXTEE ;INCREMENT THE 1/60 SECOND COUNTER
5612 021440 001076 BNE 1$ ;BR IF A SECOND NOT COUNTED
5613 021442 013737 001222 021720 MOV HZ,SIXTEE ;RESTORE THE VALUE
5614 021450 005437 021720 NEG SIXTEE ;MAKE IT INTO 2'S COMP
5615 021454 105237 021713 INCB SECOND+1 ;INCREMENT THE SECONDS
5616 021460 122737 000072 021713 CMPB #72,SECOND+1 ;SEE IF COUNTER AT MAX
5617 021466 001063 BNE 1$ ;BR IF NOT
5618 021470 112737 000060 021713 MOVB #60,SECOND+1 ;RESTORE VALUE
5619 021476 105237 021712 INCB SECOND ;INCREMENT UPPER
5620 021502 122737 000066 021712 CMPB #66,SECOND ;UPPER AT MAX
5621 021510 001052 BNE 1$ ;BR IF NOT
5622 021512 112737 000060 021712 MOVB #60,SECOND ;RESET UPPER
5623 021520 105337 001305 DECB INTRVL+1 ;DECREMENT STATISTICS TYPEOUT INTERVAL
5624 021524 105237 021707 INCB MINUTE+1 ;INCREMENT MINUTE
5625 021530 122737 000072 021707 CMPB #72,MINUTE+1 ;AT MAX ?
5626 021536 001037 BNE 1$ ;BR IF NOT
5627 021540 112737 000060 021707 MOVB #60,MINUTE+1 ;RESET LOWER
5628 021546 105237 021706 INCB MINUTE ;INCREMENT UPPER HALF
5629 021552 122737 000066 021706 CMPB #66,MINUTE ;AT MAX ?
5630 021560 001026 BNE 1$ ;BR IF NOT
5631 021562 112737 000060 021706 MOVB #60,MINUTE ;RESET UPPER
5632 021570 105237 021703 INCB HOUR+1 ;INCREMENT HOURS
5633 021574 022737 035071 021702 CMP #35071,HOUR ;AT MAX (99) ?
5634 021602 001412 BEQ 2$ ;BR IF IT IS
5635 021604 122737 000072 021703 CMPB #72,HOUR+1 ;AT MAX ?
5636 021612 001011 BNE 1$ ;BR IF NOT
5637 021614 112737 000060 021703 MOVB #60,HOUR+1 ;RESET VALUE
5638 021622 105237 021702 INCB HOUR ;INCREMENT UPPER
5639 021626 000403 BR 1$
5640 021630 012737 030060 021702 2$: MOV #30060,HOUR ;RESET HOUR FIELD
5641 021636 012746 000021 1$: MOV #21,-(SP) ;17 MS ON THE STACK
5642 021642 004037 035126 JSR RO,RPTMR ;DRIVER TIMER ROUTINE
5643 021646 105737 001304 TSTB INTRVL ;DISPLAY THE PERFORMANCE SUMMARY ?
5644 021652 001411 BEQ 3$ ;BR IF NOT
5645 021654 105737 001305 TSTB INTRVL+1 ;DISPLAY INTERVAL FINISHED ?
5646 021660 001006 BNE 3$ ;BR IF NOT
5647 021662 113737 001304 001305 MOVB INTRVL,INTRVL+1 ;RESTORE THE INTERVAL
5648 021670 012737 177777 001230 MOV #-1,STATIN ;SET ERROR RATE DATA DISPLAY FLAG
5649 021676 012600 3$: MOV (SP)+,RO ;RESTORE RO
5650 021700 000002 RTI
  
```

5651											
5652	021702	060	060	HOUR:	.BYTE	60,60		:HOURS			
5653	021704	072	000		.BYTE	72,0		: ' ' & TERMINATOR			
5654	021706	060	060	MINUTE:	.BYTE	60,60		:MINUTES			
5655	021710	072	000		.BYTE	72,0		: ' ' & TERMINATOR			
5656	021712	060	060	SECOND:	.BYTE	60,60		:SECONDS			
5657	021714	040	040		.BYTE	40,40		:SPACES			
5658	021716	000	040		.BYTE	0		:TERMINATOR			
5659		021720			.EVEN						
5660											
5661	021720	000000		SIXTEE:	.WORD	0		:1/60TH OR 1/50TH OF A SECOND COUNTER			
5662											
5663											
5664											
5665	021722			KSR:							
5666	021722	010046			MOV	R0,-(SP)		::PUSH R0 ON STACK			
5667	021724	010146			MOV	R1,-(SP)		::PUSH R1 ON STACK			
5668	021726	010246			MOV	R2,-(SP)		::PUSH R2 ON STACK			
5669	021730	010346			MOV	R3,-(SP)		::PUSH R3 ON STACK			
5670	021732	010446			MOV	R4,-(SP)		::PUSH R4 ON STACK			
5671	021734	010546			MOV	R5,-(SP)		::PUSH R5 ON STACK			
5672	021736	012705	026265		MOV	#STKQSR+1,R5		:ADDR OF INPUT			
5673	021742	104401	001165		TYPE	,SCRLF		:CR-LF			
5674	021746	122715	000101		CMPB	#101,(R5)		:EQ TO AN 'A'			
5675	021752	001410			BEG	1\$:BR IF IT IS			
5676	021754	121527	000075		CMPB	(R5),#75		:UNIT NUMBER GREATER THAN AN ASCII 7 ?			
5677	021760	103054			BHIS	6\$:BR IF IT IS			
5678	021762	121527	000060		CMPB	(R5),#60		:UNIT NUMBER LESS THAN AN ASCII 0 ?			
5679	021766	103451			BLO	6\$:BR IF IT IS			
5680	021770	142715	177770		BICB	#7,(R5)		:LEAVE ONLY LOWER 3 BITS IF CHAR NOT 'A'			
5681	021774	122765	000124	177777	1\$:	CMPB	#'T,-1(R5)	:EQ TO 'T'			
5682	022002	001003			BNE	2\$:BR IF NOT EQ			
5683	022004	004737	022134		JSR	PC,NEWASN		:ASSIGN UNIT FOR TEST			
5684	022010	000442			BR	7\$:EXIT			
5685	022012	122765	000104	177777	2\$:	CMPB	#'D,-1(R5)	:EQ TO 'D' ?			
5686	022020	001003			BNE	3\$:BR IF NOT EQ			
5687	022022	004737	022452		JSR	PC,DEASGN		:DEASSIGN UNIT			
5688	022026	000433			BR	7\$:EXIT			
5689	022030	122765	000123	177777	3\$:	CMPB	#'S,-1(R5)	:EQ TO 'S'			
5690	022036	001003			BNE	4\$:BR IF NOT EQ			
5691	022040	004737	022554		JSR	PC,SCMND		:TYPE STATISTICS			
5692	022044	000424			BR	7\$:EXIT			
5693	022046	122765	000127	177777	4\$:	CMPB	#'W,-1(R5)	:EQ TO 'W'			
5694	022054	001007			BNE	5\$:BR IF NOT EQ			
5695	022056	032737	000001	001140		BIT	#SWD,SWR	:IS SWITCH D SET ?			
5696	022064	001012			BNE	6\$:BR IF SET, CAN'T DO 'W' COMMAND			
5697	022066	004737	023026		JSR	PC,DATAPK		:WRITE A DATA PACK			
5698	022072	000411			BR	7\$:EXIT			
5699	022074	122765	000122	177777	5\$:	CMPB	#'R,-1(R5)	:EQ TO 'R' ?			
5700	022102	001003			BNE	6\$:BR IF NOT EQ			
5701	022104	004737	023040		JSR	PC,REDAPK		:READ A DATA PACK			
5702	022110	000402			BR	7\$:EXIT			
5703	022112	104401	052405		6\$:	TYPE	,INVLD	:TYPE 'INVALID COMMAND' MESSAGE			
5704	022116				7\$:						
5705	022116	012605			MOV	(SP)+,R5		::POP STACK INTO R5			
5706	022120	012604			MOV	(SP)+,R4		::POP STACK INTO R4			

```

5707 022122 012603          MOV      (SP)+,R3          ;;POP STACK INTO R3
5708 022124 012602          MOV      (SP)+,R2          ;;POP STACK INTO R2
5709 022126 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
5710 022130 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
5711 022132 000207          RTS      PC
5712
5713          ;ROUTINE TO ASSIGN A DRIVE ('T' COMMAND)
5714
5715 022134 005037 001232      NEWASN: CLR      PACK          ;CLEAR 'W' COMMAND INDICATOR
5716 022140 005004          ASGN0: CLR      R4           ;R4 IS TABLE INDEX
5717 022142 012703 000010      MOV      #8, R3           ;SET INITIAL COUNT TO 8
5718 022146 122715 000101      CMPB    #101,(R5)        ;ASSIGN ALL UNITS?
5719 022152 001416          BEQ      ASGN2           ;BR IF ALL UNITS
5720 022154 111504          ASGN1: MOVB    (R5),R4     ;PUT UNIT # IN R4
5721 022156 012737 051437 023612      MOV      #UNTA$N,ASNMSG   ;'UNIT ASSIGNED' MESSAGE ADDR
5722 022164 012703 000001      MOV      #1,R3           ;RELOAD R3 FOR 1 UNIT
5723 022170 105764 041640      TSTB    ASNLST(R4)       ;UNIT ALREADY ASSIGNED
5724 022174 001002          BNE      1$             ;BR IF IT IS
5725 022176 004737 022232      JSR      PC,ASGN3        ;SEE IF UNIT ON THE SYSTEM
5726 022202 000137 023562      1$:     JMP      ASNERR     ;RETURN HERE IF DRIVE NOT AVAIL
5727 022206 000207          RTS      PC             ;EXIT
5728 022210 004737 022232      ASGN2: JSR      PC,ASGN3   ;ASSIGN ALL UNASSIGNED, AVAIL UNITS
5729 022214 000137 022222      JMP      1$             ;UNIT NOT ON SYSTEM
5730 022220 000207          RTS      PC             ;EXIT
5731 022222 012746 022214      1$:     MOV      #ASGN2+4,-(SP) ;PUT RETURN ADDRESS ON THE STACK
5732 022226 000137 022342      JMP      ASGN4          ;LOOK FOR MORE UNITS
5733 022232 105764 041640      ASGN3: TSTB    ASNLST(R4)  ;UNIT ALREADY ASSIGNED?
5734 022236 001041          BNE      ASGN4          ;BR IF IT IS
5735 022240 005737 030700      1$:     TST      DTUW          ;DATA TRANSFER UNDER WAY ?
5736 022244 100375          BPL      1$             ;BR IF IT IS
5737 022246 110437 041526      MOVB    R4,GENDPB        ;DRIVE NUMBER
5738 022252 004737 013012      JSR      PC,RECALD       ;RECALIBRATE DRIVE
5739 022256 105764 030556      TSTB    DRVSTA(R4)      ;DRIVE AVAILABLE?
5740 022262 001436          BEQ      ASGN6          ;BR IF DRIVE OFFLINE
5741 022264 100441          BMI      ASGN7          ;BR DRIVE NOT AVAILABLE
5742 022266 006304          ASL      R4             ;MAKE R4 INTO WORD INDEX
5743 022270 016464 042124 041672      MOV      BLKADR(R4),NEWUNT(R4) ;DPB ADDRESS
5744 022276 016400 042124      MOV      BLKADR(R4),R0   ;PUT BLOCK'S ADDR INTO R0
5745 022302 004737 013272      JSR      PC,CLRDPB       ;CLEAR BLOCK FOR UNIT JUST ASSIGNED
5746 022306 004737 023052      JSR      PC,GETID        ;GET DRIVE I.D.
5747 022312 004737 023132      JSR      PC,GETADR       ;GET BAD SECTOR ADDRESSES
5748 022316 012760 000001 000074      MOV      #1,$PASSC(R0)   ;PRESET PASS COUNT TO 1
5749 022324 005737 001232      TST      PACK          ;WRITE DATA PACK ?
5750 022330 001403          BEQ      2$             ;BR IF NOT
5751 022332 113760 001232 000031      MOVB    PACK,$PACK(R0)  ;SET READ/WRITE DATA PACK INDICATOR
5752 022340 006204          2$:     ASR      R4           ;RESTORE UNIT ADDRESS
5753 022342 005303          ASGN4: DEC      R3           ;DEC UNIT COUNT
5754 022344 001402          BEQ      ASGN5          ;BR IF FINISHED
5755 022346 005204          INC      R4           ;INCREMENT UNIT NUMBER
5756 022350 000730          BR       ASGN3          ;CONTINUE
5757 022352 062716 000004          ASGN5: ADD      #4,(SP)    ;INCREMENT RETURN
5758 022356 000207          RTS      PC             ;RETURN
5759 022360 012737 051374 023612      ASGN6: MOV      #UNTOFF,ASNMSG ;'OFFLINE' MESSAGE ADDRESS
5760 022366 000207          RTS      PC             ;RETURN
5761 022370 006304          ASGN7: ASL      R4           ;R4 INTO WORD INDEX
5762 022372 005764 030566      TST      DRVTYP(R4)     ;UNIT PRESENT?

```

E11

```

5763 022376 001414      BEQ      1$      ;BR IF NOT
5764 022400 022764 020020 030566      CMP      #20020,DRV TYP(R4) ;UNIT RPO4?
5765 022406 001414      BEQ      2$      ;BR IF SINGLE PORT RPO4
5766 022410 022764 024020 030566      CMP      #24020,DRV TYP(R4) ;UNIT DUAL PORT RPO4?
5767 022416 001410      BEQ      2$      ;BR IF DUAL PORT RPO4
5768 022420 012737 051465 023612      MOV      #NOTRP,ASNMSG ;ADDRESS OF 'NOT RPO4' MSG
5769 022426 000407      BR       3$      ;EXIT
5770 022430 012737 051502 023612 1$:      MOV      #NOTPRS,ASNMSG ;ADDRESS OF 'NOT PRESENT' MSG
5771 022436 000403      BR       3$      ;EXIT
5772 022440 012737 051517 023612 2$:      MOV      #NOTSAF,ASNMSG ;ADDR OF 'UNIT UNSAFE' MESSAGE
5773 022446 006204      3$:      ASR      R4      ;RESTORE R4
5774 022450 000207      RTS      PC      ;ERROR RETURN
5775
5776
5777      ;ROUTINE TO DEASSIGN A UNIT ('D' COMMAND)
5778 022452 005004      DEASGN: CLR      R4
5779 022454 122715 000101      CMPB     #101,(R5) ;DEASSIGN ALL UNITS ?
5780 022460 001432      BEQ      5$      ;BR IF YES
5781 022462 012703 000001      MOV      #BIT00,R3 ;SET R3 FOR ONE UNIT
5782 022466 111504      MOVB     (R5),R4 ;GET UNIT NUMBER
5783 022470 105764 041640 1$:      TSTB     ASNLST(R4) ;UNIT ASSIGNED ?
5784 022474 001413      BEQ      3$      ;BR IF NOT
5785 022476 105064 041640      CLRB     ASNLST(R4) ;DELETE THE UNIT FROM THE ASSIGNED LIST
5786 022502 006304      ASL      R4      ;MAKE ADDR INTO A WORD INDEX
5787 022504 016464 042124 041650      MOV      BLKADR(R4),DUNIT(R4) ;PUT ADDRESS IN DEASSIGN LIST
5788 022512 006204      ASR      R4
5789 022514 005303      2$:      DEC      R3      ;ANY MORE UNITS ?
5790 022516 001412      BEQ      4$      ;BR IF NOT
5791 022520 005204      INC      R4
5792 022522 000762      BR       1$
5793 022524 122715 000101 3$:      CMPB     #101,(R5) ;DEASSIGN ALL UNITS ?
5794 022530 001771      BEQ      2$      ;BR IF YES
5795 022532 012737 051415 023612      MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MESSAGE
5796 022540 004737 023562      JSR      PC,ASNEAR ;REPORT IT
5797 022544 000207      4$:      RTS      PC
5798 022546 012703 000010 5$:      MOV      #8.,R3 ;SET UNIT COUNT TO 8
5799 022552 000746      BR       1$
5800
5801      ;ROUTINE TO TYPE UNIT PERFORMANCE SUMMARY ('S' COMMAND)
5802
5803 022554 005004      SCMND: CLR      R4
5804 022556 122715 000101      CMPB     #101,(R5) ;ALL STATISTICS ?
5805 022562 001420      BEQ      2$      ;BR IF YES
5806 022564 111504      MOVB     (R5),R4 ;GET UNIT #
5807 022566 105764 041640      TSTB     ASNLST(R4) ;SEE IF UNIT ASSIGNED
5808 022572 001406      BEQ      1$      ;BR IF NOT
5809 022574 006304      ASL      R4      ;MAKE UNIT ADDR INTO WORD INDEX
5810 022576 016400 042124      MOV      BLKADR(R4),R0 ;ADDR OF BLOCK
5811 022602 004737 021176      JSR      PC,TYPEST ;TYPE DRIVE STATISTICS
5812 022606 000506      BR       9$      ;EXIT
5813 022610 012737 051415 023612 1$:      MOV      #UNTNOT,ASNMSG ;ADDR OF 'NOT ASSIGNED' MSG
5814 022616 004737 023562      JSR      PC,ASNEAR ;TYPE ERROR MESSAGE
5815 022622 000500      BR       9$      ;EXIT
5816 022624 012703 000010 2$:      MOV      #8.,R3 ;UNIT COUNT
5817 022630 105764 041640 3$:      TSTB     ASNLST(R4) ;SEE IF ANY UNIT ASSIGNED
5818 022634 001004      BNE     4$
  
```


5819	022636	005204			INC	R4	: INCREMENT UNIT ADDRESS
5820	022640	005303			DEC	R3	: DECREMENT COUNTER
5821	022642	001372			BNE	3\$: MORE TO CHECK
5822	022644	000467			BR	9\$: NONE ASSIGNED, RETURN
5823	022646	004737	020450	4\$:	JSR	PC, STATPR	: TYPE ALL STATISTICS
5824	022652	105737	001234		TSTB	DATE	: SEE IF 'DATE' ENTERED
5825	022656	001404			BEQ	11\$: BR IF NOT
5826	022660	104401	052526		TYPE	, DATEIS	: 'DATE:'
5827	022664	104401	001234		TYPE	, DATE	: THE OPERATOR ENTERED DATE
5828	022670	105737	001246	11\$:	TSTB	OPERID	: SEE IF OPERATOR I.D. ENTERED
5829	022674	001404			BEQ	12\$: BR IF NOT
5830	022676	104401	052537		TYPE	, IDIS	: 'OPERATOR I.D.: '
5831	022702	104401	001246		TYPE	, OPERID	: THE OPERATOR I.D.
5832	022706	104401	052561	12\$:	TYPE	, HEDLIN	: HEADER LINE
5833	022712	012737	037322	022774	MOV	#BLKO+\$DRVID, 6\$: DRIVE I.D. FIELD ADDRESS
5834	022720	005004			CLR	R4	: DRIVE ADDRESS
5835	022722	012703	000010		MOV	#8, R3	: COUNTER
5836	022726	105764	041640	5\$:	TSTB	ASNLST(R4)	: SEE IF DRIVE ASSIGNED
5837	022732	001423			BEQ	7\$: BR IF NOT ASSIGNED
5838	022734	142737	000007	001263	BICB	#7, UNIT+1	: CLEAR OLD DRIVE NUMBER
5839	022742	150437	001263		BISB	R4, UNIT+1	: LOAD NEW DRIVE NUMBER
5840	022746	104401	001262		TYPE	, UNIT	: TYPE THE DRIVE NUMBER
5841	022752	104401	051362		TYPE	, LIN4SP	: 4 SPACES
5842	022756	105777	000012		TSTB	36\$: SEE IF DRIVE I.D. ENTERED
5843	022762	001003			BNE	10\$: BR IF DRIVE I.D. PRESENT
5844	022764	104401	052604		TYPE	, NONE	: TYPE 'NONE'
5845	022770	000404			BR	7\$: CONTINUE
5846	022772	104401		10\$:	TYPE		: TYPE THE DRIVE I.D.
5847	022774	000000		6\$:	.WORD	0	: ADDRESS OF DRIVE I.D. FIELD HERE
5848	022776	104401	001165		TYPE	, \$CRLF	: CR-LF
5849	023002	005303		7\$:	DEC	R3	: DECREMENT THE COUNTER
5850	023004	001405			BEQ	8\$: BR IF AT END
5851	023006	062737	000236	022774	ADD	#SRHEC2+4, 6\$: INCREMENT THE MESSAGE FIELD ADDRESS
5852	023014	005204			INC	R4	: INCREMENT DRIVE ADDRESS
5853	023016	000743			BR	5\$: CONTINUE
5854	023020	104401	001165	8\$:	TYPE	, \$CRLF	: CR-LF
5855	023024	000207		9\$:	RTS	PC	
5856							
5857							: ROUTINE TO WRITE A DATA PACK ('W' COMMAND)
5858							
5859	023026	012737	177777	001232	DATAPK: MOV	#-1, PACK	: SET THE 'W' COMMAND INDICATOR
5860	023034	000137	022140		JMP	ASGNO	: ASSIGN REQUESTED UNIT
5861							
5862							
5863							: ROUTINE TO READ A DATA PACK ('R' COMMAND)
5864							
5865	023040	012737	000001	001232	REDAPK: MOV	#1, PACK	: SET THE 'READ' INDICATOR
5866	023046	000137	022140		JMP	ASGNO	: ASSIGN THE REQUESTED UNIT
5867							
5868							: ROUTINE TO GET THE DRIVE ID NUMBER FROM THE OPERATOR
5869							
5870	023052	010546			GETID: MOV	R5, -(SP)	: SAVE R5
5871	023054	104401	052474		TYPE	, ENDRV	: 'ENTER DRV I.D.: '
5872	023060	142737	000007	001263	BICB	#7, UNIT+1	: CLEAR OLD UNIT NUMBER
5873	023066	010446			MOV	R4, -(SP)	: CONVERT INDEX TO DRIVE NUMBER
5874	023070	006216			ASR	(SP)	: SHIFT IT

5875	023072	152637	001263	B1SB	(SP)+,UNIT+1	;LOAD NEW UNIT NUMBER	
5876	023076	104401	001262	TYPE	,UNIT	;TYPE THE DRIVE NUMBER	
5877	023102	104401	052523	TYPE	,COLON	;TYPE A COLON ':'	
5878	023106	104407		RDLIN		;READ THE ENTRY	
5879	023110	012605		MOV	(SP)+,R5	;GET THE ENTRY ADDRESS	
5880	023112	012560	000154	MOV	(R5)+,\$DRVID(R0)	;STORE THE I.D.	
5881	023116	012560	000156	MOV	(R5)+,\$DRVID+2(R0)	;STORE THE I.D.	
5882	023122	012560	000160	MOV	(R5)+,\$DRVID+4(R0)	;STORE THE I.D.	
5883	023126	012605		MOV	(SP)+,R5	;RESTORE R5	
5884	023130	000207		RTS	PC	;RETURN	
5885							
5886						;ROUTINE TO GET THE ADDRESSES OF ANY BAD SECTORS (UP TO A MAX OF 8)	
5887							
5888	023132			GETADR:			
5889	023132	010146		MOV	R1,-(SP)	;PUSH R1 ON STACK	
5890	023134	010246		MOV	R2,-(SP)	;PUSH R2 ON STACK	
5891	023136	010346		MOV	R3,-(SP)	;PUSH R3 ON STACK	
5892	023140	104401	052613	TYPE	,ENTADR	;ENTER SECTOR ADDRESSES	
5893	023144	104401	001262	TYPE	,UNIT	;TYPE THE UNIT NUMBER	
5894	023150	104401	051365	TYPE	,LINSPO	;1 SPACE	
5895	023154	104401	001164	TYPE	,\$QUES	;QUESTION MARK	
5896	023160	012703	000020	MOV	#16.,R3	;NUMBER OF LOCATIONS IN THE TABLE TO PRESET	
5897	023164	012701	000114	MOV	#\$BDSEC,R1	;TABLE INCREMENT	
5898	023170	060001		ADD	R0,R1	;BLOCK STARTING ADDRESS	
5899	023172	012721	177777	1\$:	MOV	#-1,(R1)+	;SET LOCATION TO 1'S
5900	023176	005303		DEC	R3	;DECREMENT TABLE SIZE COUNT	
5901	023200	001374		BNE	1\$;BR IF NOT FINISHED WITH TABLE	
5902	023202	162701	000040	SUB	#32.,R1	;SET POINTER TO BEGINNING OF TABLE	
5903	023206	012703	000010	MOV	#8.,R3	;NUMBER OF ADDRESSES IN TABLE	
5904	023212	104407		2\$:	RDLIN	;GET ADDRESS FROM OPERATOR	
5905	023214	012602		MOV	(SP)+,R2	;TEXT POINTER	
5906	023216	121227	000003	CMPB	(R2),#3	;CONTROL 'C' ENTERED ?	
5907	023222	001476		BEQ	4\$;BR IF IT WAS	
5908	023224	012737	000054	MOV	#,SEPART	;COMMA IS THE SEPARATOR	
5909	023232	004537	023432	JSR	R5,CONVEA	;GET THE CYLINDER ADDRESS	
5910	023236	000460		BR	3\$;TERMINATOR FOUND (INVALID)	
5911	023240	005737	023556	TST	HOLD	;SEE IF INVALID TEXT ENTERED	
5912	023244	100455		BMI	3\$;BR IF ANY CYLINDER TEXT INVALID	
5913	023246	023727	023556	CMP	HOLD,#410.	;IS CYLINDER VALUE OK ?	
5914	023254	101051		BHI	3\$;BR IF NOT	
5915	023256	013711	023556	MOV	HOLD,(R1)	;MOVE CYLINDER VALUE TO THE TABLE	
5916	023262	005037	023430	CLR	6\$;CLEAR THE TRACK 'CR' INDICATOR	
5917	023266	004537	023432	JSR	R5,CONVEA	;GET THE TRACK ADDRESS	
5918	023272	000402		BR	.+6	;CR ENTERED; ENTIRE TRACK SPECIFIED	
5919	023274	005237	023430	INC	6\$;SET THE 'SECTOR' SWITCH	
5920	023300	005737	023556	TST	HOLD	;SEE IF INVALID CHARACTERS ENTERED	
5921	023304	100435		BMI	3\$;BR IF ANY WERE	
5922	023306	023727	023556	CMP	HOLD,#18.	;SEE IF TRACK ADDRESS TOO LARGE	
5923	023314	101031		BHI	3\$;BR IF IT IS	
5924	023316	113761	023556	MOVB	HOLD,3(R1)	;PUT TRACK ADDRESS INTO TABLE	
5925	023324	005737	023430	TST	6\$;SEE IF <CR> ENTERED	
5926	023330	001416		BEQ	5\$;GET OUT IF IT WAS	
5927	023332	004537	023432	JSR	R5,CONVEA	;GET SECTOR ADDRESS	
5928	023336	000401		BR	.+4	;CR MUST BE ENTERED	
5929	023340	000417		BR	3\$;INVALID SEPARATOR USED	
5930	023342	005737	023556	TST	HOLD	;SEE IF INVALID CHARACTER ENTERED	

```

5931 023346 100414          BMI      3$          ;BR IF ONE WAS
5932 023350 023727 023556 000025  CMP      HOLD,#21.  ;SEE IF SECTOR ADDRESS TOO LARGE
5933 023356 101010          BHI      3$          ;BR IF IT IS
5934 023360 113761 023556 000002  MOVB    HOLD,2(R1) ;PUT SECTOR ADDRESS INTO TABLE
5935 023366 005303          5$: DEC      R3          ;MORE ENTRIES ?
5936 023370 001413          BEQ      4$          ;BR IF NOT
5937 023372 062701 000004  ADD     #4,R1       ;INCREMENT THE TABLE POINTER
5938 023376 000705          BR       2$          ;CONTINUE
5939 023400 012711 177777 3$: MOV     #-1,(R1)    ;CLEAR PRESENT TABLE ENTRY
5940 023404 012761 177777 000002  MOV     #-1,2(R1)  ;CLEAR PRESENT TABLE ENTRY
5941 023412 104401 052646  TYPE    ,BADENT    ;'INVALID ENTRY'
5942 023416 000675          BR       2$          ;TRY AGAIN
5943 023420          4$:
5944 023420 012603          MOV     (SP)+,R3    ;:POP STACK INTO R3
5945 023422 012602          MOV     (SP)+,R2    ;:POP STACK INTO R2
5946 023424 012601          MOV     (SP)+,R1    ;:POP STACK INTO R1
5947 023426 000207          RTS     PC          ;RETURN
5948
5949 023430 000000          6$: .WORD 0          ;TRACK <CR> INDICATOR
5950
5951          ;ROUTINE TO CONVERT INPUT DECIMAL ASCII NUMBERS TO BINARY
5952          ;: RETURN IF TERMINATOR (CR); RETURN +2 IF SEPARATOR (,).
5953          ;: R2 CONTAINS ADDRESS OF ASCII STRING
5954
5955 023432 005037 023556  CONVEA: CLR     HOLD      ;CLEAR WORKING LOCATION
5956 023436 105712          1$: TST    (R2)      ;SEE IF TERMINATOR
5957 023440 001445          BEQ     5$          ;BR IF TERMINATOR
5958 023442 121237 023560  CMPB   (R2),SEPART ;LOOK FOR THE SEPARATOR
5959 023446 001437          BEQ     4$          ;BR IF SEPARATOR
5960 023450 121227 000060  CMPB   (R2),#'0    ;CHARATER GREATER OR EQUAL TO 0 ?
5961 023454 103430          BLO    3$          ;BR IF NOT
5962 023456 121227 000072  CMPB   (R2),#' :   ;IF CHARACTER GREATER THAN 9 ?
5963 023462 103025          BHIS   3$          ;BR IF IT IS
5964 023464 005737 023556  TST    HOLD        ;VALUE IN WORKING LOCATION
5965 023470 001412          BEQ     2$          ;BR IF NONE
5966 023472 006337 023556  ASL    HOLD        ;MULTIPLY 'HOLD' BY 2
5967 023476 013746 023556  MOV    HOLD,-(SP)  ;THE THE '2 TIMES' VALUE
5968 023502 006337 023556  ASL    HOLD        ;MULTIPLY 'HOLD' BY 2 AGAIN
5969 023506 006337 023556  ASL    HOLD        ;MULTIPLY 'HOLD' BY 2 AGAIN
5970 023512 062637 023556  ADD    (SP)+,HOLD  ;ADD THE '2 TIMES' VALUE
5971 023516 005046          2$: CLR     -(SP)      ;CLEAR THE STACK
5972 023520 111216          MOVB   (R2),(SP)   ;PUT THE CHARACTER ON THE STACK
5973 023522 005202          INC    R2          ;INCREMENT THE POINTER
5974 023524 142716 000060  BICB   #60,(SP)    ;CLEAR THE UPPER BITS
5975 023530 062637 023556  ADD    (SP)+,HOLD  ;ADD THE NEW CHARACTER
5976 023534 000740          BR     1$          ;GET THE NEXT CHARACTER
5977 023536 012737 177777 023556  3$: MOV     #-1,HOLD ;SET BAD CHARACTER INDICATOR
5978 023544 000403          BR     5$          ;EXIT
5979 023546 062705 000002  4$: ADD     #2,R5    ;INCREMENT THE RETURN
5980 023552 005202          INC    R2          ;INCREMENT THE POINTER FOR SEPARATOR THEI
5981 023554 000205          5$: RTS     R5          ;RETURN
5982
5983 023556 000000          HOLD: .WORD 0      ;WORKING LOCATION FOR THE CONVERSION
5984 023560 000000          SEPART: .WORD      ;SEPARATOR CHARACTER (IN ASCII) GOES HERE
5985
5986          ;TYPEOUT ASSIGN/DEASSIGN ERROR MESSAGE

```

```

5987
5988 023562 142737 000007 001263 ASNERR: BICB #7,UNIT+1 ;CLEAR PREVIOUS DRIVE NUMBER
5989 023570 150437 001263 BISB R4,UNIT+1 ;PRESENT DRIVE NUMBER
5990 023574 104401 052331 TYPE ,QUES ;QUESTION MARK
5991 023600 104401 051367 TYPE ,UNMSG ;TYPE 'UNIT'
5992 023604 104401 001262 TYPE ,UNIT ;UNIT NUMBER
5993 023610 104401 TYPE ;TYPE SPECIFIC MESSAGE
5994 023612 000000 ASNMSG: .WORD 0 ;MESSAGE ADDRESS
5995 023614 104401 001165 TYPE ,SCRLF
5996 023620 000207 RTS PC
5997
5998 ;DEASSIGN A UNIT IF A FATAL ERROR OCCURS
5999
6000 DROP: CLR R4 ;CLEAR R4 FOR UNIT NUMBER
6001 023624 111004 MOVB (R0),R4 ;MOVE UNIT NUMBER TO R4
6002 023626 105064 041640 CLRB ASNLST(R4) ;REMOVE UNIT FROM ASSIGNED LIST
6003 023632 006304 ASL R4 ;MAKE UNIT NUMBER INTO A TABLE INDEX
6004 023634 010064 041650 MOV R0,DUNIT(R4) ;PUT UNIT IN DROP LIST
6005 023640 104401 001165 TYPE ,SCRLF
6006 023644 104401 001165 TYPE ,SCRLF
6007 023650 104401 051727 TYPE ,DROPNG ;TYPE 'DROPPING UNIT'
6008 023654 104401 052040 TYPE ,DRNUM ;'DRIVE #'
6009 023660 142737 000007 001263 BICB #7,UNIT+1 ;CLEAR THE UNIT NUMBER
6010 023666 151037 001263 BISB (R0),UNIT+1 ;SET THE UNIT NUMBER
6011 023672 104401 001262 TYPE ,UNIT ;TYPE THE UNIT NUMBER
6012 023676 104401 001165 TYPE ,SCRLF
6013 023702 000207 RTS PC
6014
6015 ;ROUTINE TO DEASSIGN A UNIT IF ERRORS BECOMES EXCESSIVE
6016
6017 023704 032737 000020 001140 ABNRML: BIT #SW04,SWR ;SEE IF SWITCH 4 SET
6018 023712 001004 BNE 1$ ;BR IF IT'S SET
6019 023714 023760 001302 000062 CMP MAXER,$TOTAL(R0) ;CHECK TOTAL ERROR VALUE
6020 023722 001737 BEQ DROP ;DROP THE UNIT IF IT EXCEEDS MAX
6021 023724 000207 1$: RTS PC ;RETURN
6022
6023 ;ROUTINE TO CHECK FOR END OF PASS AND END OF TEST
6024
6025 023726 005737 001322 NRML: TST ENDET ;END OF PASS DETERMINED BY SEEKS OR WORDS ?
6026 023732 001412 BEQ NRML1 ;BR IF SEEKS
6027 023734 026037 000060 001270 CMP $READ+2(R0),ENDCON+2 ;CHECK MSW OF WORDS READ COUNT
6028 023742 101017 BHI NRML2 ;BR IF MSW GREATER THAN LIMIT
6029 023744 103405 BLO NRML1 ;BR IF MSW LESS THAN LIMIT
6030 023746 026037 000056 001266 CMP $READ(R0),ENDCON ;CHECK LSW AGAINST LIMIT
6031 023754 103012 BHS NRML2 ;BR IF EQUAL OR GREATER
6032 023756 000506 BR NRMLX ;EXIT
6033 023760 026037 000050 001274 NRML1: CMP $POSIT+2(R0),ENDSEK+2 ;CHECK MSW OF SEEK COUNT
6034 023766 101005 BHI NRML2 ;BR IF MSW GREATER THAN LIMIT
6035 023770 103501 BLO NRMLX ;EXIT IF MSW LESS THAN LIMIT
6036 023772 026037 000046 001272 CMP $POSIT(R0),ENDSEK ;CHECK LSW OF SEEK COUNT
6037 024000 103475 BLO NRMLX ;EXIT IF LSW LESS THAN LIMIT
6038 024002 104401 001165 NRML2: TYPE ,SCRLF ;CR-LF
6039 024006 104401 051763 TYPE ,ENDPAS ;END OF PASS FOR THE UNIT
6040 024012 016046 000074 MOV $PASSC(R0),-(SP) ;PUT PASS COUNT ON THE STACK
6041 024016 104405 TYPDS ;CONVERT PASS COUNT TO DECIMAL AND TYPE IT
6042 024020 142737 000007 001263 BICB #7,UNIT+1 ;CLEAR THE UNIT NUMBER

```

```

6043 024026 151037 001263      B1SB      (R0),UNIT+1      ;SET THE UNIT NUMBER
6044 024032 032737 000020 001140      BIT        #SW04,SWR      ;SWITCH 4 SET ?
6045 024040 001015                BNE        1$           ;BR IF SET
6046 024042 026037 000074 001276      CMP        $PASSC(R0),PASCNT ;SEE IF AT END OF TEST
6047 024050 103411                BLO        1$           ;BR IF NOT
6048 024052 104401 051777      TYPE      ,ENDTST       ;TYPE 'END OF TEST'
6049 024056 104401 052040      TYPE      ,DRNUM        ;'DRIVE #'
6050 024062 104401 001262      TYPE      ,UNIT         ;DRIVE NUMBER
6051 024066 104401 001165      TYPE      ,$CRLF        ;CR-LF
6052 024072 000427                BR         3$           ;DEASSIGN THE DRIVE
6053 024074 104401 052040      1$:      TYPE      ,DRNUM        ;'DRIVE #'
6054 024100 104401 001262      TYPE      ,UNIT         ;DRIVE NUMBER
6055 024104 104401 001165      TYPE      ,$CRLF        ;CR-LF
6056 024110 004737 021176      JSR       PC,TYPEST     ;TYPE THE UNIT'S STATISTICS
6057 024114 010346      MOV       R3,-(SP)      ;SAVE R3
6058 024116 010446      MOV       R4,-(SP)      ;SAVE R4
6059 024120 010004      MOV       R0,R4         ;UNIT'S BLOCK ADDRESS
6060 024122 062704 000042      ADD       #SOPERC,R4    ;ADD THE STARTING ADDR OF SECTIONS TO CLEAR
6061 024126 012703 000010      MOV       #8.,R3        ;NUMBER OF LOCNS TO BE CLEARED
6062                                ;(ERROR COUNTERS NOT CLEARED)
6063 024132 005024      2$:      CLR        (R4)+      ;CLEAR THE LOCN
6064 024134 005303      DEC       R3            ;DECREMENT THE LOCATION COUNTER
6065 024136 001375      BNE       2$           ;BR IF MORE TO GO
6066 024140 012604      MOV       (SP)+,R4      ;RESTORE R4
6067 024142 012603      MOV       (SP)+,R3      ;RESTORE R3
6068 024144 005260 000074      INC       $PASSC(R0)    ;INCREMENT THE PASS COUNT
6069 024150 000411      BR        NRMLX         ;EXIT
6070 024152 104401 001165      3$:      TYPE      , $CRLF
6071 024156 005004      CLR       R4            ;CLEAR R4 FOR DRIVE NUMBER
6072 024160 111004      MOVB     (R0),R4        ;MOVE DRIVE NUMBER
6073 024162 105064 041640      CLRB     ASNLST(R4)     ;DELETE DRIVE FROM ASSIGNED LIST
6074 024166 006304      ASL      R4            ;MAKE DRIVE NUMBER INTO TABLE INDEX
6075 024170 010064 041650      MOV      R0,DUNIT(R4)  ;PUT BLOCK ADDRESS INTO DROP LIST
6076 024174 000207      NRMLX:   RTS           ;RETURN
6077
6078                                ;ROUTINE TO GET THE REMAINDER OF THE RANDOM NUMBER
6079
6080 024176 013746 027736      GETREM:  MOV      $LONUM,-(SP) ;STORE RANDOM NUMBER ON THE STACK FOR DIVIDE
6081 024202 013746 027734      MOV      $HINUM,-(SP)    ;UPPER PART
6082 024206 010546      MOV      R5,-(SP)        ;PUT THE DIVISOR ONTO THE STACK
6083 024210 004737 024224      JSR     PC,LINKDV       ;DIVIDE THE RANDOM NUMBERS
6084 024214 012605      MOV      (SP)+,R5        ;PUT THE REMAINDER INTO R5
6085 024216 005726      TST     (SP)+           ;ADJUST THE STACK POINTER
6086 024220 000240      NOP
6087 024222 000207      RTS        PC           ;FOR DEBUGGING HALT
6088
6089                                ;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE
6090                                ; THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE
6091                                ; CALLING SEQUENCE TO BE USED
6092
6093 024224      LINKDV:
6094 024224 010546      MOV      R5,-(SP)        ;;PUSH R5 ON STACK
6095 024226 010446      MOV      R4,-(SP)        ;;PUSH R4 ON STACK
6096 024230 010346      MOV      R3,-(SP)        ;;PUSH R3 ON STACK
6097 024232 010246      MOV      R2,-(SP)        ;;PUSH R2 ON STACK
6098 024234 010146      MOV      R1,-(SP)        ;;PUSH R1 ON STACK
  
```

6099	024236	010046		MOV	RO, -(SP)	:: PUSH RO ON STACK
6100	024240	016605	000016	MOV	16(SP), R5	:: DIVISOR
6101	024244	005004		CLR	R4	:: OTHER DIVISOR WORD
6102	024246	016602	000020	MOV	20(SP), R2	:: UPPER DIVIDEND WORD
6103	024252	016603	000022	MOV	22(SP), R3	:: LOWER DIVIDEND WORD
6104	024256	005000		CLR	RO	:: CLEAR OTHER DIVIDEND REGISTERS
6105	024260	005001		CLR	R1	
6106	024262	004737	024316	JSR	PC, M.DPID	:: GO TO THE DIVIDE ROUTINE
6107	024266	010166	000020	MOV	R1, 20(SP)	:: REMAINDER ON THE STACK
6108	024272	010366	000022	MOV	R3, 22(SP)	:: QUOTIENT ON THE STACK
6109	024276	012600		MOV	(SP)+, RO	:: POP STACK INTO RO
6110	024300	012601		MOV	(SP)+, R1	:: POP STACK INTO R1
6111	024302	012602		MOV	(SP)+, R2	:: POP STACK INTO R2
6112	024304	012603		MOV	(SP)+, R3	:: POP STACK INTO R3
6113	024306	012604		MOV	(SP)+, R4	:: POP STACK INTO R4
6114	024310	012605		MOV	(SP)+, R5	:: POP STACK INTO R5
6115	024312	012616		MOV	(SP)+, (SP)	:: MOVE RETURN UP THE STACK
6116	024314	000207		RTS	PC	
6117						
6118						
6119						
6120						
6121						
6122						
6123						
6124						
6125	024316	012746	000040	M.DPID: MOV	#40, -(SP)	:: COUNTER FOR DIVISION CYCLES
6126	024322	010446		MOV	R4, -(SP)	:: HIGH ORDER
6127	024324	010546		MOV	R5, -(SP)	:: LOW ORDER DIVISOR TO THE STACK
6128	024326	005466	000002	NEG	2(SP)	:: FORM NEGATIVE
6129	024332	005416		NEG	2SP	:: VERSION OF THE DIVISOR
6130	024334	005666	000002	SBC	2(SP)	
6131	024340	061601		ADD	2SP, R1	
6132	024342	005500		ADC	RO	:: PERFORM THE INITIAL SUBTRACTION
6133	024344	066600	000002	ADD	2(SP), RO	
6134	024350	103445		BCS	M.DP50	:: IF CARRY THEN OVERFLOW HAS OCCURRED
6135	024352	005046		CLR	-(SP)	:: THIS IS A LONGER LASTING CARRY BIT
6136	024354	006103		M.DP40: ROL	R3	
6137	024356	006102		ROL	R2	
6138	024360	006101		ROL	R1	
6139	024362	006100		ROL	RO	
6140	024364	005716		TST	2SP	:: TEST "CARRY" INDICATOR
6141	024366	001410		BEQ	M.DP41	:: IF NO "CARRY" THEN ADD ELSE SUBTRACT
6142	024370	005016		CLR	2SP	:: CLEAR UP FOR NEXT TIME
6143	024372	066601	000002	ADD	2(SP), R1	
6144	024376	005500		ADC	RO	:: ADD -(DIVISOR)
6145	024400	005516		ADC	2SP	:: SET "CARRY"
6146	024402	066600	000004	ADD	4(SP), RO ; <- I	
6147	024406	000404		BR	M.DP42	
6148	024410	060501		M.DP41: ADD	R5, R1	
6149	024412	005500		ADC	RO	:: ADD +(DIVISOR)
6150	024414	005516		ADC	2SP	:: SET "CARRY"
6151	024416	060400		ADD	R4, RO ; <- I	
6152	024420	005516		M.DP42: ADC	2SP	:: SET "CARRY"
6153	024422	005716		TST	2SP	:: TEST THE UPDATE INDICATOR
6154	024424	001401		BEQ	.+4 ; ->	:: IF ZERO FORGET IT

DIVISION UTILITY SUBROUTINE
 RO-R1-R2-R3=DIVIDEND
 R4-R5=DIVISOR
 RO-R1=REMAINDER AFTER DIVISION
 R2-R3=QUOTIENT AFTER DIVISION
 ENTER WITH JSR PC, M.DPID

```

6155 024426 005203          INC      R3      ; I      ;NO CARRY POSSIBLE HERE
6156 024430 005366 000006  DEC      6(SP)  ; <-    ;DECREMENT COUNTER
6157 024434 003347          BGT      M.DP40 ;        ;BRANCH IF MORE TO DO
6158 024436 006003          ROR      R3
6159 024440 103404          BCS      M.DP44
6160 024442 060501          ADD      R5,R1
6161 024444 005500          ADC      R0
6162 024446 060400          ADD      R4,R0
6163 024450 000241          CLC
6164 024452 006103          M.DP44: ROL      R3
6165 024454 062706 000010  ADD      #10,SP ;ADJUST STACK BY 4 WORDS
6166 024460 000242          CLV
6167 024462 000207          RTS      PC
6168 024464 062706 000006  M.DP50: ADD      #6,SP
6169 024470 000262          SEV
6170 024472 000207          RTS      PC
6171
6172
6173
6174          ;LINK SUBROUTINE TO MULTIPLY ROUTINE 'M.DPIM'
6175
6176 024474          LINKML:
6177 024474 010046          MOV      R0,-(SP) ;:;PUSH R0 ON STACK
6178 024476 010146          MOV      R1,-(SP) ;:;PUSH R1 ON STACK
6179 024500 010246          MOV      R2,-(SP) ;:;PUSH R2 ON STACK
6180 024502 010346          MOV      R3,-(SP) ;:;PUSH R3 ON STACK
6181 024504 010446          MOV      R4,-(SP) ;:;PUSH R4 ON STACK
6182 024506 010546          MOV      R5,-(SP) ;:;PUSH R5 ON STACK
6183 024510 005002          CLR      R2      ;PUT MULTIPLIER IN R2 & R3
6184 024512 016603 000020  MOV      20(SP),R3
6185 024516 005004          CLR      R4      ;PUT MULTIPLICAND IN R4 & R5
6186 024520 016605 000016  MOV      16(SP),R5
6187 024524 004737 024556  JSR      PC,M.DPIM ;MULTIPLY R0 & R1 BY R2 & R3
6188 024530 010266 000016  MOV      R2,16(SP) ;HIGH ORDER PRODUCT
6189 024534 010366 000020  MOV      R3,20(SP) ;LOW ORDER PRODUCT
6190 024540 012605          MOV      (SP)+,R5 ;:;POP STACK INTO R5
6191 024542 012604          MOV      (SP)+,R4 ;:;POP STACK INTO R4
6192 024544 012603          MOV      (SP)+,R3 ;:;POP STACK INTO R3
6193 024546 012602          MOV      (SP)+,R2 ;:;POP STACK INTO R2
6194 024550 012601          MOV      (SP)+,R1 ;:;POP STACK INTO R1
6195 024552 012600          MOV      (SP)+,R0 ;:;POP STACK INTO R0
6196 024554 000207          RTS      PC
6197
6198          ;
6199          ;
6200          ;
6201          ;
6202          ;
6203          ;
6204          ;
6205          ;
6206 024556 005000          M.DPIM: CLR      R0      ;CLEAR HIGH ORDER WORDS
6207 024560 005001          CLR      R1
6208 024562 012746 000041  MOV      #41,-(SP) ;MOVE 33 DEC TO COUNTER
6209 024566 006000          M.DP01: ROR      R0
6210 024570 006001          ROR      R1
  
```

```

6211 024572 006002          ROR      R2          ;SHIFT TO ADD
6212 024574 006003          ROR      R3
6213 024576 103003          BCC     M.DP02      ;NO CARRY NO ADD
6214 024600 060501          ADD     R5,R1
6215 024602 005500          ADC     R0          ;ADD DOUBLE PRECISION TO OBTAIN NEW PARTIAL
6216 024604 060400          ADD     R4,R0      ;PRODUCT
6217 024606 005316          M.DP02: DEC     R5P    ;DECREMENT COUNTER
6218 024610 001366          BNE     M.DP01
6219 024612 005726          TST     (SP)+      ;REMOVE THE COUNTER
6220 024614 000207          RTS     PC
  
```

;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES

```

6224 024616 012737 000004 024736 $RPZR8: MOV     #4,RPZRC ;SETUP TO TYPE 8 SIGNIFICANT DIGITS
6225 024624 000412          BR      RPZERO    ;TYPE THE VALUE
6226 024626 012737 000006 024736 $RPZR4: MOV     #6,RPZRC ;SETUP TO TYPE 4 SIGNIFICANT DIGITS
6227 024634 000406          BR      RPZERO    ;TYPE THE VALUE
6228 024636 012737 000007 024736 $RPZR5: MOV     #7,RPZRC ;SETUP TO TYPE 3 SIGNIFICANT DIGITS
6229 024644 000402          BR      RPZERO    ;TYPE THE VALUE
6230 024646 005037 024736 $RPZR0: CLR     RPZRC ;SETUP TO TYPE 10 SIGNIFICANT DIGITS
6231 024652 010046          RPZERO: MOV    RO,-(SP) ;SAVE RO
6232 024654 016600 000004          MOV    4(SP),RO ;ADDRESS OF NUMBER TO RO
6233 024660 122710 000060 1$:      CMPB   #'0,(RO) ;BYTE EQUAL TO ASCII '0' ?
6234 024664 001004          BNE     2$
6235 024666 112710 000040          MOVB   #40,(RO) ;REPLACE THE ZERO WITH A SPACE
6236 024672 005200          INC     RO        ;INCREMENT THE BYTE ADDRESS
6237 024674 000771          BR      1$
6238 024676 105710          2$:      TSTB   (RO)      ;GO BACK AND LOOK FOR MORE LEADING ZEROS
6239 024700 001003          BNE     3$
6240 024702 005300          DEC     RO        ;SEE IF ZERO BYTE TERMINATOR
6241 024704 112710 000060          MOVB   #'0,(RO) ;BR IF NOT
6242 024710 016637 000004 024726 3$:      MOV    4(SP),4$ ;BACKUP STRING POINTER
6243 024716 063737 024736 024726          ADD    RPZRC,4$  ;PUT A ZERO BACK IN
6244 024724 104401          TYPE   ;PUT ADDRESS IN LOCATION FOR TYPEOUT
6245 024726 000000          4$:      .WORD 0 ;BEGINNING OF SIGNIFICANT DIGITS
6246 024730 012600          MOV    (SP)+,RO  ;TYPE THE NUMBER
6247 024732 012616          MOV    (SP)+,(SP) ;ADDRESS OF NUMBER
6248 024734 000207          RTS     PC        ;RESTORE RO
6249          ;MOVE RETURN ADDRESS
6250 024736 000000          RPZRC: .WORD 0 ;RETURN
6251          ;OFFSET FOR CONVERTED DIGITS HERE
  
```

;CONVERT AN OCTAL NUMBER TO ASCII AND PRINT IT
 ; (ROUTINE ADAPTED FROM '\$TYPOC')

```

6255 024740 112737 000001 025131 PRT0CT: MOVB   #1,FILLO ;SET THE ZERO FILL SWITCH
6256 024746 112737 000006 025133          MOVB   #6,OMODE+1 ;SET FOR SIX(6) DIGITS
6257 024754 112737 000005 025130          MOVB   #5,OCNT   ;SET THE ITERATION COUNT
6258 024762 010346          MOV    R3,-(SP) ;SAVE R3
6259 024764 010446          MOV    R4,-(SP) ;SAVE R4
6260 024766 010546          MOV    R5,-(SP) ;SAVE R5
6261 024770 113704 025133          MOVB   OMODE+1,R4 ;GET THE NUMBER OF DIGITS TO TYPE
6262 024774 005404          NEG    R4
6263 024776 062704 000006          ADD    #6,R4     ;SUBTRACT IT FOR MAX. ALLOWED
6264 025002 110437 025132          MOVB   R4,OMODE ;SAVE IT FOR USE
6265 025006 113704 025131          MOVB   FILLO,R4 ;GET THE ZERO FILL SWITCH
6266 025012 016605 000010          MOV    10(SP),R5 ;PICKUP THE INPUT NUMBER
  
```



```

6267 025016 005003          CLR      R3          ;CLEAR THE OUTPUT WORD
6268 025020 006105          1$:     ROL      R5          ;ROTATE MSB INTO "C"
6269 025022 000404          BR       3$          ;GO DO MSB
6270 025024 006105          2$:     ROL      R5          ;FORM THIS DIGIT
6271 025026 006105          ROL      R5
6272 025030 006105          ROL      R5
6273 025032 010503          MOV      R5,R3
6274 025034 006103          3$:     ROL      R3          ;GET LSB OF THIS DIGIT
6275 025036 105337 025132    DECB     OMODE        ;TYPE THIS DIGIT?
6276 025042 100016          BPL      7$          ;BR IF NO
6277 025044 042703 177770    BIC      #177770,R3   ;GET RID OF JUNK
6278 025050 001002          BNE      4$          ;TEST FOR 0
6279 025052 005704          TST      R4          ;SUPPRESS THIS 0?
6280 025054 001403          BEQ      5$          ;BR IF YES
6281 025056 005204          4$:     INC      R4          ;DON'T SUPPRESS ANYMORE 0'S
6282 025060 052703 000060    BIS      #'0,R3      ;MAKE THIS DIGIT ASCII
6283 025064 052703 000040    BIS      #' ,R3      ;MAKE ASCII IF NOT ALREADY
6284 025070 110337 025126    MOVB     R3,8$        ;SAVE FOR TYPING
6285 025074 104412 025126    DISPLY   ,8$          ;GO PRINT THIS DIGIT
6286 025100 105337 025130    7$:     DECB     0CNT        ;COUNT BY 1
6287 025104 003347          BGT      2$          ;BR IF MORE TO DO
6288 025106 002402          BLT      6$          ;BR IF DONE
6289 025110 005204          INC      R4          ;INSURE LAST DIGIT ISN'T A BLANK
6290 025112 000744          BR       2$          ;GO DO THE LAST DIGIT
6291 025114 012605          6$:     MOV      (SP)+,R5 ;RESTORE R5
6292 025116 012604          MOV      (SP)+,R4   ;RESTORE R4
6293 025120 012603          MOV      (SP)+,R3   ;RESTORE R3
6294 025122 012616          MOV      (SP)+,(SP) ;SET UP THE STACK FOR RETURNING
6295 025124 000207          RTS      PC          ;RETURN
6296 025126 000          8$:     .BYTE   0          ;STORAGE FOR ASCII DIGIT
6297 025127 000          .BYTE   0          ;TERMINATOR FOR TYPE ROUTINE
6298 025130 000          0CNT:   .BYTE   0          ;OCTAL DIGIT COUNTER
6299 025131 000          FILL0:  .BYTE   0          ;ZERO FILL SWITCH
6300 025132 000000          0MODE:  0          ;NUMBER OF DIGITS TO TYPE
6301
6302          ;ROUTINE TO TYPE AT PRIORITY 4
6303
6304 025134 013746 177776    TYPRI4: MOV      0#PS,-(SP) ;SAVE THE PRESENT STATUS
6305 025140 012737 000200 177776    MOV      #200,0#PS    ;CHANGE THE PRIORITY TO 4
6306 025146 012537 025156    MOV      (R5)+,1$     ;MESSAGE ADDRESS
6307 025152 004737 025636    JSR      PC,$TYPE     ;TYPE THE MESSAGE
6308 025156 000000          1$:     .WORD   0          ;MESSAGE ADDRESS GOES HERE
6309 025160 000205          RTS      R5          ;RETURN
6310
6311          ;ROUTINE TO ROUTE MESSAGES TO EITHER THE PRINTER (IF AVAILABLE) OR
6312          ;THE TELETYPE
6313          ;NOTE: $SDSPY MUST BE DEFINED BY 'SETTRAP'
6314
6315 025162 032737 020000 001140 $SDSPY: BIT      #SW13,SWR ;INHIBIT TYPEOUTS
6316 025170 001027          BNE      5$          ;BR IF SET
6317 025172 012737 000200 177776 1$:     MOV      #200,0#PS    ;CHANGE THE PRIORITY TO 4
6318 025200 005737 001212    TST      $PRFLG      ;PRINTER ON THE SYSTEM ?
6319 025204 100411          BMI      3$          ;BR IF NOT
6320 025206 017637 000000 025224    MOV      0(SP),2$     ;GET THE MESSAGE ADDRESS
6321 025214 013746 177776    MOV      0#PS,-(SP)  ;PUT THE PROC STATUS ON THE STACK
6322 025220 004737 025744    JSR      PC,$PRINT   ;PRINT THE MESSAGE

```

```

6323 025224 000000          2$: .WORD 0          ;MESSAGE ADDRESS GOES HERE
6324 025226 000410          BR 5$          ;EXIT
6325 025230 017637 000000 025246 3$: MOV 2(SP),4$    ;SET UP TO TYPE THE MESSAGE
6326 025236 013746 177776          MOV 2#PS,-(SP) ;PUT THE PROC STATUS ON THE STACK
6327 025242 004737 025636          JSR PC,$TYPE   ;TYPE THE MESSAGE
6328 025246 000000          4$: .WORD 0          ;MESSAGE ADDRESS GOES HERE
6329 025250 062716 000002          5$: ADD 2,(SP)   ;RETURN ADDRESS
6330 025254 000002
6331
6332          ;SAVE THE REGISTERS IF A DRIVER ERROR CALL
6333
6334 025256 010537 001260          ERENT: MOV R5,ATTN ;SAVE THE ATTENTION REGISTER CONTENTS
6335 025262 010137 001256          MOV R1,DRIVE   ;DRIVE NUMBER
6336 025266 032737 020000 001140          BIT #SW13,SWR ;INHIBIT PRINTOUTS ?
6337 025274 001002          BNE 1$        ;BR IF YES
6338 025276 004737 021402          JSR PC,$TIME  ;TYPE THE TIME
6339 025302 000207          1$: RTS PC
6340
6341          ;ROUTINE TO GET AN OCTAL PARAMETER
6342          ;RETURN +2 IF CARRIAGE RETURN ENTERED
6343
6344 025304 011646          GETOCT: MOV (SP),-(SP) ;PROVIDE SPACE FOR THE
6345 025306 104407          1$: ROLIN      ;READ AN ASCIZ LINE
6346 025310 012600          MOV (SP)+,R0   ;GET ADDRESS OF 1ST CHARACTER
6347 025312 010037 025416          MOV R0,5$     ;AND SAVE IT
6348 025316 105710          TST3 (R0)     ;FIRST CHARACTER A 'CR'
6349 025320 001442          BEQ 6$        ;BR IF IT IS
6350 025322 122710 000003          CMPB #3,(R0) ;CONTROL C ?
6351 025326 001443          BEQ 8$        ;BR IF IT IS
6352 025330 005001          CLR R1       ;CLEAR DATA WORD
6353 025332 005002          CLR R2
6354 025334 112046          2$: MOV B (R0)+,-(SP) ;PICKUP THIS CHARACTER
6355 025336 001420          BEQ 3$       ;IF ZERO GET OUT
6356 025340 122716 000060          CMPB #'0,(SP) ;MAKE SURE THIS CHARACTER
6357 025344 003021          BGT 4$       ;IS AN OCTAL DIGIT
6358 025346 122716 000067          CMPB #'7,(SP)
6359 025352 002416          BLT 4$
6360 025354 006301          ASL R1       ;*2
6361 025356 006102          ROL R2
6362 025360 006301          ASL R1       ;*4
6363 025362 006102          ROL R2
6364 025364 006301          ASL R1       ;*8
6365 025366 006102          ROL R2
6366 025370 042716 177770          BIC #1C7,(SP) ;STRIP THE ASCII JUNK
6367 025374 062601          ADD (SP)+,R1 ;ADD IN THIS DIGIT
6368 025376 000756          BR 2$        ;LOOP
6369 025400 005726          3$: TST (SP)+   ;CLEAN TERMINATOR FROM STACK
6370 025402 010166 000002          MOV R1,2(SP) ;SAVE THE RESULT
6371 025406 000412          BR 7$        ;EXIT
6372 025410 005726          4$: TST (SP)+   ;CLEAN PARTIAL FROM STACK
6373 025412 105010          CLRB (R0)   ;SET A TERMINATOR
6374 025414 104401          TYPE       ;TYPE UP THRU THE BAD CHAR.
6375 025416 000000          5$: .WORD 0
6376 025420 104401 001164          TYPE $QUES  ;"? "CR" & "LF"
6377 025424 000730          BR 1$       ;TRY AGAIN
6378 025426 012616          6$: MOV (SP)+,(SP) ;RESTORE THE PC

```

```

6379 025430 062716 000002          ADD    #2,(SP)          ;INCREMENT RETURN
6380 025434 000207          7$:   RTS    PC          ;RETURN
6381 025436 000137 002250          8$:   JMP    ENTPR       ;CONTROL C ENTERED
6382
6383          ;ROUTINE TO GET DECIMAL PARAMETERS
6384          ;RETURN +2 IF CARRIAGE RETURN ENTERED
6385
6386 025442 011646          GETDEC: MOV    (SP),-(SP) ;PROVIDE SPACE FOR
6387 025444 104407          1$:   RDLIN          ;READ AN ASCIZ LINE
6388 025446 012600          MOV    (SP)+,R0        ;ADDRESS OF 1ST CHAR.
6389 025450 010037 025552          MOV    R0,6$          ;SAVE IN CASE OF BAD INPUT
6390 025454 105710          TSTB   (R0)            ;FIRST CHARACTER A 'CR' ?
6391 025456 001441          BEQ    7$              ;BR IF IT IS
6392 025460 122710 000003          CMPB   #3,(R0)        ;CONTROL C ?
6393 025464 001442          BEQ    9$              ;BR IF IT IS
6394 025466 005046          CLR    -(SP)          ;CLEAR DATA WORD
6395 025470 112001          2$:   MOVB   (R0)+,R1    ;PICKUP THIS CHARACTER
6396 025472 001421          BEQ    4$              ;GET OUT IF ZERO
6397 025474 122701 000060          CMPB   #'0',R1        ;MAKE SURE THIS CHARACTER
6398 025500 003021          BGT    5$              ;IS A DIGIT BETWEEN 0 & 9
6399 025502 122701 000071          CMPB   #'9',R1
6400 025506 002416          BLT    5$
6401 025510 006316          ASL    (SP)            ;*2
6402 025512 011646          MOV    (SP),-(SP)    ;SAVE FOR LATER
6403 025514 006316          ASL    (SP)            ;*4
6404 025516 006316          ASL    (SP)            ;*8
6405 025520 062616          ADD    (SP)+,(SP)    ;*10
6406 025522 102410          BVS    5$              ;OVERFLOW ISN'T ALLOWED
6407 025524 162701 000060          SUB    #'0',R1        ;STRIP AWAY THE ASCII JUNK
6408 025530 060116          ADD    R1,(SP)        ;ADD IN THIS DIGIT
6409 025532 102404          BVS    5$              ;OVERFLOW ISN'T ALLOWED
6410 025534 000755          BR     2$              ;LOOP
6411 025536 012666 000002          4$:   MOV    (SP)+,2(SP) ;SAVE THE RESULT
6412 025542 000412          BR     8$              ;EXIT
6413 025544 005726          5$:   TST    (SP)+        ;CLEAN PARTIAL NUMBER FROM STACK
6414 025546 105010          CLRB   (R0)            ;SET A TERMINATOR
6415 025550 104401          TYPE   ;TYPE THE INPUT UP TO BAD CHAR.
6416 025552 000000          6$:   .WORD 0            ;POINTER GOES HERE
6417 025554 104401 001164          TYPE   $QUES          ;"?" "CR" & "LF"
6418 025560 000731          BR     1$              ;TRY AGAIN
6419 025562 012616          7$:   MOV    (SP)+,(SP)  ;RESTORE THE PC
6420 025564 062716 000002          ADD    #2,(SP)        ;INCREMENT THE RETURN
6421 025570 000207          8$:   RTS    PC          ;RETURN
6422 025572 000137 002250          9$:   JMP    ENTPR       ;CONTROL C ENTERED
6423
6424          ;*****
6425
6426          .SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
6427
6428          ;*CALL
6429          ;*   MOV    #NUMADR, -(SP) ;FIRST ADDRESS OF ASCIZ STRING
6430          ;*   JSR    PC, @#SSUPRS
6431
6432          ;NOTE: ROUTINE MODIFIED FROM THE 'SYSMAC' ROUTINE
6433
6434

```

6435	025576	010046		SSUPRS:	MOV	RO, -(SP)		:SAVE RO
6436	025600	016600	000004		MOV	4(SP), RO		:PICKUP THE POINTER
6437	025604	105710		1\$:	TSTB	(RO)		:TERMINATEOR?
6438	025606	001403			BEQ	2\$:BR IF YES
6439	025610	122720	000060		CMPB	#'0, (RO)+		:IS THIS AN ASCII "0" ?
6440	025614	001773			BEQ	1\$:BR IF YES
6441	025616	005300		2\$:	DEC	RO		:BACKUP BY "1"
6442	025620	010037	025626		MOV	RO, 3\$:SAVE FOR TYPING
6443	025624	104412			DISPLY			:GO PRINT
6444	025626	000000		3\$:	.WORD	0		:ASCIZ POINTER GOES HERE
6445	025630	012600			MOV	(SP)+, RO		:RESTORE RO
6446	025632	012616			MOV	(SP)+, (SP)		:RESTORE THE STACK
6447	025634	000207			RTS	PC		:RETURN
6448								
6449								
6450								
6451								
6452								
6453								
6454								
6455								
6456								
6457								
6458								
6459								
6460								
6461								
6462								
6463								
6464								
6465								
6466								
6467								
6468								
6469								
6470								
6471	025636	105737	001157	\$TYPE:	TSTB	\$TPFLG		:IS THERE A TERMINAL?
6472	025642	100002			BPL	1\$:BR IF YES
6473	025644	000000			HALT			:HALT HERE IF NO TERMINAL
6474	025646	000407			BR	3\$:LEAVE
6475	025650	010046		1\$:	MOV	RO, -(SP)		:SAVE RO
6476	025652	017600	000002		MOV	32(SP), RO		:GET ADDRESS OF ASCIZ STRING
6477	025656	112046		2\$:	MOVB	(RO)+, -(SP)		:PUSH CHARACTER TO BE TYPED ONTO STACK
6478	025660	001005			BNE	4\$:BR IF IT ISN'T THE TERMINATOR
6479	025662	005726			TST	(SP)+		:IF TERMINATOR POP IT OFF THE STACK
6480	025664	012600			MOV	(SP)+, RO		:RESTORE RO
6481	025666	062716	000002	3\$:	ADD	#2, (SP)		:ADJUST RETURN PC
6482	025672	000002			RTI			:RETURN
6483	025674	004737	025726	4\$:	JSR	PC, 7\$:GO TYPE THIS CHARACTER
6484	025700	123726	001156	5\$:	CMPB	\$FILLC, (SP)+		:IS IT TIME FOR FILLER CHARS.?
6485	025704	001364			BNE	2\$:IF NO GO GET NEXT CHAR.
6486	025706	013746	001154		MOV	\$NULL, -(SP)		:GET # OF FILLER CHARS. NEEDED
6487								:AND THE NULL CHAR.
6488	025712	105366	000001	6\$:	DECB	1(SP)		:DOES A NULL NEED TO BE TYPED?
6489	025716	002770			BLT	5\$:BR IF NO--GO POP THE NULL OFF OF STACK
6490	025720	004737	025726		JSR	PC, 7\$:GO TYPE A NULL

;;*****

.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
 ;*OR
 ;* TYPE
 ;* MESADR
 ;*2) USING A JSR INSTRUCTION
 ;* MOV PS, -(SP)
 ;* JSR PC, \$TYPE
 ;* MESADDR

;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING

;PUSH PROCESSOR STATUS WORD ON THE STACK
 ;CALL TYPE ROUTINE
 ;FIRST ADDRESS OF MESSAGE

```

6491 025724 000772          BR      6$      ;LOOP
6492 025726 105777 153216 7$:  TSTB   2$TPS   ;WAIT UNTIL PRINTER IS READY
6493 025732 100375          BPL    7$
6494 025734 116677 000002 153210 MOVB   2(SP),2$TPB ;LOAD CHAR TO BE TYPED INTO DATA REG.
6495 025742 000207          RTS     PC
6496
6497 ;:*****
6498
6499 .SBTTL PRINT ROUTINE
6500
6501 ;*ROUTINE TO PRINT ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE
6502 ;*
6503 ;*CALL:
6504 ;*1) USING A TRAP INSTRUCTION
6505 ;*   DISPLY ,MESADR      ;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
6506 ;*
6507 ;*OR
6508 ;*   PRINT
6509 ;*   MESADR
6510 ;*
6511 ;*2) USING A JSR INSTRUCTION
6512 ;*   MOV   PS,-(SP)      ;PUSH PROCESSOR STATUS WORD ON THE STACK
6513 ;*   JSR   PC,$PRINT    ;CALL PRINT ROUTINE
6514 ;*   MESADDR            ;FIRST ADDRESS OF MESSAGE
6515
6516 025744 005737 001212 $PRINT: TST   $PRFLG   ;IS THERE A PRINTER
6517 025750 100002          BPL    1$      ;BR IF YES
6518 025752 000000          HALT
6519 025754 000412          BR     7$      ;RETURN & TYPE INSTEAD
6520 025756 005777 153232 1$:  TST   2$LSCS   ;PRINTER ERROR ?
6521 025762 100012          BPL    6$      ;BR IF NO ERROR
6522 025764 005737 001220 TST   $PRTER    ;WAS ERROR SET BEFORE ?
6523 025770 001004          BNE   7$      ;BR IF YES, EXIT & TYPE INSTEAD
6524 025772 005137 001220 COM   $PRTER    ;SET PRINTER ERROR FLAG
6525 025776 104401 052333 TYPE  ,PRTATN   ;TYPE 'PRINTER ATTENTION'
6526 026002 062716 000004 7$:  ADD   4,(SP)   ;INCREMENT RETURN TO GO TO TYPE ROUTINE
6527 026006 000002          RTI
6528 026010 005037 001220 6$:  CLR   $PRTER   ;CLEAR PRINTER ERROR FLAG
6529 026014 010046          MOV   RO,-(SP) ;SAVE RO
6530 026016 017600 000002 MOV   2(SP),RO  ;GET ADDRESS OF ASCIZ STRING
6531 026022 005746          TST   -(SP)    ;MOVE THE STACK POINTER DOWN
6532 026024 112016          2$:  MOVB  (RO)+,(SP) ;PUSH CHARACTER TO BE TYPED ONTO THE STACK.
6533 026026 001005          BNE   4$      ;BR IF IT ISN'T THE TERMINATOR
6534 026030 005726          TST   (SP)+   ;IF TERMINATOR, POP IT OFF THE STACK
6535 026032 012600          MOV   (SP)+,RO ;RESTORE RO
6536 026034 062716 000002 3$:  ADD   2,(SP)   ;ADJUST RETURN PC
6537 026040 000002          RTI
6538 026042 004737 026050 4$:  JSR   PC,5$   ;PRINT THIS CHARACTER
6539 026046 000766          BR    2$      ;GET NEXT CHAR
6540 026050 105777 153140 5$:  TSTB  2$LSCS   ;WAIT UNTIL PRINTER READY
6541 026054 100375          BPL    5$
6542 026056 116677 000002 153132 MOVB  2(SP),2$LSDB ;LOAD CHAR TO BE PRINTED INTO DATA REG
6543 026064 000207          RTS     PC
6544
6545 ;:*****
6546

```

6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557 026066 005037 026260
6558 026072 012737 026264 026262
6559 026100 012737 026130 000060
6560 026106 012737 000200 000062
6561 026114 005777 153026
6562 026120 012777 000100 153016
6563 026126 000207
6564
6565
6566
6567
6568 026130 117746 153012
6569 026134 042716 177600
6570 026140 005737 026260
6571 026144 001004
6572 026146 004737 021402
6573 026152 104401 001165
6574 026156 111637 026256
6575 026162 104401 026256
6576 026166 022716 000015
6577 026172 001416
6578 026174 023727 026262 026266
6579 026202 001003
6580 026204 104401 001160
6581 026210 000420
6582 026212 111677 000044
6583 026216 005237 026260
6584 026222 005237 026262
6585 026226 000411
6586 026230 004737 021722
6587 026234 005037 026260
6588 026240 012737 026264 026262
6589 026246 005037 026264
6590 026252 005726
6591 026254 000002
6592 026256 000
6593 026257 000
6594 026260 000000
6595 026262 000000
6596 026264 000002
6597 026266
6598
6599
6600
6601
6602

```

.SBTTL TTY INPUT ROUTINE
;*TK INITIALIZE ROUTINE
; *THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; *SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
;
;*CALL
; *
; * JSR PC,$TKINT
; * RETURN
;
$TKINT: CLR $TKCNT ;CLEAR COUNT OF ITEMS IN QUEUE
MOV $TKQSR, $TKQIN ;MOVE THE STARTING ADDRESS
MOV $TKSRV, @TKVEC ;INITIALIZE THE KEYBOARD VECTOR
MOV #200, @TKVEC+2 ;"BR" LEVEL 4
TST @TKB ;CLEAR DONE FLAG
MOV @BIT06, @STKS ;ENABLE INTERRUPT
RTS PC ;RETURN TO CALLER

;*TK SERVICE ROUTINE
; *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;
$TKSRV: MOVB @TKB, -(SP) ;PICKUP THE CHARACTER
BIC #C177, (SP) ;STRIP THE JUNK
TST $TKCNT ;SEE IF FIRST CHARACTER
BNE 1$ ;BR IF NOT
JSR PC, $TIME ;TYPE THE TIME
TYPE $CRLF ;CR-LF
1$: MOVB (SP), 5$ ;ECHO THE CHARACTER
TYPE 5$
CMP #15, (SP) ;CARRIAGE RETURN ?
BEQ 3$ ;BR IF CR
CMP $TKQIN, $TKQEND ;AT THE END ?
BNE 2$ ;BR IF NOT AT END
TYPE $BELL ;RING THE TTY BELL
BR 4$ ;EXIT
2$: MOVB (SP), @TKQIN ;PUT IN QUEUE
INC $TKCNT ;COUNT THE CHARACTER
INC $TKQIN ;UPDATE THE POINTER
BR 4$ ;EXIT
3$: JSR PC, KSR ;PROCESS THE ENTRY
CLR $TKCNT ;CLEAR THE CHARACTER COUNTER
MOV $TKQSR, $TKQIN ;RESET THE POINTER
CLR $TKQSR ;CLEAR THE BUFFER
4$: TST (SP)+ ;RESET THE STACK POINTER
RTI ;RETURN
5$: .BYTE 0 ;STORAGE FOR ASCII CHARACTER TO TYPE
.BYTE 0 ;TERMINATOR
$TKCNT: .WORD 0 ;NUMBER OF ITEMS IN THE QUEUE
$TKQIN: .WORD 0 ;INPUT POINTER
$TKQSR: .BLKB 2 ;TTY KEYBOARD QUEUE
$TKQEND = .
.EVEN

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

```

```

6603      : *      RDCHR      ; INPUT A SINGLE CHARACTER FROM THE TTY
6604      : *      RETURN HERE ; CHARACTER IS ON THE STACK
6605      :
6606      :
6607      026266 011646      $RDCHR: MOV      (SP), -(SP)      ; PUSH DOWN THE PC
6608      026270 016666 000004 000002      MOV      4(SP), 2(SP)      ; SAVE THE PS
6609      026276 105777 152642      1$:      TSTB     @STKS      ; WAIT FOR
6610      026302 100375      BPL      1$      ; A CHARACTER
6611      026304 117766 152636 000004      MOVB     @STKB, 4(SP)      ; READ THE TTY
6612      026312 042766 177600 000004      BIC     #'C<177>, 4(SP) ; GET RID OF JUNK IF ANY
6613      026320 000002      RTI      ; GO BACK TO USER
6614
6615      : *****
6616      : *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
6617      : *CALL:
6618      : *      RDLIN      ; INPUT A STRING FROM THE TTY
6619      : *      RETURN HERE ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
6620      : *      ; TERMINATOR WILL BE A BYTE OF ALL 0'S
6621
6622      026322 010346      $RDLIN: MOV      R3, -(SP)      ; SAVE R3
6623      026324 005046      CLR      -(SP)      ; CLEAR THE RUBOUT KEY
6624      026326 012703 026570      1$:      MOV      #STTYIN, R3      ; GET ADDRESS
6625      026332 022703 026602      2$:      CMP      #STTYIN+10., R3      ; BUFFER FULL?
6626      026336 101450      BLOS     4$      ; BR IF YES
6627      026340 104406      RDCHR    ; GO READ ONE CHARACTER FROM THE TTY
6628      026342 112613      MOVB     (SP)+, (R3)      ; GET CHARACTER
6629      026344 122713 000177      CMPB     #'177, (R3)      ; IS IT A RUBOUT
6630      026350 001022      BNE      5$      ; BR IF NO
6631      026352 005716      TST      (SP)      ; IS THIS THE FIRST RUBOUT?
6632      026354 001007      BNE      6$      ; BR IF NO
6633      026356 112737 000134 026554      MOVB     #' \, 9$      ; TYPE A BACK SLASH
6634      026364 104401 026554      TYPE     9$
6635      026370 012716 177777      MOV      #-1, (SP)      ; SET THE RUBOUT KEY
6636      026374 005303      6$:      DEC      R3      ; BACKUP BY ONE
6637      026376 020327 026570      CMP      R3, #STTYIN      ; STACK EMPTY?
6638      026402 103426      BLO      4$      ; BR IF YES
6639      026404 111337 026554      MOVB     (R3), 9$      ; SETUP TO TYPEOUT THE DELETED CHAR.
6640      026410 104401 026554      TYPE     9$      ; GO TYPE
6641      026414 000746      BR       2$      ; GO READ ANOTHER CHAR.
6642      026416 005716      5$:      TST      (SP)      ; RUBOUT KEY SET?
6643      026420 001406      BEQ      7$      ; BR IF NO
6644      026422 112737 000134 026554      MOVB     #' \, 9$      ; TYPE A BACK SLASH
6645      026430 104401 026554      TYPE     9$
6646      026434 005016      CLR      (SP)      ; CLEAR THE RUBOUT KEY
6647      026436 122713 000003      7$:      CMPB     #3, (R3) ; IS CHARACTER A CTRL C ?
6648      026442 001425      BEQ      8$      ; BR IF IT IS
6649      026444 122713 000025      CMPB     #25, (R3)      ; IS CHARACTER A CTRL U?
6650      026450 001006      BNE      3$      ; BR IF NO
6651      026452 104401 026556      TYPE     $CNTLU      ; TYPE A CONTROL "U"
6652      026456 000723      BR       1$      ; GO START OVER
6653      026460 104401 001164      4$:      TYPE     $QUES      ; TYPE A '?'
6654      026464 000720      BR       1$      ; CLEAR THE BUFFER AND LOOP
6655      026466 111337 026554      3$:      MOVB     (R3), 9$      ; ECHO THE CHARACTER
6656      026472 104401 026554      TYPE     9$
6657      026476 122723 000015      CMPB     #15, (R3)+      ; CHECK FOR RETURN
6658      026502 001313      BNE      2$      ; LOOP IF NOT RETURN
  
```

```

6659 026504 105063 177777          CLRB   -1(R3)          ;CLEAR RETURN (THE 15)
6660 026510 104401 001166          TYPE   $LF            ;TYPE A LINE FEED
6661 026514 000405                    BR     10$            ;
6662 026516 104401 026563          TYPE   $CNTLC         ;TYPE A CONTROL C
6663 026522 112737 000003 026570 8$:   MOVB   #3,$TTYIN     ;FORCE 1ST CHARACTER IN BUF TO CTRL C
6664 026530 005726                    TST    (SP)+          ;CLEAN RUBOUT KEY FROM THE STACK
6665 026532 012603                    MOV    (SP)+,R3      ;RESTORE R3
6666 026534 011646                    MOV    (SP),-(SP)    ;ADJUST THE STACK AND PUT ADDRESS OF THE
6667 026536 016666 000004 000002    MOV    4(SP),2(SP)   ;FIRST ASCII CHARACTER ON IT
6668 026544 012766 026570 000004    MOV    #$TTYIN,4(SP)
6669 026552 000002                    RTI                    ;RETURN
6670 026554 000          9$:   .BYTE  0              ;STORAGE FOR ASCII CHAR. TO TYPE
6671 026555 000          .BYTE  0              ;TERMINATOR
6672 026556 052536 005015 000    $CNTLU: .ASCIZ  /↑U/<15><12> ;CONTROL "U"
6673 026563 136 006503 000012    $CNTLC: .ASCIZ  /↑C/<15><12> ;CONTROL C
6674 026570 000012    $TTYIN: .BLKB  10.    ;RESERVE 10. BYTES FOR TTY INPUT
6675
6676
6677
6678
6679
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696
6697
6698
6699
6700
6701
6702
6703
6704
6705
6706
6707
6708
6709
6710
6711
6712
6713
6714
026602 004737 025256          JSR    PC,ERENTR     ;
026602 105237 001103          INCB   $ERFLG        ;SET THE ERROR FLAG
026612 001775                    BEQ    7$            ;DON'T LET THE FLAG GO TO ZERO
026614 013777 001102 152320    MOV    $TSTNM,$DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
026622 032777 002000 152310    BIT    #BIT10,$SWR   ;BELL ON ERROR?
026630 001402                    BEQ    1$            ;NO - SKIP
026632 104401 001160          TYPE   $BELL         ;RING BELL
026636 005237 001112          INC    $ERTTL        ;COUNT THE NUMBER OF ERRORS
026642 011637 001116          MOV    (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
026646 162737 000002 001116    SUB    #2,$ERRPC     ;
026654 117737 152236 001114    MOVB  $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
026662 032777 020000 152250    BIT    #BIT13,$SWR   ;SKIP TYPEOUT IF SET
026670 001004                    BNE   20$           ;SKIP TYPEOUTS
026672 004737 026714          JSR    PC,$ERRTYP    ;GO TO USER ERROR ROUTINE
026676 104401 001165          TYPE   $CRLF         ;
026702 005777 152232          TST    $SWR          ;HALT ON ERROR
026706 100001                    BPL   3$            ;SKIP IF CONTINUE
026710 000000                    HALT                    ;HALT ON ERROR!
026712

```

;;*****

.SBTTL MACRO ROUTINES

.SBTTL ERROR HANDLER ROUTINE

;;*****

```

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

```

\$ERROR:

```

7$:   JSR    PC,ERENTR     ;
      INCB   $ERFLG        ;SET THE ERROR FLAG
      BEQ    7$            ;DON'T LET THE FLAG GO TO ZERO
      MOV    $TSTNM,$DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
      BIT    #BIT10,$SWR   ;BELL ON ERROR?
      BEQ    1$            ;NO - SKIP
      TYPE   $BELL         ;RING BELL
1$:   INC    $ERTTL        ;COUNT THE NUMBER OF ERRORS
      MOV    (SP),$ERRPC   ;GET ADDRESS OF ERROR INSTRUCTION
      SUB    #2,$ERRPC     ;
      MOVB  $ERRPC,$ITEMB ;STRIP AND SAVE THE ERROR ITEM CODE
      BIT    #BIT13,$SWR   ;SKIP TYPEOUT IF SET
      BNE   20$           ;SKIP TYPEOUTS
      JSR    PC,$ERRTYP    ;GO TO USER ERROR ROUTINE
      TYPE   $CRLF         ;
20$:  TST    $SWR          ;HALT ON ERROR
2$:   BPL   3$            ;SKIP IF CONTINUE
3$:   HALT                    ;HALT ON ERROR!

```



```

6715 026712 000002
6716
6717
6718
6719
6720
6721
6722
6723 026714
6724 026714 104401 001165
6725 026720 010046
6726 026722 005000
6727 026724 153700 001114
6728 026730 001004
6729
6730 026732 013746 001116
6731
6732 026736 104402
6733 026740 000445
6734 026742 005300
6735 026744 006300
6736 026746 006300
6737 026750 006300
6738 026752 062700 001346
6739 026756 012037 026766
6740 026762 001404
6741 026764 104401
6742 026766 000000
6743 026770 104401 001165
6744 026774 012037 027004
6745 027000 001404
6746 027002 104401
6747 027004 000000
6748 027006 104401 001165
6749 027012 010146
6750 027014 012001
6751 027016 001415
6752 027020 012000
6753 027022 105720
6754 027024 001003
6755 027026 013146
6756 027030 104402
6757 027032 000402
6758 027034
6759 027034 013146
6760 027036 104405
6761 027040 005711
6762 027042 001403
6763 027044 104401 027064
6764 027050 000764
6765
6766 027052 012601
6767 027054 012600
6768 027056 104401 001165
6769 027062 000207
6770 027064 020040 000

```

```

RTI ;:RETURN
.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.

$ERRTYP:
TYPE $SCLF ;:"CARRIAGE RETURN" & "LINE FEED"
MOV RO,-(SP) ;:SAVE RO
CLR RO ;:PICKUP THE ITEM INDEX
BISB @#$ITEMB,RO
BNE 1$ ;:IF ITEM NUMBER IS ZERO, JUST
;:TYPE THE PC OF THE ERROR
MOV $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT
;:ERROR ADDRESS
;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1$: BR 10$ ;:GET OUT
DEC RO ;:ADJUST THE INDEX SO THAT IT WILL
ASL RO ;:WORK FOR THE ERROR TABLE
ASL RO
ASL RO
ADD #$ERRTB,RO ;:FORM TABLE POINTER
MOV (RO)+,2$ ;:PICKUP "ERROR MESSAGE" POINTER
BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER
TYPE ;:TYPE THE "ERROR MESSAGE"
WORD 0 ;:"ERROR MESSAGE" POINTER GOES HERE
2$: ;:"CARRIAGE RETURN" & "LINE FEED"
TYPE $SCLF ;:PICKUP "DATA HEADER" POINTER
MOV (RO)+,4$ ;:SKIP TYPEOUT IF 0
BEQ 5$ ;:TYPE THE "DATA HEADER"
TYPE ;:"DATA HEADER" POINTER GOES HERE
WORD 0 ;:"CARRIAGE RETURN" & "LINE FEED"
3$: ;:"CARRIAGE RETURN" & "LINE FEED"
TYPE $SCLF ;:SAVE R1
MOV R1,-(SP) ;:PICKUP "DATA TABLE" POINTER
MOV (RO)+,R1 ;:BR IF NO DATA TO BE TYPED
BEQ 9$ ;:PICKUP "DATA FORMAT" POINTER
MOV (RO)+,RO ;:"OCTAL" OR "DECIMAL"
TSTB (RO)+ ;:BR IF DECIMAL
BNE 7$ ;:SAVE @(R1)+ FOR TYPEOUT
MOV @(R1)+,-(SP) ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
TYPDS ;:
BR 8$ ;:SAVE @(R1)+ FOR TYPEOUT
7$: ;:GO TYPE--DECIMAL ASCII WITH SIGN
MOV @(R1)+,-(SP) ;:IS THERE ANOTHER NUMBER?
TST (R1) ;:BR IF NO
BEQ 9$ ;:TYPE TWO(2) SPACES
TYPE 11$ ;:LOOP
BR 6$ ;:
6$: ;:
TSTB (RO)+
BNE 7$
MOV @(R1)+,-(SP)
TYPDS
BR 8$
8$: ;:
TST (R1)
BEQ 9$
TYPE 11$
BR 6$
9$: ;:RESTORE R1
MOV (SP)+,R1
10$: ;:RESTORE RO
MOV (SP)+,RO
TYPE $SCLF ;:"CARRIAGE RETURN" & "LINE FEED"
RTS PC ;:RETURN
11$: .ASCIZ / / ;:TWO(2) SPACES

```

6771 027070

.SBTTL .EVEN
BINARY TO OCTAL (ASCII) AND TYPE

6772
6773
6774
6775
6776
6777
6778
6779
6780
6781
6782
6783
6784
6785
6786
6787
6788
6789
6790
6791
6792
6793
6794
6795
6796
6797
6798
6799
6800
6801
6802
6803
6804
6805
6806
6807
6808
6809
6810
6811
6812
6813
6814
6815
6816
6817
6818
6819
6820
6821
6822
6823
6824
6825
6826

027070 017646 000000
027074 116637 000001 027313
027102 112637 027315
027106 062716 000002
027112 000406
027114 112737 000001 027313
027122 112737 000006 027315
027130 112737 000005 027312
027136 010346
027140 010446
027142 010546
027144 113704 027315
027150 005404
027152 062704 000006
027156 110437 027314
027162 113704 027313
027166 016605 000012
027172 005003
027174 006105
027176 000404
027200 006105
027202 006105
027204 006105
027206 010503
027210 006103
027212 105337 027314
027216 100016
027220 042703 177770
027224 001002
027226 005704

```

*****
*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
*$TYPOS: MOV      2(SP),-(SP)    ;;PICKUP THE MODE
        MOV      1(SP),SOFILL    ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1  ;;NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)         ;;ADJUST RETURN ADDRESS
        BR      $TYPON
*$TYPOC: MOV      #1,SOFILL      ;;SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1    ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV      #5,SOCNT      ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)       ;;SAVE R3
        MOV      R4,-(SP)       ;;SAVE R4
        MOV      R5,-(SP)       ;;SAVE R5
        MOV      SOMODE+1,R4    ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,SOMODE      ;;SAVE IT FOR USE
        MOV      SOFILL,R4     ;;GET THE ZERO FILL SWITCH
        MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
        CLR      R3            ;;CLEAR THE OUTPUT WORD
        ROL     R5             ;;ROTATE MSB INTO "C"
        BR      3$            ;;GO DO MSB
        ROL     R5             ;;FORM THIS DIGIT
        ROL     R5
        ROL     R5
        MOV      R5,R3
        ROL     R3            ;;GET LSB OF THIS DIGIT
        DECB    SOMODE        ;;TYPE THIS DIGIT?
        BPL     7$            ;;BR IF NO
        BIC     #177770,R3    ;;GET RID OF JUNK
        BNE     4$            ;;TEST FOR 0
        TST     R4            ;;SUPPRESS THIS 0?

```

```

6827 027230 001403          BEQ      5$          ;;BR IF YES
6828 027232 005204          4$: INC      R4          ;;DON'T SUPPRESS ANYMORE 0'S
6829 027234 052703 000060  BIS      #'0,R3      ;;MAKE THIS DIGIT ASCII
6830 027240 052703 000040  5$: BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
6831 027244 110337 027310  MOVB     R3,8$        ;;SAVE FOR TYPING
6832 027250 104401 027310  TYPE     8$          ;;GO TYPE THIS DIGIT
6833 027254 105337 027312  7$: DECB   $OCNT      ;;COUNT BY 1
6834 027260 003347          BGT      2$          ;;BR IF MORE TO DO
6835 027262 002402          BLT      6$          ;;BR IF DONE
6836 027264 005204          INC      R4          ;;INSURE LAST DIGIT ISN'T A BLANK
6837 027266 000744          BR       2$          ;;GO DO THE LAST DIGIT
6838 027270 012605          6$: MOV     (SP)+,R5    ;;RESTORE R5
6839 027272 012604          MOV     (SP)+,R4    ;;RESTORE R4
6840 027274 012603          MOV     (SP)+,R3    ;;RESTORE R3
6841 027276 016666 000002 000004  MOV     2(SP),4(SP) ;;SET THE STACK FOR RETURNING
6842 027304 012616          MOV     (SP)+,(SP)
6843 027306 000002          RTI          ;;RETURN
6844 027310          000          8$: .BYTE   0          ;;STORAGE FOR ASCII DIGIT
6845 027311          000          .BYTE   0          ;;TERMINATOR FOR TYPE ROUTINE
6846 027312          000          $OCNT: .BYTE  0          ;;OCTAL DIGIT COUNTER
6847 027313          000          $OFILL: .BYTE 0          ;;ZERO FILL SWITCH
6848 027314 000000          $OMODE: .WORD  0        ;;NUMBER OF DIGITS TO TYPE
6849          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
6850
6851          ;;*****
6852          ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
6853          ;;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
6854          ;;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
6855          ;;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
6856          ;;*REPLACED WITH SPACES.
6857          ;;*CALL:
6858          ;;*      MOV     NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
6859          ;;*      TYPDS          ;;GO TO THE ROUTINE
6860
6861          $TYPDS:
6862          MOV     R0,-(SP)          ;;PUSH R0 ON STACK
6863          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
6864          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
6865          MOV     R3,-(SP)          ;;PUSH R3 ON STACK
6866          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
6867          MOV     #20200,-(SP)      ;;SET BLANK SWITCH AND SIGN
6868          MOV     20(SP),R5         ;;GET THE INPUT NUMBER
6869          BPL     1$              ;;BR IF INPUT IS POS.
6870          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
6871          MOVB   #'-,1(SP)         ;;MAKE THE ASCII NUMBER NEG.
6872          CLR     R0              ;;ZERO THE CONSTANTS INDEX
6873          MOV     #$DBLK,R3        ;;SETUP THE OUTPUT POINTER
6874          MOVB   #' ,(R3)+         ;;SET THE FIRST CHARACTER TO A BLANK
6875          CLR     R2              ;;CLEAR THE BCD NUMBER
6876          MOV     $DTBL(R0),R1     ;;GET THE CONSTANT
6877          SUB     R1,R5            ;;FORM THIS BCD DIGIT
6878          BLT     4$              ;;BR IF DONE
6879          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
6880          BR     3$
6881          4$: ADD     R1,R5          ;;ADD BACK THE CONSTANT
6882          TST     R2              ;;CHECK IF BCD DIGIT=0

```

```

6883 027406 001002          BNE      5$          ;; FALL THROUGH IF 0
6884 027410 105716          TSTB     (SP)        ;; STILL DOING LEADING 0'S?
6885 027412 100407          BMI      7$          ;; BR IF YES
6886 027414 106316          5$: ASLB     (SP)        ;; MSD?
6887 027416 103003          BCC      6$          ;; BR IF NO
6888 027420 116663 000001 177777 MOVVB    1(SP),-1(R3) ;; YES--SET THE SIGN
6889 027426 052702 000060 6$: BIS     #'0,R2    ;; MAKE THE BCD DIGIT ASCII
6890 027432 052702 000040 7$: BIS     #' ,R2    ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
6891 027436 110223          MOVVB    R2,(R3)+    ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
6892 027440 005720          TST      (R0)+       ;; JUST INCREMENTING
6893 027442 020027 000010  CMP      R0,#10     ;; CHECK THE TABLE INDEX
6894 027446 002746          BLT      2$          ;; GO DO THE NEXT DIGIT
6895 027450 003002          BGT      8$          ;; GO TO EXIT
6896 027452 010502          MOV      R5,R2      ;; GET THE LSD
6897 027454 000764          BR       6$          ;; GO CHANGE TO ASCII
6898 027456 105726          8$: TSTB     (SP)+    ;; WAS THE LSD THE FIRST NON-ZERO?
6899 027460 100003          BPL      9$          ;; BR IF NO
6900 027462 116663 177777 177776 MOVVB    -1(SP),-2(R3) ;; YES--SET THE SIGN FOR TYPING
6901 027470 105013          9$: CLRB     (R3)     ;; SET THE TERMINATOR
6902 027472 012605          MOV      (SP)+,R5   ;; POP STACK INTO R5
6903 027474 012603          MOV      (SP)+,R3   ;; POP STACK INTO R3
6904 027476 012602          MOV      (SP)+,R2   ;; POP STACK INTO R2
6905 027500 012601          MOV      (SP)+,R1   ;; POP STACK INTO R1
6906 027502 012600          MOV      (SP)+,R0   ;; POP STACK INTO R0
6907 027504 104401 027532 TYPE     $DBLK      ;; NOW TYPE THE NUMBER
6908 027510 016666 000002 000004 MOV      2(SP),4(SP) ;; ADJUST THE STACK
6909 027516 012616          MOV      (SP)+,(SP)
6910 027520 000002          RTI
6911 027522 023420          $DTBL: 10000.      ;; RETURN TO USER
6912 027524 001750          1000.
6913 027526 000144          100.
6914 027530 000012          10.
6915 027532 000004          $DBLK: .BLKW 4
6916                                     .SBTTL SAVE AND RESTORE R0-R5 ROUTINES
6917
6918                                     ;; *****
6919                                     ;; *SAVE R0-R5
6920                                     ;; *CALL:
6921                                     ;; * SAVREG
6922                                     ;; *UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
6923                                     ;; *
6924                                     ;; *TOP---(+16)
6925                                     ;; * +2---(+18)
6926                                     ;; * +4---R5
6927                                     ;; * +6---R4
6928                                     ;; * +8---R3
6929                                     ;; *+10---R2
6930                                     ;; *+12---R1
6931                                     ;; *+14---R0
6932
6933 027542          $SAVREG:
6934 027542 010046          MOV      R0,-(SP)   ;; PUSH R0 ON STACK
6935 027544 010146          MOV      R1,-(SP)   ;; PUSH R1 ON STACK
6936 027546 010246          MOV      R2,-(SP)   ;; PUSH R2 ON STACK
6937 027550 010346          MOV      R3,-(SP)   ;; PUSH R3 ON STACK
6938 027552 010446          MOV      R4,-(SP)   ;; PUSH R4 ON STACK

```

```

6939 027554 010546          MOV      R5, -(SP)          ;; PUSH R5 ON STACK
6940 027556 016646 000022  MOV      22(SP), -(SP)      ;; SAVE PS OF MAIN FLOW
6941 027562 016646 000022  MOV      22(SP), -(SP)      ;; SAVE PC OF MAIN FLOW
6942 027566 016646 000022  MOV      22(SP), -(SP)      ;; SAVE PS OF CALL
6943 027572 016646 000022  MOV      22(SP), -(SP)      ;; SAVE PC OF CALL
6944 027576 000002          RTI
6945
6946          ;*RESTORE RO-R5
6947          ;*CALL:
6948          ;*
6949          ;* RESREG
6949 027600          $RESREG:
6950 027600 012666 000022  MOV      (SP)+, 22(SP)      ;; RESTORE PC OF CALL
6951 027604 012666 000022  MOV      (SP)+, 22(SP)      ;; RESTORE PS OF CALL
6952 027610 012666 000022  MOV      (SP)+, 22(SP)      ;; RESTORE PC OF MAIN FLOW
6953 027614 012666 000022  MOV      (SP)+, 22(SP)      ;; RESTORE PS OF MAIN FLOW
6954 027620 012605          MOV      (SP)+, R5          ;; POP STACK INTO R5
6955 027622 012604          MOV      (SP)+, R4          ;; POP STACK INTO R4
6956 027624 012603          MOV      (SP)+, R3          ;; POP STACK INTO R3
6957 027626 012602          MOV      (SP)+, R2          ;; POP STACK INTO R2
6958 027630 012601          MOV      (SP)+, R1          ;; POP STACK INTO R1
6959 027632 012600          MOV      (SP)+, R0          ;; POP STACK INTO R0
6960 027634 000002          RTI
6961          .SBTTL RANDOM NUMBER GENERATOR ROUTINE
6962
6963          ;*****
6964          ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
6965          ;*WITH A RANGE OF 0 TO 2(+33)-1.
6966          ;*CALL:
6967          ;*
6968          ;* JSR      PC, $RAND          ;; CALL THE ROUTINE
6969          ;* RETURN          ;; RETURN HERE THE RANDOM
6970          ;*          ;; NUMBER WILL BE IN
6971          ;*          ;; $HINUM, $LONUM
6972
6972 027636          $RAND:
6973 027636 010046          MOV      R0, -(SP)          ;; PUSH R0 ON STACK
6974 027640 010146          MOV      R1, -(SP)          ;; PUSH R1 ON STACK
6975 027642 010246          MOV      R2, -(SP)          ;; PUSH R2 ON STACK
6976 027644 013700 027736  MOV      $LONUM, R0          ;; SET R0 WITH LOW
6977 027650 013701 027734  MOV      $HINUM, R1          ;; SET R1 WITH HIGH
6978 027654 012702 177771  MOV      #-7, R2            ;; SET SHIFT COUNT
6979 027660 096300          1$: ASL      R0                ;; SHIFT R0 LEFT AND
6980 027662 006101          ROL      R1                ;; ROTATE CARRY INTO R1 AND
6981 027664 005202          INC      R2                ;; CHECK FOR DONE
6982 027666 001374          BNE     1$                 ;; CONTINUE SHIFT LOOP
6983 027670 063700 027736  ADD      $LONUM, R0          ;; ADD NUMBER TO MAKE X 129
6984 027674 005501          ADC      R1                ;; PROPOGATE CARRY
6985 027676 063701 027734  ADD      $HINUM, R1          ;; ADD NUMBER TO MAKE X 129
6986 027702 062700 001057  ADD      #1057, R0           ;; ADD LOW CONSTANT
6987 027706 005501          ADC      R1                ;; PROPOGATE CARRY
6988 027710 062701 047401  ADD      #47401, R1          ;; ADD HIGH CONSTANT
6989 027714 010037 027736  MOV      R0, $LONUM          ;; SAVE R0
6990 027720 010137 027734  MOV      R1, $HINUM          ;; SAVE R1
6991 027724 012602          MOV      (SP)+, R2          ;; POP STACK INTO R2
6992 027726 012601          MOV      (SP)+, R1          ;; POP STACK INTO R1
6993 027730 012600          MOV      (SP)+, R0          ;; POP STACK INTO R0
6994 027732 000207          RTS      PC                ;; RETURN

```

6995 027734 176543
6996 027736 123456
6997
6998
6999
7000
7001
7002
7003
7004
7005 027740 010046
7006 027742 010146
7007 027744 013746 000004
7008 027750 013746 000006
7009 027754 010600
7010
7011 027756 104400
7012 027760 012637 000006
7013 027764 012737 030004 000004
7014 027772 012701 020000
7015 027776 005711
7016 030000 005721
7017 030002 000775
7018 030004 162701 000002
7019 030010 010006
7020 030012 012637 000006
7021 030016 012637 000004
7022 030022 010137 030034
7023 030026 012601
7024 030030 012600
7025 030032 000207
7026 030034 000000
7027
7028
7029
7030
7031
7032
7033
7034
7035
7036
7037
7038
7039
7040 030036 104410
7041 030040 016602 000002
7042 030044 012700 030216
7043 030050 010066 000002
7044 030054 012201
7045 030056 012202
7046 030060 012737 000012 030134
7047 030066 012704 030146
7048 030072 012705 030150
7049 030076 005003
7050 030100 161401

```

$HINUM: .WORD 176543
$LONUM: .WORD 123456
.SBTTL ROUTINE TO SIZE MEMORY

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

$SIZE: MOV RO,-(SP) ;;SAVE RO ON THE STACK
MOV R1,-(SP) ;;SAVE R1 ON THE STACK
MOV @#ERRVEC,-(SP) ;;SAVE PRESENT ERROR VECTOR PS & PC
MOV @#ERRVEC+2,-(SP)
MOV SP,RO ;;SAVE THE STACK POINTER
;;SET THE ERRVEC PS TO THE PRESENT PS
TRAP ;;PUSH OLD PSW AND PC ON STACK
MOV (SP)+,@#ERRVEC+2 ;;SAVE THE PSW IN @#ERRVEC+2
MOV #2,@#ERRVEC ;;SET FOR TIMEOUT
MOV #20000,R1 ;;FIRST ADDRESS
1$: TST (R1) ;;TEST THIS ADDRESS
TST (R1)+ ;;STEP TO NEXT ADDRESS
BR 1$ ;;TRY ANOTHER
2$: SUB #2,R1 ;;DROP BACK
MOV RO,SP ;;RESTORE THE STACK
MOV (SP)+,@#ERRVEC+2 ;;RESTORE ERROR VECTOR
MOV (SP)+,@#ERRVEC
MOV R1,$LSTAD ;;LAST ADDRESS
MOV (SP)+,R1 ;;RESTORE R1
MOV (SP)+,RO ;;RESTORE RO
RTS PC
$LSTAD: .WORD 0 ;;CONTAINS THE LAST ADDRESS
.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
* RETURN ;;THE FIRST ADDRESS OF ASCIZ
;;IS ON THE STACK

$DB2D: SAVREG ;;SAVE REGISTERS
MOV 2(SP),R2 ;;PICKUP THE DATA POINTER
MOV #$DECVL,RO ;;GET ADDRESS OF "$DECVL" STRING
MOV RO,2(SP) ;;PUT ADDRESS OF ASCIZ STRING ON STACK
MOV (R2)+,R1 ;;PICKUP THE BINARY NUMBER
MOV (R2)+,R2
MOV #10,4$ ;;SET UP TO DO 10 CONVERSIONS
MOV #STNPWR,R4 ;;ADDRESS OF TEN POWER
MOV #STNPWR+2,R5
1$: CLR R3 ;;CLEAR PARTIAL
2$: SUB (R4),R1 ;;SUBTRACT TEN POWER

```

```

7051 030102 005602          SBC      R2
7052 030104 161502          SUB      (R5),R2
7053 030106 002402          BLT     3$      ;;BR IF TEN POWER TO LARGE
7054 030110 005203          INC     R3      ;;ADD 1 TO PARTIAL
7055 030112 000772          BR      2$      ;;LOOP
7056 030114 062401          3$:    ADD      (R4)+,R1  ;;RESTORE SUBTRACTED VALUE
7057 030116 005502          ADC     R2
7058 030120 062402          ADD      (R4)+,R2
7059 030122 022525          CMP     (R5)+,(R5)+  ;;MOVE TO NEXT TEN POWER
7060 030124 052703 000060  BIS     #'0,R3      ;;CHANGE PARTIAL TO ASCII
7061 030130 110320          MOVB   R3,(R0)+    ;;SAVE IT
7062 030132 005327          DEC     (PC)+      ;;DONE?
7063 030134 000000          4$:    .WORD   0
7064 030136 001357          BNE    1$          ;;BR IF NO
7065 030140 105020          CLAB   (R0)+      ;;TERMINATOR
7066 030142 104411          RESREG                ;;RESTORE REGISTERS
7067 030144 000207          RTS     PC        ;;RETURN
7068 030146 145000          $TNPWR: 145000     ;;1.0E09
7069 030150 035632          35632
7070 030152 160400          160400     ;;1.0E08
7071 030154 002765          2765
7072 030156 113200          113200     ;;1.0E07
7073 030160 000230          230
7074 030162 041100          041100     ;;1.0E06
7075 030164 000017          17
7076 030166 103240          103240     ;;1.0E05
7077 030170 000001          1
7078 030172 023420          23420     ;;1.0E04
7079 030174 000000          0
7080 030176 001750          1750     ;;1.0E03
7081 030200 000000          0
7082 030202 000144          144     ;;1.0E02
7083 030204 000000          0
7084 030206 000012          12     ;;1.0E01
7085 030210 000000          0
7086 030212 000001          1     ;;1.0E00
7087 030214 000000          0
7088 030216 000014          $DECVL: .BLKB 12.  ;;RESERVE STORAGE FOR ASCII STRING
7089          .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCII ROUTINE
7090
7091          ;;*****
7092          ;;THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
7093          ;;UNSIGNED DECIMAL ASCII NUMBER.
7094          ;;CALL
7095          ;;*   MOV     NUMBER,-(SP)  ;;PUT BINARY NUMBER ON THE STACK
7096          ;;*   JSR     PC,@#$SB2D  ;;CALL
7097          ;;*   RETURN                ;;ADDRESS OF THE 1ST ASCII CHAR.IS ON THE STACK
7098
7099
7100 030232 016637 000002 030262 $SB2D: MOV     2(SP),1$  ;;SAVE BINARY NUMBER
7101 030240 012746 030262 MOV     #1$,-(SP)  ;;SET POINTER
7102 030244 004737 030036 JSR     PC,@#$DB2D  ;;CALL DOUBLE LENGTH CONVERT
7103 030250 062716 000005 ADD     #5,(SP)    ;;ONLY ALLOW FIVE CHARACTERS
7104 030254 012666 000002 MOV     (SP)+,2(SP)  ;;PICKUP POINTER
7105 030260 000207          RTS     PC        ;;RETURN
7106 030262 000000          1$:    .WORD   0,0

```

```

7107      .SBTTL  DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE
7108
7109      ;:*****
7110      ;:THIS ROUTINE WILL CONVERT A 32-BIT UNSIGNED BINARY NUMBER TO AN
7111      ;:UNSIGNED OCTAL ASCII NUMBER.
7112      ;:CALL
7113      ;:
7114      ;:   MOV     #PNTR, -(SP)      ;: POINTER TO LOW WORD OF BINARY NUMBER
7115      ;:   JSR     PC, @#$DB20     ;: CALL THE ROUTINE
7116      ;:   RETURN                    ;: THE ADDRESS OF THE FIRST ASCII CHAR. IS ON THE STACK
7117
7118      030266 104410      $DB20: SAVREG      ;: SAVE ALL REGISTERS
7119      030270 016601 000002  MOV     2(SP), R1      ;: PICKUP THE POINTER TO LOW WORD
7120      030274 012705 030435  MOV     #SOCTVL+13., R5 ;: POINTER TO DATA TABLE
7121      030300 012704 000014  MOV     #12., R4      ;: DO ELEVEN CHARACTERS
7122      030304 012703 177770  MOV     #1C7, R3     ;: MASK
7123      030310 012100  MOV     (R1)+, R0    ;: LOWER WORD
7124      030312 012101  MOV     (R1)+, R1    ;: HIGH WORD
7125      030314 005002  CLR     R2           ;: TERMINATOR
7126      030316 110245      1$: MOVB  R2, -(R5)    ;: PUT CHARACTER IN DATA TABLE
7127      030320 010002  MOV     R0, R2      ;: GET THIS DIGIT
7128      030322 005304  DEC     R4          ;: COUNT THIS CHARACTER
7129      030324 003007  BGT    3$,          ;: BR IF NOT THE LAST DIGIT
7130      030326 001405  BEQ    2$,          ;: BR IF IT IS THE LAST DIGIT
7131      030330 005205  INC     R5          ;: ALL DIGITS DONE-ADJUST POINTER FOR FIRST
7132      030332 010566 000002  MOV     R5, 2(SP)   ;: ASCII CHAR. & PUT IT ON THE STACK
7133      030336 104411  RESREG                    ;: RESTORE ALL REGISTERS
7134      030340 000207  RTS     PC          ;: RETURN TO USER
7135      030342 006203      2$: ASR  R3          ;: POSITION THE MASK FOR THE LAST DIGIT
7136      030344 006001      3$: ROR  R1          ;: POSITION THE BINARY NUMBER FOR
7137      030346 006000  ROR  R0          ;: THE NEXT OCTAL DIGIT
7138      030350 006001  ROR  R1
7139      030352 006000  ROR  R0
7140      030354 006001  ROR  R1
7141      030356 006000  ROR  R0
7142      030360 040302  BIC  R3, R2      ;: MASK OUT ALL JUNK
7143      030362 062702 000060  ADD  #'0, R2     ;: MAKE THIS CHAR. ASCII
7144      030366 000753  BR    1$,        ;: GO PUT IT IN THE DATA TABLE
7145      030370 000016  $SOCTVL: .BLKB 14. ;: RESERVE DATA TABLE
7146      .SBTTL  SINGLE LENGTH BINARY TO OCTAL ASCII ROUTINE
7147
7148      ;:*****
7149      ;:THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
7150      ;:UNSIGNED OCTAL ASCII NUMBER.
7151      ;:CALL
7152      ;:
7153      ;:   MOV     NUMBER, -(SP)    ;: PUT BINARY NUMBER ON THE STACK
7154      ;:   JSR     PC, @#$SB20     ;: CALL
7155      ;:   RETURN                    ;: ADDRESS OF 1ST ASCII CHAR. IS ON THE STACK
7156      030406 016637 000002 030436  $SB20: MOV  2(SP), 1$      ;: SAVE THE BINARY NUMBER
7157      030414 012746 030436  MOV  #1$, -(SP)    ;: SET POINTER
7158      030420 004737 030266  JSR  PC, @#$DB20  ;: CALL DOUBLE LENGTH CONVERT ROUTINE
7159      030424 062716 000005  ADD  #5, (SP)     ;: ONLY ALLOW SIX CHARACTERS
7160      030430 012666 000002  MOV  (SP)+, 2(SP) ;: PICKUP POINTER
7161      030434 000207  RTS  PC          ;: RETURN
7162      030436 000000  1$: .WORD 0,0
    
```


7163
7164
7165
7166
7167
7168
7169
7170
7171
7172
7173
7174
7175
7176
7177
7178
7179
7180
7181
7182
7183
7184
7185
7186
7187
7188
7189
7190
7191
7192
7193
7194
7195
7196
7197
7198
7199
7200
7201
7202
7203
7204
7205
7206
7207
7208
7209
7210
7211
7212
7213
7214
7215
7216
7217
7218

.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.
  
```

```

$TRAP:  MOV    RD,-(SP)           ;;SAVE RD
        MOV    2(SP),RD         ;;GET TRAP ADDRESS
        TST    -(RD)           ;;BACKUP BY 2
        MOVB   (RD),RD         ;;GET RIGHT BYTE OF TRAP
        CMP    $TERM,RD        ;;CHECK FOR OUT OF BOUNDS
        BGT    .+6             ;;BR IF OK
        HALT                               ;;OUT OF BOUNDS
        BR     -2              ;;HANGUP
        ASL    RD              ;;POSITION FOR INDEXING
        MOV    $TRPAD(RD),RD    ;;INDEX TO TABLE
        RTS    RD              ;;GO TO ROUTINE
  
```

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

```

$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 ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.
  
```

```

; ROUTINE
;-----
$TRPAD: .WORD  $TRAP2           TRAP+1(104401)  TTY TYPEOUT ROUTINE
        $TYPE  ;;CALL=TYPE     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
        $TYPOC ;;CALL=TYPOC   TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
        $TYPOS ;;CALL=TYPOS   TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
        $TYPON ;;CALL=TYPON   TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
        $TYPDS ;;CALL=TYPDS
        $RDCHR ;;CALL=RDCHR    TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
        $RDLIN ;;CALL=RDLIN    TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
        $SAVREG ;;CALL=SAVREG  TRAP+10(104410) SAVE RD-R5 ROUTINE
        $RESREG ;;CALL=RESREG  TRAP+11(104411) RESTORE RD-R5 ROUTINE
        $SDSPLY ;;CALL=DISPLY  TRAP+12(104412) PRINTER-TELETYPE MESSAGE ROUTING ROUTINE
$TERM=.-$TRPAD
  
```

;;*****

.SBTTL RH11/RP04 DRIVER - SINGLE/DUAL PORT VERSION
 ;(MODIFIED FROM REV. 0.3 FOR MD-11-DERPN-B)

;*COPYRIGHT (C) 1974
 ;*DIGITAL EQUIPMENT CORP.

```

7219
7220
7221
7222
7223
7224
7225
7226
7227
7228
7229
7230 030536 000000
7231 030540 000000
7232 030542 000000
7233 030544 000000
7234
7235
7236
7237
7238
7239
7240 030546 000
7241 030547 000
7242 030550 000
7243 030551 000
7244 030552 000
7245 030553 000
7246 030554 000
7247 030555 000
7248
7249
7250
7251
7252
7253
7254 030556 000
7255 030557 000
7256 030560 000
7257 030561 000
7258 030562 000
7259 030563 000
7260 030564 000
7261 030565 000
7262
7263
7264
7265
7266
7267 030566 000000
7268 030570 000000
7269 030572 000000
7270 030574 000000
7271 030576 000000
7272 030600 000000
7273 030602 000000
7274 030604 000000

```

```

;*MAYNARD, MA 01754
;*AUTHOR: JIM LACEY/CHUCK HESS

;*****

;*STORAGE FOR RHDS1, RHER1, RHER2, AND RHER3 ON AN ERROR "2" OR "5"
;*RPERRS = RHDS1
;*RPERRS+2 = RHER1
;*RPERRS+4 = RHER2
;*RPERRS+6 = RHER3

RPERRS: .WORD 0
        .WORD 0
        .WORD 0
        .WORD 0

;*TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
;*DRVACT=0 IMPLIES DRIVE IS IDLE
;*DRVACT>0 IMPLIES DRIVE IS ACTIVE WITH A COMMAND
;*DRVACT<0 IMPLIES DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION

DRVACT: .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 DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
;*DRVSTA=0 IMPLIES DRIVE IS OFFLINE
;*DRVSTA>0 IMPLIES DRIVE IS ONLINE
;*DRVSTA<0 IMPLIES DRIVE IS UNSAFE OR NONEXISTENT

DRVSTA: .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 DRIVE TYPES (DRV TYP=8 WORDS)
;*DRV TYP WILL CONTAIN THE DRIVE TYPE OF ALL ONLINE, OFFLINE, AND
;*UNSAFE DRIVES. IF A DRIVE IS NONEXISTENT DRV TYP WILL BE ZERO.

DRV TYP: .WORD 0           ;DRIVE 0
         .WORD 0           ;DRIVE 1
         .WORD 0           ;DRIVE 2
         .WORD 0           ;DRIVE 3
         .WORD 0           ;DRIVE 4
         .WORD 0           ;DRIVE 5
         .WORD 0           ;DRIVE 6
         .WORD 0           ;DRIVE 7

```

```

7275
7276      ;*TABLE OF DUAL PORT INITIALIZATION INDICATORS
7277      ;*DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
7278      ;*DPINT<0 IF INITIALIZATION IS IN PROGRESS
7279
7280      DPINT:  .BYTE  0           ;DRIVE 0
7281      .BYTE  0           ;DRIVE 1
7282      .BYTE  0           ;DRIVE 2
7283      .BYTE  0           ;DRIVE 3
7284      .BYTE  0           ;DRIVE 4
7285      .BYTE  0           ;DRIVE 5
7286      .BYTE  0           ;DRIVE 6
7287      .BYTE  0           ;DRIVE 7
7288
7289      ;*TABLE OF PENDING DUAL PORT REQUESTS
7290      ;*DPRQS=0 IMPLIES THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
7291      ;*DPRQS<0 IMPLIES THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
7292
7293      DPRQS:  .BYTE  0           ;DRIVE 0
7294      .BYTE  0           ;DRIVE 1
7295      .BYTE  0           ;DRIVE 2
7296      .BYTE  0           ;DRIVE 3
7297      .BYTE  0           ;DRIVE 4
7298      .BYTE  0           ;DRIVE 5
7299      .BYTE  0           ;DRIVE 6
7300      .BYTE  0           ;DRIVE 7
7301
7302      ;*TRANSFER WAIT FLAG (TRNSWT=1 WORD)
7303      ;*THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
7304      ;*"DPB" OF THE I/O OPERATION.
7305
7306      TRNSWT: .WORD  0
7307
7308      ;*SEARCH WAIT KEYS (SRCHWT=1 WORD)
7309      ;*THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
7310      ;*THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
7311      ;*REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
7312      ;*EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
7313
7314      SRCHWT: .WORD  0
7315
7316      ;*RP04 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
7317      ;*ACTDRV=0 IMPLIES DRIVER IS INACTIVE
7318      ;*ACTDRV>0 IMPLIES DRIVER IS ACTIVE
7319
7320      ACTDRV: .BYTE  0
7321
7322      ;*SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
7323      ;*ACTSTR=0 IMPLIES SOFTWARE TIMER ROUTINE IS INACTIVE
7324      ;*ACTSTR>0 IMPLIES SOFTWARE TIMER ROUTINE IS ACTIVE
7325
7326      ACTSTR: .BYTE  0
7327
7328      ;*UNLOAD FLAG (ULDFLG=8 BYTES)
7329      ;*ULDFLG=0 IMPLIES NO UNLOAD COMMAND
7330      ;*ULDFLG>0 IMPLIES UNLOAD COMMAND IN PROGRESS

```

```

7331                                     ;*ULDFLG<0 IMPLIES UNLOAD COMMAND IN WAIT QUEUE
7332
7333 030634      000      ULDFLG: .BYTE 0          ;DRIVE 0
7334 030635      000      .BYTE 0          ;DRIVE 1
7335 030636      000      .BYTE 0          ;DRIVE 2
7336 030637      000      .BYTE 0          ;DRIVE 3
7337 030640      000      .BYTE 0          ;DRIVE 4
7338 030641      000      .BYTE 0          ;DRIVE 5
7339 030642      000      .BYTE 0          ;DRIVE 6
7340 030643      000      .BYTE 0          ;DRIVE 7
7341
7342                                     ;*LOOK AHEAD COUNT (LACNT=8 BYTES)
7343                                     ;*LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
7344
7345 030644      000      LACNT: .BYTE 0          ;DRIVE 0
7346 030645      000      .BYTE 0          ;DRIVE 1
7347 030646      000      .BYTE 0          ;DRIVE 2
7348 030647      000      .BYTE 0          ;DRIVE 3
7349 030650      000      .BYTE 0          ;DRIVE 4
7350 030651      000      .BYTE 0          ;DRIVE 5
7351 030652      000      .BYTE 0          ;DRIVE 6
7352 030653      000      .BYTE 0          ;DRIVE 7
7353
7354                                     ;*SAVE REGISTERS FLAG (SAVEFG =1 WORD)
7355                                     ;*SAVEFG <0 IMPLIES SAVE THE RH11/RP04 REGISTERS WHEN THE
7356                                     ;*OPERATION IS COMPLETED AS PER (DPB+14).
7357                                     ;*SAVEFG=0 IMPLIES SAVE THE RH11/RP04 REGISTERS, AS PER
7358                                     ;*(DPB+14), AFTER AN ERROR.
7359
7360 030654      000000      SAVEFG: .WORD 0
7361
7362                                     ;*SEEK FLAG (SEEKFG=1 WORD)
7363                                     ;*SEEKFG=0 IMPLIES WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
7364                                     ;*FOR A DATA TRANSFER START A SEARCH COMMAND
7365                                     ;*SEEKFG<0 IMPLIES DATA TRANSFER WILL DO IMPLIED SEEKS,
7366                                     ;*DISREGARD THE WINDOW
7367
7368 030656      000000      SEEKFG: .WORD 0
7369
7370                                     ;*TIMEOUT TABLE (TIMER=8 WORDS)
7371                                     ;*THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
7372
7373 030660      177777      TIMER: .WORD -1      ;DRIVE 0
7374 030662      177777      .WORD -1      ;DRIVE 1
7375 030664      177777      .WORD -1      ;DRIVE 2
7376 030666      177777      .WORD -1      ;DRIVE 3
7377 030670      177777      .WORD -1      ;DRIVE 4
7378 030672      177777      .WORD -1      ;DRIVE 5
7379 030674      177777      .WORD -1      ;DRIVE 6
7380 030676      177777      .WORD -1      ;DRIVE 7
7381
7382                                     ;*DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
7383                                     ;*DTUW<0 IMPLIES NO DATA TRANSFER UNDERWAY
7384                                     ;*DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
7385
7386 030700      177777      DTUW: .WORD -1

```

```

7387
7388
7389
7390
7391
7392 030702 001
7393 030703 002
7394 030704 004
7395 030705 010
7396 030706 020
7397 030707 040
7398 030710 100
7399 030711 200
7400
7401
7402
7403
7404 030712 000003
7405
7406
7407
7408 030714 176700
7409 030716 000254 000240
7410
7411
7412 030722 000004
7413
7414 030724 001000
7415
7416 030726 000200
7417
7418 030730 000005
7419
7420
7421
7422 000000
7423 000002
7424 000004
7425 000006
7426 000010
7427 000012
7428 000014
7429 000016
7430 000020
7431 000022
7432 000024
7433 000026
7434 000030
7435 000032
7436 000034
7437 000036
7438 000040
7439 000042
7440 000044
7441 000046
7442

```

```

;*ATTENTION BITS TABLE (ATABIT=8 BYTES)
;*THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
;*ATTENTION BIT

```

```

ATABIT: .BYTE 1 ;DRIVE 0
        .BYTE 2 ;DRIVE 1
        .BYTE 4 ;DRIVE 2
        .BYTE 10 ;DRIVE 3
        .BYTE 20 ;DRIVE 4
        .BYTE 40 ;DRIVE 5
        .BYTE 100 ;DRIVE 6
        .BYTE 200 ;DRIVE 7

```

```

;*RP04 TO RH11 "MASS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
;*CALLING IT FATAL (MCPEMX=1 WORD)

```

```
MCPEMX: .WORD 3
```

```

;*STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04),
;*RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

```

```
RPADR: .WORD 176700
RPVEC: .WORD 254,5*32.
```

```

;*MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)

```

```
MXLACT: .WORD 4
```

```

;*MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

```

```
MXDLTA: .WORD 8*64.
```

```

;*MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

```

```
MNDLTA: .WORD 2*64.
```

```

;*MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

```

```
MXWNDW: .WORD 5
```

```

;*DEFINITIONS OF THE RH11/RP04 ADDRESS INDEXES

```

```

RHCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
RHWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
RHBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
RHDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
RHCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
RHDS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
RHER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
RHAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
RHLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
RHDB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
RHMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
RHDT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
RHSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
RHOF=32 ;OFFSET REGISTER (DRIVE REG. 11)
RHCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
RHCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
RHER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)
RHER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)
RHEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)
RHEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)

```

```

7443 ;*RH11/RP04 DRIVER INIT. CODE
7444 ;*THIS ROUTINE WILL DETERMINE WHICH RP04 DRIVES ARE
7445 ;*AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
7446 ;*TO THE PROPER STATE FOR EACH DRIVE.
7447 ;*NOTE: THIS ROUTINE CALLS DRVINT
7448
7449 ;*CALL
7450
7451 ;* JSR PC,RPINIT
7452 ;* RETURN
7453 ;*
7454 ;*NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
7455
7456 030732 004037 037104 RPINIT: JSR RO,SAVR15 ;GO SAVE R1-R5
7457 030736 013746 177776 MOV @#PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS
7458 030742 012737 000240 177776 MOV #(<5*32.>,@#PS) ;CHANGE THE PRIORITY TO 5
7459 030750 004737 036642 JSR PC,CLRQUE ;CLEAR ALL REQUEST QUEUES
7460 030754 012701 030536 MOV #RPERRS,R1 ;FIRST ADDRESS TO BE CLEARED
7461 030760 012702 030656 MOV #SEEKFG,R2 ;LAST ADDRESS TO BE CLEARED
7462 030764 005021 1$: CLR (R1)+ ;CLEAR
7463 030766 020102 CMP R1,R2 ;ARE WE DONE?
7464 030770 101775 BLOS 1$ ;BRANCH IF NO
7465 030772 012702 030700 MOV #DTUW,R2 ;LAST ADDRESS
7466 030776 012721 177777 2$: MOV #-1,(R1)+ ;INITIALIZE
7467 031002 020102 CMP R1,R2 ;DONE?
7468 031004 101774 BLOS 2$ ;LOOP IF NO
7469 031006 012737 177777 030556 MOV #-1,DRVSTA ;SET ALL DRIVES TO UNAVAILABLE
7470 031014 012737 177777 030560 MOV #-1,DRVSTA+2
7471 031022 012737 177777 030562 MOV #-1,DRVSTA+4
7472 031030 012737 177777 030564 MOV #-1,DRVSTA+6
7473 031036 013703 030716 MOV RPVEC,R3 ;SETUP THE RH11/RP04 VECTOR
7474 031042 012723 033364 MOV #ISR,(R3)+
7475 031046 013713 030720 MOV RPVEC+2,(R3)
7476 031052 013704 030714 MOV RPADR,R4 ;FIRST ADDRESS OF RH11/RP04
7477 031056 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;MASSBUS INIT
7478 031064 005001 CLR R1 ;START WITH DRIVE 0
7479 031066 004037 031160 3$: JSR RO,DRVINT ;INIT THE DRIVE
7480 031072 031152 B$
7481 031074 005201 4$: INC R1 ;GO TO NEXT DRIVE
7482 031076 042701 177770 BIC #1C7,R1 ;MASK OUT UNUSED BITS
7483 031102 001371 BNE 3$ ;BR IF MORE DRIVES TO GO
7484 031104 012701 000007 MOV #7,R1 ;START WITH DRIVE 7
7485 031110 005037 177776 CLR @#PS ;CLEAR THE PROCESSOR STATUS
7486 031114 105761 030606 5$: TSTB DPINT(R1) ;WAITING FOR DRIVE TO SWITCH PORTS ?
7487 031120 001405 BEQ 7$ ;BR NOT WAITING
7488 031122 004737 036276 JSR PC,SET.IE ;SET INTERRUPT
7489 031126 105761 030606 6$: TSTB DPINT(R1) ;DRIVE SWITCHED PORTS ?
7490 031132 001375 BNE 6$ ;BR IF NOT
7491 031134 005301 7$: DEC R1 ;GO TO THE NEXT DRIVE
7492 031136 100366 BPL 5$ ;CHECK NEXT DRIVE
7493 031140 012637 177776 MOV (SP)+,@#PS ;RESTORE THE PROCESSOR STATUS
7494 031144 004037 037124 JSR RO,GETR15 ;RESTORE R1-R5
7495 031150 000207 RTS PC ;BYE-BYE
7496 031152 105060 030556 8$: CLRB DRVSTA(RO) ;SET DRIVE STATUS TO OFFLINE
7497 031156 000746 BR 4$ ;CONTINUE WITH THE INIT
7498

```

```

7499          ;*DRIVE INIT. ROUTINE
7500          ;*THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
7501          ;*AN RPO4. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
7502          ;*IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
7503          ;*INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
7504          ;*DRVSTA IS SET TO THE PROPER CONDITION.
7505          ;*CALL
7506          ;*
7507          ;*
7508          ;*
7509          ;*
7510          ;*
7511          ;*
7512          ;*
7513          ;*
7514          ;*
7515          ;*
7516          ;*
7517          ;*
7518          ;*
7519          ;*
7520          ;*
7521          ;*
7522          ;*
7523          ;*
7524          ;*
7525          ;*
7526          ;*
7527          ;*
7528          ;*
7529          ;*
7530          ;*
7531          ;*
7532          ;*
7533          ;*
7534          ;*
7535          ;*
7536          ;*
7537          ;*
7538          ;*
7539          ;*
7540          ;*
7541          ;*
7542          ;*
7543          ;*
7544          ;*
7545          ;*
7546          ;*
7547          ;*
7548          ;*
7549          ;*
7550          ;*
7551          ;*
7552          ;*
7553          ;*
7554          ;*
7555          ;*
7556          ;*
7557          ;*
7558          ;*
7559          ;*
7560          ;*
7561          ;*
7562          ;*
7563          ;*
7564          ;*
7565          ;*
7566          ;*
7567          ;*
7568          ;*
7569          ;*
7570          ;*
7571          ;*
7572          ;*
7573          ;*
7574          ;*
7575          ;*
7576          ;*
7577          ;*
7578          ;*
7579          ;*
7580          ;*
7581          ;*
7582          ;*
7583          ;*
7584          ;*
7585          ;*
7586          ;*
7587          ;*
7588          ;*
7589          ;*
7590          ;*
7591          ;*
7592          ;*
7593          ;*
7594          ;*
7595          ;*
7596          ;*
7597          ;*
7598          ;*
7599          ;*
7600          ;*
7601          ;*
7602          ;*
7603          ;*
7604          ;*
7605          ;*
7606          ;*
7607          ;*
7608          ;*
7609          ;*
7610          ;*
7611          ;*
7612          ;*
7613          ;*
7614          ;*
7615          ;*
7616          ;*
7617          ;*
7618          ;*
7619          ;*
7620          ;*
7621          ;*
7622          ;*
7623          ;*
7624          ;*
7625          ;*
7626          ;*
7627          ;*
7628          ;*
7629          ;*
7630          ;*
7631          ;*
7632          ;*
7633          ;*
7634          ;*
7635          ;*
7636          ;*
7637          ;*
7638          ;*
7639          ;*
7640          ;*
7641          ;*
7642          ;*
7643          ;*
7644          ;*
7645          ;*
7646          ;*
7647          ;*
7648          ;*
7649          ;*
7650          ;*
7651          ;*
7652          ;*
7653          ;*
7654          ;*
7655          ;*
7656          ;*
7657          ;*
7658          ;*
7659          ;*
7660          ;*
7661          ;*
7662          ;*
7663          ;*
7664          ;*
7665          ;*
7666          ;*
7667          ;*
7668          ;*
7669          ;*
7670          ;*
7671          ;*
7672          ;*
7673          ;*
7674          ;*
7675          ;*
7676          ;*
7677          ;*
7678          ;*
7679          ;*
7680          ;*
7681          ;*
7682          ;*
7683          ;*
7684          ;*
7685          ;*
7686          ;*
7687          ;*
7688          ;*
7689          ;*
7690          ;*
7691          ;*
7692          ;*
7693          ;*
7694          ;*
7695          ;*
7696          ;*
7697          ;*
7698          ;*
7699          ;*
7700          ;*
7701          ;*
7702          ;*
7703          ;*
7704          ;*
7705          ;*
7706          ;*
7707          ;*
7708          ;*
7709          ;*
7710          ;*
7711          ;*
7712          ;*
7713          ;*
7714          ;*
7715          ;*
7716          ;*
7717          ;*
7718          ;*
7719          ;*
7720          ;*
7721          ;*
7722          ;*
7723          ;*
7724          ;*
7725          ;*
7726          ;*
7727          ;*
7728          ;*
7729          ;*
7730          ;*
7731          ;*
7732          ;*
7733          ;*
7734          ;*
7735          ;*
7736          ;*
7737          ;*
7738          ;*
7739          ;*
7740          ;*
7741          ;*
7742          ;*
7743          ;*
7744          ;*
7745          ;*
7746          ;*
7747          ;*
7748          ;*
7749          ;*
7750          ;*
7751          ;*
7752          ;*
7753          ;*
7754          ;*
7755          ;*
7756          ;*
7757          ;*
7758          ;*
7759          ;*
7760          ;*
7761          ;*
7762          ;*
7763          ;*
7764          ;*
7765          ;*
7766          ;*
7767          ;*
7768          ;*
7769          ;*
7770          ;*
7771          ;*
7772          ;*
7773          ;*
7774          ;*
7775          ;*
7776          ;*
7777          ;*
7778          ;*
7779          ;*
7780          ;*
7781          ;*
7782          ;*
7783          ;*
7784          ;*
7785          ;*
7786          ;*
7787          ;*
7788          ;*
7789          ;*
7790          ;*
7791          ;*
7792          ;*
7793          ;*
7794          ;*
7795          ;*
7796          ;*
7797          ;*
7798          ;*
7799          ;*
7800          ;*
7801          ;*
7802          ;*
7803          ;*
7804          ;*
7805          ;*
7806          ;*
7807          ;*
7808          ;*
7809          ;*
7810          ;*
7811          ;*
7812          ;*
7813          ;*
7814          ;*
7815          ;*
7816          ;*
7817          ;*
7818          ;*
7819          ;*
7820          ;*
7821          ;*
7822          ;*
7823          ;*
7824          ;*
7825          ;*
7826          ;*
7827          ;*
7828          ;*
7829          ;*
7830          ;*
7831          ;*
7832          ;*
7833          ;*
7834          ;*
7835          ;*
7836          ;*
7837          ;*
7838          ;*
7839          ;*
7840          ;*
7841          ;*
7842          ;*
7843          ;*
7844          ;*
7845          ;*
7846          ;*
7847          ;*
7848          ;*
7849          ;*
7850          ;*
7851          ;*
7852          ;*
7853          ;*
7854          ;*
7855          ;*
7856          ;*
7857          ;*
7858          ;*
7859          ;*
7860          ;*
7861          ;*
7862          ;*
7863          ;*
7864          ;*
7865          ;*
7866          ;*
7867          ;*
7868          ;*
7869          ;*
7870          ;*
7871          ;*
7872          ;*
7873          ;*
7874          ;*
7875          ;*
7876          ;*
7877          ;*
7878          ;*
7879          ;*
7880          ;*
7881          ;*
7882          ;*
7883          ;*
7884          ;*
7885          ;*
7886          ;*
7887          ;*
7888          ;*
7889          ;*
7890          ;*
7891          ;*
7892          ;*
7893          ;*
7894          ;*
7895          ;*
7896          ;*
7897          ;*
7898          ;*
7899          ;*
7900          ;*
7901          ;*
7902          ;*
7903          ;*
7904          ;*
7905          ;*
7906          ;*
7907          ;*
7908          ;*
7909          ;*
7910          ;*
7911          ;*
7912          ;*
7913          ;*
7914          ;*
7915          ;*
7916          ;*
7917          ;*
7918          ;*
7919          ;*
7920          ;*
7921          ;*
7922          ;*
7923          ;*
7924          ;*
7925          ;*
7926          ;*
7927          ;*
7928          ;*
7929          ;*
7930          ;*
7931          ;*
7932          ;*
7933          ;*
7934          ;*
7935          ;*
7936          ;*
7937          ;*
7938          ;*
7939          ;*
7940          ;*
7941          ;*
7942          ;*
7943          ;*
7944          ;*
7945          ;*
7946          ;*
7947          ;*
7948          ;*
7949          ;*
7950          ;*
7951          ;*
7952          ;*
7953          ;*
7954          ;*
7955          ;*
7956          ;*
7957          ;*
7958          ;*
7959          ;*
7960          ;*
7961          ;*
7962          ;*
7963          ;*
7964          ;*
7965          ;*
7966          ;*
7967          ;*
7968          ;*
7969          ;*
7970          ;*
7971          ;*
7972          ;*
7973          ;*
7974          ;*
7975          ;*
7976          ;*
7977          ;*
7978          ;*
7979          ;*
7980          ;*
7981          ;*
7982          ;*
7983          ;*
7984          ;*
7985          ;*
7986          ;*
7987          ;*
7988          ;*
7989          ;*
7990          ;*
7991          ;*
7992          ;*
7993          ;*
7994          ;*
7995          ;*
7996          ;*
7997          ;*
7998          ;*
7999          ;*
8000          ;*
  
```

```

7555 031364 001004          BNE      5$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
7556 031366 112761 000001 030556  MOVB    #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
7557 031374 000402          BR       6$
7558 031376 105061 030556          5$:   CLRB    DRVSTA(R1)   ;DRIVE STATUS = OFFLINE
7559 031402 005720          6$:   TST     (R0)+      ;STEP OVER THE ERROR RETURN
7560 031404 004037 037124          7$:   JSR     R0,GETR15  ;RESTORE R1-R5
7561 031410 000200          RTS     R0           ;RETURN
7562 031412 012763 003720 030660  8$:   MOV     #2000.,TIMER(R3) ;START 2 SEC TIMER
7563 031420 012764 000000 000012  MOV     #0,RHDS1(R4)  ;SET PORT REQUEST
7564 031426 105061 030556          CLRB    DRVSTA(R1)   ;SET DRIVE TO OFFLINE
7565 031432 112761 177777 030606  MOVB    #-1,DPINT(R1) ;SET PORT INITIALIZE INIDICATOR
7566 031440 000760          BR       6$          ;EXIT
7567
7568          ;*REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
7569          ;*CALL
7570
7571          ;*
7572          ;*   JSR     R0,#RP04      ;CALL THE RP04 DRIVER
7573          ;*   PNTADR  ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
7574          ;*   RETURN1 ;RETURN HERE IF QUEUE IS FULL
7575          ;*   RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
7576          ;*           ;IS AN ERROR CONDITION
7577
7578 031442 013746 177776          RP04: MOV     @#PS,-(SP)    ;SAVE THE CALLING STATUS
7579 031446 013737 030720 177776  MOV     RPVEC+2,@#PS ;DON'T ALLOW ANY RP04 INTERRUPTS
7580 031454 112737 000001 030632  MOVB    #1,ACTDRV    ;SET "ACTIVE DRIVER" FLAG
7581 031462 004037 037104          JSR     R0,SAVR15   ;SAVE R1-R5
7582 031466 012002          MOV     (R0)+,R2    ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
7583 031470 005062 000016          CLR     16(R2)     ;CLEAR THE STATUS/ERROR INDICATOR
7584 031474 111201          MOVB    (R2),R1     ;PICKUP THE DRIVE NUMBER
7585 031476 013704 030714          MOV     RPADR,R4   ;UNIBUS ADDRESS OF RHCS1
7586 031502 105761 030556          TSTB   DRVSTA(R1)  ;CHECK DRIVES STATUS
7587 031506 003021          BGT     1$         ;BRANCH IF ONLINE
7588 031510 105761 030634          TSTB   ULDFLG(R1)  ;UNLOAD COMMAND IN QUEUE?
7589 031514 001043          BNE     3$         ;BRANCH IF YES
7590 031516 105761 030606          TSTB   DPINT(R1)   ;TRYING TO INIT THE DRIVE
7591 031522 001056          BNE     7$         ;BR IF YES
7592 031524 013704 030714          MOV     RPADR,R4   ;UNIBUS ADDRESS OF RHCS1
7593 031530 004037 031160          JSR     R0,DRVINT   ;GO INIT. THE DRIVE
7594 031534 000446          BR       6$        ;ERROR RETURN
7595 031536 105761 030606          TSTB   DPINT(R1)   ;SEE IF PORT REGEST OUTSTANDING
7596 031542 001046          BNE     7$         ;BR IF IT IS
7597 031544 105761 030556          TSTB   DRVSTA(R1)  ;IS DRIVE STATUS ONLINE?
7598 031550 003454          BLE     8$         ;BR IF NOT
7599 031552 105761 030616          1$:   TSTB   DPRQS(R1)  ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
7600 031556 001040          BNE     7$         ;BR IF YES
7601 031560 010164 000010          MOV     R1,RHCS2(R4) ;SELECT THE DRIVE
7602 031564 004037 036744          JSR     R0,DRVQUE   ;PUT THIS REQUEST IN QUEUE
7603 031570 000421          BR       5$        ;QUEUE IS FULL
7604 031572 122762 000103 000002  CMPB    #103,2(R2)  ;IS THIS REQ. FOR AN UNLOAD?
7605 031600 001003          BNE     2$         ;BR IF NO
7606 031602 112761 177777 030634  MOVB    #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
7607 031610 105761 030546          2$:   TSTB   DRVACT(R1)  ;IS THIS DRIVE ACTIVE?
7608 031614 001006          BNE     4$         ;BR IF YES
7609 031616 004737 031730          JSR     PC,OPT      ;CALL THE OPTIMIZER
7610 031622 000403          BR       4$

```



```

7611 031624 052762 120000 000016 3$: BIS #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
7612 031632 005720 4$: TST (R0)+
7613 031634 004037 037124 5$: JSR R0,GETR15 ;RESTORE R1-R5
7614 031640 105037 030632 CLR B ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
7615 031644 012637 177776 MOV (SP)+,Q#PS ;RETURN "PS" TO USER LEVEL
7616 031650 000200 RTS R0 ;RETURN TO CALLER
7617 031652 004737 032770 6$: JSR PC,C17 ;GO HANDLE THE PARITY ERROR
7618 031656 000765 BR 4$
7619 031660 004037 036744 7$: JSR R0,DRVQUE ;PUT REQUEST IN QUEUE
7620 031664 000763 BR 5$ ;QUEUE IS FULL
7621 031666 032714 000100 BIT #BIT06,(R4) ;IS 'IE' SET ALREADY ?
7622 031672 001357 BNE 4$ ;BR IF IT IS
7623 031674 004737 036276 JSR PC,SET.IE ;SET INTERRUPT
7624 031700 000754 BR 4$ ;RETURN, REQUEST IN QUEUE
7625 031702 105761 030556 8$: TSTB DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
7626 031706 002404 BLT 9$ ;BR IF UNSAFE
7627 031710 052762 140000 000016 BIS #BIT15!BIT14,16(R2) ;ERROR--DRIVE IS OFFLINE
7628 031716 000745 BR 4$ ;GO TO EXIT
7629 031720 052762 110000 000016 9$: BIS #BIT15!BIT12,16(R2) ;ERROR--DRIVE IS UNSAFE
7630 031726 000741 BR 4$ ;GO TO EXIT
7631
7632 ;*OPTIMIZER-CALLED FOR A PARTICULAR DRIV
7633
7634 ;*CALL
7635 ;* MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
7636 ;* JSR PC,OPT ;SETUP A COMMAND
7637
7638 031730 004037 037104 OPT: JSR R0,SAVR15 ;SAVE R1-R5
7639 031734 013746 177776 MOV Q#PS, -(SP) ;SAVE PROC. STATUS
7640 031740 146137 030702 030630 BIC B ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
7641 031746 004737 037020 JSR PC,GETREQ ;GET "DPB" POINTER OF REQUEST
7642 031752 005702 TST R2 ;IS THERE A REQUEST IN QUEUE?
7643 031754 001452 BEQ 6$ ;NO--BRANCH TO EXIT
7644 031756 105761 030556 TSTB DRVSTA(R1) ;IS DRIVE ONLINE?
7645 031762 003006 BGT 1$ ;YES--BRANCH
7646 031764 004737 037042 JSR PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
7647 031770 052762 140000 000016 BIS #BIT15!BIT14,16(R2) ;SET ERROR BIT OF STATUS/ERROR INDICATOR
7648 031776 000434 BR 5$ ;BRANCH TO EXIT
7649 032000 012764 000000 000012 1$: MOV #0,RHDS1(R4) ;SEIZE THE DRIVE OR SET REQUEST
7650 032006 032714 004000 BIT #BIT11,(R4) ;DRIVE AVAILABLE ?
7651 032012 001441 BEQ 7$ ;BR IF NOT
7652 032014 122762 000150 000002 CMP B #150,2(R2) ;IS THE REQUEST FOR I/O?
7653 032022 002403 BLT 2$ ;YES--BRANCH
7654 032024 004737 032354 JSR PC,C14 ;CALL THE COMMAND INITIATOR
7655 032030 000417 BR 5$ ;BRANCH TO EXIT
7656 032032 005737 030700 2$: TST DTUW ;DATA TRANSFER UNDERWAY?
7657 032036 002012 BGE 4$ ;YES--GO START A SEARCH
7658 032040 005737 030656 TST SEEKFG ;DO IMPLIED SEEKS?
7659 032044 100404 BMI 3$ ;YES---BRANCH
7660 032046 004037 033226 JSR R0,LA ;NO--DO LOOK AHEAD
7661 032052 000406 BR 5$ ;RETURN HERE ON A PARITY ERROR
7662 032054 000403 BR 4$ ;GO START A SEARCH
7663 032056 004737 032140 3$: JSR PC,C11 ;START A DATA TRANSFER
7664 032062 000402 BR 5$
7665 032064 004737 032246 4$: JSR PC,C13 ;START A SEARCH
7666 032070 012637 177776 5$: MOV (SP)+,Q#PS ;RESTORE PROC. STATUS
  
```

```

7667 032074 004037 037124      JSR      RD,GETR15      ;RESTORE R1-R5
7668 032100 000207              RTS      PC
7669 032102 032714 000100      6$:     BIT      #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
7670 032106 001370              BNE     5$             ;BR IF SET
7671 032110 004737 036276      JSR      PC,SET.IE     ;SET "IE" WITHOUT A "TRE"
7672 032114 000765              BR      5$             ;EXIT THE ROUTINE
7673 032116 1:2761 177777 030616 7$:     MOVB   #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
7674 032124 010103              MOV     R1,R3         ;SET UP TO ADDRESS WORDS
7675 032126 006303              ASL     R3            ;CONVERT TO WORD INDEX
7676 032130 012763 023420 030660  MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
7677 032136 000761              BR      6$            ;SET 'IE'
7678
7679                      ;*COMMAND INITIATOR
7680
7681                      ;*CALL
7682                      ;*
7683                      ;*     MOV     #DRVNUM,R1      ;DRIVE NUMBER
7684                      ;*     MOV     #DPB,R2        ;ADDRESS OF DPB
7685                      ;*     JSR     PC,CI?       ;CI?= CI1,CI3, OR CI4
7686                      ;*
7687                      ;*
7688                      ;*
7689                      ;*
7690                      ;*
7690 032140 004737 037042      CI1:    JSR      PC,POPQUE ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
7691 032144 010237 030626      MOV     R2,TRNSWT    ;PUT REQ. IN TRANSFER WAIT QUEUE
7692 032150 010203              CI2:    MOV     R2,R3   ;DPB ADDRESS TO R3
7693 032152 013704 030714      MOV     RPADR,R4     ;RHCS1 ADDRESS
7694 032156 010164 000010      MOV     R1,RHCS2(R4) ;SELECT DRIVE
7695 032162 062703 000004      ADD     #4,R3        ;DESIRED WORD COUNT
7696 032166 062704 000002      ADD     #2,R4        ;RHC ADDRESS
7697 032172 012324              MOV     (R3)+,(R4)+  ;LOAD WORD COUNT
7698 032174 012324              MOV     (R3)+,(R4)+  ;LOAD BUFFER ADDRESS
7699 032176 012346              MOV     (R3)+,-(SP)  ;LOAD SECTOR AND TRACK
7700 032200 004037 036036      JSR     RD,WRT.RP    ;CALL THE LOAD(WRITE) ROUTINE
7701 032204 000006              RHDA   CI7           ;INDEX OF REGISTER TO LOAD
7702 032206 032770              CI7
7703 032210 012346              MOV     (R3)+,-(SP)  ;ERROR RETURN ADDRESS
7704 032212 004037 036036      JSR     RD,WRT.RP    ;LOAD CYLINDER ADDRESS
7705 032216 000034              RHCA   CI7
7706 032220 032770              CI7
7707 032222 016246 000002      MOV     2(R2),-(SP)  ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
7708 032226 004037 036036      JSR     RD,WRT.RP
7709 032232 000000              RHCS1  CI7
7710 032234 032770              CI7
7711 032236 010137 030700      MOV     R1,DTUW     ;SET "DATA TRANSFER UNDERWAY"
7712 032242 000137 032732              JMP     CI5
7713 032246 013704 030714      CI3:    MOV     RPADR,R4     ;RHCS1 ADDRESS
7714 032252 010164 000010      MOV     R1,RHCS2(R4) ;SELECT DRIVE
7715 032256 016246 000012      MOV     12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
7716 032262 004037 036036      JSR     RD,WRT.RP
7717 032266 000034              RHCA   CI7
7718 032270 032770              CI7
7719 032272 116203 000010      MOVB   10(R2),R3    ;PICKUP SECTOR ADDRESS
7720 032276 163703 030730      SUB     MXWINDW,R3   ;BACKUP BY MAX. SEARCH FOR I/O WINDOW
7721 032302 002002              BGE    1$
7722 032304 062703 000026      ADD     #22.,R3

```

7723	032310	010346			1\$:	MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
7724	032312	116266	000011	000001		MOVB	11(R2),1(SP)	;THE DESIRED TRACK
7725	032320	004037	036036			JSR	RO,WRT.RP	;LOAD DESIRED TRACK & SECTOR
7726	032324	000006				RHDA		
7727	032326	032770				CI7		
7728	032330	012746	000131			MOV	#131,-(SP)	;START A SEARCH
7729	032334	004037	036036			JSR	RO,WRT.RP	
7730	032340	000000				RHCS1		
7731	032342	032770				CI7		
7732	032344	156137	030702	030630		BISB	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
7733	032352	000567				BR	CI5	
7734	032354	013704	030714		CI4:	MOV	RPADR,R4	;RHCS1 ADDRESS
7735	032360	010164	000010			MOV	R1,RHCS2(R4)	;SELECT DRIVE
7736	032364	116203	000002			MOVB	2(R2),R3	;PICKUP THE REQUESTED COMMAND
7737	032370	122703	000131			CMPB	#131,R3	;IS IT A SEARCH COMMAND?
7738	032374	001007				BNE	1\$;BRANCH IF NO
7739	032376	016246	000010			MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
7740	032402	004037	036036			JSR	RO,WRT.RP	
7741	032406	000006				RHDA		
7742	032410	032770				CI7		
7743	032412	000403				BR	2\$;GO LOAD CYLINDER
7744	032414	122703	000105		1\$:	CMPB	#105,R3	;IS IT A SEEK COMMAND
7745	032420	001007				BNE	3\$;BRANCH IF NO
7746	032422	016246	000012		2\$:	MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER
7747	032426	004037	036036			JSR	RO,WRT.RP	
7748	032432	000034				RHCA		
7749	032434	032770				CI7		
7750	032436	000546				BR	CI6	
7751	032440	122703	000115		3\$:	CMPB	#115,R3	;IS IT AN "OFFSET" COMMAND?
7752	032444	001013				BNE	4\$;BR IF NO
7753	032446	004037	035700			JSR	RO,RD.RP	;MERGE THE OFFSET VALUE INTO RHOF
7754	032452	000032				RHOF		;BUT DON'T CHANGE THE UPPER
7755	032454	032770				CI7		
7756	032456	116216	000001			MOVB	1(R2),(SP)	;BYTE WHEN LOADING THE
7757	032462	004037	036036			JSR	RO,WRT.RP	;REGISTER (RHOF)
7758	032466	000032				RHOF		
7759	032470	032770				CI7		
7760	032472	000530				BR	CI6	;GO START THE COMMAND
7761	032474	122703	000107		4\$:	CMPB	#107,R3	;IS IT A "RECALIBRATE" COMMAND?
7762	032500	001525				BEQ	CI6	;BRANCH IF YES
7763	032502	122703	000117			CMPB	#117,R3	;IS IT A RETURN TO CENTER?
7764	032506	001522				BEQ	CI6	;BRANCH IF YES
7765	032510	122703	000103			CMPB	#103,R3	;IS IT AN "UNLOAD" COMMAND?
7766	032514	001016				BNE	5\$;BRANCH IF NO
7767	032516	112761	000001	030546		MOVB	#1,DRVACT(R1)	;SET THE DRIVE ACTIVE INDICATOR
7768	032524	105061	030556			CLRB	DRVSTA(R1)	;PUT DRIVE STATUS TO OFFLINE
7769	032530	112761	000001	030634		MOVB	#1,ULDFLG(R1)	;SET "UNLOAD IN PROGRESS" FLAG
7770	032536	010346				MOV	R3,-(SP)	;START THE "UNLOAD" COMMAND
7771	032540	004037	036036			JSR	RO,WRT.RP	
7772	032544	000000				RHCS1		
7773	032546	032770				CI7		
7774	032550	000207				RTS	PC	;RETURN TO USER
7775	032552	122703	000143		5\$:	CMPB	#143,R3	;IS IT A "SET FORMAT" COMMAND?
7776	032556	001014				BNE	6\$;BRANCH IF NO
7777	032560	004037	035700			JSR	RO,RD.RP	;READ THE OFFSET REGISTER
7778	032564	000032				RHOF		

7779	032566	032770			CI7		
7780	032570	116266	000001	000001	MOVB	1(R2),1(SP)	:COMBINE "FMT22" "ECI" AND "HCI"
7781	032576	004037	036036		JSR	RO,WRT.RP	:LOAD "FMT22", "ECI", AND/OR "HCI".
7782	032602	000032			RHOF		
7783	032604	032770			CI7		
7784	032606	000436			BR	12\$	
7785	032610	122703	000141		CMPB	#141,R3	:IS IT A "GET REGISTER" COMMAND?
7786	032614	001023			BNE	10\$:BRANCH IF NO
7787	032616	016203	000006		MOV	6(R2),R3	:POINTS TO 1ST ADDRESS OF WHERE
7789							:TO PUT THE REGISTER(S)
7789	032622	116237	000010	032640	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
7790	032630	116205	000011		MOVB	11(R2),R5	:INDEX OF LAST REG. TO MOVE
7791	032634	004037	035700		JSR	RO,RO.RP	:READ RP04 REGISTER
7792	032640	000000			RHCS1		:INDEX OF REG. TO READ
7793	032642	032770			CI7		
7794	032644	012623			MOV	(SP)+,(R3)+	:GET THE CONTENTS OF RH11/RP04 REG.
7795	032646	023705	032640		CMP	9\$,R5	:LAST REG. BEEN READ?
7796	032652	001414			BEQ	12\$:GET OUT IF YES
7797	032654	062737	000002	032640	ADD	#2,9\$:INCREASE THE INDEX BY 2
7798	032662	000764			BR	8\$:LOOP--MORE TO READ
7799	032664	122703	000145		CMPB	#145,R3	:IS IT A "SELECT DRIVE" COMMAND?
7800	032670	001405			BEQ	12\$:BRANCH IF YES
7801	032672	010346			MOV	R3,-(SP)	:LOAD THE COMMAND
7802	032674	004037	036036		JSR	RO,WRT.RP	
7803	032700	000000			RHCS1		
7804	032702	032770			CI7		
7805	032704	004737	037042		JSR	PC,POPQUE	:REMOVE REG. FROM QUEUE
7806	032710	052762	000200	000016	BIS	#BIT07,16(R2)	:SET THE "DONE" BIT
7807	032716	005737	030654		TST	SAVEFG	:SAVE THE RH11/RP04 REGISTERS?
7808	032722	100002			BPL	13\$:BRANCH IF NO
7809	032724	004737	036200		JSR	PC,SVRH11	:YES--GO SAVE THE REGISTERS
7810	032730	000207			RTS	PC	:RETURN TO USER
7811	032732	006301			ASL	R1	
7812	032734	012761	001750	030660	MOV	#1000.,TIMER(R1)	:SET A ONE SECOND TIMER
7813	032742	006201			ASR	R1	
7814	032744	112761	000001	030546	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE
7815	032752	000207			RTS	PC	:RETURN TO THE USER
7816	032754	010346			MOV	R3,-(SP)	:LOAD THE COMMAND
7817	032756	004037	036036		JSR	RO,WRT.RP	
7818	032762	000000			RHCS1		
7819	032764	032770			CI7		
7820	032766	000761			BR	CI5	
7821	032770	005702			TST	R2	:ANYTHING IN QUEUE ?
7822	032772	001405			BEQ	CI7B	:BR IF NOT
7823	032774	012762	104000	000016	MOV	#BIT15:BIT11,16(R2)	:SET "PARITY" ERROR INDICATOR
7824	033002	004737	036200		JSR	PC,SVRH11	:GO SAVE THE RH11/RP04 REGISTERS
7825	033006	012746	000111		MOV	#111,-(SP)	:DO A "DRIVE CLEAR"
7826	033012	004037	036036		JSR	RO,WRT.RP	
7827	033016	000000			RHCS1		
7828	033020	033060			CI8		
7829	033022	004737	036724		JSR	PC,EMPTYQ	:EMPTY THE QUEUE
7830	033026	105061	030634		CLRB	UNDFLG(R1)	:CLEAR THE UNLOAD IN QUEUE FLAG
7831	033032	105061	030546		CLRB	DRVACT(R1)	:DRIVE IS IDLE
7832	033036	020137	030700		CMP	R1,DTUW	:IF THIS DRIVE HAD AN I/O REQUEST
7833	033042	001005			BNE	1\$:IN PROGRESS CLEAR ALL OF THE FLAGS
7834	033044	005037	030626		CLR	TRNSWT	

```

7835 033050 012737 177777 030700      MOV      #-1,DTUW
7836 033056 000207      1$:      RTS      PC
7837 033060 004037 037104      C18:     JSR      RD,SAVR15      ;SAVE R1-R5
7838 033064 013704 030714      MOV      RPADR,R4      ;PICKUP THE ADDRESS OF THE FIRST REGISTER
7839 033070 005001      CLR      R1
7840 033072 005003      CLR      R3
7841 033074 105761 030546      1$:      TSTB     DRVACT(R1)    ;DRIVE ACTIVE?
7842 033100 001423      BEQ      4$            ;BRANCH IF NO
7843 033102 013702 030626      MOV      TRANSW,R2     ;GET THE "TRANSFER WAIT" QUEUE
7844 033106 020137 030700      CMP      R1,DTUW      ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
7845 033112 001402      BEQ      2$            ;BRANCH IF YES
7846 033114 004737 037020      JSR      PC,GETREQ     ;GET THE DPB POINTER
7847 033120 005702      2$:      TST      R2           ;QUEUE ENTRY FOR DRIVE ?
7848 033122 001405      BEQ      3$            ;BR IF NOT
7849 033124 052762 102000 000016      BIS      #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
7850 033132 004737 036200      JSR      PC,SVRH11     ;SAVE RH11/RP04 REGISTERS
7851 033136 012763 177777 030660      3$:      MOV      #-1,TIMER(R3) ;STOP THE TIMER
7852 033144 105061 030546      CLRB     DRVACT(R1)    ;SET "DRIVE ACTIVE" TO IDLE
7853 033150 105061 030634      4$:      CLRB     ULDFLG(R1)    ;CLEAR UNLOAD FLAG
7854 033154 005201      INC      R1           ;MOVE TO THE NEXT DRIVE
7855 033156 062703 000002      ADD      #2,R3
7856 033162 042701 177770      BIC      #1<7,R1
7857 033166 001342      BNE      1$           ;BRANCH IF MORE DRIVES
7858 033170 012737 177777 030700      MOV      #-1,DTUW     ;NO DATA TRANSFERS UNDERWAY
7859 033176 005037 030626      CLR      TRANSW      ;CLEAR THE "TRANSFER WAIT" QUEUE
7860 033202 004737 036642      JSR      PC,CLRQUE     ;CLEAR ALL OF THE REQUEST QUEUES
7861 033206 012764 000040 000010      MOV      #BIT05,RHCS2(R4) ;DO A MASSBUS INIT.
7862 033214 004737 036276      JSR      PC,SET.IE     ;SET "IE" WITHOUT "TRE"
7863 033220 004037 037124      JSR      RD,GETR15     ;RESTORE THE REGISTERS
7864 033224 000207      RTS      PC           ;RETURN
7865
7866      ;*LOOK AHEAD ROUTINE
7867      ;*CALL
7868      ;*
7869      ;*      MOV      #DRVNUM,R1      ;DRIVE NUMBER
7870      ;*      MOV      #DPB,R2        ;POINT TO DPB
7871      ;*      JSR      RD,LA         ;GO CHECK THE WINDOW
7872      ;*      RETURN1      ;ERROR RETURN
7873      ;*      RETURN2      ;START A SEARCH
7874      ;*      RETURN3      ;START A DATA TRANSFER
7875
7876 033226 013704 030714      LA:      MOV      RPADR,R4      ;GET RHCS1'S ADDRESS
7877 033232 010164 000010      MOV      R1,RHCS2(R4) ;SELECT DRIVE
7878 033236 004037 035700      JSR      RD,RD.RP      ;READ CURRENT CYLINDER
7879 033242 000036      RHCC
7880 033244 033356      4$
7881 033246 022662 000012      CMP      (SP)+,12(R2)  ;ERROR RETURN ADDRESS
7882      ;IS CURRENT CYLINDER=DESIRED
7883      ;CYLINDER?
7883 033252 001037      BNE      3$            ;EXIT IF NO
7884 033254 105261 030644      INCB     LACNT(R1)     ;INCREMENT THE LOOK AHEAD COUNT
7885 033260 126137 030644 030722      CMPB     LACNT(R1),MXLACT ;EXCEED MAX?
7886 033266 003026      BGT      2$            ;BRANCH IF YES
7887 033270 116203 000010      MOVB     10(R2),R3    ;GET DESIRED SECTOR ADDRESS AND
7888 033274 000303      SWAB     R3           ;MULT. BY 64--ALIGN WITH
7889 033276 006203      ASR      R3           ;LOOK AHEAD REGISTER
7890 033300 006203      ASR      R3

```

```

7891 033302 012737 000340 177776      MOV      #340,DP#PS      ;PRIORITY LEVEL "7"
7892 033310 004037 035700                JSR      RD,RD.RP      ;READ LOOK AHEAD REGISTER
7893 033314 000020                RHLA
7894 033316 033356                4$
7895 033320 162603                SUB      (SP)+,R3      ;CALCULATE THE DELTA
7896 033322 002002                BGE     1$
7897 033324 062703 002600                ADD      #<22.*64.>,R3  ;MAKE THE DELTA POSITIVE
7898 033330 023703 030724                1$:     CMP      MXDLTA,R3  ;CHECK THE DELTA TO SEE
7899 033334 002406                BLT     3$              ;IF IT IS WITHIN THE
7900 033336 023703 030726                CMP      MNDLTA,R3    ;WINDOW---IF YES, ZERO
7901 033342 002003                BGE     3$              ;THE LOOK AHEAD COUNT
7902 033344 105061 030644                2$:     CLRB   LACNT(R1) ;AND TAKE THE I/O EXIT
7903 033350 005720                TST     (RD)+
7904 033352 005720                3$:     TST     (RD)+    ;ADJUST THE RETURN ADDRESS
7905 033354 000200                RTS     RD
7906 033356 004737 032770                4$:     JSR      PC,CI7   ;PROCESS THE ERROR
7907 033362 000200                RTS     RD              ;TAKE ERROR RETURN
7908
7909                ;*INTERRUPT SERVICE ROUTINE
7910
7911 033364 112737 000001 030632  ISR:   MOVB   #1,ACTDRV      ;SET "ACTIVE DRIVER" FLAG
7912 033372 004037 037104                JSR      RD,SAVR15     ;SAVE R1-R5
7913 033376 013704 030714                MOV      RPADR,R4     ;ADDRESS OF RHSCS1
7914 033402 013701 030700                MOV      DTUW,R1      ;GET "DATA TRANSFER UNDERWAY" INDICATOR
7915 033406 002403                BLT     1$              ;BRANCH IF NO DATA TRANSFER UNDERWAY
7916 033410 004737 033434                JSR      PC,TD         ;CALL TRANSFER DONE
7917 033414 000402                BR      2$              ;EXIT
7918 033416 004737 033716                1$:     JSR      PC,SC   ;CALL SPECIAL CONDITIONS
7919 033422 004037 037124                2$:     JSR      RD,GETR15 ;RESTORE R1-R5
7920 033426 105037 030632                CLRB   ACTDRV         ;CLEAR "ACTIVE DRIVER" FLAG
7921 033432 000002                RTI
7922
7923                ;*TRANSFER DONE ROUTINE
7924
7925 033434 105061 030546 030700  TD:   CLRB   DRVACT(R1)   ;SET DRIVE ACTIVE INDICATOR TO IDLE
7926 033440 012737 177777                MOV      #-1,DTUW     ;NO DATA TRANSFERS UNDERWAY
7927 033446 006301                ASL     R1
7928 033450 012761 177777 030660                MOV      #-1,TIMER(R1) ;CANCEL TIMEOUT
7929 033456 006201                ASR     R1
7930 033460 013702 030626                MOV      TRNSWT,R2    ;GET "DPB" ADDRESS FROM THE
7931 033464 005037 030626                CLR     TRNSWT        ;TRANSFER WAIT QUEUE--CLEAR QUEUE
7932 033470 052762 000200 000016                BIS     #BIT07,16(R2) ;SET DONE
7933 033476 010164 000010                MOV     R1,RHCS2(R4)  ;SELECT THE DRIVE
7934 033502 004037 035700                JSR     RD,RD.RP      ;TRANSFER ERROR(TRE=1)?
7935 033506 000000                RHCS1
7936 033510 032770                CI7
7937 033512 006126                ROL     (SP)+
7938 033514 100421                BMI     3$              ;BR IF YES
7939 033516 005737 030654                TST     SAVEFG        ;SAVE THE RH11/RP04 REGISTERS?
7940 033522 100002                BPL     1$              ;BRANCH IF NO
7941 033524 004737 036200                JSR     PC,SVRH11     ;YES--SAVE THE REGISTERS
7942 033530 004737 033610                1$:     JSR     PC,WC   ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
7943 033534 004737 037020                JSR     PC,GETREQ     ;GET DPB POINTER
7944 033540 005702                TST     R2            ;ENTRY FOR DRIVE ?
7945 033542 001403                BEQ     2$              ;BR IF NOT
7946 033544 004737 031730                JSR     PC,OPT        ;CALL OPTIMIZER

```

```

7947 033550 000520          BR      SC2          ;SPECIAL CONDITION (ENTRY #2)
7948 033552 012714 000113 2$:      MOV      #113,(R4)      ;RELEASE THE DRIVE
7949 033556 000515          BR      SC2
7950 033560 052762 100100 000016 3$:      BIS      #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
7951 033566 004737 036724          JSR      PC,EMPTYQ      ;EMPTY THE "DRIVES WAIT" QUEUE
7952 033572 004737 036200          JSR      PC,SVRH11     ;SAVE THE RH11/RP04 REGISTERS
7953 033576 012714 040111          MOV      #40111,(R4)   ;ISSUE A "DRIVE CLEAR"
7954 033602 012714 000113          MOV      #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
7955 033606 000501          BR      SC2          ;SPECIAL CONDITION (ENTRY #2)
7956
7957          ;*FORCED WRITE CHECK ROUTINE
7958
7959 033610 005737 001316  WC:      TST      AUTOCK      ;AUTOMATIC WRITE CHECKS ?
7960 033614 001437          BEQ      2$          ;BR IF NOT
7961 033616 122762 000002 000030  CMPB     #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
7962 033624 001404          BEQ      1$          ;BR IF IT WAS
7963 033626 122762 000003 000030  CMPB     #3,$CODE(R2) ;LAST OPERATION WRITE HEADER & DATA ?
7964 033634 001027          BNE      2$          ;BR IF NOT
7965 033636 004037 036744  1$:      JSR      RD,DRVQUE    ;PUT THE OPERATION IN THE QUEUE
7966 033642 000424          BR      2$          ;QUEUE IS FULL
7967 033644 005062 000016          CLR      16(R2)      ;CLEAR 'DONE' BIT IN DPB
7968 033650 116262 000030 000032  MOVB     $CODE(R2),$PREV0(R2) ;SAVE WRITE OPERATION CODE
7969 033656 016262 000012 000040  MOV      $CYL(R2),$PREVA+2(R2) ;SAVE CYLINDER
7970 033664 016262 000010 000036  MOV      $SEC(R2),$PREVA(R2)  ;SAVE SECTOR AND TRACK ADDRESSES
7971 033672 142762 000002 000030  BICB     #2,$CODE(R2)  ;CHANGE WRITE TO CHECK
7972 033700 142762 000020 000002  BICB     #20,$COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
7973 033706 152762 000010 000002  BISB     #10,$COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
7974 033714 000207  2$:      RTS      PC          ;EXIT
7975
7976          ;*SPECIAL CONDITION ROUTINE
7977
7978 033716 005001  SC:      CLR      R1          ;START WITH DRIVE 0
7979 033720 012702 000001          MOV      #1,R2
7980 033724 105761 030556  1$:      TSTB     DRVSTA(R1) ;NONEXISTENT?
7981 033730 002030          BGE      SC2        ;NO--BRANCH
7982 033732 105761 030606          TSTB     DPINT(R1)   ;TRYING TO INIT THE DRIVE ?
7983 033736 001025          BNE      SC2        ;YES--BRANCH
7984 033740 005201          INC      R1         ;YES--MOVE TO THE NEXT DRIVE
7985 033742 106302          ASLB     R2         ;MORE DRIVES?
7986 033744 103367          BCC      1$        ;YES--BRANCH
7987 033746 012764 000040 000010  MOV      #BITS,RHCS2(R4) ;INIT THE SYSTEM
7988 033754 000410          BR      SC1A       ;TAKE ERROR EXIT
7989 033756 012701 000010  SC1:     MOV      #8.,R1   ;DO EIGHT DRIVES
7990 033762 005301  1$:      DEC      R1         ;NEXT DRIVE
7991 033764 002754          BLT      SC        ;BRANCH IF OUT OF DRIVES
7992 033766 105761 030546          TSTB     DRVACT(R1) ;IS THIS DRIVE IDLE?
7993 033772 001373          BNE      1$        ;BRANCH IF NO
7994 033774 104001          ERROR    1         ;ILLEGAL INTERRUPT
7995 033776 112764 177777 000016  SC1A:   MOVB     #-1,RHAS(R4) ;CLEAR ANY OUTSTANDING ATTN BITS
7996 034004 004737 036276          JSR      PC,SET.IE  ;GO SET INTERRUPT ENABLE
7997 034010 000207          RTS      PC
7998 034012 012701 000003  SC2:     MOV      #3,R1     ;READ RHAS UP TO THREE TIMES
7999 034016 116403 000016  1$:      MOVB     RHAS(R4),R3 ;READ "RHAS"
8000 034022 001011          BNE      2$        ;BRANCH IF ANY ATA BITS = 1
8001 034024 005301          DEC      R1         ;COUNT THIS READ
8002 034026 003373          BGT      1$        ;LOOP IF MORE READS ALLOWED

```

8003	034030	004037	035700		JSR	RD, RD.RP	; READ CONTROL AND STATUS REGISTER
8004	034034	000000			RHCS1		
8005	034036	033060			CIB		
8006	034040	106126			ROLB	(SP)+	; IS "IE"=1?
8007	034042	100345			BPL	SC1	; NO--TAKE ERROR EXIT
8008	034044	000207			RTS	PC	; YES--RETURN
8009	034046	005046		2\$:	CLR	-(SP)	; PROCESS ALL DRIVES THAT HAVE
8010	034050	110316			MOVB	R3, (SP)	; AN "ATA"=1
8011	034052	012703	000001		MOV	#1, R3	
8012	034056	005001			CLR	R1	
8013	034060	030316		SC4:	BIT	R3, (SP)	; ATA=1?
8014	034062	001005			BNE	SC5	; YES--BRANCH
8015	034064	005201		SC3:	INC	R1	; MOVE TO THE NEXT DRIVE
8016	034066	106303			ASLB	R3	
8017	034070	001373			BNE	SC4	; BRANCH IF MORE TO CHECK?
8018	034072	005726			TST	(SP)+	; CLEAN OFF THE STACK
8019	034074	000207			RTS	PC	; RETURN TO USER
8020	034076	105761	030606	SC5:	TSTB	DPINT(R1)	; INITIALIZING THE DRIVE ?
8021	034102	001402			BEQ	+.6	; BR IF NOT
8022	034104	000137	034766		JMP	SC13	; PROCESS THE DRIVE
8023	034110	105761	030616		TSTB	DPRQS(R1)	; PORT REQUEST OUTSTANDING ?
8024	034114	001402			BEQ	+.6	; BR IF NOT
8025	034116	000137	034766		JMP	SC13	; START THE OUTSTANDING COMMAND
8026	034122	023701	030700		CMP	DTUW, R1	; IS THIS DRIVE SETUP FOR I/O?
8027	034126	001003			BNE	1\$; NO---BRANCH
8028	034130	005726			TST	(SP)+	; YES---CLEAN OFF THE STACK (RHAS)
8029	034132	000137	033434		JMP	TD	; JUMP TO 'TRANSFER DONE'
8030	034136	105761	030556	1\$:	TSTB	DRVSTA(R1)	; CHECK THE DRIVE STATUS
8031	034142	003041			BGT	SC6	; BRANCH IF ONLINE
8032	034144	105761	030634		TSTB	ULDFLG(R1)	; UNLOAD IN PROGRESS?
8033	034150	003422			BLE	2\$; BRANCH IF NO
8034	034152	004737	037020		JSR	PC, GETREQ	; GET DPB POINTER
8035	034156	004737	036200		JSR	PC, SVRH11	; SAVE THE RH11/RP04 REGISTERS
8036	034162	004737	034716		JSR	PC, SC12	; SAVE RHDS1, RHER1, RHER2, AND RHER3
8037							; ALSO DO A DRIVE INIT (DRVINT)
8038	034166	105761	030556		TSTB	DRVSTA(R1)	; DID DRIVE COME ONLINE?
8039	034172	003413			BLE	3\$; NO---BRANCH
8040	034174	032737	040000	030536	BIT	#BIT14, RPERRS	; WAS THERE AN ERROR?
8041	034202	001002			BNE	+.6	; BR IF ERROR
8042	034204	000137	034626		JMP	SC11	; NO ERROR
8043	034210	013705	030540		MOV	RPERRS+2, R5	; YES -- PICKUP RHER1 AND
8044	034214	000463			BR	SC6A	; GO PROCESS THE ERROR
8045	034216	004737	034716	2\$:	JSR	PC, SC12	; SAVE RHDS1, RHER1, RHER2, AND RHER3
8046							; ALSO DO A DRVINT
8047	034222	116164	030702	000016	3\$:	MOVB	ATABIT(R1), RHAS(R4) ; CLEAR THE ATTENTION BIT
8048	034230	012714	000111		MOV	#111, (R4)	; ISSUE A 'DRIVE CLEAR'
8049	034234	012714	000113		MOV	#113, (R4)	; RELEASE THE DRIVE
8050	034240	011605			MOV	(SP), R5	; PICKUP (RHAS) BEFORE THE ERROR CALL
8051	034242	104005			ERROR	5	; REPORT THE ERROR
8052	034244	000707			BR	SC3	; GO CHECK FOR MORE ATA'S
8053	034246	105761	030546	SC6:	TSTB	DRVACT(R1)	; CHECK THE DRIVE ACTIVE INDICATOR
8054	034252	001552			BEQ	SC10	; BR IF IDLE
8055	034254	006301			ASL	R1	
8056	034256	012761	177777	030660	MOV	#-1, TIMER(R1)	; STOP THE TIMER
8057	034264	006201			ASR	R1	
8058	034266	004737	037020		JSR	PC, GETREQ	; GET THE DPB POINTER FROM THE QUEUE

8059	034272	010164	000010		MOV	R1,RHCS2(R4)	;SELECT DRIVE
8060	034276	004037	035700		JSR	RO,RD.RP	;READ THE RPO4'S STATUS REG.
8061	034302	000012			RHDS1		
8062	034304	034554			SCB		
8063	034306	011605			MOV	(SP),R5	;AND PUT IT IN R5
8064	034310	006126			ROL	(SP)+	;WAS THERE AN ERROR?
8065	034312	100407			BMI	1\$;BR IF ERROR
8066	034314	105761	030546		TSTB	DRVACT(R1)	;CHECK DRIVE'S STATE
8067	034320	003142			BGT	SC11	;BR IF DRIVE ACTIVE WITH ORDER
8068	034322	052762	100210	000016	BIS	#BIT15!BIT07!BIT03,16(R2)	;INFORM USER OF ERROR RECOVER COMPLETION
8069	034330	000472			BR	SC7	
8070	034332	004037	035700	1\$:	JSR	RO,RD.RP	;READ ERROR REGISTER #1
8071	034336	000014			RHER1		
8072	034340	034554			SCB		
8073	034342	012605			MOV	(SP)+,R5	;AND SAVE IT IN R5
8074	034344	004737	036200		JSR	PC,SVRH11	;SAVE RH11/RPO4 REGISTERS
8075	034350	012746	000111		MOV	#11,-(SP)	;ISSUE A DRIVE CLEAR
8076	034354	004037	036036		JSR	RO,WRT.RP	
8077	034360	000000			RHCS1		
8078	034362	034554			SCB		
8079	034364	006105		SC6A:	ROL	R5	;WAS "UNSAFE" CONDITION =1?
8080	034366	100406			BMI	1\$;BRANCH IF YES
8081	034370	005702			TST	R2	;ANYTHING IN QUEUE ?
8082	034372	001403			BEQ	.+10	;BR IF NOT
8083	034374	052762	100240	000016	BIS	#BIT15!BIT07!BIT05,16(R2)	;INFORM USER OF ERROR
8084	034402	000445			BR	SC7	
8085	034404	004037	035700	1\$:	JSR	RO,RD.RP	;READ DRIVE STATUS REG. #1
8086	034410	000012			RHDS1		
8087	034412	034554			SCB		
8088	034414	011605			MOV	(SP),R5	;SAVE RHDS1 IN R5
8089	034416	006126			ROL	(SP)+	; "ERR"=1?
8090	034420	100014			BPL	2\$;BR IF NO--UNSAFE CLEARED
8091	034422	112761	177777	030556	MOVB	#-1,DRVSTA(R1)	;DRIVE IS UNSAFE
8092	034430	004737	036200		JSR	PC,SVRH11	;SAVE RH11/RPO4 REGISTERS
8093	034434	116164	030702	000016	MOVB	ATABIT(R1),RHAS(R4)	;CLEAR ATTENTION BIT
8094	034442	052762	110000	000016	BIS	#BIT15!BIT12,16(R2)	;INFORM USER OF UNSAFE ERROR
8095	034450	000422			BR	SC7	
8096	034452	105705		2\$:	TSTB	R5	; "DRY" =1?
8097	034454	100415			BMI	3\$;BRANCH IF YES
8098	034456	112761	177777	030546	MOVB	#-1,DRVACT(R1)	;ACTIVE ERROR RECOVER
8099	034464	112761	000001	030556	MOVB	#1,DRVSTA(R1)	;ONLINE
8100	034472	006301			ASL	R1	
8101	034474	012761	072460	030660	MOV	#30000.,TIMER(R1)	;START 30 SECOND TIMER
8102	034502	006201			ASR	R1	
8103	034504	000137	034064		JMP	SC3	
8104	034510	052762	100220	000016	BIS	#BIT15!BIT07!BIT04,16(R2)	;INFORM USER OF ERROR
8105	034516	105061	030546	SC7:	CLRB	DRVACT(R1)	;DRIVE IS IDLE
8106	034522	004737	036724		JSR	PC,EMPTYQ	;DUMP THE QUEUE
8107	034526	012714	000113		MOV	#13,(R4)	;RELEASE THE DRIVE
8108	034532	105761	030634		TSTB	ULDFLG(R1)	;UNLOAD IN PROGRESS OR QUEUE?
8109	034536	001002			BNE	.+6	;BR IF NO
8110	034540	000137	034064		JMP	SC3	;NO UNLOAD
8111	034544	105061	030634		CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
8112	034550	000137	034064		JMP	SC3	
8113	034554	005726		SC8:	TST	(SP)+	;REMOVE (RHAS) FROM THE STACK
8114	034556	105761	030546		TSTB	DRVACT(R1)	;IS DRIVE IDLE?

8115	034562	001404			BEQ	1\$;YES--BRANCH
8116	034564	004737	037020		JSR	PC,GETREQ		;GET DPB POINTER
8117	034570	000137	032770		JMP	CI7		;PROCESS THE PARITY ERROR
8118	034574	000137	033006		JMP	CI7B		;PROCESS THE PARITY ERROR
8119	034600	011605			1\$:			
					SC10:	MOV	(SP),R5 ;PUT (RHAS) IN R5	
8120	034602	004737	034716		JSR	PC,SC12		;SAVE RHDS1, RHER1, RHER2, AND RHER3
8121	034606	116164	030702	000016	MOVB	ATABIT(R1),RHAS(R4)		;CLEAR ATTENTION BIT
8122	034614	012714	000113		MOV	#113,(R4)		;ISSUE A RELEASE
8123	034620	104002			ERROR	2		;REPORT ILLEGAL DRIVE INTERRUPT
8124	034622	000137	034064		JMP	SC3		;GO CHECK FOR MORE ATA'S
8125	034626	105761	030634		SC11:	TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS"?
8126	034632	003402			BLE	1\$;BRANCH IF NO
8127	034634	105061	030634		CLRB	ULDFLG(R1)		;CLEAR UNLOAD FLAG
8128	034640	105061	030546		1\$:	CLRB	DRVACT(R1)	;SET DRIVE IDLE
8129	034644	136137	030702	030630	BITB	ATABIT(R1),SRCHWT		;DOING A SEARCH OPERATION FOR ;AN I/O COMMAND?
8130								
8131	034652	001012			BNE	2\$;BRANCH IF YES
8132	034654	004737	037042		JSR	PC,POPQUE		;REMOVE REQUEST FROM QUEUE
8133	034660	052762	000200	000016	BIS	#BIT07,16(R2)		;SET "DONE" BIT
8134	034666	005737	030654		TST	SAVEFG		;SAVE THE RH11/RP04 REGISTERS?
8135	034672	100002			BPL	2\$;BRANCH IF NO
8136	034674	004737	036200		JSR	PC,SVRH11		;YES--SAVE ALL OF THE RH11/RP04 REG'S
8137	034700	116164	030702	000016	2\$:	MOVB	ATABIT(R1),RHAS(R4)	;CLEAR ATTENTION BIT
8138	034706	004737	031730		JSR	PC,OPT		;START A REQUEST
8139	034712	000137	034064		JMP	SC3		
8140	034716	010164	000010		SC12:	MOV	R1,RHCS2(R4)	;SELECT DRIVE
8141	034722	016437	000012	030536	MOV	RHDS1(R4),RPERRS		;SAVE THE FOUR REGISTERS THAT
8142	034730	016437	000014	030540	MOV	RHER1(R4),RPERRS+2		;WILL TELL US SOMETHING
8143	034736	016437	000040	030542	MOV	RHER2(R4),RPERRS+4		
8144	034744	016437	000042	030544	MOV	RHER3(R4),RPERRS+6		
8145	034752	004037	031160		JSR	RD,DRVINT		;INIT. THE STATE OF THE DRIVE
8146	034756	000401			BR	1\$;TAKE ERROR EXIT
8147	034760	000207			RTS	PC		;RETURN
8148	034762	005726			1\$:	TST	(SP)+	;POP PC OFF OF THE STACK
8149	034764	000673			BR	SC8		;PROCESS THE PARITY ERROR
8150	034766	006301			SC13:	ASL	R1	;SETUP TO ADDRESS WORDS
8151	034770	012761	177777	030660	MOV	#-1,TIMER(R1)		;STOP THE TIMER
8152	034776	006201			ASR	R1		
8153	035000	010164	000010		MOV	R1,RHCS2(R4)		;SELECT THE DRIVE
8154	035004	116164	030702	000016	MOVB	ATABIT(R1),RHAS(R4)		;CLEAR THE ATTENTION BIT
8155	035012	032714	004000		BIT	#BIT11,(R4)		;DRIVE AVAILABLE ?
8156	035016	001006			BNE	1\$;BR IF AVAILABLE
8157	035020	006301			ASL	R1		
8158	035022	012761	023420	030660	MOV	#10000.,TIMER(R1)		;START 10 SEC TIMER AGAIN
8159	035030	006201			ASR	R1		
8160	035032	000433			BR	3\$;EXIT
8161	035034	105761	030606		1\$:	TSTB	DPINT(R1)	;INITIALIZING THE DRIVE ?
8162	035040	001424			BEQ	2\$;BR IF NOT
8163	035042	105061	030606		CLRB	DPINT(R1)		;CLEAR THE INIT INDICATOR
8164	035046	004037	031160		JSR	RD,DRVINT		;GO INIT THE DRIVE
8165	035052	000240			NOP			;DUMMY PARITY ERROR RETURN
8166	035054	105761	030556		TSTB	DRVSTA(R1)		;DRIVE ONLINE ?
8167	035060	003014			BGT	2\$;BR IF YES -- START ORDER
8168	035062	005702			TST	R2		;QUEUE ENTRY FOR THE DRIVE
8169	035064	001416			BEQ	3\$;BR IF NOT
8170	035066	004737	037020		JSR	PC,GETREQ		;GET DPB ADDRESS

```

8171 035072 052762 140000 000016      BIS      #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
8172 035100 004737 036200      JSR      PC,SVRH11      ;SAVE THE REGISTERS
8173 035104 004737 036724      JSR      PC,EMPTYQ      ;EMPTY THE REQUEST QUEUE
8174 035110 000404      BR       3$
8175 035112 105061 030616      2$:     CLRB   DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
8176 035116 004737 031730      JSR      PC,OPT        ;START THE PENDING REQUEST
8177 035122 000137 034064      3$:     JMP     SC3        ;PROCESS OTHER DRIVES
8178
8179      ;*RPO4 TIMER ROUTINE
8180      ;*CALL
8181      ;*
8182      ;*     MOV     #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
8183      ;*     JSR     RD,RPTMR   ;CALL RPO4 TIME ROUTINE
8184 035126 005737 030632      RPTMR:  TST     ACTDRV    ;CHECK "ACTDRV & ACTSTR"
8185 035132 001032      BNE     4$            ;IF NON ZERO EXIT
8186 035134 112737 000001 030633      MOVVB   #1,ACTSTR    ;SET "ACTSTR"
8187 035142 004037 037104      JSR      RD,SAVR15    ;SAVE R1-R5
8188 035146 005001      CLR     R1            ;START WITH DRIVE 0
8189 035150 005003      CLR     R3
8190 035152 005763 030660      1$:     TST     TIMER(R3) ;IS THE TIMER RUNNING?
8191 035156 002407      BLT     2$            ;BRANCH IF NO
8192 035160 166663 000016 030660      SUB     16(SP),TIMER(R3) ;COUNT THE INTERVAL
8193 035166 003003      BGT     2$            ;BR IF NO SOFTWARE TIMEOUT
8194 035170 004037 035224      JSR      RD,STO       ;CALL SOFTWARE TIMEOUT ROUTINE
8195 035174 000405      BR      3$            ;GO TO THE EXIT
8196 035176 005201      2$:     INC     R1            ;MOVE TO NEXT DRIVE
8197 035200 005723      TST     (R3)+
8198 035202 022701 000010      CMP     #8.,R1        ;OUT OF DRIVES?
8199 035206 003361      BGT     1$            ;BRANCH IF NO
8200 035210 004037 037124      3$:     JSR      RD,GETR15 ;RESTORE R1-R5
8201 035214 105037 030633      CLRB   ACTSTR        ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
8202 035220 012616      4$:     MOV     (SP)+,(SP) ;ADJUST THE STACK
8203 035222 000200      RTS     RD            ;RETURN
8204
8205      ;*SOFTWARE TIMEOUT ROUTINE
8206      ;*
8207      ;*CALL:  STO
8208      ;*     MOV     #DRVNUM,R1 ;DRIVE NUMBER
8209      ;*     JSR     RD,STO     ;CALL--DRVACT MUST BE NONZERO
8210      ;*     RETURN
8211
8212 035224 013746 177776      STO:    MOV     @#PS,-(SP) ;SAVE THE PROCESSOR STATUS
8213 035230 013737 030720 177776      MOV     RPVEC+2,@#PS   ;SET THE "PS" TO RPO4 BR LEVEL
8214 035236 004037 037104      JSR      RD,SAVR15    ;SAVE R1-R5
8215 035242 013704 030714      MOV     RPADR,R4      ;GET ADDRESS OF "RHCS1"
8216 035246 010164 000010      MOV     R1,RHCS2(R4)  ;SELECT THE DRIVE
8217 035252 004037 035700      JSR      RD,RD.RP     ;READ "DRIVE STATUS REG"
8218 035256 000012      RHDS1
8219 035260 035572      STOS
8220 035262 105726      TSTB   (SP)+
8221 035264 100477      BMI    ST02
8222 035266 105761 030606      ST01:  TSTB   DPINT(R1)  ;TRYING TO INITIALIZE THE DRIVE ?
8223 035272 001074      BNE    ST02
8224 035274 105761 030616      TSTB   DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8225 035300 001071      BNE    ST02
8226 035302 013702 030626      MOV     TRNSWT,R2    ;PICKUP TRANSFER WAIT QUEUE

```

```

8227 035306 020137 030700          CMP      R1,DTUW          ;TRANSFER UNDERWAY ON THIS DRIVE?
8228 035312 001402          BEQ      1$              ;BRANCH IF YES
8229 035314 004737 037020          JSR      PC,GETREQ       ;GET DPB ADDRESS
8230 035320 052762 101000 000016 1$:  BIS      #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
8231 035326 004737 036200          JSR      PC,SVRH11       ;SAVE RH11/RP04 REGISTERS
8232 035332 012764 000040 000010  MOV      #BIT05,RHCS2(R4) ;"INIT" THE MASS BUS
8233 035340 105061 030546          CLRB     DRVACT(R1)      ;DRIVE IS IDLE
8234 035344 105061 030634          CLRB     ULDFLG(R1)     ;CLEAR THE UNLOAD FLAG
8235 035350 0050J1          CLR      R1              ;START WITH DRIVE 0
8236 035352 005003          CLR      R3
8237 035354 004037 031160          JSR      RD,DRVINT      ;INIT. THIS DRIVE
8238 035360 000504          BR       ST05            ;PARITY ERROR RETURN
8239 035362 105761 030546          TSTB     DRVACT(R1)     ;DRIVE IDLE BEFORE THE INIT.?
8240 035366 001414          BEQ      4$              ;YES--BRANCH
8241 035370 013702 030626          MOV      TRNSWT,R2      ;GET TRANSFER WAIT QUEUE
8242 035374 023701 030700          CMP      DTUW,R1        ;WAS THERE I/O ON THIS DRIVE?
8243 035400 001402          BEQ      3$              ;YES--BRANCH
8244 035402 004737 037020          JSR      PC,GETREQ       ;GET THE DPB POINTER FROM QUEUE
8245 035406 052762 100400 000016 3$:  BIS      #BIT15!BIT08,16(R2) ;INFORM USER OF INIT.
8246 035414 105061 030546          CLRB     DRVACT(R1)     ;SET DRIVE ACTIVE TO IDLE
8247 035420 105061 030634          CLRB     ULDFLG(R1)     ;NO UNLOAD
8248 035424 012763 177777 030660  MOV      #-1,TIMER(R3)   ;STOP THE TIMER
8249 035432 005723          TST      (R3)+          ;UPDATE THE INDEX
8250 035434 005201          INC      R1              ;INCREMENT THE DRIVE NUMBER
8251 035436 022701 000010          CMP      #8.,R1         ;LAST DRIVE BEEN CHECKED?
8252 035442 003344          BGT      2$              ;NO--LOOP
8253 035444 012737 177777 030700  MOV      #-1,DTUW       ;NO DATA TRANSFERS UNDERWAY
8254 035452 005037 030626          CLR      TRNSWT        ;CLEAR TRANSFER WAIT QUEUE
8255 035456 004737 036642          JSR      PC,CLRQUE      ;CLEAR ALL REQUEST QUEUES
8256 035462 000436          BR       ST04            ;EXIT
8257 035464 116405 000016          ST02:  MOVB     RHAS(R4),R5 ;READ ATTENTION REG
8258 035470 136105 030702          BITB     ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
8259 035474 001017          BNE      ST03            ;YES--BRANCH
8260 035476 105761 030606          TSTB     DPINT(R1)     ;TRYING TO INITIALIZE THE DRIVE ?
8261 035502 001036          BNE      ST06            ;BR IF YES - DRIVE NOT ONLINE
8262 035504 105761 030616          TSTB     DPRQS(R1)     ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
8263 035510 001052          BNE      ST07            ;BR IF YES - NO RESPONSE TO REQUEST
8264 035512 020137 030700          CMP      R1,DTUW       ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
8265 035516 001263          BNE      ST01            ;BR IF NO
8266 035520 004037 03570C          JSR      RD,RD.RP       ;YES--CHECK "RDY"
8267 035524 000000          RHCS1
8268 035526 035572          ST05
8269 035530 105726          TSTB     (SP)+
8270 035532 100255          BPL      ST01            ;BR IF "RDY"=0
8271 035534 005720          ST03:  TST      (R0)+          ;ADJUST FOR THE PROPER RETURN
8272 035536 105761 030606          TSTB     DPINT(R1)     ;INITIALIZING THE DRIVE ?
8273 035542 001003          BNE      1$              ;BR IF INIT PENDING
8274 035544 105761 030616          TSTB     DPRQS(R1)     ;PORT REQUEST PENDING ?
8275 035550 001403          BEQ      ST04            ;BR IF NOT
8276 035552 012763 177777 030660 1$:  MOV      #-1,TIMER(R3)   ;STOP THE TIMER
8277 035560 004037 037124          ST04:  JSR      RD,GETR15   ;RESTORE R1-R5
8278 035564 012637 177776          MOV      (SP)+,2#PS     ;RESTORE PROCESSOR STATUS
8279 035570 000200          RTS      RD              ;RETURN
8280 035572 004737 033060          ST05:  JSR      PC,C18     ;GO HANDLE THE PARITY ERROR
8281 035576 000770          BR       ST04
8282 035600 105061 030606          ST06:  CLRB     DPINT(R1)  ;CLEAR THE INITIALIZE INDICATOR
  
```

```

8283 035604 105061 030556          CLRB   DRVSTA(R1)      ;SET UNIT OFFLINE
8284 035610 012763 177777 030660    MOV    #-1,TIMER(R3) ;STOP THE TIMER
8285 035616 004737 037020          JSR    PC,GETREQ     ;GET THE DPB ADDRESS
8286 035622 005702          TST    R2            ;REQUEST IN QUEUE ?
8287 035624 001755          BEQ    ST04         ;BR IF NOT
8288 035626 052762 140000 000016    BIS    #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
8289 035634 000414          BR     ST08         ;FINISH
8290 035636 012763 177777 030660  ST07:  MOV    #-1,TIMER(R3) ;STOP THE TIMER
8291 035644 105061 030616          CLRB   DPRQS(R1)    ;CLEAR PORT REQUEST INDICATOR
8292 035650 004737 037020          JSR    PC,GETREQ     ;GET DPB ADDRESS
8293 035654 005702          TST    R2            ;QUEUE ENTRY FOR DRIVE ?
8294 035656 001403          BEQ    ST08         ;BR IF NONE
8295 035660 052762 100004 000016    BIS    #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
8296 035666 004737 036724          JSR    PC,EMPTYQ    ;CLEAR THE QUEUE FOR THE DRIVE
8297 035672 004737 036200          JSR    PC,SVRH11    ;SAVE THE REGISTERS
8298 035676 000730          BR     ST04         ;RETURN
8299
8300                                     ;*ROUTINE TO READ A RH11/RP04 REGISTER
8301                                     ;*CALL
8302                                     ;*
8303                                     ;*   JSR    RD, RD.RP      ;GO READ A REGISTER
8304                                     ;*   INDEX  ;REG. INDEX FROM BASE
8305                                     ;*   ERRADR  ;ERROR ADDRESS--PROCESS ERROR STARTING
8306                                     ;*           ;AT THIS ADDRESS
8307                                     ;*   RETURN ;CONTENTS OF REG. IS ON THE STACK
8308
8309 035700 013737 030712 036024  RD.RP:  MOV    MCPEMX, RD.RP2 ;MAX. RETRYS ALLOWED
8310 035706 011646          MOV    (SP), -(SP)  ;SAVE RD FOR RETURN
8311 035710 013737 030714 035724    MOV    RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
8312 035716 062037 035724          ADD    (RD)+, RD.ADR ;USING THE BASE AND THE INDEX
8313 035722 013727          RD.RP1: MOV    @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RP04
8314 035724 000000          RD.ADR: .WORD 0    ;ADDRESS IS FORMED HERE
8315 035726 000000          RD.WRD: .WORD 0    ;REG. CONTENTS PUT HERE
8316 035730 013766 035726 000002    MOV    RD.WRD, 2(SP) ;RETURN IT TO THE USER
8317 035736 017746 172752          MOV    @RPADR, -(SP) ;READ RHCS1
8318 035742 032716 020000          BIT    #BIT13, (SP) ;DID MCPE SET?
8319 035746 001002          BNE    1$         ;BRANCH IF YES
8320 035750 022620          CMP    (SP)+, (RD)+ ;ADJUST FOR RETURN
8321 035752 000200          RTS    RD         ;RETURN IF NO
8322 035754 104003          1$:  ERROR 3        ;REPORT "MCPE" ERROR
8323 035756 005737 030700          TST    DTUW        ;DATA TRANSFER UNDERWAY?
8324 035762 100405          BMI    2$         ;NO--BRANCH
8325 035764 032716 040000          BIT    #BIT14, (SP) ;NO--"TRE"=1?
8326 035770 001402          BEQ    2$         ;NO--BRANCH
8327 035772 005726          TST    (SP)+      ;YES--CLEAN OFF THE STACK AND
8328 035774 000415          BR     RD.RP3     ;TAKE THE FATAL ERROR EXIT
8329 035776 052716 040000          2$:  BIS    #BIT14, (SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
8330 036002 000316          SWAB   (SP)       ;POSITION BEFORE WRITING
8331 036004 013737 030714 036020    MOV    RPADR, 3$   ;FORM ADDRESS OF HIGH BYTE
8332 036012 005237 036020          INC    3$         ;
8333 036016 112637          MOVB   (SP)+, @ (PC)+ ;WRITE THE HIGH BYTE OF RHCS1
8334 036020 000000          3$:  .WORD 0        ;ADDRESS STORAGE
8335 036022 005327          DEC    (PC)+      ;EXCEEDED MAX. RETRYS
8336 036024 000003          RD.RP2: .WORD 3    ;
8337 036026 002335          BGE    RD.RP1     ;BRANCH IF NO
8338 036030 011000          RD.RP3: MOV    (RD), RD ;FATAL ERROR EXIT

```

```

8339 036032 012616          MOV      (SP)+,(SP)
8340 036034 000200          RTS      RO
8341
8342          ;*ROUTINE TO WRITE A RH11/RP04 REGISTER
8343          ;*CALL
8344          ;*
8345          ;*      MOV      DATA,-(SP)          ;DATA TO BE LOADED ON THE STACK
8346          ;*      JSR      RO,WRT.RP          ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
8347          ;*      INDEX   ERRADR          ;INDEX OF THE REGISTER TO BE LOADED
8348          ;*      ERRADR  RETURN          ;ADDRESS TO RETURN TO ON AN ERROR
8349          ;*
8350          ;*      RETURN
8351 036036 016637 000002 036124 WRT.RP: MOV      2(SP),WRT.WD          ;SAVE THE WORD TO WRITE
8352 036044 012616          MOV      (SP)+,(SP)          ;ADJUST THE STACK
8353 036046 012037 036126          MOV      (RO)+,WRT.AD          ;GET INDEX OF REGISTER TO BE WRITTEN
8354 036052 001020          BNE      2$
8355 036054 122737 000150 036124          CMPB    #150,WRT.WD          ;IS THE COMMAND FOR DATA TRANSFERS?
8356 036062 002414          BLT      2$
8357 036064 004037 035700          JSR      RO,RD.RP          ;YES---DON'T GET THE OLD A16&A17, & PSEL
8358 036070 000000          RHCS1
8359 036072 036110          1$
8360 036074 000316          SWAB    (SP)
8361 036076 042716 177770          BIC     #1C7,(SP)
8362 036102 112637 036125          MOVB   (SP)+,WRT.WD+1
8363 036106 000402          BR      2$
8364 036110 011000          1$: MOV      (RO),RO          ;TAKE THE ERROR EXIT
8365 036112 000200          RTS      RO
8366 036114 063737 030714 036126 2$: ADD     RPADR,WRT.AD          ;FORM THE ADDRESS OF THE DISK REG.
8367 036122 012737          MOV     (PC)+,@(PC)+          ;LOAD THE DESIRED REG.
8368 036124 000000          WRT.WD: .WORD 0          ;WORD TO WRITE GOES HERE
8369 036126 000000          WRT.AD: .WORD 0          ;ADDRESS IS FORMED HERE
8370 036130 004037 035700          JSR     RO,RD.RP          ;CHECK FOR PARITY ERROR ON WRITE
8371 036134 000014          RHER1
8372 036136 036170          2$
8373 036140 032726 000010          BIT     #BIT03,(SP)+
8374 036144 001413          BEQ     3$
8375 036146 016037 177776 036160          MOV     -2(RO),1$
8376 036154 004037 035700          JSR     RO,RD.RP          ;BRANCH IF "PAR=0"
8377 036160 000000          1$: .WORD 0          ;PICKUP THE INDEX
8378 036162 036170          2$          ;READ THE REG.
8379 036164 104004          ERROR 4          ;REG. INDEX
8380 036166 005726          TST     (SP)+          ;RETURN TO THIS ADDRESS ON ERROR
8381 036170 011000          2$: MOV     (RO),RO          ;REPORT THE PARITY ON WRITE ERROR
8382 036172 000200          RTS     RO          ;CLEAN OFF THE STACK
8383 036174 005720          3$: TST     (RO)+          ;TAKE THE "PARITY ON WRITE" ERROR EXIT
8384 036176 000200          RTS     RO          ;ADJUST FOR ERROR FREE EXIT
8385
8386          ;*ROUTINE TO SAVE THE RH11/RP04 REGISTERS AS PER DPB+14
8387          ;*CALL
8388          ;*
8389          ;*      MOV      #DPBNUM,R2          ;DPB POINTER TO R2
8390          ;*      JSR      PC,SVRH11          ;SAVE THE DRIVES REG'S
8391
8392 036200 004037 037104          SVRH11: JSR     RO,SAVR15          ;SAVE REG.'S R1-R5
8393 036204 005702          TST     R2          ;QUEUE ENTRY FOR THE DRIVE ?
8394 036206 001430          BEQ     4$          ;BR IF NONE
  
```

```

8395 036210 013704 030714      MOV      RPADR,R4
8396 036214 111264 000010      MOVVB   (R2),RHCS2(R4) ;SELECT DRIVE
8397 036220 016202 000014      MOV      14(R2),R2 ;GET THE ERROR TABLE POINTER
8398 036224 001421 ;EXIT IF 0
8399 036226 005003 ;COUNTER & POINTER
8400 036230 012705 000022      MOV      #RHDB,R5 ;PROBLEM REGISTER
8401 036234 ;1$: CMP      R3,R5 ;REACHED RHDB?
8402 036236 001005 ;BNE     2$ ;BR IF NO
8403 036240 105764 000010      TSTB   RHCS2(R4) ;CHECK "OR"
8404 036244 100402 ;BMI     2$ ;BRANCH IF RHDB CAN BE READ
8405 036246 005022 ;CLR     (R2)+ ;ELSE SAVE IT AS 0'S
8406 036250 000403 ;BR      3$
8407 036252 010446 ;2$: MOV      R4,-(SP) ;FORM RH11/RP04 ADDRESS THAT IS
8408 036254 060316 ;ADD     R3,(SP) ;TO BE READ
8409 036256 013622 ;MOV     2(SP)+,(R2)+ ;AND READ IT
8410 036260 005723 ;3$: TST     (R3)+ ;MOVE TO NEXT REG INDEX
8411 036262 020327 000046      CMP     R3,#RHEC2 ;DONE?
8412 036266 003762 ;BLE     1$ ;BRANCH IF NO
8413 036270 004037 037124      ;4$: JSR     R0,GETR15 ;GET REGISTERS R1-R5
8414 036274 000207 ;RTS     PC ;RETURN TO USER
8415
8416 ;*ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
8417 ;*CALL
8418 ;* MOV      #DRVNUM,R1 ;DRIVE NUMBER TO R1
8419 ;* JSR     PC,SET.IE ;SET "IE"
8420 ;* RETURN
8421
8422 036276 010446 ;SET.IE: MOV     R4,-(SP) ;SAVE R4
8423 036300 013704 030714      MOV     RPADR,R4 ;PICKUP ADDRESS OF RHCS1
8424 036304 010164 000010      MOV     R1,RHCS2(R4) ;SELECT DRIVE
8425 036310 011446 ;MOV     (R4),-(SP) ;READ RHCS1
8426 036312 052716 040000      BIS     #BIT14,(SP) ;SET THE "TRE" BIT OF THE WORD READ
8427 036316 000316 ;SWAB   (SP) ;ADJUST FOR DATO
8428 036320 112714 000100      MOVVB  #BIT06,(R4) ;SET "IE"
8429 036324 032764 010000 000010      BIT     #BIT12,RHCS2(R4) ;IS "NED"=1?
8430 036332 001002 ;BNE     1$ ;YES--CLEAR "TRE"
8431 036334 005726 ;TST     (SP)+ ;CLEAN OFF THE STACK
8432 036336 000402 ;BR      2$
8433 036340 112664 000001      ;1$: MOVVB  (SP)+,1(R4) ;CLEAR "TRE"
8434 036344 012604 ;2$: MOV     (SP)+,R4 ;RESTORE R4
8435 036346 000207 ;RTS     PC ;RETURN TO CALLER
8436
8437 ;*QUEUE COUNT
8438 036350 000 ;QCNT: .BYTE 0 ;DRIVE 0
8439 036351 000 ;.BYTE 0 ;DRIVE 1
8440 036352 000 ;.BYTE 0 ;DRIVE 2
8441 036353 000 ;.BYTE 0 ;DRIVE 3
8442 036354 000 ;.BYTE 0 ;DRIVE 4
8443 036355 000 ;.BYTE 0 ;DRIVE 5
8444 036356 000 ;.BYTE 0 ;DRIVE 6
8445 036357 000 ;.BYTE 0 ;DRIVE 7
8446
8447 ;QUEUE INPUT POINTERS
8448
8449 036360 036442 ;QINPT: .WORD QDRV0 ;DRIVE 0
8450 036362 036462 ;.WORD QDRV1 ;DRIVE 1

```

```

8451 036364 036502 .WORD QDRV2 ;DRIVE 2
8452 036366 036522 .WORD QDRV3 ;DRIVE 3
8453 036370 036542 .WORD QDRV4 ;DRIVE 4
8454 036372 036562 .WORD QDRV5 ;DRIVE 5
8455 036374 036602 .WORD QDRV6 ;DRIVE 6
8456 036376 036622 .WORD QDRV7 ;DRIVE 7
8457
8458 ;QUEUE OUTPUT POINTERS
8459
8460 036400 036442 QOUTPT: .WORD QDRV0 ;DRIVE 0
8461 036402 036462 .WORD QDRV1 ;DRIVE 1
8462 036404 036502 .WORD QDRV2 ;DRIVE 2
8463 036406 036522 .WORD QDRV3 ;DRIVE 3
8464 036410 036542 .WORD QDRV4 ;DRIVE 4
8465 036412 036562 .WORD QDRV5 ;DRIVE 5
8466 036414 036602 .WORD QDRV6 ;DRIVE 6
8467 036416 036622 .WORD QDRV7 ;DRIVE 7
8468
8469 036420 036442 QSTART: .WORD QDRV0 ;DRIVE 0 START ADDRESS
8470 036422 036462 QSTOP: .WORD QDRV1 ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
8471 036424 036502 .WORD QDRV2 ;STOP DRIVE 1--START DRIVE 2
8472 036426 036522 .WORD QDRV3 ;STOP DRIVE 2--START DRIVE 3
8473 036430 036542 .WORD QDRV4 ;STOP DRIVE 3--START DRIVE 4
8474 036432 036562 .WORD QDRV5 ;STOP DRIVE 4--START DRIVE 5
8475 036434 036602 .WORD QDRV6 ;STOP DRIVE 5--START DRIVE 6
8476 036436 036622 .WORD QDRV7 ;STOP DRIVE 6--START DRIVE 7
8477 036440 036642 .WORD QTERM ;STOP DRIVE 7
8478
8479 ;DRIVE REQUEST QUEUES
8480
8481 036442 000010 QDRV0: .BLKW 10
8482 036462 000010 QDRV1: .BLKW 10
8483 036502 000010 QDRV2: .BLKW 10
8484 036522 000010 QDRV3: .BLKW 10
8485 036542 000010 QDRV4: .BLKW 10
8486 036562 000010 QDRV5: .BLKW 10
8487 036602 000010 QDRV6: .BLKW 10
8488 036622 000010 QDRV7: .BLKW 10
8489 036642 000010 QTERM=.
8490
8491 ;*ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
8492 ;
8493 ;*CALL
8494 ;* JSR PC,CLRQUE
8495
8496 036642 004037 037104 CLRQUE: JSR R0,SAVR15 ;SAVE R1-R5
8497 036646 012702 036350 MOV #QCNT,R2 ;ZERO THE QUEUE COUNTS
8498 036652 005022 CLR (R2)+ ;DRIVES 0 & 1
8499 036654 005022 CLR (R2)+ ;DRIVES 2 & 3
8500 036656 005022 CLR (R2)+ ;DRIVES 4 & 5
8501 036660 005022 CLR (R2)+ ;DRIVES 6 & 7
8502 036662 012703 000010 MOV #8,R3 ;MOVE THE STARTING
8503 036666 012701 036420 MOV #QSTART,R1 ;ADDRESS OF THE QUEUE INTO
8504 036672 012122 1$: MOV (R1)+,(R2)+ ;THE QUEUE INPUT POINTER
8505 036674 005303 DEC R3
8506 036676 001375 BNE 1$
  
```



```

8507 036700 012703 000010      MOV      #8,R3      ;MOVE THE STARTING ADDRESS
8508 036704 012701 036420      MOV      #QSTART,R1 ;OF THE QUEUE INTO THE
8509 036710 012122 2$:      MOV      (R1)+,(R2)+ ;QUEUE OUTPUT POINTER
8510 036712 005303      DEC      R3
8511 036714 001375      BNE     2$
8512 036716 004037 037124      JSR     RD,GETR15   ;RESTORE R1-R5
8513 036722 000207      RTS     PC
8514
8515      ;*EMPTY THE QUEUE SPECIFIED BY R1
8516      ;*CALL
8517      ;*      MOV      DRVNUM,R1      ;DRIVE NUMBER TO R1
8518      ;*      JSR     PC,EMPTYQ
8519
8520
8521 036724 105061 036350      EMPTYQ: CLR      QCNT(R1)      ;CLEAR NUMBER OF ITEMS IN QUEUE
8522 036730 006301      ASL     R1
8523 036732 016161 036360 036400      MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
8524 036740 006201      ASR     R1
8525 036742 000207      RTS     PC
8526
8527      ;*ROUTINE TO PUT A REQUEST IN QUEUE
8528      ;*CALL
8529      ;*      MOV      #DRVNUM,R1      ;DRIVE NUMBER
8530      ;*      MOV      #DPB,R2      ;ADDRESS OF PARAMETER BLOCK
8531      ;*      JSR     RD,DRVQUE      ;GO PUT REQUEST IN QUEUE
8532      ;*      RETURN1      ;RETURN HERE IF QUEUE IS FULL
8533      ;*      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
8534
8535
8536 036744 122761 000010 036350      DRVQUE: CMP      #10,QCNT(R1)      ;IS QUEUE FULL?
8537 036752 001421      BEQ     2$          ;BR IF YES-TAKE RETURN1
8538 036754 105261 036350      INCB   QCNT(R1)      ;INCREMENT QUEUE COUNT
8539 036760 006301      ASL     R1
8540 036762 010271 036360      MOV     R2,QINPT(R1) ;PUT THIS REQUEST IN QUEUE
8541 036766 062761 000002 036360      ADD     #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
8542 036774 026161 036360 036422      CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
8543 037002 001003      BNE     1$          ;BRANCH IF NO
8544 037004 016161 036420 036360      MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
8545 037012 006201      1$:     ASR     R1
8546 037014 005720      TST     (R0)+      ;TAKE RETURN 2
8547 037016 000200      2$:     RTS     R0      ;RETURN TO USER
8548
8549      ;*ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
8550      ;*CALL
8551      ;*      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
8552      ;*      JSR     PC,GETREQ      ;GO GET THE REQUEST
8553      ;*      RETURN      ;R2="DPB" ADDRESS OF THE REQUEST
8554      ;*      ;R2=0 IF NO REQUEST IN QUEUE
8555
8556
8557 037020 005032      GETREQ: CLR      R2
8558 037022 105761 036350      TSTB   QCNT(R1)      ;IS THERE ANY REQUEST IN QUEUE?
8559 037026 001404      BEQ     2$          ;NO---BRANCH
8560 037030 006301      1$:     ASL     R1
8561 037032 017102 036400      MOV     QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
8562 037036 006201      ASR     R1
  
```

```

8563 037040 000207 2S: RTS PC ;RETURN TO USER
8564
8565 ;*ROUTINE TO "POP" THE REQUEST FROM QUEUE
8566
8567 ;*CALL
8568 ;* MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
8569 ;* JSR PC,POPQUE ;CALL TO REMOVE REQUEST
8570 ;* RETURN ;R2=ADDRESS OF DPB REMOVED
8571
8572 037042 105361 036350 POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
8573 037046 006301 ASL R1
8574 037050 017102 036400 MOV QOUTPT(R1),R2 ;GET THE "DPB" POINTER
8575 037054 062761 000002 036400 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
8576 037062 026161 036400 036422 CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
8577 037070 001003 BNE 1S ;NO--BRANCH TO EXIT
8578 037072 016161 036420 036400 MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
8579 037100 006201 1S: ASR R1
8580 037102 000207 RTS PC ;RETURN TO USER
8581
8582 ;*ROUTINES TO SAVE R0-R5 AND R1-R5
8583 ;*
8584 ;*CALL: SAVROS
8585 ;* JSR RO,SAVROS
8586 ;* RETURN ;R0-R5 IS ON THE STACK
8587 ;*
8588 ;*CALL: SAVR15
8589 ;* JSR R0,SAVR15
8590 ;* RETURN ;R1-R5 IS ON THE STACK
8591 ;*
8592 ;*UPON RETURN FROM SAVROS AND SAVR15 THE STACK WILL LOOK LIKE:
8593 ;*
8594 ;*+12 R0
8595 ;*+10 R1
8596 ;*+06 R2
8597 ;*+04 R3
8598 ;*+02 R4
8599 ;*TOP R5
8600
8601 037104 SAVROS: MOV R0,RO ;SAVE R0 BY THE JSR INST.
8602 037104 010146 SAVR15: MOV R1,-(SP) ;SAVE R1
8603 037106 010246 MOV R2,-(SP) ;SAVE R2
8604 037110 010346 MOV R3,-(SP) ;SAVE R3
8605 037112 010446 MOV R4,-(SP) ;SAVE R4
8606 037114 010546 MOV R5,-(SP) ;SAVE R5
8607 037116 016646 000012 MOV 12(SP),-(SP) ;GET R0
8608 037122 000200 RTS RO
8609
8610 ;*ROUTINES TO RESTORE R0-R5 AND R1-R5
8611 ;*
8612 ;*CALL: GETROS
8613 ;* JSR RO,GETROS
8614 ;* RETURN ;R0-R5 HAVE BEEN RESTORED
8615 ;*
8616 ;*CALL: GETR15
8617 ;* JSR R0,GETR15
8618 ;* RETURN ;R1-R5 HAVE BEEN RESTORED

```

```

0619
0620 037124 011666 000014      GETR15: MOV      (SP),14(SP)      ;POSITION RD
0621 037130 005726      GETR05: TST      (SP)+          ;POP RD OF THE JSR OFF OF THE STACK
0622 037132 012605          MOV      (SP)+,R5              ;RESTORE R5
0623 037134 012604          MOV      (SP)+,R4              ;RESTORE R4
0624 037136 012603          MOV      (SP)+,R3              ;RESTORE R3
0625 037140 012602          MOV      (SP)+,R2              ;RESTORE R2
0626 037142 012601          MOV      (SP)+,R1              ;RESTORE R1
0627 037144 000200          RTS      RD                    ;RESTORE RD
0628
0629
0630
0631      ;*****
0632      .SBTTL DATA, CONTROL, & STATUS BLOCKS
0633
0634      ;*****
0635      ;BLOCK LOCATION EQUATE STATEMENTS
0636
0637      000001      $FMT      =          1          ;FMT,HCI,ECI OR OFFSET CODE
0638      000002      $COMND   =          $FMT+1      ;OPERATION CODE
0639      000003      $PSEL    =          $COMND+1     ;PORT SELECT & BITS A16, A17
0640      000004      $WRDM    =          $PSEL+1      ;WORD COUNT (2'S COMP)
0641      000006      $BUF     =          $WRDM+2      ;BUFFER ADDR OR REGISTER TABLE POINTER
0642      000010      $SEC     =          $BUF+2       ;SECTOR ADDRESS OR 1ST REG ADDR
0643      000011      $TRK    =          $SEC+1       ;TRACK ADDRESS OF LAST REG ADDR
0644      000012      $CYL    =          $TRK+1       ;CYLINDER ADDR
0645      000014      $REG     =          $CYL+2      ;REGISTER STORAGE (IF ERROR)
0646      000016      $STATUS  =          $REG+2      ;STATUS WORD (SET BY DRIVER)
0647      000020      $WRDL   =          $STATUS+2    ;WORD COUNT (NOT 2'S COMP)
0648      000022      $RETRY  =          $WRDL+2     ;LOWER BYTE = RETRY COUNT, UPPER BYTE = RETRY LIMIT
0649      000024      $MASK   =          $RETRY+2    ;ERROR CHECK MASK
0650      000026      $$SEC   =          $MASK+2     ;SECTOR SIZE FOR CURRENT OPERATION
0651      000030      $CODE   =          $$SEC+2     ;PRESENT COMMAND SELECTION CODE
0652      000031      $PACK   =          $CODE+1     ;WRITE DATA PACK INDICATOR
0653      000032      $PREVO  =          $PACK+1     ;PREVIOUS COMMAND SELECTION CODE
0654      000034      $PATTC  =          $PREVO+2    ;PATTERN CODE
0655      000036      $PREVA  =          $PATTC+2    ;PREVIOUS ADDRESS - TRACK, SECTOR, CYLINDER
0656      000042      $OPERC  =          $PREVA+4    ;OPERATION COUNT
0657      000046      $POSIT  =          $OPERC+4    ;SEEK COUNT
0658      000052      $TRANS  =          $POSIT+4    ;TOTAL BITS XFERED COUNT (R & W)
0659      000056      $READ   =          $TRANS+4   ;TOTAL BITS READ COUNT
0660      000062      $TOTAL  =          $READ+4     ;TOTAL ERRORS (ALL TYPES) COUNT
0661      000064      $$SOFT  =          $TOTAL+2   ;'SOFT' ERROR COUNT
0662      000066      $HARD   =          $$SOFT+2   ;'HARD' ERROR COUNT
0663      000070      $SKI    =          $HARD+2    ;'SKI' OR 'OCYL' ERROR COUNT
0664      000072      $MISPO  =          $SKI+2     ;PROG DETECTED MISPOSITIONING ERROR S COUNT
0665      000074      $PASSC  =          $MISPO+2   ;PASS COUNTER
0666      000076      $RSV    =          $PASSC+2   ;STORE THE RANDOM NUMBERS HERE
0667      000102      $NCODE  =          $RSV+4     ;NEXT OPERATION
0668      000103      $NPATC  =          $NCODE+1   ;NEXT PATTERN
0669      000104      $NSEC   =          $NPATC+1   ;NEXT SECTOR
0670      000105      $NTRK   =          $NSEC+1   ;NEXT TRACK
0671      000106      $NCYL   =          $NTRK+1   ;NEXT CYLINDER
0672      000110      $NWRDL  =          $NICYL+2   ;NEXT BUFFER SIZE
0673      000112      $NEXT   =          $NWRDL+2   ;PARAMETER SELECTION INDICATOR
0674      000114      $BDSEC   =          $NEXT+2    ;BAD SECTOR STORAGE TABLE
  
```

8675	000154	\$DRVID	=	\$B0SEC+40	:RPO4 REGISTER STORAGE
8676	000164	\$RHCS1	=	\$DRVID+10	:RPO4 REGISTER STORAGE
8677	000166	\$RHWC	=	\$RHCS1+2	
8678	000170	\$RHBA	=	\$RHWC+2	
8679	000172	\$RHDA	=	\$RHBA+2	
8680	000174	\$RHCS2	=	\$RHDA+2	
8681	000176	\$RHDS1	=	\$RHCS2+2	
8682	000200	\$RHER1	=	\$RHDS1+2	
8683	000202	\$RHAS	=	\$RHER1+2	
8684	000204	\$RHLA	=	\$RHAS+2	
8685	000206	\$RHDB	=	\$RHLA+2	
8686	000210	\$RHMR	=	\$RHDB+2	
8687	000212	\$RHDT	=	\$RHMR+2	
8688	000214	\$RHSN	=	\$RHDT+2	
8689	000216	\$RHOF	=	\$RHSN+2	
8690	000220	\$RHCA	=	\$RHOF+2	
8691	000222	\$RHCC	=	\$RHCA+2	
8692	000224	\$RHER2	=	\$RHCC+2	
8693	000226	\$RHER3	=	\$RHER2+2	
8694	000230	\$RHEC1	=	\$RHER3+2	
8695	000232	\$RHEC2	=	\$RHEC1+2	

;BLOCK FOR DRIVE 0

8700	037146	000	000	BLKO:	.BYTE	0,0	;DRIVE NUMBER
8701	037150	000000			.WORD	00	
8702	037152	000000			.WORD	00	
8703	037154	000000			.WORD	00	
8704	037156	000000			.WORD	00	
8705	037160	000000			.WORD	0	
8706	037162	037332			.WORD	+150	
8707	037164	000000			.WORD	00	
8708	037166	000000			.WORD	00	
8709	037170	000000			.WORD	00	
8710	037172	000000			.WORD	00	
8711	037174	000000			.WORD	00	
8712	037176	000000			.WORD	00	
8713	037200	000000			.WORD	00	
8714	037202	000000			.WORD	00	
8715	037204	000000			.WORD	00	
8716	037206	000000			.WORD	00	
8717	037210	000000			.WORD	00	
8718	037212	000000			.WORD	00	
8719	037214	000000			.WORD	00	
8720	037216	000000			.WORD	00	
8721	037220	000000			.WORD	00	
8722	037222	000000			.WORD	00	
8723	037224	000000			.WORD	00	
8724	037226	000000			.WORD	00	
8725	037230	000000			.WORD	00	
8726	037232	000000			.WORD	00	
8727	037234	000000			.WORD	00	
8728	037236	000000			.WORD	00	
8729	037240	000000			.WORD	00	
8730	037242	000000			.WORD	0	

8731	037244	000000	.WORD	0
8732	037246	000000	.WORD	0
8733	037250	000000	.WORD	0
8734	037252	000000	.WORD	0
8735	037254	000000	.WORD	0
8736	037256	000000	.WORD	0
8737	037260	000000	.WORD	0
8738	037262	000000	.WORD	0
8739	037264	000000	.WORD	0
8740	037266	000000	.WORD	0
8741	037270	000000	.WORD	0
8742	037272	000000	.WORD	0
8743	037274	000000	.WORD	0
8744	037276	000000	.WORD	0
8745	037300	000000	.WORD	0
8746	037302	000000	.WORD	0
8747	037304	000000	.WORD	0
8748	037306	000000	.WORD	0
8749	037310	000000	.WORD	0
8750	037312	000000	.WORD	0
8751	037314	000000	.WORD	0
8752	037316	000000	.WORD	0
8753	037320	000000	.WORD	0
8754	037322	000000	.WORD	0
8755	037324	000000	.WORD	0
8756	037326	000000	.WORD	0
8757	037330	000000	.WORD	0
8758	037332	000000	.WORD	0
8759	037334	000000	.WORD	0
8760	037336	000000	.WORD	0
8761	037340	000000	.WORD	0
8762	037342	000000	.WORD	0
8763	037344	000000	.WORD	0
8764	037346	000000	.WORD	0
8765	037350	000000	.WORD	0
8766	037352	000000	.WORD	0
8767	037354	000000	.WORD	0
8768	037356	000000	.WORD	0
8769	037360	000000	.WORD	0
8770	037362	000000	.WORD	0
8771	037364	000000	.WORD	0
8772	037366	000000	.WORD	0
8773	037370	000000	.WORD	0
8774	037372	000000	.WORD	0
8775	037374	000000	.WORD	0
8776	037376	000000	.WORD	0
8777	037400	000000	.WORD	0
8778	037402	000000	.WORD	0

;BLOCK FOR DRIVE 1

8781						
8782	037404	001	000	BLK1:	.BYTE	1,0 ;DRIVE NUMBER
8783	037406	000000			.WORD	0
8784	037410	000000			.WORD	0
8785	037412	000000			.WORD	0
8786	037414	000000			.WORD	0

8787	037416	000000	.WORD	0
8788	037420	037570	.WORD	.+150
8789	037422	000000	.WORD	0
8790	037424	000000	.WORD	0
8791	037426	000000	.WORD	0
8792	037430	000000	.WORD	0
8793	037432	000000	.WORD	0
8794	037434	000000	.WORD	0
8795	037436	000000	.WORD	0
8796	037440	000000	.WORD	0
8797	037442	000000	.WORD	0
8798	037444	000000	.WORD	0
8799	037446	000000	.WORD	0
8800	037450	000000	.WORD	0
8801	037452	000000	.WORD	0
8802	037454	000000	.WORD	0
8803	037456	000000	.WORD	0
8804	037460	000000	.WORD	0
8805	037462	000000	.WORD	0
8806	037464	000000	.WORD	0
8807	037466	000000	.WORD	0
8808	037470	000000	.WORD	0
8809	037472	000000	.WORD	0
8810	037474	000000	.WORD	0
8811	037476	000000	.WORD	0
8812	037500	000000	.WORD	0
8813	037502	000000	.WORD	0
8814	037504	000000	.WORD	0
8815	037506	000000	.WORD	0
8816	037510	000000	.WORD	0
8817	037512	000000	.WORD	0
8818	037514	000000	.WORD	0
8819	037516	000000	.WORD	0
8820	037520	000000	.WORD	0
8821	037522	000000	.WORD	0
8822	037524	000000	.WORD	0
8823	037526	000000	.WORD	0
8824	037530	000000	.WORD	0
8825	037532	000000	.WORD	0
8826	037534	000000	.WORD	0
8827	037536	000000	.WORD	0
8828	037540	000000	.WORD	0
8829	037542	000000	.WORD	0
8830	037544	000000	.WORD	0
8831	037546	000000	.WORD	0
8832	037550	000000	.WORD	0
8833	037552	000000	.WORD	0
8834	037554	000000	.WORD	0
8835	037556	000000	.WORD	0
8836	037560	000000	.WORD	0
8837	037562	000000	.WORD	0
8838	037564	000000	.WORD	0
8839	037566	000000	.WORD	0
8840	037570	000000	.WORD	0
8841	037572	000000	.WORD	0
8842	037574	000000	.WORD	0

8843	037576	000000	.WORD	0
8844	037600	000000	.WORD	0
8845	037602	000000	.WORD	0
8846	037604	000000	.WORD	0
8847	037606	000000	.WORD	0
8848	037610	000000	.WORD	0
8849	037612	000000	.WORD	0
8850	037614	000000	.WORD	0
8851	037616	000000	.WORD	0
8852	037620	000000	.WORD	0
8853	037622	000000	.WORD	0
8854	037624	000000	.WORD	0
8855	037626	000000	.WORD	0
8856	037630	000000	.WORD	0
8857	037632	000000	.WORD	0
8858	037634	000000	.WORD	0
8859	037636	000000	.WORD	0
8860	037640	000000	.WORD	0

;BLOCK FOR DRIVE 2

8861					
8862					
8863					
8864	037642	002	000	BLK2: .BYTE 2,0	;DRIVE NUMBER
8865	037644	000000		.WORD	0
8866	037646	000000		.WORD	0
8867	037650	000000		.WORD	0
8868	037652	000000		.WORD	0
8869	037654	000000		.WORD	0
8870	037656	040026		.WORD	+150
8871	037660	000000		.WORD	0
8872	037662	000000		.WORD	0
8873	037664	000000		.WORD	0
8874	037666	000000		.WORD	0
8875	037670	000000		.WORD	0
8876	037672	000000		.WORD	0
8877	037674	000000		.WORD	0
8878	037676	000000		.WORD	0
8879	037700	000000		.WORD	0
8880	037702	000000		.WORD	0
8881	037704	000000		.WORD	0
8882	037706	000000		.WORD	0
8883	037710	000000		.WORD	0
8884	037712	000000		.WORD	0
8885	037714	000000		.WORD	0
8886	037716	000000		.WORD	0
8887	037720	000000		.WORD	0
8888	037722	000000		.WORD	0
8889	037724	000000		.WORD	0
8890	037726	000000		.WORD	0
8891	037730	000000		.WORD	0
8892	037732	000000		.WORD	0
8893	037734	000000		.WORD	0
8894	037736	000000		.WORD	0
8895	037740	000000		.WORD	0
8896	037742	000000		.WORD	0
8897	037744	000000		.WORD	0
8898	037746	000000		.WORD	0

8899	037750	000000	.WORD	0
8900	037752	000000	.WORD	0
8901	037754	000000	.WORD	0
8902	037756	000000	.WORD	0
8903	037760	000000	.WORD	0
8904	037762	000000	.WORD	0
8905	037764	000000	.WORD	0
8906	037766	000000	.WORD	0
8907	037770	000000	.WORD	0
8908	037772	000000	.WORD	0
8909	037774	000000	.WORD	0
8910	037776	000000	.WORD	0
8911	040000	000000	.WORD	0
8912	040002	000000	.WORD	0
8913	040004	000000	.WORD	0
8914	040006	000000	.WORD	0
8915	040010	000000	.WORD	0
8916	040012	000000	.WORD	0
8917	040014	000000	.WORD	0
8918	040016	000000	.WORD	0
8919	040020	000000	.WORD	0
8920	040022	000000	.WORD	0
8921	040024	000000	.WORD	0
8922	040026	000000	.WORD	0
8923	040030	000000	.WORD	0
8924	040032	000000	.WORD	0
8925	040034	000000	.WORD	0
8926	040036	000000	.WORD	0
8927	040040	000000	.WORD	0
8928	040042	000000	.WORD	0
8929	040044	000000	.WORD	0
8930	040046	000000	.WORD	0
8931	040050	000000	.WORD	0
8932	040052	000000	.WORD	0
8933	040054	000000	.WORD	0
8934	040056	000000	.WORD	0
8935	040060	000000	.WORD	0
8936	040062	000000	.WORD	0
8937	040064	000000	.WORD	0
8938	040066	000000	.WORD	0
8939	040070	000000	.WORD	0
8940	040072	000000	.WORD	0
8941	040074	000000	.WORD	0
8942	040076	000000	.WORD	0

;BLOCK FOR DRIVE 3

8943					
8944					
8945					
8946	040100	003	000	BLK3: .BYTE 3,0	;DRIVE NUMBER
8947	040102	000000		.WORD 0	
8948	040104	000000		.WORD 0	
8949	040106	000000		.WORD 0	
8950	040110	000000		.WORD 0	
8951	040112	000000		.WORD 0	
8952	040114	040264		.WORD +150	
8953	040116	000000		.WORD 0	
8954	040120	000000		.WORD 0	

8955	040122	000000	.WORD	0
8956	040124	000000	.WORD	0
8957	040126	000000	.WORD	0
8958	040130	000000	.WORD	0
8959	040132	000000	.WORD	0
8960	040134	000000	.WORD	0
8961	040136	000000	.WORD	0
8962	040140	000000	.WORD	0
8963	040142	000000	.WORD	0
8964	040144	000000	.WORD	0
8965	040146	000000	.WORD	0
8966	040150	000000	.WORD	0
8967	040152	000000	.WORD	0
8968	040154	000000	.WORD	0
8969	040156	000000	.WORD	0
8970	040160	000000	.WORD	0
8971	040162	000000	.WORD	0
8972	040164	000000	.WORD	0
8973	040166	000000	.WORD	0
8974	040170	000000	.WORD	0
8975	040172	000000	.WORD	0
8976	040174	000000	.WORD	0
8977	040176	000000	.WORD	0
8978	040200	000000	.WORD	0
8979	040202	000000	.WORD	0
8980	040204	000000	.WORD	0
8981	040206	000000	.WORD	0
8982	040210	000000	.WORD	0
8983	040212	000000	.WORD	0
8984	040214	000000	.WORD	0
8985	040216	000000	.WORD	0
8986	040220	000000	.WORD	0
8987	040222	000000	.WORD	0
8988	040224	000000	.WORD	0
8989	040226	000000	.WORD	0
8990	040230	000000	.WORD	0
8991	040232	000000	.WORD	0
8992	040234	000000	.WORD	0
8993	040236	000000	.WORD	0
8994	040240	000000	.WORD	0
8995	040242	000000	.WORD	0
8996	040244	000000	.WORD	0
8997	040246	000000	.WORD	0
8998	040250	000000	.WORD	0
8999	040252	000000	.WORD	0
9000	040254	000000	.WORD	0
9001	040256	000000	.WORD	0
9002	040260	000000	.WORD	0
9003	040262	000000	.WORD	0
9004	040264	000000	.WORD	0
9005	040266	000000	.WORD	0
9006	040270	000000	.WORD	0
9007	040272	000000	.WORD	0
9008	040274	000000	.WORD	0
9009	040276	000000	.WORD	0
9010	040300	000000	.WORD	0

9011	040302	000000	.WORD	0
9012	040304	000000	.WORD	0
9013	040306	000000	.WORD	0
9014	040310	000000	.WORD	0
9015	040312	000000	.WORD	0
9016	040314	000000	.WORD	0
9017	040316	000000	.WORD	0
9018	040320	000000	.WORD	0
9019	040322	000000	.WORD	0
9020	040324	000000	.WORD	0
9021	040326	000000	.WORD	0
9022	040330	000000	.WORD	0
9023	040332	000000	.WORD	0
9024	040334	000000	.WORD	0

;BLOCK FOR DRIVE 4

9025					
9026					
9027					
9028	040336	004	000	BLK4: .BYTE	4,0 ;DRIVE NUMBER
9029	040340	000000		.WORD	0
9030	040342	000000		.WORD	0
9031	040344	000000		.WORD	0
9032	040346	000000		.WORD	0
9033	040350	000000		.WORD	0
9034	040352	040522		.WORD	+150
9035	040354	000000		.WORD	0
9036	040356	000000		.WORD	0
9037	040360	000000		.WORD	0
9038	040362	000000		.WORD	0
9039	040364	000000		.WORD	0
9040	040366	000000		.WORD	0
9041	040370	000000		.WORD	0
9042	040372	000000		.WORD	0
9043	040374	000000		.WORD	0
9044	040376	000000		.WORD	0
9045	040400	000000		.WORD	0
9046	040402	000000		.WORD	0
9047	040404	000000		.WORD	0
9048	040406	000000		.WORD	0
9049	040410	000000		.WORD	0
9050	040412	000000		.WORD	0
9051	040414	000000		.WORD	0
9052	040416	000000		.WORD	0
9053	040420	000000		.WORD	0
9054	040422	000000		.WORD	0
9055	040424	000000		.WORD	0
9056	040426	000000		.WORD	0
9057	040430	000000		.WORD	0
9058	040432	000000		.WORD	0
9059	040434	000000		.WORD	0
9060	040436	000000		.WORD	0
9061	040440	000000		.WORD	0
9062	040442	000000		.WORD	0
9063	040444	000000		.WORD	0
9064	040446	000000		.WORD	0
9065	040450	000000		.WORD	0
9066	040452	000000		.WORD	0

9067	040454	000000	.WORD	0
9068	040456	000000	.WORD	0
9069	040460	000000	.WORD	0
9070	040462	000000	.WORD	0
9071	040464	000000	.WORD	0
9072	040466	000000	.WORD	0
9073	040470	000000	.WORD	0
9074	040472	000000	.WORD	0
9075	040474	000000	.WORD	0
9076	040476	000000	.WORD	0
9077	040500	000000	.WORD	0
9078	040502	000000	.WORD	0
9079	040504	000000	.WORD	0
9080	040506	000000	.WORD	0
9081	040510	000000	.WORD	0
9082	040512	000000	.WORD	0
9083	040514	000000	.WORD	0
9084	040516	000000	.WORD	0
9085	040520	000000	.WORD	0
9086	040522	000000	.WORD	0
9087	040524	000000	.WORD	0
9088	040526	000000	.WORD	0
9089	040530	000000	.WORD	0
9090	040532	000000	.WORD	0
9091	040534	000000	.WORD	0
9092	040536	000000	.WORD	0
9093	040540	000000	.WORD	0
9094	040542	000000	.WORD	0
9095	040544	000000	.WORD	0
9096	040546	000000	.WORD	0
9097	040550	000000	.WORD	0
9098	040552	000000	.WORD	0
9099	040554	000000	.WORD	0
9100	040556	000000	.WORD	0
9101	040560	000000	.WORD	0
9102	040562	000000	.WORD	0
9103	040564	000000	.WORD	0
9104	040566	000000	.WORD	0
9105	040570	000000	.WORD	0
9106	040572	000000	.WORD	0

;BLOCK FOR DRIVE 5

9107						
9108						
9109						
9110	040574	005	000	BLKS:	.BYTE	5,0 ;DRIVE NUMBER
9111	040576	000000			.WORD	0
9112	040600	000000			.WORD	0
9113	040602	000000			.WORD	0
9114	040604	000000			.WORD	0
9115	040606	000000			.WORD	0
9116	040610	040760			.WORD	+150
9117	040612	000000			.WORD	0
9118	040614	000000			.WORD	0
9119	040616	000000			.WORD	0
9120	040620	000000			.WORD	0
9121	040622	000000			.WORD	0
9122	040624	000000			.WORD	0

9123	040626	000000	.WORD	0
9124	040630	000000	.WORD	0
9125	040632	000000	.WORD	0
9126	040634	000000	.WORD	0
9127	040636	000000	.WORD	0
9128	040640	000000	.WORD	0
9129	040642	000000	.WORD	0
9130	040644	000000	.WORD	0
9131	040646	000000	.WORD	0
9132	040650	000000	.WORD	0
9133	040652	000000	.WORD	0
9134	040654	000000	.WORD	0
9135	040656	000000	.WORD	0
9136	040660	000000	.WORD	0
9137	040662	000000	.WORD	0
9138	040664	000000	.WORD	0
9139	040666	000000	.WORD	0
9140	040670	000000	.WORD	0
9141	040672	000000	.WORD	0
9142	040674	000000	.WORD	0
9143	040676	000000	.WORD	0
9144	040700	000000	.WORD	0
9145	040702	000000	.WORD	0
9146	040704	000000	.WORD	0
9147	040706	000000	.WORD	0
9148	040710	000000	.WORD	0
9149	040712	000000	.WORD	0
9150	040714	000000	.WORD	0
9151	040716	000000	.WORD	0
9152	040720	000000	.WORD	0
9153	040722	000000	.WORD	0
9154	040724	000000	.WORD	0
9155	040726	000000	.WORD	0
9156	040730	000000	.WORD	0
9157	040732	000000	.WORD	0
9158	040734	000000	.WORD	0
9159	040736	000000	.WORD	0
9160	040740	000000	.WORD	0
9161	040742	000000	.WORD	0
9162	040744	000000	.WORD	0
9163	040746	000000	.WORD	0
9164	040750	000000	.WORD	0
9165	040752	000000	.WORD	0
9166	040754	000000	.WORD	0
9167	040756	000000	.WORD	0
9168	040760	000000	.WORD	0
9169	040762	000000	.WORD	0
9170	040764	000000	.WORD	0
9171	040766	000000	.WORD	0
9172	040770	000000	.WORD	0
9173	040772	000000	.WORD	0
9174	040774	000000	.WORD	0
9175	040776	000000	.WORD	0
9176	041000	000000	.WORD	0
9177	041002	000000	.WORD	0
9178	041004	000000	.WORD	0

9179	041006	000000	.WORD	0
9180	041010	000000	.WORD	0
9181	041012	000000	.WORD	0
9182	041014	000000	.WORD	0
9183	041016	000000	.WORD	0
9184	041020	000000	.WORD	0
9185	041022	000000	.WORD	0
9186	041024	000000	.WORD	0
9187	041026	000000	.WORD	0
9188	041030	000000	.WORD	0

;BLOCK FOR DRIVE 6

9191						
9192	041032	006	000	BLK6: .BYTE	6,0	;DRIVE NUMBER
9193	041034	000000		.WORD	0	
9194	041036	000000		.WORD	0	
9195	041040	000000		.WORD	0	
9196	041042	000000		.WORD	0	
9197	041044	000000		.WORD	0	
9198	041046	041216		.WORD	+150	
9199	041050	000000		.WORD	0	
9200	041052	000000		.WORD	0	
9201	041054	000000		.WORD	0	
9202	041056	000000		.WORD	0	
9203	041060	000000		.WORD	0	
9204	041062	000000		.WORD	0	
9205	041064	000000		.WORD	0	
9206	041066	000000		.WORD	0	
9207	041070	000000		.WORD	0	
9208	041072	000000		.WORD	0	
9209	041074	000000		.WORD	0	
9210	041076	000000		.WORD	0	
9211	041100	000000		.WORD	0	
9212	041102	000000		.WORD	0	
9213	041104	000000		.WORD	0	
9214	041106	000000		.WORD	0	
9215	041110	000000		.WORD	0	
9216	041112	000000		.WORD	0	
9217	041114	000000		.WORD	0	
9218	041116	000000		.WORD	0	
9219	041120	000000		.WORD	0	
9220	041122	000000		.WORD	0	
9221	041124	000000		.WORD	0	
9222	041126	000000		.WORD	0	
9223	041130	000000		.WORD	0	
9224	041132	000000		.WORD	0	
9225	041134	000000		.WORD	0	
9226	041136	000000		.WORD	0	
9227	041140	000000		.WORD	0	
9228	041142	000000		.WORD	0	
9229	041144	000000		.WORD	0	
9230	041146	000000		.WORD	0	
9231	041150	000000		.WORD	0	
9232	041152	000000		.WORD	0	
9233	041154	000000		.WORD	0	
9234	041156	000000		.WORD	0	

9291	041332	000000	.WORD	0
9292	041334	000000	.WORD	0
9293	041336	000000	.WORD	0
9294	041340	000000	.WORD	0
9295	041342	000000	.WORD	0
9296	041344	000000	.WORD	0
9297	041346	000000	.WORD	0
9298	041350	000000	.WORD	0
9299	041352	000000	.WORD	0
9300	041354	000000	.WORD	0
9301	041356	000000	.WORD	0
9302	041360	000000	.WORD	0
9303	041362	000000	.WORD	0
9304	041364	000000	.WORD	0
9305	041366	000000	.WORD	0
9306	041370	000000	.WORD	0
9307	041372	000000	.WORD	0
9308	041374	000000	.WORD	0
9309	041376	000000	.WORD	0
9310	041400	000000	.WORD	0
9311	041402	000000	.WORD	0
9312	041404	000000	.WORD	0
9313	041406	000000	.WORD	0
9314	041410	000000	.WORD	0
9315	041412	000000	.WORD	0
9316	041414	000000	.WORD	0
9317	041416	000000	.WORD	0
9318	041420	000000	.WORD	0
9319	041422	000000	.WORD	0
9320	041424	000000	.WORD	0
9321	041426	000000	.WORD	0
9322	041430	000000	.WORD	0
9323	041432	000000	.WORD	0
9324	041434	000000	.WORD	0
9325	041436	000000	.WORD	0
9326	041440	000000	.WORD	0
9327	041442	000000	.WORD	0
9328	041444	000000	.WORD	0
9329	041446	000000	.WORD	0
9330	041450	000000	.WORD	0
9331	041452	000000	.WORD	0
9332	041454	000000	.WORD	0
9333	041456	000000	.WORD	0
9334	041460	000000	.WORD	0
9335	041462	000000	.WORD	0
9336	041464	000000	.WORD	0
9337	041466	000000	.WORD	0
9338	041470	000000	.WORD	0
9339	041472	000000	.WORD	0
9340	041474	000000	.WORD	0
9341	041476	000000	.WORD	0
9342	041500	000000	.WORD	0
9343	041502	000000	.WORD	0
9344	041504	000000	.WORD	0
9345	041506	000000	.WORD	0
9346	041510	000000	.WORD	0

```

9347 041512 000000 .WORD 0
9348 041514 000000 .WORD 0
9349 041516 000000 .WORD 0
9350 041520 000000 .WORD 0
9351 041522 000000 .WORD 0
9352 041524 000000 .WORD 0
9353
9354 ;GENERAL PURPOSE DPB - USED BY 'READHD', 'RECALT', 'OFFSET', & 'RTNCTR'
9355
9356 041526 000000 000000 177774 GENDPB: .WORD 0,0,-4,CYLDER
9357 041534 053644
9358 041536 000000 000000 041546 .WORD 0,0,GENREG,0
9359 041544 000000
9360 041546 000024 GENREG: .BLKW 24 ;REGISTER STORAGE IF ERROR
9361
9362 ;*****
9363
9364 .SBTTL TABLES, CONSTANTS, AND VARIABLE LOCATIONS
9365
9366 ;*****
9367
9368 041616 000000 000000 000000 ORDERQ: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF DRIVES PERFORMING COMMANDS
9369 041624 000000 000000 000000
9370 041632 000000 000000 000000
9371
9372 041640 000 000 000 ASNLST: .BYTE 0,0,0,0,0,0,0,0 ; -1 IS ASSIGNED UNIT
9373 041643 000 000 000
9374 041646 000 000
9375
9376 041650 000000 000000 000000 DUNIT: .WORD 0,0,0,0,0,0,0,0,0 ;ADDRESSES OF UNIT TO BE DEASSIGNED
9377 041656 000000 000000 000000
9378 041664 000000 000000 000000
9379
9380 041672 000000 000000 000000 NEWUNT: .WORD 0,0,0,0,0,0,0,0,0 ;ADDRESSES OF NEWLY ASSIGNED UNITS
9381 041700 000000 000000 000000
9382 041706 000000 000000 000000
9383
9384 041714 000000 000000 000000 AVAIL: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF UNITS WAITING FOR PARAMETERS
9385 041722 000000 000000 000000
9386 041730 000000 000000 000000
9387
9388 041736 000000 000000 000000 WAIT: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF UNITS WAITING FOR BUFFERS
9389 041744 000000 000000 000000
9390 041752 000000 000000 000000
9391 041760 000000 000000 000000 PARQ: .WORD 0,0,0,0,0,0,0,0,0 ;LIST OF UNITS WAITING FOR PARAMETERS
9392 041766 000000 000000 000000
9393 041774 000000 000000 000000
9394
9395 042002 000000 BUFTBL: .WORD 0 ;BUFFER ALLOCATION TABLE ENTRY COUNT
9396 042004 000000 000000 .WORD 0,0
9397 042010 000000 000000 .WORD 0,0
9398 042014 000000 000000 .WORD 0,0
9399 042020 000000 000000 .WORD 0,0
9400 042024 000000 000000 .WORD 0,0
9401 042030 000000 000000 .WORD 0,0
9402 042034 000000 000000 .WORD 0,0

```


9403	042040	000000	000000	.WORD	0,0
9404	042044	000000	000000	.WORD	0,0
9405	042050	000000	000000	.WORD	0,0
9406	042054	000000	000000	.WORD	0,0
9407	042060	000000	000000	.WORD	0,0
9408	042064	000000	000000	.WORD	0,0
9409	042070	000000	000000	.WORD	0,0
9410	042074	000000	000000	.WORD	0,0
9411	042100	000000	000000	.WORD	0,0
9412	042104	000000	000000	.WORD	0,0
9413	042110	000000	000000	.WORD	0,0
9414	042114	000000	000000	.WORD	0,0
9415	042120	000000	000000	.WORD	0,0
9416					
9417	042124			BLKADR:	;ADDRESS OF EACH UNIT BLOCK START
9418	042124	037146		.WORD	BLK0
9419	042126	037404		.WORD	BLK1
9420	042130	037642		.WORD	BLK2
9421	042132	040100		.WORD	BLK3
9422	042134	040336		.WORD	BLK4
9423	042136	040574		.WORD	BLK5
9424	042140	041032		.WORD	BLK6
9425	042142	041270		.WORD	BLK7
9426					

9427	042144	151			COMTBL: .BYTE	WCKD	;WRITE CHECK DATA
9428	042145	153			.BYTE	WCKHD	;WRITE CHECK HEADER AND DATA
9429	042146	161			.BYTE	WRTDAT	;WRITE DATA
9430	042147	163			.BYTE	WRTHD	;WRITE HEADER AND DATA
9431	042150	171			.BYTE	RDDAT	;READ DATA
9432	042151	173			.BYTE	RDHD	;READ HEADER AND DATA
9433							
9434	042152	041527	042113	020040	OPMNEM: .ASCIZ	/WCKD /	;WRITE CHECK DATA
9435	042160	000040					
9436	042162	041527	044113	020104	.ASCIZ	/WCKHD /	;WRITE CHECK HEADER & DATA
9437	042170	000040					
9438	042172	051127	042124	052101	.ASCIZ	/WRTDAT /	;WRITE DATA
9439	042200	000040					
9440	042202	051127	044124	020104	.ASCIZ	/WRTHD /	;WRITE HEADER & DATA
9441	042210	000040					
9442	042212	042122	040504	020124	.ASCIZ	/RDDAT /	;READ DATA
9443	042220	000040					
9444	042222	042122	042110	020040	.ASCIZ	/RDHD /	;READ HEADER & DATA
9445	042230	000040					
9446							
9447	042232	000			OFFCOD: .BYTE	0	;OFFSET CODE TABLE
9448	042233	020			.BYTE	20	;+400 U INCHES
9449	042234	220			.BYTE	220	; -400 U INCHES
9450	042235	040			.BYTE	40	;+800 U INCHES
9451	042236	240			.BYTE	240	; -800 U INCHES
9452	042237	060			.BYTE	60	;+1200 U INCHES
9453	042240	260	000		.BYTE	260,0	; -1200 U INCHES, TERMINATOR
9454							
9455	042242	042260			OFMTBL: .WORD	OFMSG0	;1ST OFFSET MESSAGE
9456	042244	042313			.WORD	OFMSG1	;2ND OFFSET MESSAGE
9457	042246	042347			.WORD	OFMSG2	;3RD OFFSET MESSAGE
9458	042250	042403			.WORD	OFMSG3	;4TH OFFSET MESSAGE
9459	042252	042437			.WORD	OFMSG4	;5TH OFFSET MESSAGE
9460	042254	042473			.WORD	OFMSG5	;6TH OFFSET MESSAGE
9461	042256	042530			.WORD	OFMSG6	;7TH OFFSET MESSAGE
9462							
9463	042260	043101	042524	020122	OFMSG0: .ASCIZ	/AFTER RETRY WITHOUT OFFSET/	
9464	042266	042522	051124	020131			
9465	042274	044527	044124	052517			
9466	042302	020124	043117	051506			
9467	042310	052105	000				
9468	042313	101	020124	043117	OFMSG1: .ASCIZ	/AT OFFSET +400 MICRO-INCHES/	
9469	042320	051506	052105	025440			
9470	042326	030064	020060	044515			
9471	042334	051103	026517	047111			
9472	042342	044103	051505	000			
9473	042347	101	020124	043117	OFMSG2: .ASCIZ	/AT OFFSET -400 MICRO-INCHES/	
9474	042354	051506	052105	026440			
9475	042362	030064	020060	044515			
9476	042370	051103	026517	047111			
9477	042376	044103	051505	000			
9478	042403	101	020124	043117	OFMSG3: .ASCIZ	/AT OFFSET +800 MICRO-INCHES/	
9479	042410	051506	052105	025440			
9480	042416	030070	020060	044515			
9481	042424	051103	026517	047111			
9482	042432	044103	051505	000			

9483	042437	101	020124	043117	OFMSG4: .ASCIZ /AT OFFSET -800 MICRO-INCHES/
9484	042444	051506	052105	026440	
9485	042452	030070	020060	044515	
9486	042460	051103	026517	047111	
9487	042466	044103	051505	000	
9488	042473	101	020124	043117	OFMSG5: .ASCIZ /AT OFFSET +1200 MICRO-INCHES/
9489	042500	051506	052105	025440	
9490	042506	031061	030060	046440	
9491	042514	041511	047522	044455	
9492	042522	041516	042510	000123	
9493	042530	052101	047440	043106	OFMSG6: .ASCIZ /AT OFFSET -1200 MICRO-INCHES/
9494	042536	042523	020124	030455	
9495	042544	030062	020060	044515	
9496	042552	051103	026517	047111	
9497	042560	044103	051505	000	

9498
9499 042566

.EVEN

;ERROR MESSAGE LINE 2 POINTER TABLE

9503	042566	046552	DT14.T: .WORD	DT14.0	;DT14 ADDRESSES
9504	042570	046574	.WORD	DT14.1	
9505	042572	046616	.WORD	DT14.2	
9506	042574	046640	.WORD	DT14.3	
9507	042576	046662	.WORD	DT14.4	
9508	042600	046704	.WORD	DT14.5	
9509	042602	046726	.WORD	DT14.6	
9510	042604	046750	.WORD	DT14.7	

9512	042606	046772	DT15.T: .WORD	DT15.0	;DT15 ADDRESSES
9513	042610	047014	.WORD	DT15.1	
9514	042612	047036	.WORD	DT15.2	
9515	042614	047060	.WORD	DT15.3	
9516	042616	047102	.WORD	DT15.4	
9517	042620	047124	.WORD	DT15.5	
9518	042622	047146	.WORD	DT15.6	
9519	042624	047170	.WORD	DT15.7	

9521	042626	047212	DT16.T: .WORD	DT16.0	;DT16 ADDRESSES
9522	042630	047226	.WORD	DT16.1	
9523	042632	047242	.WORD	DT16.2	
9524	042634	047256	.WORD	DT16.3	
9525	042636	047272	.WORD	DT16.4	
9526	042640	047306	.WORD	DT16.5	
9527	042642	047322	.WORD	DT16.6	
9528	042644	047336	.WORD	DT16.7	

;*****

.SBTTL DATA PATTERNS

;*****

9536	042646	000000	STNDAT: .WORD	0	;STANDARD DATA PATTERN POINTER TABLE
9537	042650	042752	.WORD	DATA1	
9538	042652	043012	.WORD	DATA1+40	

9539	042654	043052	.WORD	DATA1+100	
9540	042656	043112	.WORD	DATA1+140	
9541	042660	043152	.WORD	DATA1+200	
9542	042662	043212	.WORD	DATA1+240	
9543	042664	043252	.WORD	DATA1+300	
9544	042666	043312	.WORD	DATA1+340	
9545	042670	043352	.WORD	DATA1+400	
9546	042672	043412	.WORD	DATA1+440	
9547	042674	043452	.WORD	DATA1+500	
9548	042676	043512	.WORD	DATA1+540	
9549	042700	043552	.WORD	DATA1+600	
9550	042702	043612	.WORD	DATA1+640	
9551	042704	043652	.WORD	DATA1+700	
9552	042706	042712	.WORD	DATA0	;ZER0ES
9553	042710	043614	.WORD	DATA1+642	;0NES
9554					
9555	042712	000000	DATA0: .WORD	0	;DUMMY DATA PATTERN
9556	042714	000000	.WORD	0	
9557	042716	000000	.WORD	0	
9558	042720	000000	.WORD	0	
9559	042722	000000	.WORD	0	
9560	042724	000000	.WORD	0	
9561	042726	000000	.WORD	0	
9562	042730	000000	.WORD	0	
9563	042732	000000	.WORD	0	
9564	042734	000000	.WORD	0	
9565	042736	000000	.WORD	0	
9566	042740	000000	.WORD	0	
9567	042742	000000	.WORD	0	
9568	042744	000000	.WORD	0	
9569	042746	000000	.WORD	0	
9570	042750	000000	.WORD	0	
9571					
9572	042752	000001	DATA1: .WORD	000001	;STANDARD PATTERN 1
9573	042754	000003	.WORD	000003	
9574	042756	000007	.WORD	000007	
9575	042760	000017	.WORD	000017	
9576	042762	000037	.WORD	000037	
9577	042764	000077	.WORD	000077	
9578	042766	000177	.WORD	000177	
9579	042770	000377	.WORD	000377	
9580	042772	000777	.WORD	000777	
9581	042774	001777	.WORD	001777	
9582	042776	003777	.WORD	003777	
9583	043000	007777	.WORD	007777	
9584	043002	017777	.WORD	017777	
9585	043004	037777	.WORD	037777	
9586	043006	077777	.WORD	077777	
9587	043010	177777	.WORD	177777	
9588					
9589	043012	177776	.WORD	177776	;STANDARD PATTERN 2
9590	043014	177774	.WORD	177774	
9591	043016	177770	.WORD	177770	
9592	043020	177760	.WORD	177760	
9593	043022	177740	.WORD	177740	
9594	043024	177700	.WORD	177700	

9595	043026	177600	.WORD	177600	
9596	043030	177400	.WORD	177400	
9597	043032	177000	.WORD	177000	
9598	043034	176000	.WORD	176000	
9599	043036	174000	.WORD	174000	
9600	043040	170000	.WORD	170000	
9601	043042	160000	.WORD	160000	
9602	043044	140000	.WORD	140000	
9603	043046	100000	.WORD	100000	
9604	043050	000000	.WORD	000000	
9605					
9606	043052	000000	.WORD	000000	;STANDARD PATTERN 3
9607	043054	000000	.WORD	000000	
9608	043056	000000	.WORD	000000	
9609	043060	177777	.WORD	177777	
9610	043062	177777	.WORD	177777	
9611	043064	177777	.WORD	177777	
9612	043066	000000	.WORD	000000	
9613	043070	000000	.WORD	000000	
9614	043072	177777	.WORD	177777	
9615	043074	177777	.WORD	177777	
9616	043076	000000	.WORD	000000	
9617	043100	177777	.WORD	177777	
9618	043102	000000	.WORD	000000	
9619	043104	177777	.WORD	177777	
9620	043106	000000	.WORD	000000	
9621	043110	177777	.WORD	177777	
9622					
9623	043112	000000	.WORD	000000	;STANDARD PATTERN 4
9624	043114	010421	.WORD	010421	
9625	043116	021042	.WORD	021042	
9626	043120	031463	.WORD	031463	
9627	043122	042104	.WORD	042104	
9628	043124	052525	.WORD	052525	
9629	043126	063146	.WORD	063146	
9630	043130	073567	.WORD	073567	
9631	043132	104210	.WORD	104210	
9632	043134	114631	.WORD	114631	
9633	043136	125252	.WORD	125252	
9634	043140	135673	.WORD	135673	
9635	043142	146314	.WORD	146314	
9636	043144	156735	.WORD	156735	
9637	043146	167356	.WORD	167356	
9638	043150	177777	.WORD	177777	
9639					
9640	043152	052525	.WORD	052525	;STANDARD PATTERN 5
9641	043154	052525	.WORD	052525	
9642	043156	052525	.WORD	052525	
9643	043160	125252	.WORD	125252	
9644	043162	125252	.WORD	125252	
9645	043164	125252	.WORD	125252	
9646	043166	052525	.WORD	052525	
9647	043170	052525	.WORD	052525	
9648	043172	125252	.WORD	125252	
9649	043174	125252	.WORD	125252	
9650	043176	052525	.WORD	052525	

9651	043200	125252	.WORD	125252	
9652	043202	052525	.WORD	052525	
9653	043204	125252	.WORD	125252	
9654	043206	052525	.WORD	052525	
9655	043210	125252	.WORD	125252	
9656					
9657	043212	007417	.WORD	007417	;STANDARD PATTERN 6
9658	043214	007417	.WORD	007417	
9659	043216	007417	.WORD	007417	
9660	043220	170360	.WORD	170360	
9661	043222	170360	.WORD	170360	
9662	043224	170360	.WORD	170360	
9663	043226	007417	.WORD	007417	
9664	043230	007417	.WORD	007417	
9665	043232	170360	.WORD	170360	
9666	043234	170360	.WORD	170360	
9667	043236	007417	.WORD	007417	
9668	043240	170360	.WORD	170360	
9669	043242	007417	.WORD	007417	
9670	043244	170360	.WORD	170360	
9671	043246	007417	.WORD	007417	
9672	043250	170360	.WORD	170360	
9673					
9674	043252	026455	.WORD	026455	;STANDARD PATTERN 7
9675	043254	026455	.WORD	026455	
9676	043256	026455	.WORD	026455	
9677	043260	151322	.WORD	151322	
9678	043262	151322	.WORD	151322	
9679	043264	151322	.WORD	151322	
9680	043266	026455	.WORD	026455	
9681	043270	026455	.WORD	026455	
9682	043272	151322	.WORD	151322	
9683	043274	151322	.WORD	151322	
9684	043276	026455	.WORD	026455	
9685	043300	151322	.WORD	151322	
9686	043302	026455	.WORD	026455	
9687	043304	151322	.WORD	151322	
9688	043306	026455	.WORD	026455	
9689	043310	151322	.WORD	151322	
9690					
9691	043312	165555	.WORD	165555	;STANDARD PATTERN 8
9692	043314	133333	.WORD	133333	
9693	043316	165555	.WORD	165555	
9694	043320	133333	.WORD	133333	
9695	043322	165555	.WORD	165555	
9696	043324	133333	.WORD	133333	
9697	043326	165555	.WORD	165555	
9698	043330	133333	.WORD	133333	
9699	043332	165555	.WORD	165555	
9700	043334	133333	.WORD	133333	
9701	043336	165555	.WORD	165555	
9702	043340	133333	.WORD	133333	
9703	043342	165555	.WORD	165555	
9704	043344	133333	.WORD	133333	
9705	043346	165555	.WORD	165555	
9706	043350	133333	.WORD	133333	

9707				
9708	043352	000001	.WORD	000001 ;STANDARD PATTERN 9
9709	043354	000002	.WORD	000002
9710	043356	000004	.WORD	000004
9711	043360	000010	.WORD	000010
9712	043362	000020	.WORD	000020
9713	043364	000040	.WORD	000040
9714	043366	000100	.WORD	000100
9715	043370	000200	.WORD	000200
9716	043372	000400	.WORD	000400
9717	043374	001000	.WORD	001000
9718	043376	002000	.WORD	002000
9719	043400	004000	.WORD	004000
9720	043402	010000	.WORD	010000
9721	043404	020000	.WORD	020000
9722	043406	040000	.WORD	040000
9723	043410	100000	.WORD	100000
9724				
9725	043412	177776	.WORD	177776 ;STANDARD PATTERN 10
9726	043414	177775	.WORD	177775
9727	043416	177773	.WORD	177773
9728	043420	177767	.WORD	177767
9729	043422	177757	.WORD	177757
9730	043424	177737	.WORD	177737
9731	043426	177677	.WORD	177677
9732	043430	177577	.WORD	177577
9733	043432	177377	.WORD	177377
9734	043434	176777	.WORD	176777
9735	043436	175777	.WORD	175777
9736	043440	173777	.WORD	173777
9737	043442	167777	.WORD	167777
9738	043444	157777	.WORD	157777
9739	043446	137777	.WORD	137777
9740	043450	077777	.WORD	077777
9741				
9742	043452	172666	.WORD	172666 ;STANDARD PATTERN 11
9743	043454	155555	.WORD	155555
9744	043456	172666	.WORD	172666
9745	043460	155555	.WORD	155555
9746	043462	172666	.WORD	172666
9747	043464	155555	.WORD	155555
9748	043466	172666	.WORD	172666
9749	043470	155555	.WORD	155555
9750	043472	172666	.WORD	172666
9751	043474	155555	.WORD	155555
9752	043476	172666	.WORD	172666
9753	043500	155555	.WORD	155555
9754	043502	172666	.WORD	172666
9755	043504	155555	.WORD	155555
9756	043506	172666	.WORD	172666
9757	043510	155555	.WORD	155555
9758				
9759	043512	077777	.WORD	077777 ;STANDARD PATTERN 12
9760	043514	137777	.WORD	137777
9761	043516	157777	.WORD	157777
9762	043520	167777	.WORD	167777

9763	043522	173777	.WORD	173777	
9764	043524	175777	.WORD	175777	
9765	043526	176777	.WORD	176777	
9766	043530	177377	.WORD	177377	
9767	043532	177577	.WORD	177577	
9768	043534	177677	.WORD	177677	
9769	043536	177737	.WORD	177737	
9770	043540	177757	.WORD	177757	
9771	043542	177767	.WORD	177767	
9772	043544	177773	.WORD	177773	
9773	043546	177775	.WORD	177775	
9774	043550	177776	.WORD	177776	
9775					
9776	043552	153333	.WORD	153333	;STANDARD PATTERN 13
9777	043554	066667	.WORD	066667	
9778	043556	153333	.WORD	153333	
9779	043560	066667	.WORD	066667	
9780	043562	153333	.WORD	153333	
9781	043564	066667	.WORD	066667	
9782	043566	153333	.WORD	153333	
9783	043570	066667	.WORD	066667	
9784	043572	153333	.WORD	153333	
9785	043574	066667	.WORD	066667	
9786	043576	153333	.WORD	153333	
9787	043600	066667	.WORD	066667	
9788	043602	153333	.WORD	153333	
9789	043604	066667	.WORD	066667	
9790	043606	153333	.WORD	153333	
9791	043610	066667	.WORD	066667	
9792					
9793	043612	000000	.WORD	000000	;STANDARD PATTERN 14
9794	043614	177777	.WORD	177777	
9795	043616	177777	.WORD	177777	
9796	043620	177777	.WORD	177777	
9797	043622	177777	.WORD	177777	
9798	043624	177777	.WORD	177777	
9799	043626	177777	.WORD	177777	
9800	043630	177777	.WORD	177777	
9801	043632	177777	.WORD	177777	
9802	043634	177777	.WORD	177777	
9803	043636	177777	.WORD	177777	
9804	043640	177777	.WORD	177777	
9805	043642	177777	.WORD	177777	
9806	043644	177777	.WORD	177777	
9807	043646	177777	.WORD	177777	
9808	043650	177777	.WORD	177777	
9809					
9810	043652	177777	.WORD	177777	;STANDARD PATTERN 15
9811	043654	000000	.WORD	000000	
9812	043656	000000	.WORD	000000	
9813	043660	000000	.WORD	000000	
9814	043662	000000	.WORD	000000	
9815	043664	000000	.WORD	000000	
9816	043666	000000	.WORD	000000	
9817	043670	000000	.WORD	000000	
9818	043672	000000	.WORD	000000	

9819	043674	000000			.WORD	000000
9820	043676	000000			.WORD	000000
9821	043700	000000			.WORD	000000
9822	043702	000000			.WORD	000000
9823	043704	000000			.WORD	000000
9824	043706	000000			.WORD	000000
9825	043710	000000			.WORD	000000

;;*****

.SBTTL PRINTER/TELETYPE MESSAGES

;;*****

9833	043712	046111	042514	040507	EM1:	.ASCIZ /ILLEGAL RH11 INTERRUPT (SC=0 OR RHAS=0)/
9834	043720	020114	044122	030461		
9835	043726	044440	052116	051105		
9836	043734	052522	052120	024040		
9837	043742	041523	030075	047440		
9838	043750	020122	044122	051501		
9839	043756	030075	000051			
9840						
9841	043762	047125	054105	042520	EM2:	.ASCIZ /UNEXPECTED ATTENTION/
9842	043770	052103	042105	040440		
9843	043776	052124	047105	044524		
9844	044004	047117	000			
9845						
9846	044007	103	047117	051124	EM3:	.ASCIZ /CONTROL BUS PARITY ERROR DETECTED BY THE RH11/
9847	044014	046117	041040	051525		
9848	044022	050040	051101	052111		
9849	044030	020131	051105	047522		
9850	044036	020122	042504	042524		
9851	044044	052103	042105	041040		
9852	044052	020131	044124	020105		
9853	044060	044122	030461	000		
9854						
9855	044065	103	047117	051124	EM4:	.ASCIZ /CONTROL BUS PARITY ERROR DETECTED BY THE RPO4/
9856	044072	046117	041040	051525		
9857	044100	050040	051101	052111		
9858	044106	020131	051105	047522		
9859	044114	020122	042504	042524		
9860	044122	052103	042105	041040		
9861	044130	020131	044124	020105		
9862	044136	050122	032060	000		
9863						
9864	044143	101	052124	047105	EM5:	.ASCIZ /ATTENTION FROM AN OFFLINE UNIT/
9865	044150	044524	047117	043040		
9866	044156	047522	020115	047101		
9867	044164	047440	043106	044514		
9868	044172	042516	052440	044516		
9869	044200	000124				
9870						
9871	044202	047125	047503	051122	EM10:	.ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
9872	044210	041505	040524	046102		
9873	044216	020105	040515	051523		
9874	044224	052502	020123	040520		

9875	044232	044522	054524	042440	
9876	044240	051122	051117	000	
9877					
9878	044245	106	052101	046101	EM11: .ASCIZ /FATAL MASSBUS PARITY ERROR/
9879	044252	046440	051501	041123	
9880	044260	051525	050040	051101	
9881	044266	052111	020131	051105	
9882	044274	047522	000122		
9883					
9884	044300	042520	051522	051511	EM12: .ASCIZ /PERSISTENT DEVICE UNSAFE/
9885	044306	042524	052116	042040	
9886	044314	053105	041511	020105	
9887	044322	047125	040523	042506	
9888	044330	000			
9889					
9890	044331	117	042520	040522	EM13: .ASCIZ /OPERATION NOT COMPLETED WITHIN TIME LIMIT/
9891	044336	044524	047117	047040	
9892	044344	052117	041440	046517	
9893	044352	046120	052105	042105	
9894	044360	053440	052111	044510	
9895	044366	020116	044524	042515	
9896	044374	046040	046511	052111	
9897	044402	000			
9898					
9899	044403	125	044516	020124	EM14: .ASCIZ /UNIT WENT OFFLINE/
9900	044410	042527	052116	047440	
9901	044416	043106	044514	042516	
9902	044424	000			
9903					
9904	044425	116	020117	042522	EM15: .ASCIZ /NO RESPONSE TO PORT REQUEST/
9905	044432	050123	047117	042523	
9906	044440	052040	020117	047520	
9907	044446	052122	051040	050505	
9908	044454	042525	052123	000	
9909					
9910	044461	110	040505	042504	EM20: .ASCIZ /HEADER CRC ERROR/
9911	044466	020122	051103	020103	
9912	044474	051105	047522	000122	
9913					
9914	044502	040504	040524	041440	EM21: .ASCIZ /DATA CHECK ('DCK') ERROR/
9915	044510	042510	045503	024040	
9916	044516	042047	045503	024447	
9917	044524	042440	051122	051117	
9918	044532	000			
9919					
9920	044533	127	044522	042524	EM22: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') SET/
9921	044540	041440	042510	045503	
9922	044546	042440	051122	051117	
9923	044554	026440	042040	052101	
9924	044562	020101	044103	041505	
9925	044570	020113	023450	041504	
9926	044576	023513	020051	042523	
9927	044604	000124			
9928					
9929	044606	051127	052111	020105	EM23: .ASCIZ /WRITE CHECK ERROR - DATA CHECK ('DCK') NOT SET/
9930	044614	044103	041505	020113	

9931	044622	051105	047522	020122		
9932	044630	020055	040504	040524		
9933	044636	041440	042510	045503		
9934	044644	024040	042047	045503		
9935	044652	024447	047040	052117		
9936	044660	051440	052105	000		
9937						
9938	044665	110	040505	042504	EM24:	.ASCIZ /HEADER READ ERROR - 'FMT' BIT DROPPED/
9939	044672	020122	042522	042101		
9940	044700	042440	051122	051117		
9941	044706	026440	023440	046506		
9942	044714	023524	041040	052111		
9943	044722	042040	047522	050120		
9944	044730	042105	000			
9945						
9946	044733	110	040505	042504	EM25:	.ASCIZ /HEADER READ ERROR - HEADER COMPARE ('HCE') ERROR/
9947	044740	020122	042522	042101		
9948	044746	042440	051122	051117		
9949	044754	026440	044040	040505		
9950	044762	042504	020122	047503		
9951	044770	050115	051101	020105		
9952	044776	023450	041510	023505		
9953	045004	020051	051105	047522		
9954	045012	000122				
9955						
9956	045014	047506	046522	052101	EM26:	.ASCIZ /FORMAT ERROR ('FER')/
9957	045022	042440	051122	051117		
9958	045030	024040	043047	051105		
9959	045036	024447	000			
9960						
9961	045041	110	040505	042504	EM27:	.ASCIZ /HEADER COMPARE ('HCE') ERROR/
9962	045046	020122	047503	050115		
9963	045054	051101	020105	023450		
9964	045062	041510	023505	020051		
9965	045070	051105	047522	000122		
9966						
9967	045076	044515	041523	046105	EM30:	.ASCIZ /MISCELLANEOUS DRIVE ERROR/
9968	045104	040514	042516	052517		
9969	045112	020123	051104	053111		
9970	045120	020105	051105	047522		
9971	045126	000122				
9972						
9973	045130	050117	051105	052101	EM31:	.ASCIZ /OPERATION INCOMPLETE ('OPI') ERROR/
9974	045136	047511	020116	047111		
9975	045144	047503	050115	042514		
9976	045152	042524	024040	047447		
9977	045160	044520	024447	042440		
9978	045166	051122	051117	000		
9979						
9980	045173	104	044522	042526	EM32:	.ASCIZ /DRIVE TIMING ('DTE') ERROR/
9981	045200	052040	046511	047111		
9982	045206	020107	023450	052104		
9983	045214	023505	020051	051105		
9984	045222	047522	000122			
9985						
9986	045226	040520	044522	054524	EM33:	.ASCIZ /PARITY ('PAR') ERROR AFTER OPERATION STARTED/

9987	045234	024040	050047	051101	
9988	045242	024447	042440	051122	
9989	045250	051117	040440	052105	
9990	045256	051105	047440	042520	
9991	045264	040522	044524	047117	
9992	045272	051440	040524	052122	
9993	045300	042105	000		
9994					
9995	045303	127	044522	042524	EM34: .ASCIZ /WRITE CLOCK FAILURE ('WCF') ERROR/
9996	045310	041440	047514	045503	
9997	045316	043040	044501	052514	
9998	045324	042522	024040	053447	
9999	045332	043103	024447	042440	
10000	045340	051122	051117	000	
10001					
10002	045345	111	053116	046101	EM35: .ASCIZ /INVALID ADDRESS ('IAE') ERROR/
10003	045352	042111	040440	042104	
10004	045360	042522	051523	024040	
10005	045366	044447	042501	024447	
10006	045374	042440	051122	051117	
10007	045402	000			
10008					
10009	045403	127	044522	042524	EM36: .ASCIZ /WRITE LOCK ('WLE') ERROR/
10010	045410	046040	041517	020113	
10011	045416	023450	046127	023505	
10012	045424	020051	051105	047522	
10013	045432	000122			
10014					
10015	045434	044122	030461	047440	EM40: .ASCIZ /RH11 OR UNIBUS TRANSFER ERROR/
10016	045442	020122	047125	041111	
10017	045450	051525	052040	040522	
10018	045456	051516	042506	020122	
10019	045464	051105	047522	000122	
10020					
10021	045472	052502	020123	042101	EM41: .ASCIZ /BUS ADDRESS OR WORD COUNT INCORRECT/
10022	045500	051104	051505	020123	
10023	045506	051117	053440	051117	
10024	045514	020104	047503	047125	
10025	045522	020124	047111	047503	
10026	045530	051122	041505	000124	
10027					
10028	045536	040504	040524	041440	EM42: .ASCIZ /DATA COMPARE ERRORS - NO RPO4 ERROR DETECTED/
10029	045544	046517	040520	042522	
10030	045552	042440	051122	051117	
10031	045560	020123	020055	047516	
10032	045566	051040	030120	020064	
10033	045574	051105	047522	020122	
10034	045602	042504	042524	052103	
10035	045610	042105	000		
10036					
10037	045613	103	047101	052047	EM43: .ASCIZ /CAN'T MATCH DATA READ WITH A PATTERN/
10038	045620	046440	052101	044103	
10039	045626	042040	052101	020101	
10040	045634	042522	042101	053440	
10041	045642	052111	020110	020101	
10042	045650	040520	052124	051105	

E01

10043	045656	000116							
10044									
10045	045660	051105	047522	020122	EM44:	.ASCIZ	/ERROR BIT(S) SET, BUT NO ERROR SIGNALLED BY THE RH11/		
10046	045666	044502	024124	024523					
10047	045674	051440	052105	020054					
10048	045702	052502	020124	047516					
10049	045710	042440	051122	051117					
10050	045716	051440	043511	040516					
10051	045724	042514	020104	054502					
10052	045732	052040	042510	051040					
10053	045740	030510	000061						
10054	045744	042523	045505	044440	EM50:	.ASCIZ	/SEEK INCOMPLETE OR OFF CYLINDER ERROR/		
10055	045752	041516	046517	046120					
10056	045760	052105	020105	051117					
10057	045766	047440	043106	041440					
10058	045774	046131	047111	042504					
10059	046002	020122	051105	047522					
10060	046010	000122							
10061									
10062	046012	051120	043517	040522	EM51:	.ASCIZ	/PROGRAM DETECTED POSITIONING ERROR/		
10063	046020	020115	042504	042524					
10064	046026	052103	042105	050040					
10065	046034	051517	052111	047511					
10066	046042	044516	043516	042440					
10067	046050	051122	051117	000					
10068									
10069	046055	104	053105	041511	EM60:	.ASCIZ	/DEVICE UNSAFE/		
10070	046062	020105	047125	040523					
10071	046070	042506	000						
10072									
10073	046073	040	020040	042040	DH2:	.ASCIZ	/ DRV RHDS1 RHER1 RHER2 RHER3 RHAS/		
10074	046100	053122	051040	042110					
10075	046106	030523	020040	051040					
10076	046114	042510	030522	020040					
10077	046122	051040	042510	031122					
10078	046130	020040	051040	042510					
10079	046136	031522	020040	051040					
10080	046144	040510	000123						
10081									
10082	046150	020040	020040	051104	DH3:	.ASCIZ	/ DRV REG ADR DATA/		
10083	046156	020126	042522	020107					
10084	046164	042101	020122	042040					
10085	046172	052101	000101						
10086									
10087	046176	020040	020040	051104	DH4:	.ASCIZ	/ DRV REG ADR GOOD BAD/		
10088	046204	020126	042522	020107					
10089	046212	042101	020122	020040					
10090	046220	043440	047517	020104					
10091	046226	020040	041040	042101					
10092	046234	000							
10093									
10094	046235	104	053122	051040	DH14:	.ASCII	/DRV RHCS1 RHCS2 RHDS1 RHER1 /		
10095	046242	041510	030523	020040					
10096	046250	051040	041510	031123					
10097	046256	020040	051040	042110					
10098	046264	030523	020040	051040					

10099	046272	042510	030522	020040							
10100	046300	040									
10101	046301	122	042510	031122		.ASCIZ	/RHER2	RHER3	RHEC1	RHEC2/<15><12>	
10102	046306	020040	051040	042510							
10103	046314	031522	020040	051040							
10104	046322	042510	030503	020040							
10105	046330	051040	042510	031103							
10106	046336	005015	000								
10107											
10108	046341	122	053510	020103	DH15:	.ASCII	/RHWC	RHBA	RHDA	RHAS	RHLA /
10109	046346	020040	051040	041110							
10110	046354	020101	020040	051040							
10111	046362	042110	020101	020040							
10112	046370	051040	040510	020123							
10113	046376	020040	051040	046110							
10114	046404	020101	020040	040							
10115	046411	122	042110	020102		.ASCIZ	/RHDB	RHMA	RHDT/<15><12>		
10116	046416	020040	051040	046510							
10117	046424	020122	020040	051040							
10118	046432	042110	006524	000012							
10119											
10120	046440	044122	047123	020040	DH16:	.ASCIZ	/RHSN	RHOF	RHCA	RHCC	STATUS/<15><12>
10121	046446	020040	044122	043117							
10122	046454	020040	020040	044122							
10123	046462	040503	020040	020040							
10124	046470	044122	041503	020040							
10125	046476	020040	052123	052101							
10126	046504	051525	005015	000							
10127											
10128		046512				.EVEN					
10129											
10130	046512	001256	030536	030540	DT2:	.WORD	DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0				
10131	046520	030542	030544	001260							
10132	046526	000000									
10133											
10134	046530	001256	035724	035726	DT3:	.WORD	DRIVE,RD.ADR,RD.WRD,0				
10135	046536	000000									
10136											
10137	046540	001256	036126	036124	DT4:	.WORD	DRIVE,WRT.AD,WRT.WD,RD.WRD,0				
10138	046546	035726	000000								
10139											
10140	046552	037332	037342	037344	DT14.0:	.WORD	BLK0+\$RHCS1,BLK0+\$RHCS2,BLK0+\$RHDS1,BLK0+\$RHER1				
10141	046560	037346									
10142	046562	037372	037374	037376		.WORD	BLK0+\$RHER2,BLK0+\$RHER3,BLK0+\$RHEC1,BLK0+\$RHEC2,0				
10143	046570	037400	000000								
10144	046574	037570	037600	037602	DT14.1:	.WORD	BLK1+\$RHCS1,BLK1+\$RHCS2,BLK1+\$RHDS1,BLK1+\$RHER1				
10145	046602	037604									
10146	046604	037630	037632	037634		.WORD	BLK1+\$RHER2,BLK1+\$RHER3,BLK1+\$RHEC1,BLK1+\$RHEC2,0				
10147	046612	037636	000000								
10148	046616	040026	040036	040040	DT14.2:	.WORD	BLK2+\$RHCS1,BLK2+\$RHCS2,BLK2+\$RHDS1,BLK2+\$RHER1				
10149	046624	040042									
10150	046626	040066	040070	040072		.WORD	BLK2+\$RHER2,BLK2+\$RHER3,BLK2+\$RHEC1,BLK2+\$RHEC2,0				
10151	046634	040074	000000								
10152	046640	040264	040274	040276	DT14.3:	.WORD	BLK3+\$RHCS1,BLK3+\$RHCS2,BLK3+\$RHDS1,BLK3+\$RHER1				
10153	046646	040300									
10154	046650	040324	040326	040330		.WORD	BLK3+\$RHER2,BLK3+\$RHER3,BLK3+\$RHEC1,BLK3+\$RHEC2,0				

10155	046656	040332	000000				
10156	046662	040522	040532	040534	DT14.4:	.WORD	BLK4+\$RHCS1, BLK4+\$RHCS2, BLK4+\$RHDS1, BLK4+\$RHER1
10157	046670	040536					
10158	046672	040562	040564	040566		.WORD	BLK4+\$RHER2, BLK4+\$RHER3, BLK4+\$RHEC1, BLK4+\$RHEC2, 0
10159	046700	040570	000000				
10160	046704	040760	040770	040772	DT14.5:	.WORD	BLK5+\$RHCS1, BLK5+\$RHCS2, BLK5+\$RHDS1, BLK5+\$RHER1
10161	046712	040774					
10162	046714	041020	041022	041024		.WORD	BLK5+\$RHER2, BLK5+\$RHER3, BLK5+\$RHEC1, BLK5+\$RHEC2, 0
10163	046722	041026	000000				
10164	046726	041216	041226	041230	DT14.6:	.WORD	BLK6+\$RHCS1, BLK6+\$RHCS2, BLK6+\$RHDS1, BLK6+\$RHER1
10165	046734	041232					
10166	046736	041256	041260	041262		.WORD	BLK6+\$RHER2, BLK6+\$RHER3, BLK6+\$RHEC1, BLK6+\$RHEC2, 0
10167	046744	041264	000000				
10168	046750	041454	041464	041466	DT14.7:	.WORD	BLK7+\$RHCS1, BLK7+\$RHCS2, BLK7+\$RHDS1, BLK7+\$RHER1
10169	046756	041470					
10170	046760	041514	041516	041520		.WORD	BLK7+\$RHER2, BLK7+\$RHER3, BLK7+\$RHEC1, BLK7+\$RHEC2, 0
10171	046766	041522	000000				
10172							
10173	046772	037334	037336	037340	DT15.0:	.WORD	BLK0+\$RHWC, BLK0+\$RHBA, BLK0+\$RHDA, BLK0+\$RHAS, BLK0+\$RHLA
10174	047000	037350	037352				
10175	047004	037354	037356	037360		.WORD	BLK0+\$RHDB, BLK0+\$RHMR, BLK0+\$RHDT, 0
10176	047012	000000					
10177	047014	037572	037574	037576	DT15.1:	.WORD	BLK1+\$RHWC, BLK1+\$RHBA, BLK1+\$RHDA, BLK1+\$RHAS, BLK1+\$RHLA
10178	047022	037606	037610				
10179	047026	037612	037614	037616		.WORD	BLK1+\$RHDB, BLK1+\$RHMR, BLK1+\$RHDT, 0
10180	047034	000000					
10181	047036	040030	040032	040034	DT15.2:	.WORD	BLK2+\$RHWC, BLK2+\$RHBA, BLK2+\$RHDA, BLK2+\$RHAS, BLK2+\$RHLA
10182	047044	040044	040046				
10183	047050	040050	040052	040054		.WORD	BLK2+\$RHDB, BLK2+\$RHMR, BLK2+\$RHDT, 0
10184	047056	000000					
10185	047060	040266	040270	040272	DT15.3:	.WORD	BLK3+\$RHWC, BLK3+\$RHBA, BLK3+\$RHDA, BLK3+\$RHAS, BLK3+\$RHLA
10186	047066	040302	040304				
10187	047072	040306	040310	040312		.WORD	BLK3+\$RHDB, BLK3+\$RHMR, BLK3+\$RHDT, 0
10188	047100	000000					
10189	047102	040524	040526	040530	DT15.4:	.WORD	BLK4+\$RHWC, BLK4+\$RHBA, BLK4+\$RHDA, BLK4+\$RHAS, BLK4+\$RHLA
10190	047110	040540	040542				
10191	047114	040544	040546	040550		.WORD	BLK4+\$RHDB, BLK4+\$RHMR, BLK4+\$RHDT, 0
10192	047122	000000					
10193	047124	040762	040764	040766	DT15.5:	.WORD	BLK5+\$RHWC, BLK5+\$RHBA, BLK5+\$RHDA, BLK5+\$RHAS, BLK5+\$RHLA
10194	047132	040776	041000				
10195	047136	041002	041004	041006		.WORD	BLK5+\$RHDB, BLK5+\$RHMR, BLK5+\$RHDT, 0
10196	047144	000000					
10197	047146	041220	041222	041224	DT15.6:	.WORD	BLK6+\$RHWC, BLK6+\$RHBA, BLK6+\$RHDA, BLK6+\$RHAS, BLK6+\$RHLA
10198	047154	041234	041236				
10199	047160	041240	041242	041244		.WORD	BLK6+\$RHDB, BLK6+\$RHMR, BLK6+\$RHDT, 0
10200	047166	000000					
10201	047170	041456	041460	041462	DT15.7:	.WORD	BLK7+\$RHWC, BLK7+\$RHBA, BLK7+\$RHDA, BLK7+\$RHAS, BLK7+\$RHLA
10202	047176	041472	041474				
10203	047202	041476	041500	041502		.WORD	BLK7+\$RHDB, BLK7+\$RHMR, BLK7+\$RHDT, 0
10204	047210	000000					
10205							
10206	047212	037362	037364	037366	DT16.0:	.WORD	BLK0+\$RHSN, BLK0+\$RHOF, BLK0+\$RHCA, BLK0+\$RHCC, BLK0+\$STATUS, 0
10207	047220	037370	037164	000000			
10208	047226	037620	037622	037624	DT16.1:	.WORD	BLK1+\$RHSN, BLK1+\$RHOF, BLK1+\$RHCA, BLK1+\$RHCC, BLK1+\$STATUS, 0
10209	047234	037626	037422	000000			
10210	047242	040056	040060	040062	DT16.2:	.WORD	BLK2+\$RHSN, BLK2+\$RHOF, BLK2+\$RHCA, BLK2+\$RHCC, BLK2+\$STATUS, 0

H01

10211	047250	040064	037660	000000		
10212	047256	040314	040316	040320	DT16.3: .WORD	BLK3+\$RHSN, BLK3+\$RHOF, BLK3+\$RHCA, BLK3+\$RHCC, BLK3+\$STATUS, 0
10213	047264	040322	040116	000000		
10214	047272	040552	040554	040556	DT16.4: .WORD	BLK4+\$RHSN, BLK4+\$RHOF, BLK4+\$RHCA, BLK4+\$RHCC, BLK4+\$STATUS, 0
10215	047300	040560	040354	000000		
10216	047306	041010	041012	041014	DT16.5: .WORD	BLK5+\$RHSN, BLK5+\$RHOF, BLK5+\$RHCA, BLK5+\$RHCC, BLK5+\$STATUS, 0
10217	047314	041016	040612	000000		
10218	047322	041246	041250	041252	DT16.6: .WORD	BLK6+\$RHSN, BLK6+\$RHOF, BLK6+\$RHCA, BLK6+\$RHCC, BLK6+\$STATUS, 0
10219	047330	041254	041050	000000		
10220	047336	041504	041506	041510	DT16.7: .WORD	BLK7+\$RHSN, BLK7+\$RHOF, BLK7+\$RHCA, BLK7+\$RHCC, BLK7+\$STATUS, 0
10221	047344	041512	041306	000000		
10222						
10223	047352	001	000	000	DF2: .BYTE	1,0,0,0,0,0
10224	047355	000	000	000		
10225						
10226	047360	001	000	000	DF3: .BYTE	1,0,0
10227						
10228	047363	001	000	000	DF4: .BYTE	1,0,0,0
10229	047366	000				
10230						
10231		047370				.EVEN
10232						
10233	047370	051120	051505	047105	LIN2C: .ASCIZ	/PRESENT ORDER = /
10234	047376	020124	051117	042504		
10235	047404	020122	020075	000		
10236	047411	040	050040	042522	LIN2P: .ASCIZ	/ PREVIOUS ORDER = /
10237	047416	044526	052517	020123		
10238	047424	051117	042504	020122		
10239	047432	020075	000			
10240	047435	105	051122	051117	LIN2M: .ASCIZ	/ERROR OCCURED DURING NON-DATA TRANSFER OPERATION/
10241	047442	047440	041503	051125		
10242	047450	042105	042040	051125		
10243	047456	047111	020107	047516		
10244	047464	026516	040504	040524		
10245	047472	052040	040522	051516		
10246	047500	042506	020122	050117		
10247	047506	051105	052101	047511		
10248	047514	000116				
10249	047516	020052	051105	047522	LIN2S: .ASCIZ	Ⓜ* ERROR AT BAD TRACK/SECTORⓂ
10250	047524	020122	052101	041040		
10251	047532	042101	052040	040522		
10252	047540	045503	051457	041505		
10253	047546	047524	000122			
10254	047552	051105	047522	020122	LINM3: .ASCIZ	/ERROR AT C/
10255	047560	052101	041440	000		
10256	047565	040	000124		T: .ASCIZ	/ T/
10257	047570	051120	051505	047105	LINN3: .ASCIZ	/PRESENT ADDR = C/
10258	047576	020124	042101	051104		
10259	047604	036440	041440	000		
10260	047611	040	000123		S: .ASCIZ	/ S/
10261	047614	020040	050040	042522	LINP3: .ASCIZ	/ PREV ADDR = C/
10262	047622	020126	042101	051104		
10263	047630	036440	041440	000		
10264	047635	123	040524	052122	LINS3: .ASCIZ	/START CYL = /
10265	047642	041440	046131	036440		
10266	047650	000040				

10267	047652	020040	047105	020104	LINEN3: .ASCIZ / END CYL = /
10268	047660	054503	020114	020075	
10269	047666	000			
10270	047667	040	040440	052103	LINA3: .ASCIZ / ACTUAL CYL = /
10271	047674	040525	020114	054503	
10272	047702	020114	020075	000	
10273	047707	040	052040	045522	LINT3: .ASCIZ / TRK = /
10274	047714	036440	000040		
10275	047720	051040	041510	020101	LINCA3: .ASCIZ / RHCA = /
10276	047726	020075	000		
10277	047731	122	042110	020101	LINDA3: .ASCIZ /RHDA = /
10278	047736	020075	000		
10279	047741	122	041110	020101	LINB3: .ASCIZ /RHBA = /
10280	047746	020075	000		
10281	047751	040	051040	053510	LINW3: .ASCIZ / RHWC = /
10282	047756	020103	020075	000	
10283	047763	123	040524	052122	LINST3: .ASCIZ /START TRK = /
10284	047770	052040	045522	036440	
10285	047776	000040			
10286	050000	052123	051101	020124	LINSS3: .ASCIZ /START SEC = /
10287	050006	042523	020103	020075	
10288	050014	000			
10289	050015	102	043125	042506	LINM4: .ASCIZ /BUFFER ADDR = /
10290	050022	020122	042101	051104	
10291	050030	036440	000040		
10292	050034	020040	044523	042532	LINS4: .ASCIZ / SIZE = /
10293	050042	036440	000040		
10294	050046	020040	041501	052524	LINX4: .ASCIZ / ACTUAL NMBR WRDS XFRD = /
10295	050054	046101	047040	041115	
10296	050062	020122	051127	051504	
10297	050070	054040	051106	020104	
10298	050076	020075	000		
10299	050101	107	047517	020104	LIND5: .ASCIZ /GOOD DATA = /
10300	050106	040504	040524	036440	
10301	050114	000040			
10302	050116	020040	040502	020104	LINB5: .ASCIZ / BAD DATA = /
10303	050124	040504	040524	036440	
10304	050132	000040			
10305	050134	020040	042523	052103	LINP5: .ASCIZ / SECT POS = /
10306	050142	050040	051517	036440	
10307	050150	000040			
10308	050152	042510	042101	051105	LINSS: .ASCIZ /HEADER FROM ERROR SECTOR = /
10309	050160	043040	047522	020115	
10310	050166	051105	047522	020122	
10311	050174	042523	052103	051117	
10312	050202	036440	000040		
10313	050206	044122	041505	020061	LINEP5: .ASCIZ /RHEC1 = /
10314	050214	020075	000		
10315	050217	040	044122	041505	LINE05: .ASCIZ / RHEC2 = /
10316	050224	020062	020075	000	
10317	050231	123	041505	047524	LINB6: .ASCIZ /SECTOR IS ECC CORRECTABLE /
10318	050236	020122	051511	042440	
10319	050244	041503	041440	051117	
10320	050252	042522	052103	041101	
10321	050260	042514	000040		
10322	050264	042523	052103	051117	LINC6: .ASCIZ /SECTOR READ CORRECTLY /

10323	050272	051040	040505	020104					
10324	050300	047503	051122	041505					
10325	050306	046124	020131	000					
10326	050313	103	051117	042522	LIN6:	.ASCIZ	/CORRECTED ON /		
10327	050320	052103	042105	047440					
10328	050326	020116	000						
10329	050331	040	042522	051124	LINR6:	.ASCIZ	/ RETRIES/		
10330	050336	042511	000123						
10331	050342	047125	047503	051122	LINU06:	.ASCIZ	/UNCORRECTABLE AFTER /		
10332	050350	041505	040524	046102					
10333	050356	020105	043101	042524					
10334	050364	020122	000						
10335	050367	040	052040	052117	LIN7M:	.ASCIZ	/ TOTAL MISPOS ERR = /		
10336	050374	046101	046440	051511					
10337	050402	047520	020123	051105					
10338	050410	020122	020075	000					
10339	050415	117	042122	051105	LIN70:	.ASCIZ	/ORDERS:/		
10340	050422	035123	000						
10341	050425	040	047524	040524	LIN7P:	.ASCIZ	/ TOTAL SEEKS = /		
10342	050432	020114	042523	045505					
10343	050440	020123	020075	000					
10344	050445	040	047524	040524	LIN7S:	.ASCIZ	/ TOTAL SKI,OCYL ERR = /		
10345	050452	020114	045523	026111					
10346	050460	041517	046131	042440					
10347	050466	051122	036440	000040					
10348	050474	020040	051105	047522	LIN7T:	.ASCIZ	/ ERRORS:/		
10349	050502	051522	000072						
10350	050506	020040	051127	051504	LIN7X:	.ASCIZ	/ WRDS XFR:/		
10351	050514	054040	051106	000072					
10352	050522	020040	051127	051504	LIN7R:	.ASCIZ	/ WRDS READ:/		
10353	050530	051040	040505	035104					
10354	050536	000							
10355	050537	104	043111	042506	LIN8M:	.ASCIZ	/DIFFERENT ERROR DURING RETRY/		
10356	050544	042522	052116	042440					
10357	050552	051122	051117	042040					
10358	050560	051125	047111	020107					
10359	050566	042522	051124	000131					
10360	050574	040504	040524	041440	LIN9B:	.ASCIZ	/DATA COMPARISON ERRORS/		
10361	050602	046517	040520	044522					
10362	050610	047523	020116	051105					
10363	050616	047522	051522	000					
10364	050623	040	020040	020040	LIN9H:	.ASCII	/ GOOD BAD/<15><12>		
10365	050630	020040	020040	047507					
10366	050636	042117	020040	020040					
10367	050644	040502	006504	012					
10368	050651	114	041517	020040		.ASCIZ	/LOC DATA DATA/<15><12>		
10369	050656	020040	020040	040504					
10370	050664	040524	020040	020040					
10371	050672	040504	040524	005015					
10372	050700	000							
10373	050701	114	041517	020040	LIN9I:	.ASCIZ	/LOC DATA/<15><12>		
10374	050706	020040	020040	040504					
10375	050714	040524	005015	000					
10376	050721	124	052117	046101	LIN9E:	.ASCIZ	/TOTAL COMPARE ERRORS = /		
10377	050726	041440	046517	040520					
10378	050734	042522	042440	051122					

K01

10379	050742	051117	020123	020075	
10380	050750	000			
10381	050751	124	042510	042040	LIN9G: .ASCIZ /THE DATA COMPARED OK/<15><12>
10382	050756	052101	020101	047503	
10383	050764	050115	051101	042105	
10384	050772	047440	006513	000012	
10385	051000	051105	047522	020122	LIN10A: .ASCIZ /ERROR BURST BEGINS AT WORD /
10386	051006	052502	051522	020124	
10387	051014	042502	044507	051516	
10388	051022	040440	020124	047527	
10389	051030	042122	000040		
10390	051034	044440	020116	040504	LIN10B: .ASCIZ / IN DATA FIELD OF ERROR SECTOR/<15><12>
10391	051042	040524	043040	042511	
10392	051050	042114	047440	020106	
10393	051056	051105	047522	020122	
10394	051064	042523	052103	051117	
10395	051072	005015	000		
10396	051075	105	051122	051117	LIN10C: .ASCII /ERROR WAS NOT IN THE DATA READ - /<15><12>
10397	051102	053440	051501	047040	
10398	051110	052117	044440	020116	
10399	051116	044124	020105	040504	
10400	051124	040524	051040	040505	
10401	051132	020104	020055	005015	
10402	051140	041505	020103	047503	.ASCIZ /ECC CORRECTION CAN'T BE PERFORMED/
10403	051146	051122	041505	044524	
10404	051154	047117	041440	047101	
10405	051162	052047	041040	020105	
10406	051170	042520	043122	051117	
10407	051176	042515	000104		
10408	051202	041505	020103	047503	LIN10H: .ASCII /ECC CORRECTION RESULTS/<15><12>
10409	051210	051122	041505	044524	
10410	051216	047117	051040	051505	
10411	051224	046125	051524	005015	
10412	051232	042101	051104	020040	.ASCIZ /ADDR BAD CORRECTED /<15><12>
10413	051240	020040	040502	020104	
10414	051246	020040	020040	047503	
10415	051254	051122	041505	042524	
10416	051262	020104	005015	000	
10417	051267	103	047117	042524	LIN11H: .ASCIZ /CONTENTS OF ERROR SECTOR (REPORTED ABOVE)/<15><12>
10418	051274	052116	020123	043117	
10419	051302	042440	051122	051117	
10420	051310	051440	041505	047524	
10421	051316	020122	051050	050105	
10422	051324	051117	042524	020104	
10423	051332	041101	053117	024505	
10424	051340	005015	000		
10425	051343	101	042104	020122	.ASCIZ /ADDR DATA/<15><12>
10426	051350	020040	042040	052101	
10427	051356	006501	000012		
10428	051362	020040			LIN4SP: .ASCII //
10429	051364	040			LIN5P: .ASCII //
10430	051365	040	000		LIN5PO: .ASCIZ //
10431	051367	125	044516	000124	UNTMSG: .ASCIZ /UNIT/
10432	051374	047440	043106	044514	UNTOFF: .ASCIZ / OFFLINE/
10433	051402	042516	000		
10434	051405	040	047117	044514	UNTON: .ASCIZ / ONLINE/

10491	052046	025052	025052	025052	
10492	052054	042040	044522	042526	
10493	052062	021440	000		
10494	052065	040	052123	051101	ASGND: .ASCIZ / STARTED/<15><12>
10495	052072	042524	006504	000012	
10496	052100	005015	050122	032060	TITLE: .ASCII <15><12>/RPO4 MULTI-DRIVE EXERCISER PROGRAM/
10497	052106	046440	046125	044524	
10498	052114	042055	044522	042526	
10499	052122	042440	042530	041522	
10500	052130	051511	051105	050040	
10501	052136	047522	051107	046501	
10502	052144	005015	051450	047111	.ASCII <15><12>(SINGLE/DUAL PORT VERSION)
10503	052152	046107	027505	052504	
10504	052160	046101	050040	051117	
10505	052166	020124	042526	051522	
10506	052174	047511	024516		
10507	052200	005015	040515	047111	.ASCIZ <15><12>/MAINDEC-11-DERPN-B/<15><12><12>
10508	052206	042504	026503	030461	
10509	052214	042055	051105	047120	
10510	052222	041055	005015	000012	
10511	052230	047077	052117	042440	NOTNUF: .ASCIZ /?NOT ENOUGH MEMORY/<15><12>
10512	052236	047516	043525	020110	
10513	052244	042515	047515	054522	
10514	052252	005015	000		
10515	052255	015	037412	023440	NEDCLK: .ASCIZ <15><12>/? 'L' OR 'P' CLOCK REQUIRED ON SYSTEM/<15><12>
10516	052262	022514	047440	020122	
10517	052270	050047	020047	046103	
10518	052276	041517	020213	042522	
10519	052304	052521	051111	042105	
10520	052312	047440	020116	054523	
10521	052320	052123	046505	005015	
10522	052326	000			
10523	052327	056	000		PERIOD: .ASCIZ /./
10524	052331	077	000		QUES: .ASCIZ /?/
10525	052333	007	050077	044522	PRTATN: .ASCIZ <07>/?PRINTER REQUIRES MANUAL INTERVENTION/<15><12><12>
10526	052340	052116	051105	051040	
10527	052346	050505	044525	042522	
10528	052354	020123	040515	052516	
10529	052362	046101	044440	052116	
10530	052370	051105	042526	052116	
10531	052376	047511	006516	005012	
10532	052404	000			
10533	052405	111	053116	046101	INVLD: .ASCIZ /INVALID COMMAND/<15><12>
10534	052412	042111	041440	046517	
10535	052420	040515	042116	005015	
10536	052426	000			
10537	052427	015	042412	052116	ENTDAT: .ASCIZ <15><12>/ENTER DATE: /
10538	052434	051105	042040	052101	
10539	052442	035105	000040		
10540	052446	047105	042524	020122	ENTID: .ASCIZ /ENTER OPERATOR I.D.: /
10541	052454	050117	051105	052101	
10542	052462	051117	044440	042056	
10543	052470	035056	000040		
10544	052474	005015	047105	042524	ENTDRV: .ASCIZ <15><12>/ENTER I.D. FOR DRV #/
10545	052502	020122	027111	027104	
10546	052510	043040	051117	042040	

10547	052516	053122	021440	000	
10548	052523	072	000040		COLON: .ASCIZ /: /
10549	052526	005015	040504	042524	DATEIS: .ASCIZ <15><12>/DATE: /
10550	052534	020072	000		
10551	052537	015	047412	042520	IDIS: .ASCIZ <15><12>/OPERATOR I.D.: /
10552	052544	040522	047524	020122	
10553	052552	027111	027104	020072	
10554	052560	000			
10555	052561	015	005012	051104	HEDLIN: .ASCIZ <15><12><12>/DRV DRV I.D./<15><12>
10556	052566	020126	042040	053122	
10557	052574	044440	042056	006456	
10558	052602	000012			
10559	052604	047516	042516	005015	NONE: .ASCIZ /NONE/<15><12>
10560	052612	000			
10561	052613	102	042101	052040	ENTADR: .ASCIZ @BAD TRK/SEC ADRS FOR DRV #@
10562	052620	045522	051457	041505	
10563	052626	040440	051104	020123	
10564	052634	047506	020122	051104	
10565	052642	020126	000043		
10566	052646	020077	047111	040526	BADENT: .ASCIZ /? INVALID ENTRY/<15><12>
10567	052654	044514	020104	047105	
10568	052662	051124	006531	000012	
10569	052670	005015	051120	043517	INTDON: .ASCIZ <15><12>/PROGRAM INITIALIZATION COMPLETE/<15><12><12>
10570	052676	040522	020115	047111	
10571	052704	052111	040511	044514	
10572	052712	040532	044524	047117	
10573	052720	041440	046517	046120	
10574	052726	052105	006505	005012	
10575	052734	000			
10576		052736			.EVEN

10577					
10578	052736	053204			PARLST: .WORD PAR1 ;PARAMETER ENTRY LIST
10579	052740	001300			.WORD MAXDL ;LOCATION
10580	052742	053214			.WORD PAR2 ;PARAMETER NAME
10581	052744	001304			.WORD INTRVL
10582	052746	053424			.WORD PAR19
10583	052750	001276			.WORD PASCNT
10584	052752	053224			.WORD PAR3
10585	052754	001306			.WORD CMPLMT
10586	052756	053234			.WORD PAR4
10587	052760	001324			.WORD MAXCYL
10588	052762	053244			.WORD PAR5
10589	052764	001326			.WORD MINCYL
10590	052766	053254			.WORD PAR6
10591	052770	001330			.WORD MAXTRK
10592	052772	053264			.WORD PAR7
10593	052774	001332			.WORD MINTRK
10594	052776	053274			.WORD PAR8
10595	053000	001334			.WORD MAXSEC
10596	053002	053304			.WORD PAR9
10597	053004	001336			.WORD MINSEC
10598	053006	053404			.WORD PAR17
10599	053010	001342			.WORD BEGCOD
10600	053012	053414			.WORD PAR18
10601	053014	001340			.WORD BEGPAT
10602	053016	177777			.WORD -1

10603	053020	053374				.WORD	PAR15	
10604	053022	001322				.WORD	ENDET	
10605	053024	053314				.WORD	PAR10	
10606	053026	001310				.WORD	FORMAT	
10607	053030	053324				.WORD	PAR11	
10608	053032	001170				.WORD	\$RPADR	
10609	053034	053334				.WORD	PAR12	
10610	053036	001172				.WORD	\$RPVEC	
10611	053040	053344				.WORD	PAR13	
10612	053042	001312				.WORD	SAVL0D	
10613	053044	053354				.WORD	PAR14	
10614	053046	001314				.WORD	RATIO	
10615	053050	053364				.WORD	PAR15	
10616	053052	001316				.WORD	AUTOCK	
10617	053054	053434				.WORD	PAR20	
10618	053056	001320				.WORD	NOTPRT	
10619	053060	000000				.WORD	D	;TABLE TERMINATOR
10620								
10621	053062	047105	042524	020122	ASKPAR:	.ASCIZ	/ENTER PARAMETERS:	/
10622	053070	040520	040522	042515				
10623	053076	042524	051522	020072				
10624	053104	000040						
10625	053106	005015	051105	047522	NGPAR:	.ASCIZ	<15><12>/ERROR IN PARAMETER:	/
10626	053114	020122	047111	050040				
10627	053122	051101	046501	052105				
10628	053130	051105	020072	000040				
10629	053136	020055	042522	042455	REPAR:	.ASCIZ	/- RE-ENTER PARAMETERS/<15><12>	
10630	053144	052116	051105	050040				
10631	053152	051101	046501	052105				
10632	053160	051105	006523	000012				
10633	053166	020040	034050	020051	OCT8:	.ASCIZ	/ (8) /	
10634	053174	000						
10635	053175	040	030450	024460	DEC10:	.ASCIZ	/ (10) /	
10636	053202	000040						
10637								
10638	053204	040515	042130	020114	PAR1:	.ASCIZ	/MAXDL /	
10639	053212	000040						
10640	053214	047111	051124	046126	PAR2:	.ASCIZ	/INTRVL /	
10641	053222	000040						
10642	053224	046503	046120	052115	PAR3:	.ASCIZ	/CMPLMT /	
10643	053232	000040						
10644	053234	040515	041530	046131	PAR4:	.ASCIZ	/MAXCYL /	
10645	053242	000040						
10646	053244	044515	041516	046131	PAR5:	.ASCIZ	/MINCYL /	
10647	053252	000040						
10648	053254	040515	052130	045522	PAR6:	.ASCIZ	/MAXTRK /	
10649	053262	000040						
10650	053264	044515	052116	045522	PAR7:	.ASCIZ	/MINTRK /	
10651	053272	000040						
10652	053274	040515	051530	041505	PAR8:	.ASCIZ	/MAXSEC /	
10653	053302	000040						
10654	053304	044515	051516	041505	PAR9:	.ASCIZ	/MINSEC /	
10655	053312	000040						
10656	053314	047506	046522	052101	PAR10:	.ASCIZ	/FORMAT /	
10657	053322	000040						
10658	053324	051044	040520	051104	PAR11:	.ASCIZ	/\$RPADR /	

```

10659 053332 000040
10660 053334 051044 053120 041505 PAR12: .ASCIZ /SRPVEC /
10661 053342 000040
10662 053344 040523 046126 042117 PAR13: .ASCIZ /SAVLOD /
10663 053352 000040
10664 053354 040522 044524 020117 PAR14: .ASCIZ /RATIO /
10665 053362 000040
10666 053364 052501 047524 045503 PAR15: .ASCIZ /AUTOCK /
10667 053372 000040
10668 053374 047105 042504 020124 PAR16: .ASCIZ /ENDET /
10669 053402 000040
10670 053404 042502 041507 042117 PAR17: .ASCIZ /BEGCOD /
10671 053412 000040
10672 053414 042502 050107 052101 PAR18: .ASCIZ /BEGPAT /
10673 053422 000040
10674 053424 040520 041523 052116 PAR19: .ASCIZ /PASCNT /
10675 053432 000040
10676 053434 047516 050124 052122 PAR20: .ASCIZ /NOTPRT /
10677 053442 000040
10678
10679 .EVEN
10680
10681 ;;*****
10682 .SBTTL PATCH AREA
10683
10684 ;;*****
10685
10686
10687 053444 000100 .BLKW 100 ;PATCH AREA
10688
10689 ;;*****
10690
10691 053644 000000 000000 000000 CYLDER: .WORD 0,0,0,0 ;HEADER BUFFER FOR 'READHD' ROUTINE
10692 053652 000000
10693
10694 053654 ENDPGM = . ;LAST LOCATION OF PROG + 2
10695
10696 000001 .END

```


CYLDER	053644	3799*	3800	3803*	3837*	5103	5211	5214	5217	5220	9356	10691*		
DATAPK	023026	5697	5859*											
DATA0	042712	9552	9555*											
DATA1	042752	9537	9538	9539	9540	9541	9542	9543	9544	9545	9546	9547	9548	9549
		9550	9551	9553	9572*									
DATE	001234	2762*	2974*	2975*	2976*	2977*	5824	5827						
DATEIS	052526	5826	10549*											
DCKER	005476	3572	3580*											
DCKER1	005514	3584*	3872											
DDISP =	177570	2568*	2730	2943										
DEASGN	022452	5687	5778*											
DEASSG	052015	3219	10486*											
DEC10	053175	3037	10635*											
DF2	047352	2878	2899	10223*										
DF3	047360	2885	10226*											
DF4	047363	2892	10228*											
DH14	046235	5024	10094*											
DH15	046341	5030	10108*											
DH16	046440	5034	10120*											
DH2	046073	2876	2897	10073*										
DH3	046150	2883	10082*											
DH4	046176	2890	10087*											
DISPLA	001142	2730*	2943*	2951*	6698*									
DISPLY=	104412	3460	3490	3508	3519	3583	3652	3657	3665	3681	3720	3744	3766	3808
		3834	3849	3871	3877	3895	3905	3914	3929	3945	3965	3976	3995	4007
		4012	4013	4014	4017	4020	4023	4026	4029	4032	4035	4038	4059	4077
		4224	4229	4230	4231	4244	4247	4250	4257	4261	4314	4319	4351	4354
		4357	4360	4363	4366	4369	4372	4374	4375	4409	4415	4420	4422	4502
		4979	4982	4983	4984	4989	4990	4994	5000	5001	5007	5012	5013	5023
		5024	5025	5026	5030	5034	5044	5046	5054	5056	5060	5067	5069	5073
		5078	5083	5089	5094	5097	5102	5105	5109	5122	5125	5128	5133	5137
		5138	5143	5144	5149	5156	5159	5160	5163	5168	5171	5174	5179	5190
		5194	5203	5205	5210	5213	5216	5219	5222	5223	5228	5231	5232	5235
		5240	5241	5246	5249	5251	5256	5261	5265	5266	5271	5276	5281	5284
		5289	5294	5295	5303	5308	5313	5316	5319	5320	5328	6285	6443	7209*
DISPRE	000174	2696*	2951											
DONE	005202	3432	3529*											
DPINT	030606	7280*	7486	7489	7565*	7590	7595	7982	8020	8161	8163*	8222	8260	8272
		8282*												
DPROS	030616	7293*	7599	7673*	8023	8175*	8224	8232	8274	8291*				
DRIVE	001256	2766*	6335*	10130	10134	10137								
DRNUM	052040	3456	3478	3504	6008	6049	6053	10490*						
DROP	023622	3466	3514	6000*	6020									
DROPNG	051727	6007	10475*											
DRVACT	030546	7240*	7607	7767*	7814*	7831*	7841	7852*	7925*	7992	8053	8066	8098*	8105*
		8114	8128*	8233*	8239	8246*								
DRVCLR=	000111	2672*												
DRVER	006572	3576	3743*											
DRVINT	031160	7479	7512*	7593	8145	8164	8237							
DRVQUE	036744	7602	7619	7965	8536*									
DRVSTA	030556	3150	5739	7254*	7469*	7470*	7471*	7472*	7496*	7513*	7556*	7558*	7564*	7586
		7597	7625	7644	7768*	7980	8030	8038	8091*	8099*	8166	8283*		
DRV TYP	030566	3158	3160	3162	5762	5764	5766	7267*	7516*	7529*				
DSWR =	177570	2567*	2729	2942										
DTEER	007440	3569	3868*											
DTUM	030700	5735	7386*	7465	7656	7711*	7832	7835*	7844	7858*	7914	7926*	8026	8227

M.DP42	024420	6147	6152#						
M.DP44	024452	6159	6164#						
M.DP50	024464	6134	6168#						
NEDCLK	052255	5399	10515#						
NEWASN	022134	5683	5715#						
NEWUNT	041672	3232	3248	3249*	5743*	9380#			
NGPAR	053106	3127	10625#						
NOMTCH	010262	4004#	4136						
NONE	052604	5844	10559#						
NOTNUF	052230	5341	10511#						
NOTPRS	051502	3166	5770	10447#					
NOTPRT	001320	2806#	4959	10618					
NOTRP	051465	3164	5768	10444#					
NOTSAF	051517	3168	5772	10450#					
NRML	023726	3364	6025#						
NRMLX	024174	6032	6035	6037	6069	6076#			
NRML1	023760	6026	6029	6033#					
NRML2	024002	4894	6028	6031	6034	6038#			
OCNT	025130	6257*	6286*	6298#					
OCT8	053166	3042	10633#						
OFFCOD	042232	3624	9447#						
OFFSET=	000115	2674#	4450						
OFFST	013040	3627	4449#						
OFLIN	005060	3447	3502#						
OFMSG0	042260	9455	9463#						
OFMSG1	042313	9456	9468#						
OFMSG2	042347	9457	9473#						
OFMSG3	042403	9458	9478#						
OFMSG4	042437	9459	9483#						
OFMSG5	042473	9460	9488#						
OFMSG6	042530	9461	9493#						
OFMTBL	042242	5248	9455#						
OMODE	025132	6256*	6261	6264*	6275*	6300#			
OPERID	001246	2764#	2981*	2982*	2983*	5828	5831		
OPIER	007330	3549	3846#						
OPIER1	007374	3855#							
OPMNE	042152	5006	9434#						
OPT	031730	7609	7638#	7946	8138	8176			
ORDERQ	041616	2961	3299	3333	3352	9368#			
PACK	001232	2761#	3058*	5715*	5749	5751	5859*	5865*	
PARER	007462	3552	3876#						
PARLST	052736	3024	10578#						
PARQ	041760	3284	3305	3339	3372	3374	3376	3402	9391#
PAR1	053204	3070	10578	10638#					
PAR10	053314	10605	10656#						
PAR11	053324	10607	10658#						
PAR12	053334	10609	10660#						
PAR13	053344	10611	10662#						
PAR14	053354	3101	10613	10664#					
PAR15	053364	10615	10666#						
PAR16	053374	10603	10668#						
PAR17	053404	3121	10598	10670#					
PAR18	053414	3105	10600	10672#					
PAR19	053424	10582	10674#						
PAR2	053214	10580	10640#						
PAR20	053434	10617	10676#						

5924*	5934*	5937*	5939*	5940*	5946*	6098	6105*	6107	6110*	6131*	6138*	6143*
6148*	6160*	6178	6194*	6207*	6210*	6214*	6335	6352*	6360*	6362*	6364*	6367*
6370	6395*	6397	6399	6407*	6408	6749	6750*	6755	6759	6761	6766*	6863
6876*	6877	6881	6905*	6935	6958*	6974	6977*	6980*	6984*	6985*	6987*	6988*
6990	6992*	7006	7014*	7015	7016	7018*	7022	7023*	7044*	7050*	7056*	7119*
7123	7124*	7136*	7138*	7140*	7460*	7462*	7463	7466*	7467	7478*	7481*	7482*
7484*	7486	7489	7491*	7513*	7514	7517	7547	7556*	7558*	7564*	7565*	7584*
7586	7588	7590	7595	7597	7599	7601	7606*	7607	7625	7640	7644	7673*
7674	7694	7711	7714	7732	7735	7767*	7768*	7769*	7811*	7812*	7813*	7814*
7830*	7831*	7832	7839*	7841	7844	7852*	7853*	7854*	7856*	7877	7884*	7885
7902*	7914*	7925*	7927*	7928*	7929*	7933	7978*	7980	7982	7984*	7989*	7990*
7992	7998*	8001*	8012*	8015*	8020	8023	8026	8030	8032	8038	8047	8053
8055*	8056*	8057*	8059	8066	8091*	8093	8098*	8099*	8100*	8101*	8102*	8105*
8108	8111*	8114	8121	8125	8127*	8128*	8129	8137	8140	8150*	8151*	8152*
8153	8154	8157*	8158*	8159*	8161	8163*	8166	8175*	8188*	8196*	8198	8216
8222	8224	8227	8233*	8234*	8235*	8239	8242	8246*	8247*	8250*	8251	8258
8260	8262	8264	8272	8274	8282*	8283*	8291*	8424	8503*	8504	8508*	8509
8521*	8522*	8523*	8524*	8536	8538*	8539*	8540*	8541*	8542	8544*	8545*	8558
8560*	8561	8562*	8572*	8573*	8574	8575*	8576	8578*	8579*	8602	8626*	
2573*	3229*	3244	3246*	3248*	3251	3267*	4040	4042*	4108*	4110*	4126*	4128
4147*	4410*	4418*	4559*	4564*	4579*	4585*	4594*	4602*	4607*	4619	4620*	4628*
4632*	4639*	4658*	4666*	4674*	4691*	4694*	4934	4937*	4956*	4964*	5409	5427*
5428*	5437*	5468	5472*	5473	5477*	5478	5482*	5483	5487*	5493*	5494	5498
5502	5506	5524	5529*	5530*	5533*	5668	5708*	5890	5905*	5906	5945*	5956
5958	5960	5962	5972	5973*	5980*	6097	6102*	6111*	6137*	6179	6183*	6188
6193*	6211*	6353*	6361*	6363*	6365*	6864	6875*	6879*	6882	6889*	6890*	6891
6896*	6904*	6936	6957*	6975	6978*	6981*	6991*	7041*	7044	7045*	7051*	7052*
7057*	7058*	7125*	7126	7127*	7142*	7143*	7461*	7463	7465*	7467	7582*	7583*
7584	7604	7611*	7627*	7629*	7642	7647*	7652	7691	7692	7707	7715	7719
7724	7736	7739	7746	7756	7780	7787	7789	7790	7806*	7821	7823*	7843*
7847	7849*	7881	7887	7930*	7932*	7944	7950*	7961	7963	7967*	7968*	7969*
7970*	7971*	7972*	7973*	7979*	7985*	8068*	8081	8083*	8094*	8104*	8133*	8168
8171*	8226*	8230*	8241*	8245*	8286	8288*	8293	8295*	8393	8396	8397*	8405*
8409*	8497*	8498*	8499*	8500*	8501*	8504*	8509*	8540	8557*	8561*	8574*	8603
8625*												
2574*	3023*	3028*	3034	3145*	3173*	3196*	3201*	3228*	3236*	4139*	4149*	4277*
4280*	4508	4513*	4515*	4518*	4520*	4524*	4581*	4582*	4583	4584	4596*	4597*
4598*	4599	4672*	4676*	4678*	5410	5413*	5419*	5432*	5436*	5669	5707*	5717*
5722*	5753*	5781*	5789*	5798*	5816*	5820*	5835*	5849*	5891	5896*	5900*	5903*
5935*	5944*	6057	6061*	6064*	6067*	6096	6103*	6108	6112*	6136*	6155*	6158*
6164*	6180	6184*	6189	6192*	6212*	6258	6267*	6273*	6274*	6277*	6282*	6283*
6284	6293*	6622	6624*	6625	6628*	6629	6636*	6637	6639	6647	6649	6655
6657	6659*	6665*	6805	6814*	6820*	6821*	6824*	6829*	6830*	6831	6840*	6865
6873*	6874*	6888*	6891*	6900*	6901*	6903*	6937	6956*	7049*	7054*	7060*	7061
7122*	7135*	7142	7473*	7474*	7475*	7514*	7515*	7516*	7529*	7562*	7674*	7675*
7676*	7692*	7695*	7697	7698	7699	7703	7719*	7720*	7722*	7723	7736*	7737
7744	7751	7761	7763	7765	7770	7775	7785	7787*	7794*	7799	7801	7816
7840*	7851*	7855*	7887*	7888*	7889*	7890*	7895*	7897*	7898	7900	7999*	8010
8011*	8013	8016*	8189*	8190	8192*	8197	8236*	8248*	8249	8276*	8284*	8290*
8399*	8401	8408	8410	8411	8502*	8505*	8507*	8510*	8604	8624*		
2575*	3024*	3025	3032	3144*	3150	3157*	3158	3160	3162	3169*	3176*	3230*
3235*	3240	3250*	4138	4271*	4273*	4274	4282*	4509	4511*	4512*	4514*	4517*
4519*	4523*	4622*	4623*	4624*	4625	4668*	4669*	4671	4679	5411	5412*	5415
5417*	5423	5431*	5435*	5461	5525	5527*	5528*	5532*	5670	5706*	5716*	5720*
5723	5733	5737	5739	5742*	5743*	5744	5752*	5755*	5761*	5762	5764	5766
5773*	5778*	5782*	5783	5785*	5786*	5787*	5788*	5791*	5803*	5806*	5807	5809*

R2 =%000002

R3 =%000003

R4 =%000004

SEARCH= 000131	2678#													
SECOND 021712	2969*	4984	5605	5615*	5616	5618*	5619*	5620	5622*	5656#				
SEEK = 000105	2670#													
SEEKFG 030656	7368#	7461	7658											
SELDIV= 000145	2681#													
SELPAR 014416	3291	3375	4716#											
SEPART 023560	5908*	5958	5984#											
SETFMT= 000143	2680#													
SETVEC 002740	3108	3110	3112	3114	3116	3120	3137#							
SET. IE 036276	7488	7521	7623	7671	7862	7996	8422#							
SHDTYP 020572	5422	5443#	5526											
SIXTEE 021720	2965*	2966*	5611*	5613*	5614*	5661#								
SIZMEM 001770	2987#													
SKIER 010074	3575	3964#												
SP =%000006	2579#	2932*	2940*	2948*	2952	2973	2980	3013	3033*	3039	3044	3056*	3070*	
	3077*	3081*	3085*	3089*	3093*	3097*	3101*	3105*	3121*	3126	3189	3293*	3295	
	3326*	3328	3349	3358*	3365	3389*	3390*	3392	3393	3596*	3597*	3599	3600	
	3607*	3609	3756*	3757*	3793*	3794*	4015*	4018*	4021*	4024*	4027*	4030*	4033*	
	4036*	4071*	4072*	4073*	4074	4208*	4209*	4216	4217	4241*	4242*	4245*	4248*	
	4258*	4270*	4272	4284*	4285	4291*	4292*	4293*	4294*	4296	4298*	4299	4305*	
	4315*	4316*	4334*	4335*	4337	4347*	4348*	4350	4352*	4355*	4358*	4364*	4367*	
	4370*	4394*	4395*	4396*	4397*	4399	4401*	4402	4403*	4411*	4416*	4459	4460	
	4469*	4470	4483*	4500	4508*	4509*	4510*	4522	4523	4524	4555*	4568	4570*	
	4592*	4615	4619*	4630	4632	4700*	4705	4772*	4773*	4774*	4777*	4779*	4780	
	4835*	4860	4897	4918*	4919*	4921	4922	4933*	4934*	4940*	4941*	4943	4947	
	4949	4952	4962*	4964	4965	5014*	5037	5042*	5057*	5061*	5062*	5070*	5074*	
	5075*	5079*	5080*	5095*	5098*	5103*	5106*	5107*	5112*	5115*	5123*	5126*	5134*	
	5139*	5140*	5145*	5146*	5157*	5161*	5169*	5172*	5175*	5176*	5177*	5182*	5185*	
	5192*	5195*	5197*	5198*	5199*	5200*	5202*	5211*	5214*	5217*	5220*	5229*	5233*	
	5262*	5263*	5277*	5278*	5282*	5285*	5286*	5290*	5291*	5304*	5305*	5309*	5310*	
	5314*	5317*	5348*	5349*	5350*	5352*	5353*	5354*	5355*	5357*	5359*	5370*	5389*	
	5398*	5408*	5409*	5410*	5411*	5435	5436	5437	5438	5464*	5468*	5473*	5478*	
	5483*	5494*	5498*	5502*	5506*	5510*	5511*	5512*	5513*	5514*	5523*	5524*	5525*	
	5532	5533	5534	5610*	5641*	5649	5666*	5667*	5668*	5669*	5670*	5671*	5705	
	5706	5707	5708	5709	5710	5731*	5757*	5870*	5873*	5874*	5875	5879	5883	
	5889*	5890*	5891*	5905	5944	5945	5946	5967*	5970	5971*	5972*	5974*	5975	
	6040*	6057*	6058*	6066	6067	6080*	6081*	6082*	6084	6085	6094*	6095*	6096*	
	6097*	6098*	6099*	6100	6102	6103	6107*	6108*	6109	6110	6111	6112	6113	
	6114	6115*	6125*	6126*	6127*	6128*	6129*	6130*	6131	6133	6135*	6140	6142*	
	6143	6145*	6146	6150*	6152*	6153	6156*	6165*	6168*	6177*	6178*	6179*	6180*	
	6181*	6182*	6184	6186	6188*	6189*	6190	6191	6192	6193	6194	6195	6208*	
	6217*	6219	6231*	6232	6242	6246	6247*	6258*	6259*	6260*	6266	6291	6292	
	6293	6294*	6304*	6320	6321*	6325	6326*	6329*	6344*	6346	6354*	6356	6358	
	6366*	6367	6369	6370*	6372	6378*	6379*	6386*	6388	6394*	6401*	6402*	6403*	
	6404*	6405*	6408*	6411*	6413	6419*	6420*	6435*	6436	6445	6446*	6475*	6476	
	6477*	6479	6480	6481*	6484	6486*	6488*	6494	6526*	6529*	6530	6531	6532*	
	6534	6535	6536*	6542	6568*	6569*	6574	6576	6582	6590	6607*	6608*	6611*	
	6612*	6622*	6623*	6628	6631	6635*	6642	6646*	6664	6665	6666*	6667*	6668*	
	6703	6725*	6730*	6749*	6755*	6759*	6766	6767	6797*	6798	6799	6800*	6905*	
	6806*	6807*	6813	6838	6839	6840	6841*	6842*	6862*	6863*	6864*	6865*	6866*	
	6867*	6868	6871*	6884	6886*	6888	6898	6900	6902	6903	6904	6905	6906	
	6908*	6909*	6934*	6935*	6936*	6937*	6938*	6939*	6940*	6941*	6942*	6943*	6950*	
	6951*	6952*	6953*	6954	6955	6956	6957	6958	6959	6973*	6974*	6975*	6991	
	6992	6993	7005*	7006*	7007*	7008*	7009	7012	7019*	7020	7021	7023	7024	
	7041	7043*	7100	7101*	7103*	7104*	7119	7132*	7156	7157*	7159*	7160*	7171*	
	7172	7186*	7187*	7457*	7493	7528	7534*	7538*	7545	7551*	7578*	7615	7639*	

.SERRO	1#	2520#	6581
.SERRT	1#	2520#	6716
.SMULT	1#		
.SPOWE	1#		
.SRAND	1#	2520#	6961
.SRDDE	1#		
.SFDOC	1#		
.SREAD	1#		
.SR2AZ	1#		
.SSAVE	1#	2520#	6916
.SSB2D	1#	2520#	7089
.SSB2O	1#	2520#	7146
.SSCOP	1#		
.SSIZE	1#	2520#	6997
.SSUPR	1#		
.STRAP	1#	2520#	7163
.STYPB	1#		
.STYPD	1#	2520#	6849
.STYPE	1#		
.STYPO	1#	2520#	6772
.S4OCA	1#		
.1170	1#		

ADC	4534	4542	4544	4548	4707	4711	6132	6144	6145	6149	6150	6152	6161	6215	6984
ADD	6987	7057													
	3050	3052	3200	3209	3216	3234	3246	3266	3267	3316	4039	4040	4073	4097	4282
	4284	4292	4302	4305	4320	4341	4395	4408	4483	4512	4517	4533	4541	4543	4547
	4566	4574	4582	4598	4601	4618	4624	4627	4633	4636	4706	4710	4727	4737	4747
	4757	4777	4858	4867	4936	4953	4962	5006	5198	5278	5286	5291	5305	5310	5370
	5389	5398	5418	5425	5428	5430	5472	5477	5482	5487	5493	5530	5757	5851	5898
	5937	5970	5975	5979	6060	6131	6133	6143	6146	6148	6151	6160	6162	6165	6169
	6214	6216	6243	6263	6329	6367	6379	6405	6408	6420	6481	6526	6536	6738	6800
	6810	6881	6983	6985	6986	6988	7056	7058	7103	7143	7159	7695	7696	7722	7797
	7855	7897	8312	8366	8408	8541	8575								
ASL	3157	4072	4298	4329	4401	4573	4597	4623	4830	5003	5004	5005	5019	5247	5742
	5761	5786	5809	5966	5968	5969	6003	6074	6360	6362	6364	6401	6403	6404	6735
	6736	6737	6979	7179	7515	7675	7811	7927	8055	8100	8150	8157	8522	8539	8560
	8573														
ASLB	3259	6886	7985	8016											
ASR	3169	4311	4312	4313	4316	4532	4575	5177	5752	5773	5788	5874	7135	7813	7889
	7890	7929	8057	8102	8152	8159	8524	8545	8562	8579					
BCC	6213	6887	7986												
BOS	6134	6159													
BEQ	3015	3017	3026	3108	3110	3112	3114	3116	3151	3159	3161	3163	3174	3190	3193
	3206	3208	3213	3215	3245	3265	3276	3281	3286	3288	3296	3315	3350	3354	3356
	3373	3387	3404	3416	3431	3483	3530	3533	3539	3542	3545	3548	3551	3554	3557
	3560	3563	3565	3568	3574	3612	3615	3617	3619	3625	3629	3687	3691	3698	3700
	3725	3754	3771	3791	3801	3813	3919	3934	3951	4057	4075	4091	4115	4118	4121
	4124	4143	4145	4148	4153	4165	4235	4256	4275	4297	4327	4339	4362	4392	4400
	4414	4444	4468	4480	4486	4493	4531	4538	4546	4560	4565	4572	4578	4580	4595
	4600	4626	4655	4660	4670	4675	4692	4695	4709	4734	4744	4754	4760	4776	4805
	4811	4857	4866	4977	4999	5011	5041	5121	5155	5297	5322	5449	5489	5634	5644
	5675	5719	5740	5750	5754	5763	5765	5767	5780	5784	5790	5794	5805	5808	5825
	5829	5837	5850	5907	5926	5936	5957	5959	5965	6020	6026	6141	6154	6280	6349
	6351	6355	6391	6393	6396	6438	6440	6577	6643	6648	6697	6700	6740	6745	6751
	6762	6827	7130	7487	7520	7524	7531	7643	7651	7762	7764	7796	7800	7822	7842
	7845	7848	7945	7960	7962	8021	8024	8054	8082	8115	8162	8169	8228	8240	8243
	8275	8287	8294	8326	8374	8394	8398	8537	8559						
BGE	7657	7721	7896	7901	7981	8337									
BGT	4722	6157	6287	6357	6398	6834	6895	7129	7176	7587	7645	7886	8002	8031	8067
	8167	8193	8199	8252											
BHI	3067	4107	4129	4322	4343	4540	4809	4888	5914	5923	5933	6028	6034		
BHIS	3073	3076	3080	3084	3088	3092	3096	3100	3104	3396	3603	4557	4613	4647	4651
	4781	4792	5545	5557	5569	5581	5593	5677	5963	6031					
BIC	3062	3799	4309	4310	4335	4336	4348	4349	4726	4818	6277	6366	6569	6612	6824
	7142	7482	7554	7856	8361										
BICB	3143	3239	3436	4813	5017	5460	5680	5838	5872	5974	5988	6009	6042	7640	7971
	7972														
BIS	3803	3837	4099	4337	4350	4662	4815	6282	6283	6829	6830	6889	6890	7060	7611
	7627	7629	7647	7806	7849	7932	7950	8068	8083	8094	8104	8133	8171	8230	8245
	8288	8295	8329	8426											
BISB	3240	3437	4806	5018	5461	5839	5875	5989	6010	6043	6727	7732	7973		
BIT	3382	3384	3415	3417	3419	3423	3430	3438	3440	3442	3444	3446	3448	3482	3529
	3532	3535	3538	3541	3544	3547	3550	3553	3556	3559	3562	3564	3567	3573	3589
	3614	3616	3618	3663	3686	3699	3701	3724	3753	3760	3770	3790	3797	3812	3918
	3933	3950	4048	4050	4090	4144	4238	4391	4485	4492	4717	4837	4844	4856	4895
	4976	4998	5028	5120	5154	5296	5321	5695	6017	6044	6315	6336	6699	6706	7519
	7523	7621	7650	7669	8013	8040	8155	8318	8325	8373	8429				
BITB	3264	4087	4535	4654	4659	4682	4775	4810	8129	8258					

BLE	4127	4169	4667	7598	8033	8039	8126	8412								
BLO	2989	3006	4174	4178	4887	5679	5961	6029	6035	6037	6047	6638				
BLOS	3069	4563	4890	6626	7464	7468										
BLT	6288	6359	6400	6489	6835	6878	6894	7053	7626	7653	7899	7915	7991	8191	8356	
BMI	3152	3414	3692	4159	4203	4221	4481	4879	4903	4955	5741	5912	5921	5931	6319	
	6885	7552	7659	7938	8065	8080	8097	8221	8324	8404						
BNE	2931	2945	2964	2998	3003	3035	3047	3049	3118	3120	3199	3202	3233	3237	3274	
	3279	3301	3304	3307	3329	3335	3338	3341	3368	3383	3385	3406	3418	3420	3424	
	3439	3441	3443	3445	3447	3449	3536	3590	3622	3650	3664	3702	3761	3798	4005	
	4049	4051	4053	4055	4070	4088	4113	4141	4150	4171	4201	4222	4237	4239	4279	
	4281	4301	4406	4419	4488	4490	4496	4516	4521	4536	4586	4603	4629	4640	4677	
	4683	4718	4768	4785	4817	4827	4838	4845	4875	4877	4896	4939	4944	4950	4957	
	4960	4981	5029	5416	5420	5424	5433	5543	5555	5567	5579	5591	5601	5612	5617	
	5621	5626	5630	5636	5646	5682	5686	5690	5694	5696	5700	5724	5734	5818	5821	
	5843	5901	6018	6045	6065	6218	6234	6239	6278	6316	6337	6478	6485	6523	6533	
	6571	6579	6630	6632	6650	6658	6707	6728	6754	6825	6883	6982	7064	7483	7490	
	7533	7555	7589	7591	7596	7600	7605	7608	7622	7670	7738	7745	7752	7766	7776	
	7786	7833	7857	7883	7964	7983	7993	8000	8014	8017	8027	8041	8109	8131	8156	
	8185	8223	8225	8259	8261	8263	8265	8273	8319	8354	8402	8430	8506	8511	8543	
	8577															
BPL	3027	3571	3613	3630	4946	5351	5356	5736	6276	6472	6493	6517	6521	6541	6610	
	6712	6823	6869	6899	7492	7546	7808	7940	8007	8090	8135	8270				
BR	2947	3000	3018	3029	3040	3045	3051	3154	3156	3165	3167	3177	3203	3210	3217	
	3224	3238	3247	3268	3283	3289	3311	3317	3345	3472	3592	3642	3644	3674	3676	
	3678	3695	3704	3706	3732	3779	3821	3857	3885	3890	3955	3957	3985	3990	4011	
	4109	4130	4135	4137	4151	4157	4163	4167	4206	4228	4283	4303	4331	4345	4373	
	4403	4421	4484	4491	4567	4576	4587	4608	4631	4641	4649	4653	4680	4686	4696	
	4724	4770	4783	4786	4807	4814	4829	4841	4868	4880	4907	4948	4951	4958	5045	
	5055	5358	5369	5388	5397	5421	5426	5639	5684	5688	5692	5698	5702	5756	5769	
	5771	5792	5799	5812	5815	5822	5845	5853	5910	5918	5928	5929	5938	5942	5976	
	5978	6032	6052	6069	6147	6225	6227	6229	6237	6269	6290	6324	6368	6371	6377	
	6410	6412	6418	6474	6491	6519	6539	6581	6585	6641	6652	6654	6661	6733	6757	
	6764	6801	6816	6837	6880	6897	7017	7055	7144	7178	7497	7522	7557	7566	7594	
	7603	7610	7618	7620	7624	7628	7630	7648	7655	7661	7662	7664	7672	7677	7733	
	7743	7750	7760	7784	7798	7820	7917	7947	7949	7955	7966	7988	8044	8052	8069	
	8084	8095	8146	8149	8160	8174	8195	8238	8256	8281	8289	8298	8328	8363	8406	
	8432															
BVS	6406	6409														
CLC	2995	6163														
CLR	2929	2959	2962	3023	3057	3058	3144	3194	3221	3222	3229	3230	3231	3249	3293	
	3312	3326	3362	3389	3390	3407	3594	3596	3597	3607	3666	3756	3757	3793	3794	
	3953	4042	4094	4104	4105	4214	4215	4293	4325	4333	4344	4396	4514	4519	4664	
	4665	4668	4773	4859	4913	4918	4919	4940	4941	4991	4995	5015	5061	5062	5074	
	5079	5106	5139	5145	5199	5262	5348	5353	5366	5368	5379	5381	5382	5392	5412	
	5527	5715	5716	5778	5803	5834	5916	5955	5971	6000	6063	6071	6101	6104	6105	
	6135	6142	6183	6185	6206	6207	6230	6267	6352	6353	6394	6528	6557	6587	6589	
	6623	6646	6726	6814	6872	6875	7049	7125	7462	7478	7485	7516	7583	7834	7839	
	7840	7859	7931	7967	7978	8009	8012	8188	8189	8235	8236	8254	8399	8405	8498	
	8499	8500	8501	8557												
CLRB	2960	2970	3131	4093	4160	4175	4179	5785	6002	6073	6373	6414	6659	6901	7065	
	7496	7558	7564	7614	7768	7830	7831	7852	7853	7902	7920	7925	8105	8111	8127	
	8128	8163	8175	8201	8233	8234	8246	8247	8282	8283	8291	8521				
CLV	6166															
CLV	2930	2944	2963	2988	3005	3066	3068	3072	3075	3079	3083	3087	3091	3095	3099	
CLV	3103	3109	3111	3113	3115	3117	3160	3162	3189	3207	3214	3285	3349	3395	3405	
CLV	3602	3800	4074	4106	4114	4117	4128	4140	4278	4321	4342	4413	4545	4556	4562	

	4599	4612	4625	4646	4650	4708	4721	4780	4791	4808	4886	4938	5544	5556	5568
	5580	5592	5633	5764	5766	5913	5922	5932	6019	6027	6030	6033	6036	6046	6576
	6578	6625	6637	6893	7059	7175	7463	7467	7530	7532	7795	7832	7844	7881	7898
CMPB	7900	8026	8198	8227	8242	8251	8264	8320	8401	8411	8542	8576			
	3014	3016	4112	4170	4173	4177	4300	4405	4539	4759	4784	4889	4943	4949	5616
	5620	5625	5629	5635	5674	5676	5678	5681	5685	5689	5693	5699	5718	5779	5793
	5804	5906	5958	5960	5962	6233	6350	6356	6358	6392	6397	6399	6439	6484	6629
	6647	6649	6657	7604	7652	7737	7744	7751	7761	7763	7765	7775	7785	7799	7885
	7961	7963	8355	8536											
COM	6524	7553													
DEC	3173	3201	3236	4147	4149	4236	4280	4307	4328	4418	4515	4564	4577	4579	4585
	4602	4628	4634	4639	4674	4676	4691	4694	4956	5419	5432	5753	5789	5820	5849
	5900	5935	6064	6156	6217	6240	6441	6636	6734	7062	7128	7491	7990	8001	8335
	8505	8510													
DECB	3621	3697	4495	5350	5355	5623	6275	6286	6488	6822	6833	8572			
EMT	2555														
HALT	2695	3191	3351	3484	4431	4442	4453	4466	4648	4652	4704	5298	5323	5343	5400
	6473	6518	6713	7177											
INC	3028	3176	3235	3623	4102	4131	4155	4180	4606	4607	4611	4735	4745	4755	4765
	5417	5431	5546	5558	5570	5582	5594	5611	5755	5791	5819	5852	5919	5973	5980
	6068	6155	6236	6281	6289	6583	6584	6702	6828	6836	6879	6981	7054	7131	7481
	7854	7984	8015	8196	8250	8332									
INCB	3175	3620	3693	3696	4172	4176	4482	4494	5615	5619	5624	5628	5632	5638	6696
	7884	8538													
IOT	2556														
JMP	2699	2700	2990	3071	3078	3082	3086	3090	3094	3098	3102	3106	3122	3132	3181
	3321	3330	3378	3432	3466	3514	3531	3534	3537	3540	3543	3546	3549	3552	3555
	3558	3561	3569	3572	3575	3576	3658	3755	3762	3792	3802	3804	3872	4181	4323
	4503	4898	5002	5092	5257	5272	5344	5401	5726	5729	5732	5860	5866	6381	6422
	7712	8022	8025	8029	8042	8103	8110	8112	8117	8118	8124	8139	8177		
JSR	2987	3038	3043	3137	3140	3141	3179	3195	3220	3223	3282	3290	3291	3292	3294
	3297	3298	3310	3320	3327	3331	3332	3344	3359	3361	3363	3364	3370	3371	3375
	3377	3391	3399	3408	3421	3422	3425	3459	3461	3462	3463	3464	3465	3481	3489
	3491	3492	3493	3494	3495	3507	3509	3510	3511	3512	3513	3518	3520	3521	3522
	3523	3524	3580	3582	3584	3585	3586	3587	3591	3595	3599	3608	3610	3627	3631
	3632	3633	3634	3635	3636	3638	3639	3640	3641	3643	3645	3646	3648	3651	3653
	3654	3656	3662	3667	3668	3669	3670	3671	3673	3675	3677	3679	3682	3683	3684
	3685	3689	3694	3703	3705	3707	3708	3710	3711	3712	3717	3719	3721	3722	3723
	3726	3727	3728	3731	3733	3734	3736	3737	3738	3743	3745	3746	3747	3748	3758
	3759	3763	3765	3767	3768	3769	3772	3773	3774	3775	3778	3780	3781	3783	3784
	3785	3795	3796	3805	3807	3809	3810	3811	3814	3815	3816	3817	3820	3822	3823
	3825	3826	3827	3832	3833	3835	3836	3838	3839	3840	3841	3846	3848	3850	3851
	3852	3853	3856	3858	3859	3861	3862	3863	3868	3870	3876	3878	3879	3880	3881
	3884	3886	3887	3889	3894	3896	3897	3898	3899	3904	3906	3907	3908	3913	3915
	3916	3917	3920	3921	3922	3923	3928	3930	3931	3932	3935	3936	3937	3938	3939
	3944	3946	3947	3948	3949	3954	3956	3958	3959	3964	3966	3967	3968	3969	3970
	3975	3977	3978	3979	3980	3981	3984	3986	3987	3989	3994	3996	3997	3998	3999
	4006	4008	4009	4010	4016	4019	4022	4025	4028	4031	4034	4037	4058	4060	4061
	4062	4063	4064	4076	4078	4079	4080	4081	4082	4116	4119	4122	4125	4132	4134
	4136	4146	4156	4166	4182	4204	4205	4212	4213	4218	4223	4225	4226	4227	4243
	4246	4249	4259	4260	4262	4263	4295	4317	4318	4353	4356	4359	4365	4368	4371
	4398	4412	4417	4429	4440	4451	4464	4478	4498	4499	4687	4702	4716	4720	4723
	4736	4746	4756	4766	4769	4778	4782	4794	4802	4825	4828	4842	4873	4882	4894
	4911	4920	4942	4993	4997	5027	5033	5036	5043	5058	5059	5063	5064	5065	5066
	5068	5071	5072	5076	5077	5081	5082	5088	5091	5096	5099	5104	5108	5113	5114
	5124	5127	5135	5136	5141	5142	5147	5148	5158	5162	5170	5173	5178	5183	5184

MOV

5193	5196	5201	5204	5212	5215	5218	5221	5230	5234	5264	5279	5280	5283	5287
5288	5292	5293	5306	5307	5311	5312	5315	5318	5329	5422	5429	5443	5444	5446
5450	5452	5454	5465	5466	5469	5470	5474	5475	5479	5480	5484	5485	5495	5496
5499	5500	5503	5504	5507	5508	5515	5516	5526	5531	5642	5683	5687	5691	5697
5701	5725	5728	5738	5745	5746	5747	5796	5811	5814	5823	5909	5917	5927	6056
6083	6106	6187	6307	6322	6327	6338	6483	6490	6538	6572	6596	6695	6708	7102
7158	7456	7459	7479	7488	7494	7512	7521	7525	7535	7539	7542	7548	7560	7581
7593	7602	7609	7613	7617	7619	7623	7638	7641	7646	7654	7660	7663	7665	7667
7671	7690	7700	7704	7708	7716	7725	7729	7740	7747	7753	7757	7771	7777	7781
7791	7802	7805	7809	7817	7824	7826	7829	7837	7846	7850	7860	7862	7863	7878
7892	7906	7912	7916	7918	7919	7934	7941	7942	7943	7946	7951	7952	7965	7996
8003	8034	8035	8036	8045	8058	8060	8070	8074	8076	8085	8092	8106	8116	8120
8132	8136	8138	8145	8164	8170	8172	8173	8176	8187	8194	8200	8214	8217	8229
8231	8237	8244	8255	8266	8277	8280	8285	8292	8296	8297	8357	8370	8376	8392
8413	8496	8512												
2928	2932	2934	2935	2936	2937	2940	2941	2942	2943	2948	2950	2951	2952	2955
2956	2957	2958	2961	2965	2967	2968	2969	2973	2974	2975	2976	2977	2980	2981
2982	2983	2991	2992	2993	3004	3007	3013	3024	3025	3032	3033	3039	3044	3056
3059	3060	3070	3074	3077	3081	3085	3089	3093	3097	3101	3105	3121	3126	3138
3139	3142	3145	3153	3155	3164	3166	3168	3196	3197	3204	3211	3218	3228	3248
3251	3254	3256	3260	3261	3263	3277	3284	3295	3299	3302	3305	3308	3309	3318
3319	3325	3328	3333	3336	3339	3342	3343	3352	3353	3358	3365	3366	3369	3374
3376	3388	3394	3397	3401	3402	3588	3593	3601	3604	3605	3609	3672	3688	3729
3730	3776	3777	3818	3819	3854	3855	3882	3883	3952	3982	3983	4015	4018	4021
4024	4027	4030	4033	4036	4071	4095	4096	4098	4100	4101	4108	4110	4138	4139
4161	4162	4208	4209	4210	4211	4216	4217	4241	4245	4248	4258	4270	4271	4272
4274	4277	4285	4290	4291	4294	4306	4308	4315	4324	4332	4334	4340	4346	4347
4352	4355	4358	4364	4367	4370	4393	4394	4397	4410	4411	4416	4462	4469	4508
4509	4510	4511	4513	4518	4522	4523	4524	4529	4555	4559	4561	4568	4570	4581
4583	4584	4592	4593	4594	4596	4604	4605	4609	4610	4615	4617	4619	4620	4621
4622	4632	4635	4637	4638	4657	4658	4661	4663	4671	4672	4673	4678	4679	4684
4685	4688	4689	4690	4693	4700	4701	4705	4719	4725	4732	4742	4752	4758	4764
4771	4772	4774	4779	4790	4793	4796	4801	4824	4835	4839	4840	4843	4850	4851
4852	4853	4855	4860	4865	4883	4884	4885	4893	4899	4900	4904	4908	4933	4934
4935	4937	4961	4964	4965	5014	5020	5021	5022	5031	5032	5035	5037	5039	5042
5057	5070	5095	5098	5103	5112	5115	5123	5126	5134	5157	5161	5169	5172	5175
5182	5185	5192	5195	5197	5200	5202	5211	5214	5217	5220	5229	5233	5248	5277
5282	5285	5290	5304	5309	5314	5317	5352	5357	5364	5365	5371	5376	5377	5378
5383	5384	5385	5386	5387	5390	5393	5394	5395	5396	5402	5408	5409	5410	5411
5413	5414	5427	5435	5436	5437	5438	5464	5468	5473	5478	5483	5494	5498	5502
5506	5510	5523	5524	5525	5529	5532	5533	5534	5610	5613	5640	5641	5648	5649
5666	5667	5668	5669	5670	5671	5672	5705	5706	5707	5708	5709	5710	5717	5721
5722	5731	5743	5744	5748	5759	5768	5770	5772	5781	5787	5795	5798	5810	5813
5816	5833	5835	5859	5865	5870	5873	5879	5880	5881	5882	5883	5889	5890	5891
5896	5897	5899	5903	5905	5908	5915	5939	5940	5944	5945	5946	5967	5977	6004
6040	6057	6058	6059	6061	6066	6067	6075	6080	6081	6082	6084	6094	6095	6096
6097	6098	6099	6100	6102	6103	6107	6108	6109	6110	6111	6112	6113	6114	6115
6125	6126	6127	6177	6178	6179	6180	6181	6182	6184	6186	6188	6189	6190	6191
6192	6193	6194	6195	6208	6224	6226	6228	6231	6232	6242	6246	6247	6258	6259
6260	6266	6273	6291	6292	6293	6294	6304	6305	6306	6317	6320	6321	6325	6326
6334	6335	6344	6346	6347	6370	6378	6386	6388	6389	6402	6411	6419	6435	6436
6442	6445	6446	6475	6476	6480	6486	6529	6530	6535	6558	6559	6560	6562	6588
6607	6608	6622	6624	6635	6665	6666	6667	6668	6698	6703	6725	6730	6739	6744
6749	6750	6752	6755	6759	6766	6767	6797	6805	6806	6807	6813	6820	6838	6839
6840	6841	6842	6862	6863	6864	6865	6866	6867	6868	6873	6876	6896	6902	6903
6904	6905	6906	6908	6909	6934	6935	6936	6937	6938	6939	6940	6941	6942	6943

	6950	6951	6952	6953	6954	6955	6956	6957	6958	6959	6973	6974	6975	6976	6977
	6978	6989	6990	6991	6992	6993	7005	7006	7007	7008	7009	7012	7013	7014	7019
	7020	7021	7022	7023	7024	7041	7042	7043	7044	7045	7046	7047	7048	7100	7101
	7104	7119	7120	7121	7122	7123	7124	7127	7132	7156	7157	7160	7171	7172	7180
	7126	7187	7457	7458	7460	7461	7465	7466	7469	7470	7471	7472	7473	7474	7475
	7476	7477	7484	7493	7514	7517	7528	7529	7534	7538	7545	7562	7563	7578	7579
	7582	7585	7592	7601	7615	7639	7649	7666	7674	7676	7691	7692	7693	7694	7697
	7698	7699	7703	7707	7711	7713	7714	7715	7723	7728	7734	7735	7739	7746	7770
	7787	7794	7801	7812	7816	7823	7825	7835	7838	7843	7851	7858	7861	7876	7877
	7891	7913	7914	7926	7928	7930	7933	7948	7953	7954	7969	7970	7979	7987	7989
	7998	8011	8043	8048	8049	8050	8056	8059	8063	8073	8075	8088	8101	8107	8119
	8122	8140	8141	8142	8143	8144	8151	8153	8158	8202	8212	8213	8215	8216	8226
	8232	8241	8248	8253	8276	8278	8284	8290	8309	8310	8311	8313	8316	8317	8331
	8338	8339	8351	8352	8353	8364	8367	8375	8381	8395	8397	8400	8407	8409	8422
	8423	8424	8425	8434	8497	8502	8503	8504	8507	8508	8509	8523	8540	8544	8561
	8574	8578	8602	8603	8604	8605	8606	8607	8620	8622	8623	8624	8625	8626	
MOV8	3061	3250	3252	3253	3255	3257	3258	3392	3393	3398	3400	3599	3600	3624	3626
	3647	4043	4103	4154	4232	4427	4428	4438	4439	4449	4450	4459	4460	4461	4463
	4669	4728	4738	4748	4795	4812	4819	4836	4846	4847	4848	4849	4881	4891	4892
	4905	4906	4909	4910	4912	4921	4922	4992	4996	5016	5075	5080	5107	5140	5146
	5263	5349	5354	5359	5528	5618	5622	5627	5631	5637	5647	5720	5737	5751	5782
	5806	5924	5934	5972	6001	6072	6235	6241	6255	6256	6257	6261	6264	6265	6284
	6354	6395	6477	6494	6532	6542	6568	6574	6582	6611	6628	6633	6639	6644	6655
	6663	6705	6798	6799	6802	6803	6804	6808	6811	6812	6831	6871	6874	6888	6891
	6900	7061	7126	7174	7513	7518	7547	7556	7565	7580	7584	7606	7673	7719	7724
	7736	7756	7767	7769	7780	7789	7790	7814	7887	7911	7968	7995	7999	8010	8047
	8091	8093	8098	8099	8121	8137	8154	8186	8257	8333	8362	8396	8428	8433	
NEG	2966	3262	3606	4854	4901	5614	6128	6129	6262	6809	6870				
NOP	3313	3360	3496	3497	4558	4614	6086	8165							
ROL	4330	6136	6137	6138	6139	6164	6268	6270	6271	6272	6274	6361	6363	6365	6815
	6817	6818	6819	6821	6980	7551	7937	8064	8079	8089					
ROLB	8006														
ROR	2996	6158	6209	6210	6211	6212	7136	7137	7138	7139	7140	7141			
RTI	2949	5650	6330	6482	6527	6537	6591	6613	6669	6715	6843	6910	6944	6960	7188
	7921														
RTS	3409	3426	3450	3485	3498	3525	3566	3581	3637	3655	3680	3709	3713	3718	3735
	3739	3749	3764	3782	3786	3806	3824	3828	3842	3847	3860	3864	3869	3888	3900
	3909	3924	3940	3960	3971	3988	4000	4044	4065	4083	4089	4092	4133	4183	4219
	4233	4240	4251	4264	4286	4376	4423	4432	4445	4454	4471	4497	4501	4525	4549
	4569	4616	4656	4681	4712	4797	4820	4831	4861	4869	4914	4923	4966	4985	5009
	5038	5047	5084	5090	5100	5110	5116	5129	5150	5164	5180	5186	5206	5224	5236
	5242	5252	5267	5299	5324	5330	5360	5372	5403	5439	5456	5518	5535	5547	5559
	5571	5583	5595	5606	5711	5727	5730	5758	5760	5774	5797	5855	5884	5947	5981
	5996	6013	6021	6076	6087	6116	6167	6170	6196	6220	6248	6295	6309	6339	6380
	6421	6447	6495	6543	6563	6769	6994	7025	7067	7105	7134	7161	7181	7495	7561
	7616	7668	7774	7810	7815	7836	7864	7905	7907	7974	7997	8008	8019	8147	8203
	8279	8321	8340	8365	8382	8384	8414	8435	8513	8525	8547	8563	8580	8608	8627
SBC	6130	7051													
SEV	6169														
SUB	2994	2999	3001	3357	4111	4126	4242	4273	4299	4304	4402	4404	4407	4520	4530
	4571	4666	4733	4743	4753	5176	5191	5511	5512	5513	5514	5902	6407	6704	6877
	7018	7050	7052	7720	7895	8192									
SWAB	7888	8330	8360	8427											
TRAP	7011	7190	7199	7200	7201	7202	7205	7206	7207	7208	7209				
TST	3002	3034	3046	3107	3119	3158	3192	3198	3205	3212	3232	3244	3273	3275	3278
	3287	3300	3306	3314	3334	3340	3355	3367	3372	3386	3403	3413	3570	3611	3628

	3649	3690	4052	4054	4056	4069	4120	4123	4142	4152	4168	4200	4234	4255	4296
	4326	4338	4361	4399	4443	4467	4470	4479	4487	4489	4500	4537	4630	4767	4803
	4816	4826	4874	4876	4897	4947	4952	4954	4959	4980	5010	5040	5367	5380	5391
	5448	5488	5542	5554	5566	5578	5590	5600	5735	5749	5762	5911	5920	5925	5930
	5964	6025	6085	6140	6153	6219	6279	6318	6369	6372	6413	6479	6516	6520	6522
	6531	6534	6561	6570	6590	6631	6642	6664	6711	6761	6826	6682	6892	7015	7016
	7173	7559	7612	7642	7656	7658	7807	7821	7847	7903	7904	7939	7944	7959	8018
	8028	8081	8113	8134	8148	8168	8184	8190	8197	8249	8271	8286	8293	8323	8327
	8380	8383	8393	8410	8431	8546	8621								
TSTB	2997	3048	3150	3280	3303	3337	4004	4158	4164	4202	4220	4878	4902	4945	5415
	5423	5643	5645	5723	5733	5739	5783	5807	5817	5824	5828	5836	5842	5956	6238
	6348	6390	6437	6471	6492	6540	6609	6753	6884	6898	7486	7489	7586	7588	7590
	7595	7597	7599	7607	7625	7644	7841	7980	7982	7992	8020	8023	8030	8032	8038
	8053	8066	8096	8108	8114	8125	8161	8166	8220	8222	8224	8239	8260	8262	8269
	8272	8274	8403	8558											
.ASCII	2740	2741	10094	10108	10364	10396	10408	10428	10429	10455	10496	10502			
.ASCIZ	2739	2742	6672	6673	6770	9434	9436	9438	9440	9442	9444	9463	9468	9473	9478
	9483	9488	9493	9833	9841	9846	9855	9864	9871	9878	9884	9890	9899	9904	9910
	9914	9920	9929	9938	9946	9956	9961	9967	9973	9980	9986	9995	10002	10009	10015
	10021	10028	10037	10045	10054	10062	10069	10073	10082	10087	10101	10115	10120	10233	10236
	10240	10249	10254	10256	10257	10260	10261	10264	10267	10270	10273	10275	10277	10279	10281
	10283	10286	10289	10292	10294	10299	10302	10305	10308	10313	10315	10317	10322	10326	10329
	10331	10335	10339	10341	10344	10348	10350	10352	10355	10360	10368	10373	10376	10381	10385
	10390	10402	10412	10417	10425	10430	10431	10432	10434	10436	10440	10444	10447	10450	10452
	10460	10468	10473	10475	10480	10483	10486	10490	10494	10507	10511	10515	10523	10524	10525
	10533	10537	10540	10544	10548	10549	10551	10555	10559	10561	10566	10569	10621	10625	10629
	10633	10635	10638	10640	10642	10644	10646	10648	10650	10652	10654	10656	10658	10660	10662
	10664	10666	10668	10670	10672	10674	10676								
.BLKB	6596	6674	7088	7145											
.BLKW	6915	8481	8482	8483	8484	8485	8486	8487	8488	9360	10687				
.BYTE	2711	2712	2717	2718	2726	2727	2735	2736	2737	2738	2768	4185	4186	4195	4196
	5652	5653	5654	5655	5656	5657	5658	6296	6297	6298	6299	6592	6593	6670	6671
	6844	6845	6846	6847	7240	7241	7242	7243	7244	7245	7246	7247	7254	7255	7256
	7257	7258	7259	7260	7261	7280	7281	7282	7283	7284	7285	7286	7287	7293	7294
	7295	7296	7297	7298	7299	7300	7320	7326	7333	7334	7335	7336	7337	7338	7339
	7340	7345	7346	7347	7348	7349	7350	7351	7352	7392	7393	7394	7395	7396	7397
	7398	7399	8438	8439	8440	8441	8442	8443	8444	8445	8700	8782	8864	8946	9028
	9110	9192	9274	9372	9427	9428	9429	9430	9431	9432	9447	9448	9449	9450	9451
	9452	9453	10223	10226	10228										
.ENABL	1														
.END	10696														
.ENDC	2526	2537	2538	2555	2647	2661	2663	2667	2700	2705	2709	2711	2739	2740	2744
	2772	2776	2817	2821	2830	2834	2851	2916	2917	2924	2932	2933	2934	2936	2938
	2954	3184	3188	4969	4973	5333	5337	6425	6450	6498	6546	6601	6616	6678	6684
	6687	6696	6703	6708	6709	6710	6711	6715	6716	6719	6734	6772	6775	6852	6919
	6964	7000	7013	7027	7030	7092	7110	7149	7166	7172	7175	7179	7198	7199	7200
	7201	7202	7203	7204	7205	7206	7207	7208	7209	7213	7223	8630	8634	9363	9367
.EQUIV	9531	9535	9828	9832	10682	10686	10690								
	2555	2556	2564	2609	2610	2611	2612	2613	2614	2615	2616	2617	2618	2637	2638
	2639	2640	2641	2642	2643	2644	2645	2646							
.EVEN	2769	5659	6598	6675	6771	9499	10128	10231	10576	10679					
.IF	2522	2537	2538	2553	2619	2647	2662	2666	2698	2704	2708	2710	2739	2743	2744
	2771	2775	2816	2820	2829	2833	2916	2923	2927	2932	2934	2936	2938	2954	3183
	3187	4968	4972	5332	5336	6424	6449	6497	6545	6600	6615	6677	6683	6686	6695
	6699	6706	6708	6709	6711	6714	6715	6716	6718	6733	6749	6774	6851	6918	6963
	6999	7003	7012	7013	7029	7091	7109	7148	7165	7171	7175	7190	7199	7200	7201

	7202	7203	7204	7205	7206	7207	7208	7209	7212	7222	8629	8633	9362	9366	9530
.IFF	9534	9827	9831	10681	10685	10689									
	2537	2538	2553	2663	2667	2705	2708	2710	2739	2744	2772	2776	2817	2821	2830
	2934	2917	2924	2932	3184	3188	4969	4973	5333	5337	6425	6450	6498	6546	6601
	6616	6678	6684	6686	6699	6714	6715	6716	6719	6733	6749	6775	6852	6919	6964
	7000	7003	7013	7030	7092	7110	7149	7166	7172	7213	7223	8630	8634	9363	9367
	9531	9535	9828	9832	10682	10686	10690								
.IFT	6709	7007	7013	7023	7027										
.IFTF	6708	7004	7007	7019	7023										
.IIF	2521	2526	2531	2535	2536	2537	2538	2539	2540	2541	2542	2543	2544	2545	2546
	2547	2548	2549	2550	2695	2743	2933	2938	6687	6688	6689	6690	6691	6695	6714
	6715	6716	6731	6756	6760	7198	7199	7200	7201	7202	7205	7206	7207	7208	7209
.IRP	2744	2916	3109	3520	3667	3682	4015	4208	4216	4352	4364	4508	4522	4933	4964
	5211	5408	5435	5468	5478	5483	5494	5511	5523	5532	5666	5705	5889	5944	6094
	6109	6177	6190	6695	6862	6902	6934	6954	6973	6991	8697	9418	10140	10173	10206
.LIST	1	2520	2661	2695	2739	2916	2938	6715	7190	7198	7199	7200	7201	7202	7203
	7205	7206	7207	7208	7209	7210									
.MACRO	1	2538	2702	5536	7190										
.MCALL	2520	2661	2938												
.NLIST	1	2520	2661	2695	2739	2916	2938	6715	7190	7198	7199	7200	7201	7202	7203
	7205	7206	7207	7208	7209	7210									
.PAGE	2702	2851													
.REM	1														
.REPT	2695	8701	8707	8783	8789	8865	8871	8947	8953	9029	9035	9111	9117	9193	9199
	9275	9281	9396	9556											
.SBTTL	2531	2551	2664	2689	2698	2702	2773	2818	2831	2851	2918	2926	3185	4970	5334
	6426	6451	6499	6547	6679	6681	6716	6772	6849	6916	6961	6997	7027	7089	7107
	7146	7163	7190	7214	8631	9364	9532	9829	10683						
.TITLE	2521														
.WORD	2695	2696	2697	2710	2713	2714	2715	2716	2719	2720	2721	2722	2723	2724	2725
	2728	2729	2730	2744	2745	2746	2747	2748	2749	2750	2751	2752	2753	2754	2755
	2756	2757	2758	2759	2760	2761	2762	2764	2766	2767	2777	2778	2779	2780	2781
	2782	2785	2786	2788	2789	2791	2793	2802	2806	2811	2822	2823	2824	2825	2826
	2827	2835	2836	2843	3031	3129	3171	4189	4190	4191	4192	4193	4194	4378	4379
	4380	4381	4382	4383	4384	4385	4386	4387	4551	5008	5049	5050	5250	5661	5847
	5949	5983	5984	5994	6245	6250	6308	6323	6328	6375	6416	6444	6594	6595	6742
	6747	6848	6995	6996	7026	7063	7106	7162	7197	7230	7231	7232	7233	7267	7268
	7269	7270	7271	7272	7273	7274	7306	7314	7360	7368	7373	7374	7375	7376	7377
	7378	7379	7380	7386	7404	7408	7409	7412	7414	7416	7418	8314	8315	8334	8336
	8368	8369	8377	8449	8450	8451	8452	8453	8454	8455	8456	8460	8461	8462	8463
	8464	8465	8466	8467	8469	8470	8471	8472	8473	8474	8475	8476	8477	8701	8702
	8703	8704	8705	8706	8707	8708	8709	8710	8711	8712	8713	8714	8715	8716	8717
	8718	8719	8720	8721	8722	8723	8724	8725	8726	8727	8728	8729	8730	8731	8732
	8733	8734	8735	8736	8737	8738	8739	8740	8741	8742	8743	8744	8745	8746	8747
	8748	8749	8750	8751	8752	8753	8754	8755	8756	8757	8758	8759	8760	8761	8762
	8763	8764	8765	8766	8767	8768	8769	8770	8771	8772	8773	8774	8775	8776	8777
	8778	8783	8784	8785	8786	8787	8788	8789	8790	8791	8792	8793	8794	8795	8796
	8797	8798	8799	8800	8801	8802	8803	8804	8805	8806	8807	8808	8809	8810	8811
	8812	8813	8814	8815	8816	8817	8818	8819	8820	8821	8822	8823	8824	8825	8826
	8827	8828	8829	8830	8831	8832	8833	8834	8835	8836	8837	8838	8839	8840	8841
	8842	8843	8844	8845	8846	8847	8848	8849	8850	8851	8852	8853	8854	8855	8856
	8857	8858	8859	8860	8865	8866	8867	8868	8869	8870	8871	8872	8873	8874	8875
	8876	8877	8878	8879	8880	8881	8882	8883	8884	8885	8886	8887	8888	8889	8890
	8891	8892	8893	8894	8895	8896	8897	8898	8899	8900	8901	8902	8903	8904	8905
	8906	8907	8908	8909	8910	8911	8912	8913	8914	8915	8916	8917	8918	8919	8920
	8921	8922	8923	8924	8925	8926	8927	8928	8929	8930	8931	8932	8933	8934	8935

8936	8937	8938	8939	8940	8941	8942	8947	8948	8949	8950	8951	8952	8953	8954
8955	8956	8957	8958	8959	8960	8961	8962	8963	8964	8965	8966	8967	8968	8969
8970	8971	8972	8973	8974	8975	8976	8977	8978	8979	8980	8981	8982	8983	8984
8985	8986	8987	8988	8989	8990	8991	8992	8993	8994	8995	8996	8997	8998	8999
9000	9001	9002	9003	9004	9005	9006	9007	9008	9009	9010	9011	9012	9013	9014
9015	9016	9017	9018	9019	9020	9021	9022	9023	9024	9029	9030	9031	9032	9033
9034	9035	9036	9037	9038	9039	9040	9041	9042	9043	9044	9045	9046	9047	9048
9049	9050	9051	9052	9053	9054	9055	9056	9057	9058	9059	9060	9061	9062	9063
9064	9065	9066	9067	9068	9069	9070	9071	9072	9073	9074	9075	9076	9077	9078
9079	9080	9081	9082	9083	9084	9085	9086	9087	9088	9089	9090	9091	9092	9093
9094	9095	9096	9097	9098	9099	9100	9101	9102	9103	9104	9105	9106	9111	9112
9113	9114	9115	9116	9117	9118	9119	9120	9121	9122	9123	9124	9125	9126	9127
9128	9129	9130	9131	9132	9133	9134	9135	9136	9137	9138	9139	9140	9141	9142
9143	9144	9145	9146	9147	9148	9149	9150	9151	9152	9153	9154	9155	9156	9157
9158	9159	9160	9161	9162	9163	9164	9165	9166	9167	9168	9169	9170	9171	9172
9173	9174	9175	9176	9177	9178	9179	9180	9181	9182	9183	9184	9185	9186	9187
9188	9193	9194	9195	9196	9197	9198	9199	9200	9201	9202	9203	9204	9205	9206
9207	9208	9209	9210	9211	9212	9213	9214	9215	9216	9217	9218	9219	9220	9221
9222	9223	9224	9225	9226	9227	9228	9229	9230	9231	9232	9233	9234	9235	9236
9237	9238	9239	9240	9241	9242	9243	9244	9245	9246	9247	9248	9249	9250	9251
9252	9253	9254	9255	9256	9257	9258	9259	9260	9261	9262	9263	9264	9265	9266
9267	9268	9269	9270	9275	9276	9277	9278	9279	9280	9281	9282	9283	9284	9285
9286	9287	9288	9289	9290	9291	9292	9293	9294	9295	9296	9297	9298	9299	9300
9301	9302	9303	9304	9305	9306	9307	9308	9309	9310	9311	9312	9313	9314	9315
9316	9317	9318	9319	9320	9321	9322	9323	9324	9325	9326	9327	9328	9329	9330
9331	9332	9333	9334	9335	9336	9337	9338	9339	9340	9341	9342	9343	9344	9345
9346	9347	9348	9349	9350	9351	9352	9356	9358	9368	9376	9380	9384	9388	9391
9395	9396	9397	9398	9399	9400	9401	9402	9403	9404	9405	9406	9407	9408	9409
9410	9411	9412	9413	9414	9415	9418	9419	9420	9421	9422	9423	9424	9425	9455
9456	9457	9458	9459	9460	9461	9503	9504	9505	9506	9507	9508	9509	9510	9512
9513	9514	9515	9516	9517	9518	9519	9521	9522	9523	9524	9525	9526	9527	9528
9536	9537	9538	9539	9540	9541	9542	9543	9544	9545	9546	9547	9548	9549	9550
9551	9552	9553	9555	9556	9557	9558	9559	9560	9561	9562	9563	9564	9565	9566
9567	9568	9569	9570	9572	9573	9574	9575	9576	9577	9578	9579	9580	9581	9582
9583	9584	9585	9586	9587	9589	9590	9591	9592	9593	9594	9595	9596	9597	9598
9599	9600	9601	9602	9603	9604	9606	9607	9608	9609	9610	9611	9612	9613	9614
9615	9616	9617	9618	9619	9620	9621	9623	9624	9625	9626	9627	9628	9629	9630
9631	9632	9633	9634	9635	9636	9637	9638	9640	9641	9642	9643	9644	9645	9646
9647	9648	9649	9650	9651	9652	9653	9654	9655	9657	9658	9659	9660	9661	9662
9663	9664	9665	9666	9667	9668	9669	9670	9671	9672	9674	9675	9676	9677	9678
9679	9680	9681	9682	9683	9684	9685	9686	9687	9688	9689	9691	9692	9693	9694
9695	9696	9697	9698	9699	9700	9701	9702	9703	9704	9705	9706	9708	9709	9710
9711	9712	9713	9714	9715	9716	9717	9718	9719	9720	9721	9722	9723	9725	9726
9727	9728	9729	9730	9731	9732	9733	9734	9735	9736	9737	9738	9739	9740	9742
9743	9744	9745	9746	9747	9748	9749	9750	9751	9752	9753	9754	9755	9756	9757
9759	9760	9761	9762	9763	9764	9765	9766	9767	9768	9769	9770	9771	9772	9773
9774	9776	9777	9778	9779	9780	9781	9782	9783	9784	9785	9786	9787	9788	9789
9790	9791	9793	9794	9795	9796	9797	9798	9799	9800	9801	9802	9803	9804	9805
9806	9807	9808	9810	9811	9812	9813	9814	9815	9816	9817	9818	9819	9820	9821
9822	9823	9824	9825	10130	10134	10137	10140	10142	10144	10146	10148	10150	10152	10154
10156	10158	10160	10162	10164	10166	10168	10170	10173	10175	10177	10179	10181	10183	10185
10187	10189	10191	10193	10195	10197	10199	10201	10203	10206	10208	10210	10212	10214	10216
10218	10220	10578	10579	10580	10581	10582	10583	10584	10585	10586	10587	10588	10589	10590
10591	10592	10593	10594	10595	10596	10597	10598	10599	10600	10601	10602	10603	10604	10605
10606	10607	10608	10609	10610	10611	10612	10613	10614	10615	10616	10617	10618	10619	10691

MAINDEC-11-DERPN-B MACY11 27(732) 27-SEP-76 15:05 PAGE 259
DERPNB.P11 CROSS REFERENCE TABLE -- PERMANENT SYMBOLS

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

*.DERPNB.SEG/SOL/CRF/PAGNUM/NL:TOC=SYSMAC.SML(400,1066),DERPNB(400,4571)
RUN-TIME: 56 92 15 SECONDS
RUN-TIME RATIO: 371/163=2.2
CORE USED: 36K (71 PAGES)

