

RM02/03/05

RM05/3/2 FORMATTER
CZRMLB0

AH-F919B-MC
FICHE 1 OF 1

AUG 1981
COPYRIGHT © 80-81
MADE IN USA



.REM %

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

IDENTIFICATION

PRODUCT CODE: AC-F918B-MC
 PRODUCT NAME: CZRMLB0 RM05/3/2 FORMATTER
 PRODUCT DATE: APRIL 1981
 MAINTAINER: CX DIAGNOSTIC GROUP
 AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980,1981 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 1. ABSTRACT
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 PRELIMINARY PROGRAMS
- 3. LOADING PROCEDURES
 - 3.1 PAPER TAPE AND XXDP
 - 3.2 APT
 - 3.3 APT ETABLE DEFINITIONS
- 4. STARTING PROCEDURES
 - 4.1 STARTING ADDRESSES
 - 4.2 OPERATOR ACTION
 - 4.3 RH11 - RH70 UNIBUS ADDRESS
- 5. SWITCH REGISTER SETTINGS
- 6. ERROR MESSAGES
- 7. MISCELLANEOUS
 - 7.1 FORMAT TIMES
 - 7.2 HALTING THE PROGRAM
 - 7.3 SURFACE VERIFICATION
- 8. PROGRAM DESCRIPTION
 - 8.1 FORMAT OPERATION
 - 8.2 POSITIONER VERIFICATION
 - 8.3 HEADER VERIFICATION
- 9. BAD SECTOR FILE
 - 9.1 BAD SECTOR FILE UTILITY
- 10. RH/RM SOFTWARE DRIVER DOCUMENT

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

1. ABSTRACT

THE RM05/3/2 FORMATTER PROGRAM PROVIDES THE FACILITIES TO FORMAT, TO CHECK THE HEADER AND DATA FIELDS OR VERIFY THE HEADER FIELD OF EACH DATA BLOCK ON THE PACK. EACH HEADER FIELD SPECIFIES THE ADDRESS OF ITS ASSOCIATED DATA BLOCK ON THE DISK PACK.

IN THE FORMAT OPERATION, THE PROGRAM WRITES THE HEADER OF EACH DATA BLOCK WITH A CYLINDER NUMBER, TRACK NUMBER AND SECTOR NUMBER. ALSO, WRITES THE DATA FIELD WITH SELECTED DATA PATTERN. THE PROGRAM THEN VERIFIES THE WRITTEN DATA BLOCKS VIA EXECUTING THE 'WRITE CHECK HEADER AND DATA' COMMAND. THE PROGRAM THEN PERFORMS A VERIFY OF SECTOR ZERO, ON EACH OF THE TRACKS SELECTED FOR FORMATTING. THE VERIFY IS ACCOMPLISHED BY EXECUTING A 'READ HEADER AND DATA' COMMAND AND COMPARING THE TWO HEADER WORDS TO AN EXPECTED VALUE.

IN THE VERIFY OPERATION, THE PROGRAM READS THE HEADER OF SECTOR 0 THRU 29. (18 BIT MODE) OR 31. (16 BIT MODE) ON EACH CYLINDER AND TRACK TO BE VERIFIED, THEN COMPARES THE TWO HEADER WORDS TO MAKE SURE THEY ARE CORRECT.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
12K MEMORY
KW11-L OR KW11-P CLOCK
PROGRAM LOADING DEVICE
TERMINAL
RH11 OR RH70 CONTROLLER
1 TO 8 DISK DRIVES (ANY COMBINATION OF RM05'S, RM03'S OR RM02'S)

2.2 PRELIMINARY PROGRAMS

THERE ARE NO PREREQUISITE PROGRAMS, PROVIDING THE RM SUBSYSTEM IS KNOWN TO BE OPERATIONAL. IF THE STATE OF THE RM SUBSYSTEM IS UNKNOWN, THE FOLLOWING PROGRAMS SHOULD BE RUN PRIOR TO USING THE FORMATTER PROGRAM.

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

RM05/3/2 FUNCTIONAL TEST, PART 1

3. LOADING PROCEDURES

3.1 PAPER TAPE AND XXDP

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM 'XXDP' MEDIA USING THE APPROPRIATE

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

LOADER.

3.2 APT

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

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

AUTOMATIC MODE (MONITOR)

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

DUMP MODE

INPUT DIALOGUE AFTER PROGRAM STARTS

3.3 APT ETABLE DEFINITIONS

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

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

115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171

NOT USED

7. INTERRUPT VECTOR 1:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254
8. BUS PRIORITY 1:
NOT USED.
9. INTERRUPT VECTOR 2:
NOT USED
10. BUS PRIORITY 2:
NOT USED
11. BASE ADDRESS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700
12. DEVICE MAP:
USED WHEN ENVIRONMENT MODE BIT 7 = 1. EACH BIT SET TO 1 IN BITS 0 TO 7 WILL SELECT THE CORRESPONDING DRIVE TO BE TESTED. BITS 8-15 ARE NOT USED.
13. CONTROLLER DESCRIPTOR WORDS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1. ANY NONE ZERO NUMBER SPECIFIES THAT PROGRAM RUNS IN CHECK MODE.
14. CONTROLLER DESCRIPTOR WORDS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1. ANY NONE ZERO NUMBER SPECIFIES THAT PROGRAM SELECTS 30.(18 BIT MODE) SECTORS FOR A TRACK IN ALL OPERATIONS.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

200 - START ADDRESS IS DEFAULT, ALL SWITCHES CLEAR (SEE SECTION 5)

204 - RESTART ADDRESS, THE RESTART ADDRESS PROVIDES THE OPERATOR WITH THE ABILITY TO CHANGE THE DEFAULT RM/RH ADDRESSES AND TO USE THE BAD SECTOR FILE UTILITY ROUTINES (SEE SECTION 9.1).

4.2 OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3).
2. LOAD THE STARTING ADDRESS - 200 OR 204
3. SET THE SWITCHES AS REQUIRED AND PRESS 'START'.

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

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

4. THE PROGRAM WILL THEN TYPE THE FOLLOWING MESSAGE:

'DO YOU WANT TO FORMAT (L) Y ?'

ENTER THE APPROPRIATE CODE: 'Y' FOLLOWED BY CARRIAGE RETURN OR JUST CARRIAGE RETURN (DEFAULT) TO FORMAT & HEADER VERIFY. IF THE RESPONSE IS 'N' FOLLOWED BY CARRIAGE RETURN, THEN A VERIFY ONLY MODE WILL BE ENTERED.

5. THE PROGRAM WILL THEN ASK FOR THE FORMATTING MODE:

'DO YOU WANT 16 BIT MODE (L) Y ?'

ENTER THE APPROPRIATE CHARACTER: 'Y' FOLLOWED BY A CARRIAGE RETURN OR JUST A CARRIAGE RETURN (DEFAULT) FOR 16 BIT MODE. IF THE RESPONSE IS 'N' FOLLOWED BY A CARRIAGE RETURN, THEN ENTER 18 BIT MODE.

6. THE PROGRAM WILL THEN ASK FOR A DRIVE:

'DRIVE: '

ENTER THE ADDRESS OF THE DRIVE TO BE FORMATTED. ENTER THE DRIVE NUMBER FOLLOWED BY A CARRIAGE RETURN. IF THE DRIVE SELECTED IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN ERROR MESSAGE AND RETURN TO THE DRIVE ADDRESS REQUEST.

NOTE: IT NOT POSSIBLE TO ENTER A MULT-DRIVE ENTRY FOR THIS PROMPT. THE ONLY WAY ALL THE DRIVES CAN BE FORMATTED, IS TO SETUP A CHAIN TO RUN THIS PROGRAM.

7. AFTER THE OPERATOR HAS SELECTED A DRIVE, THE OPERATOR WILL BE ASKED TO CHANGE THE DRIVE PARAMETERS WITH THE FOLLOWING PROMPT:

'CHANGE DRIVE PARAMETERS (L) N ?'

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

THE PROGRAM WILL ASK FOR ADDRESS LIMITS FOR THE SELECTED DRIVE:

'ENTER ADDRESS LIMITS: '

THE PREVIOUSLY SELECTED OR DEFAULT VALUES FOR BEGINNING CYLINDER AND TRACK AND FOR ENDING CYLINDER AND TRACK WILL BE TYPED OUT. IF A 'CARRIAGE RETURN' IS TYPED AS A RESPONSE, THE PRESENT VALUE WILL BE USED; IF A 'PERIOD' IS TYPED, THE PROGRAM WILL BYPASS THE REMAINING ENTRIES AND WILL USE THEIR PRESENT VALUES.

NOTE: THAT THE CYLINDER AND TRACK VALUES ARE DECIMAL NUMBERS. THE ADDRESS SPECIFIED BY THE BEGINNING TRACK MUST BE LESS THAN THE ENDING TRACK ADDRESS IF THESE TWO ADDRESSES ARE ON THE SAME CYLINDER. ON THE OTHER HAND, THE ENDING CYLINDER ADDRESS MAY

229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285

BE LESS THAN THE STARTING CYLINDER ADDRESS. IN THIS CASE, THE PROGRAM WILL FORMAT OR VERIFY THE DISK PACK FROM THE STARTING CYLINDER TO CYLINDER 822 AND THEN FROM CYLINDER 0 TO THE ENDING CYLINDER.

9. THE PROGRAM WILL THEN TYPE:

'STARTING FORMAT'

OR

'STARTING HEADER VERIFY'

NOTE: IF A NEW PACK IS UNFORMATTED, THE PROGRAM ALSO ASKS YOU TO ENTER TWO SERIAL NUMBERS. (EXCEPT IF OPERATING IN VERIFY ONLY MODE) THESE TWO SERIAL NUMBER MUST BE NON-ZERO DECIMAL NUMBERS.

10. THE OPERATOR CAN DETERMINE WHERE THE DRIVE IS DURING THE OPERATION BY TYPING A 'CONTROL 0'. THE PROGRAM WILL TYPE THE FOLLOWING MESSAGE:

'PRESENT ADDRESS IS: CXXX TXX'

11. TO CHANGE EITHER THE ADDRESSING MODE OR THE OPERATION MODE, THE PROGRAM MUST BE STARTED FROM LOCATION 200 OR 204 AGAIN.

4.3 RH11 - RH70 UNIBUS ADDRESS

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT 176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204. ENTER THE RH11/RH70 ADDRESS IN RESPONSE TO THE REQUEST FROM THE PROGRAM.

5. SWITCH REGISTER SETTINGS

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

'SWR = NNNNNN NEW ='

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

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

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

SW<15>=1	HALT ON ERROR
SW<13>=1	INHIBIT ERROR TYPEOUTS
SW<10>=1	BELL ON ERROR
SW<09>=1	LOOP ON ERROR
SW<02>=1	DCN'T DISPLAY SYSTEM STATUS AFTER INITIAL START
SW<01>=1	LOOP ON THE CURRENT TRACK
SW<00>=1	LOOP THE PROGRAM ON THE SELECTED DRIVE

6. ERROR MESSAGES

- 'RH INTERRUPT OCCURRED (RMAS=0) - AN INTERRUPT OCCURRED, BUT NOTHING ON THE MASSBUS IS INDICATING AN ATTENTION.
- 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED, BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.
- 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH WHEN THE INDICATED REGISTER WAS READ.
- 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.
- 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE' BIT SET FOR THE INDICATED DRIVE.
- 'RH DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.
- 'DRIVE OFFLINE' - THE INDICATED DRIVE HAS GONE OFFLINE
- 'PERSISTENT DRIVE UNSAFE ERROR' - THE INDICATED DRIVE HAS BECOME UNSAFE AND THE CONDITION CANNOT BE CLEARED BY ISSUING 'DRIVE CLEAR' INSTRUCTIONS.
- 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE DRIVE AND A PARITY ERROR OCCURRED.
- 'SOFTWARE TIMEOUT' - THE OPERATION FAILED TO COMPLETE WITHIN 1 SECOND.
- 'DRIVE UNSAFE ERROR' - A NON-PERSISTENT UNSAFE ERROR OCCURRED DURING THE OPERATION.
- 'CONTROLLER/DRIVE ERROR DURING WRITE' - THE INDICATED NON-DATA ERROR WAS DETECTED DURING A FORMAT OPERATION.

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399

13. 'CONTROLLER/DRIVE ERROR DURING WRITE CHECK' - A NON-DATA ERROR OCCURRED DURING THE WRITE CHECK.
14. 'DATA ERROR DURING WRITE CHECK' - A DATA RELATED ERROR OCCURRED DURING A WRITE CHECK OPERATION. A DATA ERROR IS CONSIDERED TO BE ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCF', 'HCE', OR 'FER'.
15. 'RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE' - THE INDICATED SECTOR FAILED DURING RETRY FOLLOWING A DATA ERROR.
16. 'CONTROLLER/DRIVE ERROR VERIFYING HEADERS' - AN ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
17. ''HCE' ERROR VERIFYING HEADERS' - A HEADER ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
18. 'CYLINDER FIELD IN HEADER IS NOT CORRECT' - THE CYLINDER FIELD FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.
19. 'TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT' - THE TRACK AND SECTOR FIELD FROM A HEADER READ DURING THE VERIFICATION IS NOT CORRECT.
20. 'WRITE CHECK ERROR' - THE RH REPORTED A WRITE CHECK ERROR AND NO DRIVE ERRORS WERE SET.

7. MISCELLANEOUS

7.1 FORMAT TIME

THE 'FORMAT' MODE WILL TAKE APPROX. 10 MINUTES FOR AN ENTIRE RM03/2 AND APPROX. 30 MINUTES FOR AN ENTIRE RM05.

THE 'VERIFY' ONLY MODE TAKES APPROX. 7 MINUTES FOR AN RM03/2 AND APPROX. 20 MINUTES FOR RM05.

7.2 HALTING THE PROGRAM

THE OPERATOR SHOULD NOT HALT THE PROGRAM VIA THE CONSOLE DURING A FORMAT OPERATION. HALTING THE PROGRAM MAY LEAVE A SECTOR INCORRECTLY FORMATTED. TO TERMINATE THE FORMAT, TYPE A 'CONTROL C'.

7.3 SURFACE VERIFICATION

THE FORMATTER PROGRAM IS NOT INTENDED TO BE USED TO PERFORM DISK PACK VERIFICATION. IF THE PROGRAM REPORTS A SECTOR AS BEING 'NOT ACCEPTABLE', THIS MAY IN FACT INDICATE A BAD SURFACE; HOWEVER, SECTORS WHICH 'PASSED' MAY OR MAY NOT BE GOOD.

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

8. PROGRAM DESCRIPTION

8.1 FORMAT OPERATION

THE PROGRAM FORMATS THE PACK ONE TRACK AT A TIME BETWEEN THE LIMITS SPECIFIED BY THE OPERATOR.

THE PACK IS WRITTEN WITH A 'WORST CASE' DATA PATTERN, WHICH IS A OCTAL NUMBER 155555, FOLLOWED BY AN OCTAL NUMBER OF 133333.

THE FORMAT OPERATION CONSISTS OF A WRITE HEADER AND DATA COMMAND FOR THE ENTIRE TRACK FOLLOWED BY A WRITE CHECK HEADER AND DATA COMMAND. IF AN ERROR OCCURS DURING THE WRITE, THE PROGRAM WILL RETRY THE WRITE BEFORE CONTINUING. IF A DATA RELATED ERROR IS DETECTED DURING THE WRITE CHECK (A DATA ERROR IS DEFINED AS ONE OF THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE', OR 'FER'), THE SECTOR IN ERROR WILL BE REWRITTEN. THE PROGRAM WILL CHECK THE SECTOR TWICE; IF A DATA ERROR IS DETECTED IN THE SECTOR DURING EITHER OF THE WRITE CHECKS, THE PROGRAM WILL DECLARE THE SECTOR AS BEING 'NONRECOVERABLE'. FOLLOWING THIS SEQUENCE, THE REMAINDER OF THE TRACK IS CHECKED. IF DATA ERRORS ARE ENCOUNTERED IN ANY OF THE REMAINING SECTORS, EACH SECTOR WILL BE HANDLED AS DESCRIBED ABOVE.

IF A NON-DATA RELATED ERROR OCCURS DURING THE WRITE CHECK, THE PROGRAM WILL RETRY THE COMMAND FOR THE ENTIRE TRACK BEFORE PROCEEDING TO THE NEXT TRACK.

IT SHOULD BE NOTED THAT THE FORMATTER PROGRAM FORMATS THE PACK ONLY AND IS NOT DIRECTLY VERIFYING THE SURFACE OF THE PACK. ERRORS THAT ARE ENCOUNTERED MAY BE THE RESULT OF BAD AREAS ON THE PACK BUT, AS THE PROGRAM IS NOT DESIGNED TO PERFORM AN EXHAUSTIVE CHECK OF THE DISK PACK SURFACE, THE PROGRAM CANNOT MAKE THIS CONCLUSION. IN GENERAL, HOWEVER, THE OPERATOR CAN ASSUME THAT SECTORS WHICH THE PROGRAM CALLS 'UNACCEPTABLE' INDICATE BAD AREAS OF THE PACK; UNFORTUNATELY, SECTORS WHICH 'PASS' CANNOT BE ASSUMED TO BE GOOD.

DURING THE FORMAT OPERATION, THE PROGRAM FILLS THE DATA FIELD WITH THE PATTERN SELECTED BY THE OPERATOR.

8.2 POSITIONER VERIFICATION (QUICK VERIFY)

AFTER THE PROGRAM COMPLETES THE FORMAT OPERATION, THE POSITIONER IS RETURNED TO THE STARTING CYLINDER AND THE HEADER FROM SECTOR 0 ON THE STARTING TRACK IS READ. THE CYLINDER ADDRESS FIELD FROM THE HEADER IS COMPARED TO THE REQUESTED CYLINDER; IF THE CYLINDER ADDRESSES DO NOT COMPARE, AN ERROR MESSAGE IS TYPED. THE PROGRAM CHECKS THE CYLINDER ADDRESS FIELD FROM THE HEADER OF SECTOR 0 ON THE BEGINNING TRACK OF EACH CYLINDER FORMATTED. THIS CHECK IS PERFORMED TO CONFIRM THAT THE DISK'S POSITIONER ADVANCED PROPERLY DURING THE FORMAT OPERATION.

8.3 HEADER VERIFICATION (VERIFY ONLY)

THE HEADER VERIFICATION OPERATION PERFORMS A READ HEADER AND DATA

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

TO AN ENTIRE TRACK AT A TIME, THEN COMPARES THE EXPECTED HEADER WORDS WITH RECEIVED HEADER WORDS. THE CYLINDER ADDRESS FIELD FROM THE HEADER IS COMPARED TO THE EXPECTED CYLINDER, IF THE CYLINDER ADDRESSES DO NOT COMPARE, AN ERROR MESSAGE IS REPORTED. NEXT, THE TRACK/SECTOR FIELD FROM THE HEADER IS COMPARED TO THE EXPECTED TRACK/SECTOR, IF THE TRACK/SECTOR ADDRESSES DO NOT COMPARE, AN ERROR IS REPORTED. VERIFICATION OF THE HEADERS STARTS FROM SECTOR 0 ON THE BEGINNING CYLINDER AND TRACK, AND CONTINUES UNTIL COMPARING SECTOR 29. (18 BIT MODE) OR 31. (16 BIT MODE) ON ENDING CYLINDER AND TRACK. THIS CHECK IS PERFORMED TO CONFIRM THAT A DISK HAS PREVIOUSLY BEEN FORMATTED AND ALL THE HEADERS ARE IN GOOD CONDITION.

9. BAD SECTOR FILE DESCRIPTION

SECTORS 0 THRU 31., ON CYLINDER 822., LAST TRACK ARE ALWAYS FORMATTED IN 16 BIT MODE. THEY CONTAIN THE BAD SECTOR FILE ON THE PACK FOR BOTH 30. SECTOR (18 BIT MODE) AND 32. SECTOR (16 BIT MODE) OPERATIONS.

THE DATA FIELD OF EACH SECTOR OF THE BAD SECTOR FILE IS DESCRIBED BELOW:

THE FIRST TWO WORDS IN THE DATA FIELD SPECIFY THE SERIAL NUMBER OF THE PACK. THE SERIAL NUMBER MUST BE A NON-ZERO NUMBER AND CONSISTS OF MAXIMUM 10 OCTAL DIGITS. THE FIRST WORD OF THE SERIAL NUMBER CONTAINS THE 5 LEAST SIGNIFICANT OCTAL DIGITS, WHILE THE SECOND WORD CONTAINS THE 5 MOST SIGNIFICANT DIGITS.

THE THIRD WORD IS ALWAYS ALL 0'S.

THE FORTH WORD IS USUALLY ALL 0'S, UNLESS THE PACK IS AN ALIGNMENT PACK, IN WHICH CASE THE WORD WILL BE 177777.

THE FOLLOWING 252. WORDS DEFINE THE BAD SECTOR ADDRESS INFORMATION. TWO WORDS ARE USED TO DEFINE A SINGLE BAD SECTOR, THE FIRST WORD OF THE PAIR, SPECIFIES THE CYLINDER ADDRESS. THE SECOND WORD OF THE PAIR, SPECIFIES THE TRACK (HIGH BYTE) AND SECTOR (LOW BYTE) ADDRESSES. ALL UNUSED LOCATIONS ARE RECORDED WITH ONES (177777).

THE SECTORS ON THE LAST TRACK (BSF) ARE DIVIDED INTO FOUR GROUPS AS FOLLOWS:

- (1) SECTORS 0, 2, 4, 6 AND 8. CONTAIN THE MANUFACTURES (MFG) DEFINED BAD SECTORS FOR 16 BIT MODE.
- (2) SECTORS 1, 3, 5, 7 AND 9. CONTAIN THE MANUFACTURES (MFG) DEFINED BAD SECTORS FOR 18 BIT MODE.
- (3) SECTORS 10., 12. THRU 30. CONTAIN THE USER (USR) DEFINED BAD SECTORS FOR 16 BIT MODE.
- (4) SECTORS 11., 13. THRU 31. CONTAIN THE USER (USR) DEFINED BAD SECTORS FOR 18 BIT MODE.

THE SECTORS IN EACH GROUP ARE IDENTICAL AND SECTORS 0, 1, 10. AND 11. ARE USED AS REFERENCE SECTORS BY THE PROGRAM.

514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570

9.1 BAD SECTOR UTILITY

THE PROGRAM PROVIDES A FEW UTILITY ROUTINES TO HANDLE THE BAD SECTOR FILE. TO ACCESS THESE ROUTINES START THE PROGRAM AT 204.

BEFORE THE PROGRAM FORMATS THE PACK, THE PROGRAM TYPES THE FOLLOWING MESSAGE TO ALLOW THE OPERATOR TO ACCESS THE UTILITIES BY ENTERING THE SELECTED FUNCTION NUMBER.

```

UPDATE 'USR' BAD SECTORS.....0
PRINT 'MFG' BAD SECTORS.....1
PRINT 'USR' BAD SECTORS.....2
INITIALIZE 'USR' BAD SECTOR FILE.....3
PRINT A SECTOR OF THE LAST TRACK.....4
PRINT THIS MESSAGE.....5
CONTINUE.....<CR>

```

ENTER UTILITY FUNCTION (HELP=5):

THE FIRST UTILITY (0) ALLOWS USER DEFINED BAD SECTORS TO BE ENTERED MANUALLY INTO THE 'USR' BAD SECTOR FILE TABLE. THIS IS ACCOMPLISHED BY SPECIFYING THE CYLINDER ADDRESS, TRACK ADDRESS AND SECTOR ADDRESS IN DECIMAL, TO THE FOLLOWING PROMPT.

```

EXAMPLE: CYL,TRK,SEC= C,T,S<CR> ;ENTERS BAD CYL,TRK,SEC IN FILE
          CYL,TRK,SEC= <CR>      ;EXIT

```

TO TERMINATE THE BAD SECTOR ADDRESS ENTRY, TYPE A 'CARRIAGE RETURN' IN RESPONSE TO THE ENTRY REQUEST.

THE SECOND (1) AND THIRD (2) UTILITIES WILL TYPE THE CONTENTS OF THE 'MFG' AND 'USR' BAD SECTOR FILES ON THE CONSOLE IN DECIMAL NUMBER. THE FORMAT OF THE TYPEOUT WILL BE AS FOLLOWS:

```

PACK S/N: XXXXXX
          *
CONTENTS OF 16 BIT 'MFG OR USR' BAD SECTOR FILE (DECIMAL)
C,T,S
CONTENTS OF 18 BIT 'MFG OR USR' BAD SECTOR FILE (DECIMAL)
C,T,S

```

WHERE 'XXXXXX' IS THE OCTAL SERIAL NUMBER OF THE PACK. WHERE 'C' IS THE DECIMAL CYLINDER ADDRESS, 'T' IS THE DECIMAL TRACK ADDRESS, AND 'S' IS DECIMAL SECTOR OF THE BAD SECTOR. IF THERE ARE NO ENTRIES CURRENTLY IN THE BAD SECTOR FILES, THE FOLLOWING TYPEOUT WILL APPEAR:

```

PACK S/N: XXXXXX
          *
CONTENTS OF 16 BIT 'MFG OR USR' BAD SECTOR FILE (DECIMAL)
* NO ENTRIES *

```


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

CONTENTS OF 18 BIT 'MFG OR USR' BAD SECTOR FILE (DECIMAL)
* NO ENTRIES *

THE FOURTH (3) UTILITY WILL RESET ALL BAD SECTOR ENTRIES IN THE 'USR' BAD SECTOR FILE TO -1. THE SERIAL NUMBER WILL NOT BE INITIALIZED BY THIS UTILITY.

THE FIFTH (4) UTILITY WILL ALLOW THE OPERATOR TO LOOK AT A SECTOR OF DATA FROM THE LAST TRACK ON THE LAST CYLINDER (BAD SECTOR FILE). THIS WILL ALLOW THE OPERATOR TO PHYSICALLY SEE WHAT IS IN THE BAD SECTOR FILE. THE DATA IS TYPED OUT IN BLOCK FORM IN OCTAL.

TYPING A '5' WILL ALLOW THE OPERATOR TO DISPLAY THE UTILITY MESSAGE AGAIN ON THE CONSOLE. TO CONTINUE THE PROGRAM OPERATION, JUST TYPE A 'CARRIAGE RETURN' <CR>.

10.1 RH/RM DRIVER

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

10.2 TO INITIALIZE THE DRIVER:

```
JSR   PC,RMINIT
RETURN
```

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

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

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

DRVTYP	CONDITION
-----	-----
0	NONEXISTENT DRIVE
4	RM03
5	RM02
7	RM05

628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684

-1

NOT AN RM05/3/2

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

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

CALL:

```
JSR    RO,RM03
PNTDPB
RETURN1
RETURN2
```

```
;MAKE THE CALL
;ADDRESS OF DPB*
;RETURN IF QUEUE IS FULL
;RETURN IF REQUEST IS IN
;QUEUE OR THERE IS AN
;ERROR CONDITION
```

*DPB (DATA PARAMETER BLOCK)

```
PNTDPB: .BYTE 0
        .BYTE 0
        .BYTE 0
        .BYTE 0
        .WORD 0
        .WORD 0
        .BYTE 0
        .BYTE 0
        .WORD 0
        .WORD 0
```

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

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

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

IT SHOULD BE NOTED THAT YOU MUST PROVIDE THE CODE TO DRIVE THE CLOCK. AND THE ELAPSED TIME MUST BE IN MILLISECONDS. THE DRIVER WILL SET THE TIMEOUT TO 1 SECOND FOR ALL POSITIONING AND DATA TRANSFER OPERATIONS AND WILL SET THE TIMOUT TO 30 SECONDS FOR ERROR RECOVERY OPERATIONS.

10.4.1 EXAMPLE - WRITE 1000. WORDS


```

685
686      1$: JSR    R0,RM03      :CALL THE DRIVER
687      FMTDPB      :DPB ADDRESS
688      BR      1$          :WAIT FOR QUEUE IF FULL
689      2$: TST    FMTDPB+16   :WAIT FOR COMMAND TO COMPLETE
690      BEQ    2$          :
691      BMI    ERROR1       :ERROR OCCURRED
692      .
693      .
694      .
695
696      FMTDPB: .BYTE  5          :DRIVE #5
697      .BYTE  0          :
698      .BYTE 161         :WRITE COMMAND
699      .BYTE  0          :
700      .WORD -1000.      :WORD COUNT
701      .WORD  WRTBUF     :BUFFER ADDRESS
702      .BYTE  3          :SECTOR
703      .BYTE  5          :TRACK
704      .WORD  400        :CYLINDER
705      .WORD  ERRTB5     :ERROR TABLE
706      .WORD  0          :STATUS/ERROR INDICATOR
707
708      ALTERNATE DPB SETUP
709
710      FMTDPB: .WORD  5          :THIS SETUP ACHIEVED
711      .WORD  WRITE      :EVERYTHING THE
712      .WORD -1000.      :ABOVE TABLE DID, BUT
713      .WORD  WRTBUF     :IN A CLEANER FORMAT
714      .BYTE  3,5
715      .WORD  400,ERRTB5,0

```

10.5 RH/RM REGISTERS

	MNEMONIC	INDEX
	-----	-----
719		
720		
721		
722	RMCS1	0
723	RMWC	2
724	RMBA	4
725	RMDA	6
726	RMCS2	10
727	RMDS	12
728	RMER1	14
729	RMAS	16
730	RMLA	20
731	RMDB	22
732	RMMR1	24
733	RMDT	26
734	RMSN	30
735	RMOF	32
736	RMDC	34
737	RMHR	36
738	RMMR2	40
739	RMER2	42
740	RMEC1	44
741	RMEC2	46

742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798

10.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P
RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRT CHK HDR & DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

10.7 DPB STATUS/ERROR INDICATOR WORD

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

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

799		
800	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
801		
802	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
803		
804	7	DONE
805		
806	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
807		
808	5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
809		
810		
811	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
812		
813	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
814		
815		
816	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 15. SECONDS.
817		
818		
819		
820	1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
821		
822		

 NOTES FOR ABOVE

823		
824		
825		
826	(1) =	REQUEST WASN'T PUT IN QUEUE. (RH/RM REGISTERS WERE NOT SAVED)
827		
828		
829	(2) =	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH/RM REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
830		
831		
832		
833		
834	(3) =	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH/RM REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
835		
836		
837		
838		
839	(4) =	A "RECALIBRATE" SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.
840		

10.8 ERROR CALLS MADE BY THE DRIVER.

THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB.

WHEN THIS TYPE OF ERROR IS DETECTED BY THE DRIVER IT WILL MAKE AN ERROR CALL OF THE FORM "ERROR N", WHERE "N" IS THE ERROR NUMBER AND THE ERROR WILL BE AN EMT INSTRUCTION.

N	TYPE	DATA AVAILABLE
-	----	-----
1	RH70 INTERRUPT	*R4= RMCS1'S ADDRESS

855

856		OCCURRED (RHAS=0)	
857			
858	2	UNEXPECTED ATTENTION	R1= DRIVE NUMBER
859		OCCURRED	R3= ATA BIT
860			*R4= RMCS1'S ADDRESS
861			R5= (RMAS)
862			RMERRS =RMDS
863			RMERRS+2=RMER1
864			RMERRS+4=RMER2
865			RMERRS+6=RMMR2
866			
867	3	MASSBUS PARITY	RD.ADR= ADDRESS OF REG. READ
868		ERROR (MCPE=1)	RD.WRD= WORD READ
869			
870	4	MASSBUS PARITY	WRT.AD= ADDRESS OF REG. WRITTEN
871		ERROR (PAR=1)	WRT.WD= WORD WRITTEN
872			RD.WRD= WORD READ BACK
873			
874	5	ADDRESS PLUG CHANGE	R1= DRIVE NUMBER
875		BIT SET ('OPE' ERROR)	R3= ATA BIT
876			*R4= RMCS1'S ADDRESS
877			R5= (RMAS)
878			RMERRS =RMDS
879			RMERRS+2=RMER1
880			RMERRS+4=RMER2
881			RMERRS+6=RMMR2
882			
883		* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)	
884			
885			

1
61
62

63

64
65
66
67
68

.*LAST REVISION 04-APR-81

.TITLE CZRMLB0 RM05/3/2 FORMATTER

.*COPYRIGHT (C) 1981

.*DIGITAL EQUIPMENT CORPORATION

.*COLORADO SPGS., CO. 80919

.*PROGRAM BY MIKE LEAVITT

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

.*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
13	INHIBIT ERROR TYPEOUTS
10	BELL ON ERROR
9	LOOP ON ERROR
2	DON'T DISPLAY SYSTEM STATUS AFTER INITIAL START
1	LOOP ON THE CURRENT TRACK
0	LOOP THE PROGRAM ON THE SELECTED DRIVE

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004

STACK = 1100

ERROR = EMT

::BASIC DEFINITION OF ERROR CALL

SCOPE = IOT

::BASIC DEFINITION OF SCOPE CALL

.*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570

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

PSW=PS

STKLMT = 177774

::STACK LIMIT REGISTER

PIRQ = 177772

::PROGRAM INTERRUPT REQUEST REGISTER

DSWR = 177570

::HARDWARE SWITCH REGISTER

DDISP = 177570

::HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007

R0 = %0

::GENERAL REGISTER

R1 = %1

::GENERAL REGISTER

R2 = %2

::GENERAL REGISTER

R3 = %3

::GENERAL REGISTER

R4 = %4

::GENERAL REGISTER

R5 = %5

::GENERAL REGISTER

R6 = %6

::GENERAL REGISTER

R7 = %7

::GENERAL REGISTER

SP = %6

::STACK POINTER

PC = %7

::PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

000000

PRO = 0

::PRIORITY LEVEL 0

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

::*'SWITCH REGISTER' SWITCH DEFINITIONS

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

::*DATA BIT DEFINITIONS (BIT00 TO BIT15)

100000	BIT15	= 100000
040000	BIT14	= 40000
020000	BIT13	= 20000
010000	BIT12	= 10000
004000	BIT11	= 4000
002000	BIT10	= 2000
001000	BIT09	= 1000
000400	BIT08	= 400
000200	BIT07	= 200
000100	BIT06	= 100
000040	BIT05	= 40
000020	BIT04	= 20
000010	BIT03	= 10
000004	BIT02	= 4
000002	BIT01	= 2
000001	BIT00	= 1
001000	BIT9=BIT09	
000400	BIT8=BIT08	
000200	BIT7=BIT07	
000100	BIT6=BIT06	

000040
 000020
 000010
 000004
 000002
 000001

BIT5=BIT05
 BIT4=BIT04
 BIT3=BIT03
 BIT2=BIT02
 BIT1=BIT01
 BIT0=BIT00

000004
 000010
 000014
 000014
 000014
 000020
 000024
 000030
 000034
 000060
 000064
 000240

```

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

.SBTTL RM REGISTERS

:CONTROL AND STATUS REGISTER 1 (RMCS1)

000100
 000200
 000400
 001000
 002000
 020000
 040000

```

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

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

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

:CONTROL AND STATUS REGISTER 2 (RMCS2)

69
 70
 71
 72
 73
 74
 75
 76
 77
 78
 79
 80
 81
 82
 83
 84
 85
 86
 87
 88
 89
 90
 91
 92
 93
 94
 95
 96
 97
 98
 99
 100
 101
 102
 103
 104
 105

000001
 000002
 000004
 000010
 000020
 000040
 000100
 000200
 000400
 001000
 002000
 004000
 010000
 020000
 040000

```

US1     = 1        ; UNIT SELECT (BIT #0)
US2     = 2        ; UNIT SELECT (BIT #1)
US4     = 4        ; UNIT SELECT (BIT #2)
BAI     = 10       ; BUS ADDRESS INCREMENT INHIBIT (BIT #3)
PAT     = 20       ; MASSBUS PARITY TEST (BIT #4)
CLR     = 40       ; CLEAR (BIT #5)
IR      = 100      ; INPUT READY (BIT #6)
OR      = 200      ; OUTPUT READY (BIT #7)
MPE     = 400      ; MASS BUS PARITY ERROR (BIT #8)
MXF     = 1000     ; MISSED TRANSFER ERROR (BIT #9)
PGE     = 2000     ; PROGRAM ERROR (BIT #10)
NEM     = 4000     ; NON EXISTENT MEMORY (BIT #11)
NED     = 10000    ; NON EXISTENT DRIVE (BIT #12)
UPE     = 20000    ; UNIBUS PARITY ERROR (BIT #13)
WCE     = 40000    ; WRITE CHECK ERROR (BIT #14)
  
```

```

106          100000          DLT      = 100000          ;DATA LATE (BIT #15)
107
108          ;DATA BUFFER REGISTER (RMDB)
109          ;(EACH BIT IS CALLED BY BIT NUMBER)
110
111          .SBTTL  RM REGISTERS
112
113          ;CONTROL AND STATUS 1 REGISTER. (#00)
114
115          000001          GO       = 1              ;GO BIT (BIT #0)
116          000002          F1       = 2              ;FUNCTION CODE BIT #1
117          000004          F2       = 4              ;FUNCTION CODE BIT #2
118          000010          F3       = 10             ;FUNCTION CODE BIT #3
119          000020          F4       = 20             ;FUNCTION CODE BIT #4
120          000040          F5       = 40             ;FUNCTION CODE BIT #5
121          004000          DVA      = 4000           ;DEVICE AVAILABLE (BIT #11)
122
123          ;DRIVE STATUS REGISTER (RMDS1) (#01)
124
125          000100          VV       = 100            ;VOLUME VALID (BIT #6)
126          000200          DRY      = 200            ;DRIVE READY (BIT #7)
127          000400          DPR      = 400            ;DRIVE PRESENT (BIT #8)
128          001000          PGM      = 1000           ;PROGRAMABLE (BIT #9)
129          002000          LST      = 2000           ;LAST SECTOR TRANSFERRED (BIT #10)
130          004000          WRL      = 4000           ;WRITE LOCK (BIT #11)
131          010000          MOL      = 10000          ;MEDIUM ON-LINE (BIT #12)
132          020000          PIP      = 20000          ;POSITIONING OPERATION IN PROGRESS (BIT #13)
133          040000          ERR      = 40000          ;COMPOSITE ERROR (BIT #14)
134          100000          ATA      = 100000         ;ATTENTION ACTIVE (BIT #15)
135
136          ;ERROR REGISTER #01 (RMER1) (#02)
137
138          000001          ILF      = 1              ;ILLEGAL FUNCTION (BIT #0)
139          000002          ILR      = 2              ;ILLEGAL REGISTER (BIT #1)
140          000004          RMR      = 4              ;REGISTER MODIFICATION REFUSED (BIT #2)
141          000010          PAR      = 10             ;PARITY ERROR (BIT #3)
142          000020          FER      = 20             ;FORMAT ERROR (BIT #4)
143          000040          WCF      = 40             ;WRITE CLOCK FAIL (BIT #5)
144          000100          ECH      = 100            ;ECC HARD ERROR (BIT #6)
145          000200          HCE      = 200            ;HEADER COMPARE ERROR (BIT #7)
146          000400          HCRC     = 400            ;HEADER CRC ERROR (BIT #8)
147          001000          AOE      = 1000           ;ADDRESS OVERFLOW ERROR (BIT #9)
148          002000          IAE      = 2000           ;INVALID ADDRESS ERROR (BIT #10)
149          004000          WLE      = 4000           ;WRITE LOCK ERROR (BIT #11)
150          010000          DTE      = 10000          ;DRIVE TIMING ERROR (BIT #12)
151          020000          OPI      = 20000          ;OPERATION INCOMPLETE (BIT #13)
152          040000          UNS      = 40000          ;DRIVE UNSAFE (BIT #14)
153          100000          DCK      = 100000         ;DATA CHECK ERROR (BIT 15)
154
155          ;MAINTAINABILITY REGISTER (RMMR1) (#03)
156
157          000001          DMD      = 1              ;DIAGINOSTIC MODE (BIT #0)
158          000002          MCLK     = 2              ;MAINTAINABILITY CLOCK (BIT #1)
159          000004          MINX     = 4              ;MAINTAINABILITY INDEX (BIT #2)
160          000010          MSTCK    = 10             ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
161          000020          MRD      = 20             ;MAINTAINABILITY READ (BIT #4)
162          000040          MWR      = 40             ;MAINTAINABILITY WRITE (BIT #5)

```



```
163      000200      DTSY      = 200      ;MAINTAINABILITY SYNC DETECTED (BIT #7)
164
165      ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
166
167      000001      AT0       = 1       ;DEVICE 0 (BIT #0)
168      000002      AT1       = 2       ;DEVICE 1 (BIT #1)
169      000004      AT2       = 4       ;DEVICE 2 (BIT #2)
170      000010      AT3       = 10      ;DEVICE 3 (BIT #3)
171      000020      AT4       = 20      ;DEVICE 4 (BIT #4)
172      000040      AT5       = 40      ;DEVICE 5 (BIT #5)
173      000100      AT6       = 100     ;DEVICE 6 (BIT #6)
174      000200      AT7       = 200     ;DEVICE 7 (BIT #7)
175
176      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
177      ;(EACH BIT IS CALLED BY BIT NUMBER)
178
179      ;DRIVE TYPE REGISTER (RMDT) (#06)
180
181      000001      DT00      = 1       ;DRIVE TYPE NUMBER BIT 1
182      000002      DT01      = 2       ;DRIVE TYPE NUMBER BIT 2
183      000004      DT02      = 4       ;DRIVE TYPE NUMBER BIT 3
184      000010      DT03      = 10      ;DRIVE TYPE NUMBER BIT 4
185      000020      DT04      = 20      ;DRIVE TYPE NUMBER BIT 5
186      000040      DT05      = 40      ;DRIVE TYPE NUMBER BIT 6
187      000100      DT06      = 100     ;DRIVE TYPE NUMBER BIT 7
188      000200      DT07      = 200     ;DRIVE TYPE NUMBER BIT 8
189      000400      DT08      = 400     ;DRIVE TYPE NUMBER BIT 9
190      004000      DRQ       = 4000    ;DRIVE REQUEST REQUIRED (BIT #11)
191      020000      MOH       = 20000   ;MOVING HEAD (BIT #13)
192      040000      TAP       = 40000   ;TAPE DRIVE (BIT #14)
193      100000      NBA       = 100000  ;NOT BLOCK ADDRESSED (BIT #15)
194
195      ;LOOK-AHEAD REGISTER (RMLA) (#07)
196
197      000100      SC01      = 100     ;SECTOR COUNT FIELD 0 (BIT #6)
198      000200      SC02      = 200     ;SECTOR COUNT FIELD 1 (BIT #7)
199      000400      SC04      = 400     ;SECTOR COUNT FIELD 2 (BIT #8)
200      001000      SC10      = 1000    ;SECTOR COUNT FIELD 3 (BIT #9)
201      002000      SC20      = 2000    ;SECTOR COUNT FIELD 4 (BIT #10)
202
203      004000      TRK1      = 4000    ;TRACK FIELD 1 (BIT #11)
204      010000      TRK2      = 10000   ;TRACK FIELD 2 (BIT #12)
205      020000      TRK4      = 20000   ;TRACK FIELD 3 (BIT #13)
206      040000      TRK10     = 40000   ;TRACK FIELD 4 (BIT #14)
207      100000      TRK20     = 100000  ;TRACK FIELD 5 (BIT #15)
208
209      ;OFFSET REGISTER (RMOF) (#11)
210
211      000001      OFDIR     = 1       ;OFFSET DIRECTION
212      002000      HCI       = 2000    ;HEADER COMPARE INHIBIT (BIT #10)
213      004000      ECI       = 4000    ;ERROR CORRECTION CODE INHIBIT (BIT #11)
214      010000      FMT16     = 10000   ;FORMAT BIT (BIT #12)
215
216      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
217      ;(EACH BIT IS CALLED BY BIT NUMBER)
218
219      ;SERIAL NUMBER REGISTER (RMSN) (#14)
```

```

220      ;(EACH IS CALLED BY BIT NUMBER)
221
222      ;RM MAINTENANCE REGISTER #02 (RMMR2) (#15)
223
224      040000      SKI      = 40000      ;SEEK INCOMPLETE (BIT #14)
225
226      ;ERROR REGISTER #02 (RMER2)
227
228      100000      BSE      = 100000      ;BAD SECTOR ERROR (BIT #15)
229
230      ;ECC POSITION REGISTER (RMEC1) (#16)
231      ;(EACH BIT IS CALLED BY BIT NUMBER)
232
233      ;ECC PATTERN REGISTER (RMEC2) (#17)
234      ;(EACH BIT IS CALLED BY BIT NUMBER)
235
236      ;SOME 1ST HEADER WORD DEFINITIONS
237
238      100000      MF      = 100000      ;MANUFACTURES BAD SECTOR FLAG (BIT #15)
239      040000      UF      = 40000      ;USERS BAD SECTOR FLAG (BIT #14)
240      010000      FMT      = 10000      ;FORMAT 16 BIT MODE (BIT #12)
241
242      .SBTTL      RM DRIVER COMMANDS
243
244      000105      SEEK      = 105      ;SEEK
245      000107      RECAL      = 107      ;RECALIBRATE
246      000111      DRVCLR      = 111      ;DRIVE CLEAR
247      000113      RELSE      = 113      ;RELEASE
248      000115      OFFSET      = 115      ;OFFSET
249      000117      RTC      = 117      ;RETURN TO CENTER LINE
250      000121      READIN      = 121      ;READ IN PRESET
251      000123      ACK      = 123      ;PACK ACKNOWLEDGE
252      000131      SEARCH      = 131      ;SEARCH
253      000141      GETREG      = 141      ;GET REGISTERS
254      000143      SETFMT      = 143      ;SET FORMAT (& ECI OR HCI)
255      000145      SELDRV      = 145      ;SELECT DRIVE
256      000151      WCKD      = 151      ;WRITE CHECK DATA
257      000153      WCKHD      = 153      ;WRITE CHECK HEADER & DATA
258      000161      WRTDAT      = 161      ;WRITE DATA
259      000163      WRTHD      = 163      ;WRITE HEADER & DATA
260      000171      RDDAT      = 171      ;READ DATA
261      000173      RDHD      = 173      ;READ HEADER & DATA
262
263      176700      ABASE      = 176700
264      000254      AVECT1      = 254
265

```


1
2
3
4
5
6
7
8
9
10

000000

000174 000174
000176 000000
000176 000000

000200 000137 007012

000204 000137 007020

000046 000210
000046 000046
000052 016352
000052 000052
000052 020000
000210 000210

001100

000024 001100
000024 000024
000044 000200
000044 000044
000044 001100
001100 001100

001100
001100 000000
001102 001210
001104 002734
001106 002734
001110 002734
001112 000032
001114

.SBTTL TRAP CATCHER

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

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

.SBTTL STARTING ADDRESS(ES)

JMP @#START ;:JUMP TO STARTING ADDRESS OF PROGRAM

JMP @#START1 ;:CHANGE THE RH/RM ADDRESS

.SBTTL ACT11 HOOKS

:HOOKS REQUIRED BY ACT11
\$SVPC= . ;:SAVE PC
. = 46
\$ENDAD ;:1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .\$EOP
. = 52
.WORD 20000 ;:2)SET LOC.52 TO 20000
. = \$SVPC ;:RESTORE PC

. = 1100
.SBTTL APT PARAMETER BLOCK

:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT

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

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

\$APTHD:
\$HIBTS: .WORD 0 ;:TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
\$MBADR: .WORD \$MAIL ;:ADDRESS OF APT MAILBOX (BITS 0-15)
\$TSTM: .WORD 1500. ;:RUN TIM OF LONGEST TEST
\$PASTM: .WORD 1500. ;:RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD 1500. ;:ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
.WORD \$ETEND-\$MAIL/2 ;:LENGTH MAILBOX-ETABLE(WORDS)
TAB.XY= . ;:CMTAGSTARTING ADDRESS

0

.SBTTL COMMON TAGS

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

001114 001114
001114 000000
001116 000
001117 000
001120 000000
001122 000000
001124 000000
001126 000000
001130 000
001131 001
001132 000000
001134 000000
001136 000000
001140 000000
001142 000000
001144 000000
001146 000000
001150 000
001151 000
001152 000000
001154 177570
001156 177570
001160 177560
001162 177562
001164 177564
001166 177566
001170 000
001171 002
001172 012
001173 000
001174 000000
001176 000000
001200 207 377 377
001204 077
001205 015
001206 012 000

.=TAB.XY
\$CMTAG: ;:START OF COMMON TAGS
\$TSTNM: .WORD 0 ;:CONTAINS THE TEST NUMBER
\$ERFLG: .BYTE 0 ;:CONTAINS ERROR FLAG
\$ICNT: .WORD 0 ;:CONTAINS SUBTEST ITERATION COUNT
\$LPADR: .WORD 0 ;:CONTAINS SCOPE LOOP ADDRESS
\$LPERR: .WORD 0 ;:CONTAINS SCOPE RETURN FOR ERRORS
\$ERTTL: .WORD 0 ;:CONTAINS TOTAL ERRORS DETECTED
\$ITEMB: .BYTE 0 ;:CONTAINS ITEM CONTROL BYTE
\$ERMAX: .BYTE 1 ;:CONTAINS MAX. ERRORS PER TEST
\$ERRPC: .WORD 0 ;:CONTAINS PC OF LAST ERROR INSTRUCTION
\$GDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'GOOD' DATA
\$BDADR: .WORD 0 ;:CONTAINS ADDRESS OF 'BAD' DATA
\$GDDAT: .WORD 0 ;:CONTAINS 'GOOD' DATA
\$BDDAT: .WORD 0 ;:CONTAINS 'BAD' DATA
 ;:RESERVED--NOT TO BE USED
\$AUTOB: .BYTE 0 ;:AUTOMATIC MODE INDICATOR
\$INTAG: .BYTE 0 ;:INTERRUPT MODE INDICATOR
 ;:ADDRESS OF SWITCH REGISTER
SWR: .WORD DSWR ;:ADDRESS OF DISPLAY REGISTER
DISPLAY: .WORD DDISP ;:TTY KBD STATUS
\$TKS: 177560 ;:TTY KBD BUFFER
\$TKB: 177562 ;:TTY PRINTER STATUS REG. ADDRESS
\$TPS: 177564 ;:TTY PRINTER BUFFER REG. ADDRESS
\$TPB: 177566 ;:CONTAINS NULL CHARACTER FOR FILLS
\$NULL: .BYTE 0 ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
\$FILLS: .BYTE 2 ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
\$FILLC: .BYTE 12 ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
\$TPFLG: .BYTE 0 ;:USER DEFINED
\$TMP0: .WORD 0 ;:ESCAPE ON ERROR ADDRESS
\$ESCAPE: 0 ;:CODE FOR BELL
\$BELL: .ASCIZ <207><377><377> ;:QUESTION MARK
\$QUES: .ASCII /?/ ;:CARRIAGE RETURN
\$CRLF: .ASCII <15> ;:LINE FEED
\$LF: .ASCIZ <12>

.SBTTL APT MAILBOX-ETABLE

001210
001210 000000
001212 000000
001214 000000
001216 000000
001220 000000
001222 000000
001224 000000
001226 000000

.EVEN
\$MAIL: ;:APT MAILBOX
\$MSGTY: .WORD AMSGTY ;:MESSAGE TYPE CODE
\$FATAL: .WORD AFATAL ;:FATAL ERROR NUMBER
\$TESTN: .WORD ATESTN ;:TEST NUMBER
\$PASS: .WORD APASS ;:PASS COUNT
\$DEVCT: .WORD ADEVCT ;:DEVICE COUNT
\$UNIT: .WORD AUNIT ;:I/O UNIT NUMBER
\$MSGAD: .WORD AMSGAD ;:MESSAGE ADDRESS
\$MSGLG: .WORD AMSLG ;:MESSAGE LENGTH


```
001230      $ETABLE:      ::APT ENVIRONMENT TABLE
001230      000          $ENV: .BYTE  AENV      ::ENVIRONMENT BYTE
001231      000          $ENVM: .BYTE  AENVM     ::ENVIRONMENT MODE BITS
001232      000000     $SWREG: .WORD  ASWREG   ::APT SWITCH REGISTER
001234      000000     $USWR:  .WORD  AUSWR    ::USER SWITCHES
001236      000000     $CPUOP: .WORD  ACPUOP   ::CPU TYPE,OPTIONS
::*
::*          BITS 15-11=CPU TYPE
::*          11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
::*          11/70=06,PDQ=07,Q=10
::*          BIT 10=REAL TIME CLOCK
::*          BIT 9=FLOATING POINT PROCESSOR
::*          BIT 8=MEMORY MANAGEMENT
001240      000          $MAMS1: .BYTE  AMAMS1  ::HIGH ADDRESS,M.S. BYTE
001241      000          $MTYP1: .BYTE  AMTYP1  ::MEM. TYPE,BLK#1
::*          MEM.TYPE BYTE -- (HIGH BYTE)
::*          900 NSEC CORE=001
::*          300 NSEC BIPOLAR=002
::*          500 NSEC MOS=003
001242      000000     $MADR1: .WORD  AMADR1  ::HIGH ADDRESS,BLK#1
::*          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001244      000          $MAMS2: .BYTE  MAMS2   ::HIGH ADDRESS,M.S. BYTE
001245      000          $MTYP2: .BYTE  AMTYP2  ::MEM. TYPE,BLK#2
001246      000000     $MADR2: .WORD  AMADR2  ::MEM.LAST ADDRESS,BLK#2
001250      000          $MAMS3: .BYTE  AMAMS3  ::HIGH ADDRESS,M.S.BYTE
001251      000          $MTYP3: .BYTE  AMTYP3  ::MEM. TYPE,BLK#3
001252      000000     $MADR3: .WORD  AMADR3  ::MEM.LAST ADDRESS,BLK#3
001254      000          $MAMS4: .BYTE  AMAMS4  ::HIGH ADDRESS,M.S.BYTE
001255      000          $MTYP4: .BYTE  AMTYP4  ::MEM. TYPE,BLK#4
001256      000000     $MADR4: .WORD  AMADR4  ::MEM.LAST ADDRESS,BLK#4
001260      000254     $VECT1: .WORD  AVECT1  ::INTERRUPT VECTOR#1,BUS PRIORITY#1
001262      000000     $VECT2: .WORD  AVECT2  ::INTERRUPT VECTOR#2BUS PRIORITY#2
001264      176700     $BASE:  .WORD  ABASE   ::BASE ADDRESS OF EQUIPMENT UNDER TEST
001266      000000     $DEVN:  .WORD  ADEVN   ::DEVICE MAP
001270      000000     $CDW1:  .WORD  ACDW1   ::CONTROLLER DESCRIPTION WORD#1
001272      000000     $CDW2:  .WORD  ACDW2   ::CONTROLLER DESCRIPTION WORD#2
001274      .MEXIT
```

.SBTTL USER DEFINED TAGS

001274	176700	\$RMADR: .WORD	176700	:RH/RM UNIBUS ADDRESS
001276	000254	\$RMVEC: .WORD	254	:RH/RM INTERRUPT VECTOR
001300	172540	\$LKCSR: .WORD	172540	:ADDRESS OF KW11-P CSR
001302	172542	\$LKCSB: .WORD	172542	:ADDRESS OF KW11-P COUNTER BUFFER
001304	000104	000106	\$LPVEC: .WORD	104,106 :ADDRESS OF KW11-P VECTOR
001310	177546	\$LKS: .WORD	177546	:ADDRESS OF KW11-L CONTROL REGISTER
001312	000100	000102	\$LLVEC: .WORD	100,102 :ADDRESS OF KW11-L VECTOR
	001222	DRIVE	=\$UNIT	:CONTAINS DRIVE NUMBER SELECTED :SAME PARAMETER USED IN APT
001316	000000	SOF SW: .WORD	0	:CONTENTS ARE FOR SOFTWARE DECISIONS
001320	000000	MODE: .WORD	0	:FORMAT= 0, VERIFY HEADERS= 1
001322	000000	SNGSEC: .WORD	0	:READ MULTI-SECTOR= 0, READ SINGLE SECTOR= 1
001324	000000	LSTRK: .WORD	0	:CONTAINS THE LAST TRACK OF THE UNIT UNDER :TEST. RM02/3= 4., RM05= 18.
001326	000000	MAXCYL: .WORD	0	:ENDING CYLINDER
001330	000000	MINCYL: .WORD	0	:STARTING CYLINDER
001332	000000	MAXTRK: .WORD	0	:ENDING TRACK
001334	000000	MINTRK: .WORD	0	:STARTING TRACK
001336	155555	PATA: .WORD	155555	:1ST WORD OF PATTERN (WCP)
001340	133333	PATB: .WORD	133333	:2ND WORD OF PATTERN (WCP)
001342	000000	RETRY: .WORD	0	:MAINTAINS # OF WRITE RETRIES MADE
001344	000000	BADSEC: .WORD	0	:CONTAINS LAST BAD SECTOR ON FORMAT
001346	000000	SAVWC: .WORD	0	:CONTAINS WC FOR REMAINING SECTORS ON ERROR
001350	000000	WC: .WORD	0	:30 OR 32 SECTOR TRACK SIZE (IN WORDS)
001352	000000	MWC: .WORD	0	:2'S COMPLEMENT OF 'WC'
001354	000000	HEDERR: .WORD	0	:POSITIONING ERROR DURING FORMAT INDICATOR
001356	000000	SEC30: .WORD	0	:30 OR 32 SECTOR MODE INDICATOR : 0= 30. SECTOR MODE (18 BIT MODE) :-1= 32. SECTOR MODE (16 BIT MODE)
001360	000000	MAXSEC: .WORD	0	:MAXIMUM SECTOR ADDRESS, 29. FOR 18. BIT MODE OR :31. FOR 16 BIT MODE
001362	000000	DS.CYL: .WORD	0	:ADDRESS OF CURRENT CYLINDER
001364	000000	DS.TRK: .WORD	0	:ADDRESS OF CURRENT TRACK
001366	000000	DDRIVE: .WORD	0	:DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
001370	000000	ATTN: .WORD	0	:ATTENTION REGISTER IMAGE FOR 'DRIVER' :ERROR MESSAGES
001372	000000	CNTLC: .WORD	0	:ADDRESS OF '^C' RETURN
001374	000001	CHGADR: .WORD	1	: 'CHANGE RH/RM ADDRESS' FLAG= 1
001376	000000	WRAP: .WORD	0	:WRAP AROUND FLAG,FORMAT FROM HIGH TO LOW :CYLINDER NUMBER
001400	000000	000000	DEVMP: .WORD	0,0 :DEVICE MAP FOR APT, DEVMP+2 IS STORAGE FOR DEVICE MAP
001404	000000	BSFAVL: .WORD	0	:IF > 0, BAD SECTOR FILE AVAILABLE FROM PACK
001406	000000	UPDBSF: .WORD	0	:IF > 0, UPDATE BAD SECTOR FILE AT 'EOP'
001410	000000	HDRBIT: .WORD	0	:HEADER BIT INDICATOR FOR FIRST HEADER WORD; :BIT15=1 AND BIT14=1, GOOD SECTOR :BIT15=0 AND BIT14=1, MFG BAD SECTOR :BIT15=1 AND BIT14=0, USR BAD SECTOR
001412	000000	XXDP: .WORD	0	:THE LOW BYTE CONTAINS THE DRIVE NUMBER FROM WHICH :THE PROGRAM WAS LOADED. THE HIGH BYTE CONTAINS THE : 'XXDP' DEVICE CODE THE RM05/3/2.
				:THIS TABLE WILL CONTAIN A COPY OF THE 16 BIT 'MFG' FILE
001414		MFG16: .BLKW	256.	:MFG 16 BIT BAD SECTOR TABLE

002414 177777
002416 177777

.WORD -1
.WORD -1

;THIS TABLE WILL CONTAIN A COPY OF THE 18 BIT 'MFG' FILE

002420
003420 177777
003422 177777

MFG18: .BLKW 256. ;MFG 18 BIT BAD SECTOR TABLE
.WORD -1
.WORD -1

;THIS TABLE WILL CONTAIN THE 16 BIT 'USR' FILE IF PACK IS BEING FORMATTED
;IN 16 BIT MODE; ELSE THIS TABLE WILL CONTAIN THE 18 BIT 'USR' FILE IF THE
;PACK IS BEING FORMATTED IN 18 BIT MODE

003424
004424 177777
004426 177777

USTAB: .BLKW 256. ;USER BAD SECTOR TABLE
.WORD -1
.WORD -1

;THIS TABLE WILL SAVE THE 18 BIT 'USR' FILE IF PACK IS BEING FORMATTED
;IN 16 BIT MODE; ELSE THIS TABLE WILL SAVE THE 16 BIT 'USR' FILE IF THE
;PACK IS BEING FORMATTED IN 18 BIT MODE

004430
005430 177777
005432 177777

USSAV: .BLKW 256. ;USER BAD SECTOR SAVED TABLE
.WORD -1
.WORD -1

;CONTAINS A LIST OF CURRENT BAD SECTORS ON THE TRACK BEING FORMATTED

005434
005534 177777
005536 000000

SECCU: .BLKW 32. ;MAXIMUM OF 32. BAD SECTORS
.WORD -1
NEXT: .WORD 0 ;FLAG, IF NEXT=-1,ALL GOOD SPOT ON
;THE CURRENT TRACK ARE FORMATTED

;DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

005540 000
005541 000
005542 000
005543 000
005544 000000
005546 000000
005550 000
005551 000
005552 000000
005554 005560
005556 000000

FMTDPB: .BYTE 0 ;DRIVE NUMBER
.BYTE 0 ;OFFSET VALUE OR FMT16, ECI, AND HCI
.BYTE 0 ;COMMAND
.BYTE 0 ;PSEL AND A17 AND A16
.WORD 0 ;WORD COUNT (NEG)
.WORD 0 ;BUFFER ADDRESS
.BYTE 0 ;SECTOR ADDRESS
.BYTE 0 ;TRACK ADDRESS
.WORD 0 ;CYLINDER ADDRESS
.WORD RM.REG ;ERROR TABLE POINTER
.WORD 0 ;STATUS-ERROR INDICATOR
;BIT 15 = 1: ERROR OCCURRED
;BIT 07 = 1: DONE
;BIT 14-10 AND BIT 06-03
;INDICATE TYPE OF ERROR

;RH/RM REGISTERS STORED HERE AFTER AN OPERATION

005560 000000
005562 000000
005564 000000
005566 000000
005570 000000

RM.REG: .WORD 0 ;RMCS1
.WORD 0 ;RMWC
.WORD 0 ;RMBA
.WORD 0 ;RMDA
.WORD 0 ;RMCS2

```

005572 000000 .WORD 0 :RMD5
005574 000000 .WORD 0 :RMR1
005576 000000 .WORD 0 :RMA5
005600 000000 .WORD 0 :RMLA
005602 000000 .WORD 0 :RMD8
005604 000000 .WORD 0 :RMMR1
005606 000000 .WORD 0 :RMDT
005610 000000 .WORD 0 :RMSN
005612 000000 .WORD 0 :RMOF
005614 000000 .WORD 0 :RMCA
005616 000000 .WORD 0 :RMDC
005620 000000 .WORD 0 :RMR2
005622 000000 .WORD 0 :RMMR2
005624 000000 .WORD 0 :RMEC1
005626 000000 .WORD 0 :RMEC2

005630 000 FMTWRT: .BYTE 0 :SPECIAL DPB TO WRITE HEADER OF BAD SECTOR
005631 000 .BYTE 0
005632 000 .BYTE 0
005633 000 .BYTE 0
005634 177776 .WORD -2 :ONLY TWO WORDS FOR HEADER
005636 046224 .WORD RBUF :BUFFER ADDRESS
005640 000 .BYTE 0 :SECTOR
005641 000 .BYTE 0 :TRACK
005642 000000 .WORD 0 :CYLINDER ADDRESS
005644 000000 .WORD 0 :ERROR TABLE
: DONT' SAVE ANYTHING
005646 000000 .WORD 0 :STATUS WORD
  
```

:PARAMETER LIMIT POINTER TABLE

```

005650 005754 001466 001330 TABLE: PAR1,822.,MINCYL
005656 005764 001466 001326 PAR2,822.,MAXCYL
005664 005774 000004 001334 PAR3,4.,MINTRK
005672 006004 000004 001332 PAR4,4.,MAXTRK,0

005702 006014 TABLE2: PAR5 :ASCIZ MESSAGE FOR INPUT
005704 001466 001362 822.,DS.CYL
005710 000004 001364 4.,DS.TRK
005714 000037 001344 31.,BADSEC

005720 006014 TABLE3: PAR5 :ASCIZ MESSAGE FOR INPUT
005722 001466 001362 822.,DS.CYL
005726 000004 001364 4.,DS.TRK
005732 000035 001344 29.,BADSEC

005736 006032 TABLE4: PAR6 :ASCIZ MESSAGE FOR INPUT
005740 001466 001362 822.,DS.CYL
005744 000004 001364 4.,DS.TRK
005750 047300 001344 20160.,BADSEC
  
```

:ASCII MESSAGES FOR ADDRESS PARAMETERS

```

005754 115 111 116 PAR1: .ASCIZ /MINCYL /
005764 115 101 130 PAR2: .ASCIZ /MAXCYL /
005774 115 111 116 PAR3: .ASCIZ /MINTRK /
006004 115 101 130 PAR4: .ASCIZ /MAXTRK /
  
```


006014 103 131 114 PAR5: .ASCIZ /CYL,TRK,SEC= /
 006032 103 131 114 PAR6: .ASCIZ /CYL,TRK,BYTE= /
 .EVEN

006052 021414 021624 021676 TABCAL: .WORD FUNCT2,FUNCT3,FUNCT4,FUNCT6,FUNCT7

:SECTOR BUFFER ADDRESS TABLE

006064	046234	ADRTBL: .WORD	BUFP	:ADDRESS OF SECTOR 0
006066	047240	.WORD	BUFP+<516.*1.>	:ADDRESS OF SECTOR 1.
006070	050244	.WORD	BUFP+<516.*2.>	:ADDRESS OF SECTOR 2.
006072	051250	.WORD	BUFP+<516.*3.>	:ADDRESS OF SECTOR 3.
006074	052254	.WORD	BUFP+<516.*4.>	:ADDRESS OF SECTOR 4.
006076	053260	.WORD	BUFP+<516.*5.>	:ADDRESS OF SECTOR 5.
006100	054264	.WORD	BUFP+<516.*6.>	:ADDRESS OF SECTOR 6.
006102	055270	.WORD	BUFP+<516.*7.>	:ADDRESS OF SECTOR 7.
006104	056274	.WORD	BUFP+<516.*8.>	:ADDRESS OF SECTOR 8.
006106	057300	.WORD	BUFP+<516.*9.>	:ADDRESS OF SECTOR 9.
006110	060304	.WORD	BUFP+<516.*10.>	:ADDRESS OF SECTOR 10.
006112	061310	.WORD	BUFP+<516.*11.>	:ADDRESS OF SECTOR 11.
006114	062314	.WORD	BUFP+<516.*12.>	:ADDRESS OF SECTOR 12.
006116	063320	.WORD	BUFP+<516.*13.>	:ADDRESS OF SECTOR 13.
006120	064324	.WORD	BUFP+<516.*14.>	:ADDRESS OF SECTOR 14.
006122	065330	.WORD	BUFP+<516.*15.>	:ADDRESS OF SECTOR 15.
006124	066334	.WORD	BUFP+<516.*16.>	:ADDRESS OF SECTOR 16.
006126	067340	.WORD	BUFP+<516.*17.>	:ADDRESS OF SECTOR 17.
006130	070344	.WORD	BUFP+<516.*18.>	:ADDRESS OF SECTOR 18.
006132	071350	.WORD	BUFP+<516.*19.>	:ADDRESS OF SECTOR 19.
006134	072354	.WORD	BUFP+<516.*20.>	:ADDRESS OF SECTOR 20.
006136	073360	.WORD	BUFP+<516.*21.>	:ADDRESS OF SECTOR 21.
006140	074364	.WORD	BUFP+<516.*22.>	:ADDRESS OF SECTOR 22.
006142	075370	.WORD	BUFP+<516.*23.>	:ADDRESS OF SECTOR 23.
006144	076374	.WORD	BUFP+<516.*24.>	:ADDRESS OF SECTOR 24.
006146	077400	.WORD	BUFP+<516.*25.>	:ADDRESS OF SECTOR 25.
006150	100404	.WORD	BUFP+<516.*26.>	:ADDRESS OF SECTOR 26.
006152	101410	.WORD	BUFP+<516.*27.>	:ADDRESS OF SECTOR 27.
006154	102414	.WORD	BUFP+<516.*28.>	:ADDRESS OF SECTOR 28.
006156	103420	.WORD	BUFP+<516.*29.>	:ADDRESS OF SECTOR 29.
006160	104424	.WORD	BUFP+<516.*30.>	:ADDRESS OF SECTOR 30.
006162	105430	.WORD	BUFP+<516.*31.>	:ADDRESS OF SECTOR 31.

:REMAINING WORD COUNT TABLE

006164	000402	WCTBL: .WORD	258.	:REMAINING WORD COUNT AFTER SECTOR 0
006166	001004	.WORD	258.+<258.*1.>	:REMAINING WORD COUNT AFTER SECTOR 1.
006170	001406	.WORD	258.+<258.*2.>	:REMAINING WORD COUNT AFTER SECTOR 2.
006172	002010	.WORD	258.+<258.*3.>	:REMAINING WORD COUNT AFTER SECTOR 3.
006174	002412	.WORD	258.+<258.*4.>	:REMAINING WORD COUNT AFTER SECTOR 4.
006176	003014	.WORD	258.+<258.*5.>	:REMAINING WORD COUNT AFTER SECTOR 5.
006200	003416	.WORD	258.+<258.*6.>	:REMAINING WORD COUNT AFTER SECTOR 6.
006202	004020	.WORD	258.+<258.*7.>	:REMAINING WORD COUNT AFTER SECTOR 7.
006204	004422	.WORD	258.+<258.*8.>	:REMAINING WORD COUNT AFTER SECTOR 8.
006206	005024	.WORD	258.+<258.*9.>	:REMAINING WORD COUNT AFTER SECTOR 9.
006210	005426	.WORD	258.+<258.*10.>	:REMAINING WORD COUNT AFTER SECTOR 10.
006212	006030	.WORD	258.+<258.*11.>	:REMAINING WORD COUNT AFTER SECTOR 11.
006214	006432	.WORD	258.+<258.*12.>	:REMAINING WORD COUNT AFTER SECTOR 12.

006216	007034	.WORD	258.<258.*13.>	:REMAINING WORD COUNT AFTER SECTOR 13.
006220	007436	.WORD	258.<258.*14.>	:REMAINING WORD COUNT AFTER SECTOR 14.
006222	010040	.WORD	258.<258.*15.>	:REMAINING WORD COUNT AFTER SECTOR 15.
006224	010442	.WORD	258.<258.*16.>	:REMAINING WORD COUNT AFTER SECTOR 16.
006226	011044	.WORD	258.<258.*17.>	:REMAINING WORD COUNT AFTER SECTOR 17.
006230	011446	.WORD	258.<258.*18.>	:REMAINING WORD COUNT AFTER SECTOR 18.
006232	012050	.WORD	258.<258.*19.>	:REMAINING WORD COUNT AFTER SECTOR 19.
006234	012452	.WORD	258.<258.*20.>	:REMAINING WORD COUNT AFTER SECTOR 20.
006236	013054	.WORD	258.<258.*21.>	:REMAINING WORD COUNT AFTER SECTOR 21.
006240	013456	.WORD	258.<258.*22.>	:REMAINING WORD COUNT AFTER SECTOR 22.
006242	014060	.WORD	258.<258.*23.>	:REMAINING WORD COUNT AFTER SECTOR 23.
006244	014462	.WORD	258.<258.*24.>	:REMAINING WORD COUNT AFTER SECTOR 24.
006246	015064	.WORD	258.<258.*25.>	:REMAINING WORD COUNT AFTER SECTOR 25.
006250	015466	.WORD	258.<258.*26.>	:REMAINING WORD COUNT AFTER SECTOR 26.
006252	016070	.WORD	258.<258.*27.>	:REMAINING WORD COUNT AFTER SECTOR 27.
006254	016472	.WORD	258.<258.*28.>	:REMAINING WORD COUNT AFTER SECTOR 28.
006256	017074	.WORD	258.<258.*29.>	:REMAINING WORD COUNT AFTER SECTOR 29.
006260	017476	.WORD	258.<258.*30.>	:REMAINING WORD COUNT AFTER SECTOR 30.
006262	020100	.WORD	258.<258.*31.>	:REMAINING WORD COUNT AFTER SECTOR 31.

006264	001166	BYTE16: .WORD	630.	
006266	002354	.WORD	630.<630.*1.>	
006270	003542	.WORD	630.<630.*2.>	
006272	004730	.WORD	630.<630.*3.>	
006274	006116	.WORD	630.<630.*4.>	
006276	007304	.WORD	630.<630.*5.>	
006300	010472	.WORD	630.<630.*6.>	
006302	011660	.WORD	630.<630.*7.>	
006304	013046	.WORD	630.<630.*8.>	
006306	014234	.WORD	630.<630.*9.>	
006310	015422	.WORD	630.<630.*10.>	
006312	016610	.WORD	630.<630.*11.>	
006314	017776	.WORD	630.<630.*12.>	
006316	021164	.WORD	630.<630.*13.>	
006320	022352	.WORD	630.<630.*14.>	
006322	023540	.WORD	630.<630.*15.>	
006324	024726	.WORD	630.<630.*16.>	
006326	026114	.WORD	630.<630.*17.>	
006330	027302	.WORD	630.<630.*18.>	
006332	030470	.WORD	630.<630.*19.>	
006334	031656	.WORD	630.<630.*20.>	
006336	033044	.WORD	630.<630.*21.>	
006340	034232	.WORD	630.<630.*22.>	
006342	035420	.WORD	630.<630.*23.>	
006344	036606	.WORD	630.<630.*24.>	
006346	037774	.WORD	630.<630.*25.>	
006350	041162	.WORD	630.<630.*26.>	
006352	042350	.WORD	630.<630.*27.>	
006354	043536	.WORD	630.<630.*28.>	
006356	044724	.WORD	630.<630.*29.>	
006360	046112	.WORD	630.<630.*30.>	
006362	047300	.WORD	630.<630.*31.>	

006364	001240	BYTE18: .WORD	672.	
006366	002500	.WORD	672.<672.*1.>	
006370	003740	.WORD	672.<672.*2.>	
006372	005200	.WORD	672.<672.*3.>	

006374 006440
006376 007700
006400 011140
006402 012400
006404 013640
006406 015100
006410 016340
006412 017600
006414 021040
006416 022300
006420 023540
006422 025000
006424 026240
006426 027500
006430 030740
006432 032200
006434 033440
006436 034700
006440 036140
006442 037400
006444 040640
006446 042100
006450 043340
006452 044600
006454 046040
006456 047300

.WORD 672.*<672.*4.>
.WORD 672.*<672.*5.>
.WORD 672.*<672.*6.>
.WORD 672.*<672.*7.>
.WORD 672.*<672.*8.>
.WORD 672.*<672.*9.>
.WORD 672.*<672.*10.>
.WORD 672.*<672.*11.>
.WORD 672.*<672.*12.>
.WORD 672.*<672.*13.>
.WORD 672.*<672.*14.>
.WORD 672.*<672.*15.>
.WORD 672.*<672.*16.>
.WORD 672.*<672.*17.>
.WORD 672.*<672.*18.>
.WORD 672.*<672.*19.>
.WORD 672.*<672.*20.>
.WORD 672.*<672.*21.>
.WORD 672.*<672.*22.>
.WORD 672.*<672.*23.>
.WORD 672.*<672.*24.>
.WORD 672.*<672.*25.>
.WORD 672.*<672.*26.>
.WORD 672.*<672.*27.>
.WORD 672.*<672.*28.>
.WORD 672.*<672.*29.>

.EVEN

.SBTTL ERROR POINTER TABLE

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

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

Index	Item 1	Item 2	EM	DH	DT	DF	Description
1	006460						
2							
3							
4	006460	042446	EM1	DH1	DT1	DF1	:RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)
5	006462	043734					
6	006464	045270					
7	006466	045642					
8							
9							
10							
11	006470	042520	EM2	DH2	DT2	DF2	:UNEXPECTED ATTENTION OCCURRED
12	006472	044002					
13	006474	045306					
14	006476	045646					
15							
16							
17							
18	006500	042556	EM3	DH3	DT3	DF3	:MASSBUS PARITY ERROR (MCPE=1)
19	006502	044054					
20	006504	045322					
21	006506	045652					
22							
23							
24							
25	006510	042614	EM4	DH4	DT4	DF4	:MASSBUS PARITY ERROR (PAR=1)
26	006512	044125					
27	006514	045336					
28	006516	045656					
29							
30							
31							
32	006520	000000	0	0	0	0	:UNUSED
33	006522	000000					
34	006524	000000					
35	006526	000000					
36							
37							
38							
39	006530	042705	EM6	DH6	DT6	DF6	:RH CONTROLLER DIDN'T RESPOND TO ADDRESSING
40	006532	044210					
41	006534	045354					
42	006536	045662					

43				
44			:ERROR 7	
45				
46	006540	042760	EM7	:TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT
47	006542	045015	DH23	
48	006544	045356	DT7	
49	006546	045752	DF25	
50				
51			:ERROR 10	
52				
53	006550	043034	EM10	:DRIVE OFFLINE
54	006552	044223	DH10	
55	006554	045366	DT10	
56	006556	045666	DF10	
57				
58			:ERROR 11	
59				
60	006560	043052	EM11	:PERSISTENT DRIVE UNSAFE ERROR
61	006562	044223	DH10	
62	006564	045366	DT10	
63	006566	045666	DF10	
64				
65			:ERROR 12	
66				
67	006570	043110	EM12	:UNCORRECTABLE MASSBUS PARITY ERROR
68	006572	044223	DH10	
69	006574	045366	DT10	
70	006576	045666	DF10	
71				
72			:ERROR 13	
73				
74	006600	043153	EM13	:SOFTWARE TIMEOUT
75	006602	044223	DH10	
76	006604	045366	DT10	
77	006606	045666	DF10	
78				
79			:ERROR 14	
80				
81	006610	043174	EM14	:DRIVE UNSAFE ERROR
82	006612	044223	DH10	
83	006614	045366	DT10	
84	006616	045666	DF10	
85				
86			:ERROR 15	
87				
88	006620	043217	EM15	:CONTROLLER/DRIVE ERROR DURING WRITE
89	006622	044223	DH10	
90	006624	045366	DT10	
91	006626	045666	DF10	
92				
93			:ERROR 16	
94				
95	006630	043263	EM16	:CONTROLLER/DRIVE ERROR DURING WRITE CHECK
96	006632	044223	DH10	
97	006634	045366	DT10	
98	006636	045666	DF10	
99				

100			:ERROR 17	
101				
102	006640	043335	EM17	:RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE
103	006642	044452	DH17	
104	006644	045440	DT17	
105	006646	045702	DF17	
106				
107			:ERROR 20	
108				
109	006650	043413	EM20	:DATA ERROR DURING WRITE CHECK
110	006652	044521	DH20	
111	006654	045452	DT20	
112	006656	045706	DF20	
113				
114			:ERROR 21	
115				
116	006660	043451	EM21	:CONTROLLER/DRIVE ERROR VERIFYING HEADERS
117	006662	044223	DH10	
118	006664	045366	DT10	
119	006666	045666	DF10	
120				
121			:ERROR 22	
122				
123	006670	043522	EM22	: 'HCE' OR 'HCRC' ERROR VERIFYING HEADERS
124	006672	044223	DH10	
125	006674	045366	DT10	
126	006676	045666	DF10	
127				
128			:ERROR 23	
129				
130	006700	043572	EM23	:CYLINDER FIELD IN HEADER IS NOT CORRECT
131	006702	045015	DH23	
132	006704	045534	DT23	
133	006706	045752	DF25	
134				
135			:ERROR 24	
136				
137	006710	043642	EM24	:WRITE CHECK ERROR
138	006712	045054	DH24	
139	006714	045544	DT24	
140	006716	045732	DF24	
141				
142			:ERROR 25	
143				
144	006720	043664	EM25	:ILLEGAL BAD SECTOR
145	006722	045227	DH25	
146	006724	045632	DT25	
147	006726	045752	DF25	


```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3 006730 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4 006732 005740      TST      -(R0)      ;ADJUST PC -2
5 006734 022626      CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
6 006736 104401 006744  TYPE      ,65$      ;:TYPE ASCIZ STRING
   006742 000417      BR       64$      ;:GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
   64$:
7 007002 010046      MOV      R0,-(SP)    ;SETUP FOR TYPING OUT PC
8 007004 104402      TYPOC
9 007006 000240      NOP
   ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
   ;TO STOP ON UNEXPECTED TIMEOUT.
10
11 007010 000411      BR       TST1      ;BRANCH TO START
12
13      .SBTTL START OF PROGRAM
14
15 007012 005037 001270  START: CLR      $CDW1      ;DISABLE USE OF UTILITY ROUTINES
16 007016 000406      BR       TST1
17
18 007020 012737 000001 001374  START1: MOV     #1,CHGADR    ;ENABLE CHANGE OF 'RH/RM BUS ADDRESS'
19 007026 012737 177777 001270  MOV     #-1,$CDW1      ;ENABLE USE OF UTILITY ROUTINES
20
21      ;:*****
   TST1: NOP
22 007034 000240      MOV     #1,$TESTN      ;:SET TEST NUMBER IN APT MAIL BOX
23 007036 012737 000001 001214  INC     #0              ;:TTY LOOP, WAIT FOR INCREMENT
24 007044 005227 000000      BNE    -4              ;:OF WORD
25 007050 001375      RESET
26 007052 000005      ;:CLEAR THE WORLD
   .SBTTL INITIALIZE THE COMMON TAGS
   ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
   MOV     #$CMTAG,R6    ;:FIRST LOCATION TO BE CLEARED
   CLR    (R6)+          ;:CLEAR MEMORY LOCATION
   CMP    #SWR,R6       ;:DONE?
   BNE    -6             ;:LOOP BACK IF NO
   MOV    #STACK,SP     ;:SETUP THE STACK POINTER
   ;:INITIALIZE A FEW VECTORS
   MOV    #ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
   MOV    #340,@#EMTVEC+2 ;:LEVEL 7
   MOV    #TRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
   MOV    #340,@#TRAPVEC+2 ;:LEVEL 7
   MOV    #PWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
   MOV    #340,@#PWRVEC+2 ;:LEVEL 7
   CLR    $ESCAPE       ;:CLEAR THE ESCAPE ON ERROR ADDRESS
   MOV    #1,$ERMAX     ;:ALLOW ONE ERROR PER TEST
   ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
   ;:EQUAL TO A '-1', SETUP FOR A SOFTWARE SWITCH REGISTER.
   MOV    @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
   MOV    #64$,@#ERRVEC ;:SET UP ERROR VECTOR
   MOV    #DSWR,SWR     ;:SETUP FOR A HARDWARE SWICH REGISTER
   MOV    #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
   CMP    #-1,@SWR      ;:TRY TO REFERENCE HARDWARE SWR
   BNE    66$          ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
   ;:AND THE HARDWARE SWR IS NOT = -1
   BR     65$          ;:BRANCH IF NO TIMEOUT
   64$:  MOV    #65$,(SP) ;:SET UP FOR TRAP RETURN

```

```

007216 000002          RTI
007220 012737 000176 001154 65$: MOV #SWREG,SWR      ;;POINT TO SOFTWARE SWR
007226 012737 000174 001156      MOV #DISPREG,DISPLAY
007234 012637 000004      MOV (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

007240 005037 001216          CLR $PASS          ;;CLEAR PASS COUNT
007244 132737 000200 001231      BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
007252 001403          BEQ 67$          ;;YES,USE NON-APT SWITCH
007254 012737 001232 001154      MOV #SWREG,SWR    ;;NO,USE APT SWITCH REGISTER
007262          67$:
27          ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
28 007262 012737 006730 000004      MOV #BADTMO,ERRVEC ;;SETUP FOR UNEXPECTED TIMEOUT
29 007270 012737 000300 000006      MOV #PR6,ERRVEC+2  ;LEVEL 6
30
31          .SBTTL TYPE PROGRAM NAME
          ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
007276 005227 177777          INC #-1           ;;FIRST TIME?
007302 001023          BNE 68$          ;;BRANCH IF NO
007304 104401 007312          TYPE ,69$       ;;TYPE ASCIZ STRING
007310 000420          BR 68$          ;;GET OVER THE ASCIZ
          ;;69$: .ASCIZ <CRLF>@CZRMLB0 - RM05/3/2 FORMATTER@<CRLF>
          68$:
          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
007352 005737 000042          TST @#42         ;;ARE WE RUNNING UNDER XXDP/ACT?
007356 001012          BNE 70$          ;;BRANCH IF YES
007360 123727 001230 000001      CMPB $ENV,#1     ;;ARE WE RUNNING UNDER APT?
007366 001406          BEQ 70$          ;;BRANCH IF YES
007370 023727 001154 000176      CMP SWR,#SWREG  ;;SOFTWARE SWITCH REG SELECTED?
007376 001005          BNE 71$          ;;BRANCH IF NO
007400 104406          GTSWR          ;;GET SOFT-SWR SETTINGS
007402 000403          BR 71$
007404 112737 000001 001150 70$: MOVB #1,$AUTOB   ;;SET AUTO-MODE INDICATOR
007412          71$:

32          ;THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
33          ;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
34
35          CLR XXDP          ;CLEAR 'XXDP' LOAD DEVICE STORAGE
36 007412 005037 001412          CMPB #16,@#41   ;LOADED FROM AN RM05/3/2 ?
37 007416 122737 000016 000041      BNE 4$          ;BR IF NOT
38 007424 001160          MOV @#40,XXDP   ;GET DEVICE INDICATOR AND NUMBER
39 007426 013737 000040 001412      CMPB #7,XXDP   ;IS IT A VALID NUMBER ?
40 007434 122737 000007 001412      BHIS 2$        ;YES
41 007442 103002          CLRB XXDP        ;NO, DEFAULT TO DRIVE 0
42 007444 105037 001412          TST @#42        ;CHAIN MODE OR ACT11 AUTO ACCEPT ?
43 007450 005737 000042 2$:        BEQ 3$          ;BR IF NEITHER
44 007454 001425          TYPE ,73$     ;;TYPE ASCIZ STRING
45 007456 104401 007464          BR 72$        ;;GET OVER THE ASCIZ
          ;;73$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
          72$:
46 007510          CLR -(SP)          ;CLEAR WORD ON STACK
47 007512 005046 113716 001412      MOVB XXDP,(SP)  ;GET DRIVE ADDRESS
48 007516 104403          TYPOS        ;TYPE THE ADDRESS
49 007520 001          .BYTE 1          ;ONLY 1 CHARACTER
50 007521 000          .BYTE 0          ;SUPPRESS LEADING ZEROS
51 007522 104401 001205          TYPE ,SCRLF   ;CR-LF
52 007526 000517          BR 4$          ;GET NUMBER OF DRIVES

```



```

53
54 007530 005227 177777 3$: INC #-1 ;FIRST TIME THRU HERE ?
55 007534 001114 BNE 4$ ;NO
56 007536 104401 007544 TYPE ,75$ ;:TYPE ASCIZ STRING
007542 000410 BR 74$ ;:GET OVER THE ASCIZ
;:75$: .ASCIZ <CRLF>/TO TEST.DRIVE /
74$:
57 007564 005046 CLR -(SP) ;CLEAR WORD ON STACK
58 007566 113716 001412 MOVB XXDP,(SP) ;GET DRIVE ADDRESS
59 007572 104403 TYPOS ;TYPE DRIVE ADDRESS
60 007574 001 .BYTE 1 ;ONLY 1 CHARACTER
61 007575 000 .BYTE 0 ;SUPPRESS LEADING ZEROS
62 007576 104401 007604 TYPE ,77$ ;:TYPE ASCIZ STRING
007602 000431 BR 76$ ;:GET OVER THE ASCIZ
;:77$: .ASCIZ /, HALT PROGRAM, REMOVE RRD PCK AND REPLACE IT/<CRLF>
76$:
63 007666 104401 007674 TYPE ,78$ ;:TYPE ASCIZ STRING
007672 000435 BR 4$ ;:GET OVER THE ASCIZ
;:78$: .ASCIZ /WITH A WORK PCK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
4$:
67 007766 012737 007012 001372 MOV #START,CNTLC ;'CONTROL C' ADDRESS
68 007774 004737 024414 JSR PC,$TKINT ;TURN ON THE KEYBOARD INTERRUPT
69
70 010000 105737 001150 TSTB $AUTOB ;RUNNING AUTO MODE ?
71 010004 001010 BNE 5$ ;BR IF YES
72 010006 004737 045756 JSR PC,BUSADR ;CHECK THE RH/RM ADDRESS
73 010012 013737 001274 030456 MOV $RMADR,RMADR ;RH/RM ADDRESS
74 010020 013737 001276 030460 MOV $RMVEC,RMVEC ;RH/RM VECTOR ADDRESS
75 010026
76 010026 105737 001230 5$: TSTB $ENV ;RUN UNDER APT MODE?
77 010032 001422 BEQ SETVEC ;NO, DO NOT LOAD FROM E-TABLE
78 010034 105737 001260 TSTB $VECT1 ;UUT INTERRUPT VECTOR
79 010040 001406 BEQ 6$ ;DO NOT CHANGE IF = 0
80 010042 113737 001260 001276 MOVB $VECT1,$RMVEC ;NEW VALUE
81 010050 013737 001276 030460 MOV $RMVEC,RMVEC ;RH/RM VECTOR
82 010056 005737 001264 6$: TST $BASE ;UUT BASE REGISTER ADDRESS
83 010062 001406 BEQ SETVEC ;DO NOT CHANGE IF = 0
84 010064 013737 001264 001274 MOV $BASE,$RMADR ;NEW BASE ADDRESS
85 010072 013737 001274 030456 MOV $RMADR,RMADR ;RH/RM ADDRESS
86
87 ;DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
88 ;PROGRAM WILL USE
89
90 010100 004737 017360 SETVEC: JSR PC,ST.CLK ;START THE CLOCK
91 010104 004737 030466 JSR PC,RMINIT ;INITIALIZE THE RM DRIVER
92 010110 013700 030456 MOV RMADR,R0 ;GET BASE ADDRESS
93 010114 012760 000040 000010 MOV #CLR,RMCS2(R0) ;CLEAR MASSBUS
94 010122 012737 177777 030416 MOV #-1,SAVEFG ;SET THE SAVE REGISTERS FLAG
95 010130 005227 177777 INC #-1 ;FIRST TIME THRU HERE ?
96 010134 001404 BEQ 1$ ;BR IF NO
97 010136 032777 000004 171010 BIT #SW02,@SWR ;TYPEOUT THE DRIVE STATUS TABLE ?
98 010144 001113 BNE 12$ ;BR IF NO
99 010146 005037 001400 1$: CLR DEVMP ;CLEAR DEVICE MAP
100 010152 005004 CLR R4 ;DRIVE TABLE POINTER
101 010154 104401 036555 TYPE ,UNSTAT ;TYPE 'UNIT STATUS'
102 010160 104401 001205 2$: TYPE ,CRLF ;CR-LF
103 010164 010446 MOV R4,-(SP) ;:SAVE R4 FOR TYPEOUT

```



```

1      .SBTTL  SIZE FOR AUTO MODE
2
3      ;IF RUNNING IN AUTO MODE, THEN AUTO SIZE DRIVES TO BE FORMATTED
4
5 010406 105737 001150      XSIZE:  TSTB   $AUTOB      ;RUNNING UNDER AUTO MODE ?
6 010412 001540              BEQ     ASKO        ;BR IF NO
7 010414 132737 000200 001231  BITB   #BIT7,$ENVM   ;APT AUTO SIZING ALLOWED ?
8 010422 001134              BNE     ASKO        ;BR IF NO
9
10 010424 005737 001266      TST     $DEVMP      ;DEVICE MAP AVAILABLE FROM APT MAIL BOX
11 010430 001406              BEQ     1$          ;BR IF NONE
12 010432 013737 001266 001400  MOV    $DEVMP,DEVMP ;LOAD IT FROM MAIL BOX
13 010440 042737 177400 001400  BIC    #177400,DEVMP ;ALLOW MAX 8 DRIVES
14 010446
15
16      ;RETURN HERE IF RUNNING IN CHAIN MODE AND MORE DRIVES TO TEST
17
18 010446 005737 001400      ST1:   TST     DEVMP      ;CHECK DEVICE MAP
19 010452 001002              BNE     1$          ;BR IF DRIVES ASSIGNED
20 010454 000137 016342      JMP     $GET42      ;GET MONITOR
21
22 010460 012737 000000 001334  1$:   MOV    #0,MINTRK   ;ALWAYS STARTS FROM 0 TRACK
23 010466 012737 000000 001330  MOV    #0,MINCYL   ;ALWAYS STARTS FROM 0 CYLINDER
24 010474 012737 001466 001326  MOV    #822.,MAXCYL ;LAST CYLINDER
25 010502 005037 001376      CLR    WRAP        ;SEEK FROM LO TO HI CYLINDER ADDRESS
26 010506 005037 001320      CLR    MODE        ;SET TO 'FORMAT' MODE
27 010512 012737 177777 001356  2$:   MOV    #-1,SEC30   ;SET 32 SECTOR
28 010520 005737 001272      TST    $CDW2       ;APT PARAMETER =0
29 010524 001012              BNE     3$          ;NO,30 SECTOR
30 010526 012737 020100 001350  MOV    #<258.*32.>,WC ;COUNT OF WORDS
31 010534 012737 157700 001352  MOV    #-<258.*32.>,MWC ;WORD COUNT
32 010542 012737 000037 001360  MOV    #31.,MAXSEC  ;MAX 32 SECTOR
33 010550 000413              BR     4$
34
35 010552 012737 017074 001350  3$:   MOV    #<258.*30.>,WC ;COUNT OF WORDS
36 010560 012737 160704 001352  MOV    #-<258.*30.>,MWC ;WORD COUNT
37 010566 005037 001356      CLR    SEC30       ;30 SECTOR
38 010572 012737 000035 001360  MOV    #29.,MAXSEC  ;MAX 30 SECTOR
39 010600 005001              CLR    R1          ;START WITH DRIVE NUMBER 0
40 010602 136137 030444 001400  4$:   BITB   ATABIT(R1),DEVMP ;IS THIS DRIVE IN DEVICE MAP ?
41 010610 001005              BNE     6$          ;BR IF YES
42 010612 005201              INC    R1          ;INCREMENT DRIVE #
43 010614 020127 000007      CMP    R1,#7       ;ALL DONE WITH MAP ?
44 010620 003770              BLE    5$          ;BR IF NO
45 010622 000711              BR     ST1         ;DONE !!
46
47 010624 010137 001222      6$:   MOV    R1,DRIVE    ;LOAD R1 INTO DRIVE
48 010630 146137 030444 001400  BICB   ATABIT(R1),DEVMP ;CLEAR THE BIT FROM DEVICE MAP
49 010636 105761 030340      TSTB   DRVSTA(R1)  ;IS DRIVE ON LINE ?
50 010642 003757              BLE    5$          ;BR IF NO
51 010644 005737 001412      TST    XXDP        ;LOADED FROM THIS DEVICE ?
52 010650 001403              BEQ    7$          ;BR IF NO
53 010652 123701 001412      CMPB   XXDP,R1     ;LOADED FROM THIS DRIVE ?
54 010656 001751              BEQ    5$          ;BR IF YES
55
56 010660 012702 000004      7$:   MOV    #4,R2       ;SETUP TRACK PARAMETER FOR AN RM03/2
57 010664 122761 000007 030350  CMPB   #7,DRVSTYP(R1) ;IS DEVICE AN RM05 ?
    
```

```

58 010672 001002          BNE      8$          ;BR IF NO
59 010674 012702 000022  MOV      #18.,R2      ;SETUP TRACK PARAMETER FOR AN RM05
60 010700 010237 001324  8$:  MOV      R2,LSTRK   ;SETUP LAST TRACK AND
61 010704 010237 001332  MOV      R2,MAXTRK   ;END TRACK FOR AUTO MODE.
62 010710 000137 011566  JMP      ST2         ;START TO FORMAT OR CHECK
63
64                      ;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
65                      ;MODES ARE: 'FORMAT & VERIFY' OR 'VERIFY HEADER FORMAT'
66
67 010714 005037 177776  ASK0:  CLR      PS          ;ASSUME 'FORMAT' MODE
68 010720 005037 001320  CLR      MODE        ;CR-LF
69 010724 104401 001205  TYPE    ,SCLF        ;TYPE 'DO YOU WANT TO FORMAT (L) Y ?'
70 010730 104401 041732  1$:  TYPE    ,MMODE    ;READ THE KEYBOARD
71 010734 104411          RDLIN           ;SAVE ADDRESS OF RESPONSE
72 010736 012600          MOV      (SP)+,R0    ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'Y')?
73 010740 105710          TSTB     (R0)        ;BR IF YES
74 010742 001416          BEQ      4$          ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
75 010744 105760 000001  TSTB     1(R0)       ;BR IF NO
76 010750 001006          BNE      2$          ;WAS IT A 'Y' RESPONSE ?
77 010752 122710 000131  CMPB     #'Y',(R0)    ;BR IF YES
78 010756 001410          BEQ      4$          ;WAS IT A 'N' RESPONSE ?
79 010760 122710 000116  CMPB     #'N',(R0)    ;BR IF YES
80 010764 001403          BEQ      3$          ;TYPE BAD ENTRY MESSAGE
81 010766 104401 037174  2$:  TYPE    ,BADENT   ;TRY AGAIN
82 010772 000756          BR       1$
83
84 010774 005237 001320  3$:  INC      MODE        ;SET 'VERIFY HEADERS ONLY' MODE
85 011000 4$:
86
87                      ;FIND OUT IF FORMAT IS TO BE IN 16. OR 18. BIT MODE
88
89 011000 104401 041771  ASK1:  TYPE    ,MSIZE    ;TYPE 'DO YOU WANT 16 BIT FORMAT MODE (L) Y ?'
90 011004 104411          RDLIN           ;READ THE ENTRY
91 011006 012600          MOV      (SP)+,R0    ;SAVE ADDRESS OF RESPONSE
92 011010 105710          TSTB     (R0)        ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'Y')?
93 011012 001414          BEQ      2$          ;BR IF YES
94 011014 105760 000001  TSTB     1(R0)       ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
95 011020 001006          BNE      1$          ;BR IF NO
96 011022 122710 000131  CMPB     #'Y',(R0)    ;WAS IT A 'Y' RESPONSE ?
97 011026 001406          BEQ      2$          ;BR IF YES
98 011030 122710 000116  CMPB     #'N',(R0)    ;WAS IT A 'N' RESPONSE ?
99 011034 001420          BEQ      3$          ;BR IF YES
100 011036 104401 037174  1$:  TYPE    ,BADENT   ;TYPE BAD ENTRY MESSAGE
101 011042 000756          BR       ASK1       ;TRY AGAIN
102
103 011044 012737 177777 001356  2$:  MOV      #-1,SEC30   ;32 SECTOR INDICATOR
104 011052 012737 020100 001350  MOV      #<258.*32.>,WC ;TRACK SIZE IN 32 SECTOR MODE
105 011060 012737 157700 001352  MOV      #-<258.*32.>,MWC ;2'S COMPLEMENT WORD COUNT
106 011066 012737 000037 001360  MOV      #31.,MAXSEC  ;MAX SECTOR ADDRESS IN 32 SECTOR MODE
107 011074 000413          BR       4$          ;CONTINUE
108
109 011076 005037 001356  3$:  CLR      SEC30       ;30 SECTOR INDICATOR
110 011102 012737 017074 001350  MOV      #<258.*30.>,WC ;TRACK SIZE IN 30 SECTOR MODE
111 011110 012737 160704 001352  MOV      #-<258.*30.>,MWC ;2'S COMPLEMENT WORD COUNT
112 011116 012737 000035 001360  MOV      #29.,MAXSEC  ;MAX SECTOR ADDRESS IN 30 SECTOR MODE
113 011124
114

```



```

115          ;GO FIND OUT WHAT DRIVE
116
117 011124 012737 017630 001372 ASK2:  MOV   #C.ENTR,CNTLC  ;CONTROL C ABORT ENTRANCE
118 011132 004737 024414          JSR   PC,$TKINT   ;INITIALIZE THE TTY KEYBOARD
119 011136 005037 001126          CLR   $ERTTL     ;CLEAR THE ERROR ACCUMULATOR
120 011142 104401 036573          TYPE ,MUNIT     ;TYPE 'DRIVE'
121 011146 104401 036604          TYPE ,COLON     ;TYPE ':'
122 011152 104411          RDLIN          ;READ THE ENTRY
123 011154 012601          MOV   (SP)+,R1   ;SAVE ADDRESS OF RESPONSE
124 011156 004537 023120          JSR   R5,CK.CHR ;CHECK ONE CHARACTER
          1$          ;ILLEGAL CHARACTER
          1$          ;CARRIAGE RETURN
          1$          ;...
          1$          ;...
          2$          ;DIGIT 0-7
          1$          ;DIGIT 8-9

125
126 011176 104401 037174          1$:   TYPE   ,BADENT  ;TYPE BAD ENTRY MESSAGE
127 011202 000750          BR     ASK2      ;ASK FOR DRIVE NUMBER AGAIN
128
129 011204 010237 001222          2$:   MOV   R2,DRIVE ;SAVE DRIVE NUMBER
130 011210 005737 001412          TST   XXDP      ;LOADED FROM THIS DEVICE ?
131 011214 001406          BEQ   3$        ;BR IF NO
132 011216 123702 001412          CMPB  XXDP,R2   ;WAS THIS THE DRIVE ?
133 011222 001003          BNE   3$        ;BR IF NO
134 011224 104401 036771          TYPE ,MLODEV   ;DRIVE IS LOAD DEVICE
135 011230 000735          BR     ASK2
136
137 011232 004737 017360          3$:   JSR   PC,ST.CLK ;START THE CLOCK
138 011236 004737 030466          JSR   PC,RMINIT ;GO SEE WHAT DRIVES ARE AVAILABLE
139 011242 013700 030456          MOV   RMADR,R0  ;GET BASE ADDRESS
140 011246 012760 000040 000010  MOV   #CLR,RMCS2(R0) ;CLEAR MASSBUS
141 011254 012737 177777 030416  MOV   #-1,SAVEFG ;SAVE THE REGISTERS
142 011262 012737 177777 030420  MOV   #-1,SEEKFG ;SET 'NO OPTIMIZATION' FLAG
143 011270 005037 177776          CLR   PS        ;PRIORITY = 0
144 011274 105762 030340          TSTB  DRVSTA(R2) ;LOOK AT DRIVE STATUS
145 011300 003015          BGT   6$        ;BR IF ONLINE
146 011302 001403          BEQ   4$        ;BR IF DRIVE NOT AVAILABLE
147 011304 104401 036751          TYPE ,MUSDR    ;'DRIVE UNSAFE'
148 011310 000705          BR     ASK2     ;GO GET DRIVE NUMBER AGAIN
149
150 011312 105762 030350          4$:   TSTB  DRVSTYP(R2) ;A DRIVE PRESENT?
151 011316 001003          BNE   5$        ;BR IF SO
152 011320 104401 036675          TYPE ,MDRNP    ;TYPE 'DRIVE NOT PRESENT'
153 011324 000677          BR     ASK2     ;GO GET DRIVE NUMBER AGAIN
154
155 011326 104401 036722          5$:   TYPE   ,MER11  ;'DRIVE NOT AVAILABLE'
156 011332 000674          BR     ASK2     ;GO GET DRIVE NUMBER AGAIN
157
158 011334 113737 001222 005540  6$:   MOVB  DRIVE,FMTDPB ;SETUP DRIVE ADDRESS
159 011342 012737 000000 001330  MOV   #0,MINCYL ;CLEAR STARTING CYLINDER ADDRESS
160 011350 012737 001466 001326  MOV   #822,MAXCYL ;SETUP END CYLINDER
161 011356 012737 000000 001334  MOV   #0,MINTRK ;CLEAR STARTING TRACK ADDRESS
162 011364 012700 000004          MOV   #4,R0     ;GET LAST TRACK FOR AN RM03/2
163 011370 122762 000007 030350  CMPB  #7,DRVSTYP(R2) ;IS DEVICE AN RM05 ?
164 011376 001002          BNE   7$        ;BR IF NO
165 011400 012700 000022          MOV   #18.,R0  ;GET LAST TRACK FOR AN RM05
    
```



```

166 011404 010037 001324      7$:  MOV    R0,LSTRK      ;SETUP LAST TRACK AND
167 011410 010037 001332      MOV    R0,MAXTRK     ;END TRACK PARAMETERS
170 011414 010037 005666      MOV    R0,TABLE+16   ;SETUP TRACK PARAMETER LIMITS IN "TABLE"
    011420 010037 005674      MOV    R0,TABLE+24   ;SETUP TRACK PARAMETER LIMITS IN "TABLE"
173 011424 010037 005710      MOV    R0,TABLE2+6   ;SETUP TRACK PARAMETER LIMITS IN "TABLE2"
    011430 010037 005726      MOV    R0,TABLE3+6   ;SETUP TRACK PARAMETER LIMITS IN "TABLE3"
    011434 010037 005744      MOV    R0,TABLE4+6   ;SETUP TRACK PARAMETER LIMITS IN "TABLE4"
174 011440 104401 001205      MOV    R0,TABLE4+6   ;SETUP TRACK PARAMETER LIMITS IN "TABLE4"
175                                TYPE    ,%CRLF        ;CR-LF
176                                ;GET ADDRESS LIMITS
177
178 011444 005037 001376      ASK3: CLR    WRAP        ;ASSUME SEEK FROM LO TO HI CYLINDERS
179 011450 104401 042362      TYPE    ,MSPRM       ;TYPE 'CHANGE DRIVE PARAMETERS ?'
180 011454 104411                                RDLIN                                ;READ THE ENTRY
181 011456 012600      MOV    (SP)+,R0      ;SAVE ADDRESS OF RESPONSE
182 011460 105710      TSTB   (R0)          ;WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
183 011462 001441      BEQ    ST2           ;BR IF YES
184 011464 105760 000001      TSTB   1(R0)        ;WAS IT TERMINATED WITH CARRIAGE RETURN ?
185 011470 001006      BNE    1$           ;BR IF NO
186 011472 122710 000131      CMPB   #'Y',(R0)     ;WAS IT A 'Y' RESPONSE ?
187 011476 001406      BEQ    2$           ;BR IF YES
188 011500 122710 000116      CMPB   #'N',(R0)     ;WAS IT A 'N' RESPONSE ?
189 011504 001430      BEQ    ST2           ;BR IF YES
190 011506 104401 037174      1$:  TYPE    ,BADENT   ;TYPE BAD ENTRY MESSAGE
191 011512 000754      BR     ASK3         ;TRY AGAIN
192 011514
193
194 011514 104401 036624      ASK3A: TYPE    ,ENTADR  ;TYPE 'ENTER ADDRESS LIMITS:'
195 011520 004437 020032      JSR    R4,PARENT    ;ENTER THE ADDRESS LIMITS
196 011524 005650      TABLE                                ;TABLE PARAMETERS
197 011526 023737 001326 001330  CMP    MAXCYL,MINCYL ;SEE IF ENDING CYL EQ TO GT THAN STARTING
198 011534 001402      BEQ    1$           ;BR IF ON THE SAME CYLINDER
199 011536 103410      BLO    3$           ;BR IF LESS
200 011540 000412      BR     4$           ;TO CHECK DRIVE
201
202 011542 023737 001332 001334 1$:  CMP    MAXTRK,MINTRK ;SEE IF ENDING TRACK EQ OR GT THAN STARTING
203 011550 103006      BHIS   4$           ;BR IF YES
204 011552 104401 037215      2$:  TYPE    ,MADRER     ;INVALID ADDRESS ENTERED
205 011556 000756      BR     ASK3A        ;TRY AGAIN
206
207 011560 012737 177777 001376 3$:  MOV    #-1,WRAP     ;ALLOW SEEK FROM HI TO LO CYLINDER ADDRESS
208 011566      4$:
    
```



```

1          .SBTTL TYPE THE DRIVE NUMBER
2
3          ;TYPE THE DRIVE TO BE FORMATTED AND START
4
5 011566 012737 017642 000060 ST2:  MOV   #0,ENTR,@#TKVEC ;VECTOR FOR CONTROL-0
6 011574 005037 177776          CLR   PS           ;PRIORITY = 0
7 011600 104401 001205          TYPE  .$CRLF       ;CR-LF
8 011604 005737 001320          TST   MODE         ;IS IT 'FORMAT' MODE ?
9 011610 001403                BEQ   1$           ;BR IF YES
10 011612 104401 037062         TYPE  ,MVERIFY     ;TYPE 'VERIFY HEADERS'
11 011616 000402                BR    2$           ;
12 011620 104401 037033         1$:  TYPE  ,MFORMT   ;TYPE 'FORMAT & VERIFY'
13 011624 005737 001356         2$:  TST   SEC30    ;IS IT 16 BIT MODE ?
14 011630 100403                BMI   3$           ;BR IF YES
15 011632 104401 037141         TYPE  ,M18BIT     ;TYPE 'OPERATE IN 18 BIT MODE'
16 011636 000402                BR    4$           ;
17 011640 104401 037106         3$:  TYPE  ,M16BIT   ;TYPE 'OPERATE IN 16 BIT MODE'
18 011644                4$:
19
20          ;THIS CODE RETRIEVES THE BAD SECTOR FILE FROM THE PACK (IF FORMATTED PACK)
21          ;SET UP DPB BLOCK READ CYL 822., LAST TRACK.
22
23 011644 005037 001342          RETBAD: CLR   RETRY      ;CLEAR THE RETRY FLAG
24 011650 012737 177777 001404  MOV   #-1,BSFAVL    ;INDICATE BAD SECTOR FILE IS NOT AVAILABLE
25 011656 013737 001222 005540  MOV   DRIVE,FMTDPB ;LOAD DRIVE #
26 011664 012737 001466 005552  MOV   #822.,FMTDPB+12 ;CYLINDER 822.
27 011672 113737 001324 005551  MOV   LSTRK,FMTDPB+11 ;LAST TRACK
28 011700 105037 005550          CLR   FMTDPB+10    ;SECTOR 0
29 011704 012737 046234 005546  MOV   #BUFP,FMTDPB+6 ;BUFFER ADDRESS
30 011712 012737 157700 005544  MOV   #-<32.*258.>,FMTDPB+4 ;WORD COUNT FULL TRACK INCLUDING HEAD
31 011720 112737 000020 005541  MOV   #20,FMTDPB+1  ;FMT SET
32 011726 112737 000143 005542  MOV   #SETFMT,FMTDPB+2 ;SET 16 BIT MODE
33 011734 004037 031214          1$:  JSR   R0,RM05     ;SET THE FORMAT BIT
34 011740 005540                FMTDPB
35 011742 000774                BR    1$
36 011744 005737 005556          2$:  TST   FMTDPB+16  ;THE COMMAND IS DONE ?
37 011750 001775                BEQ   2$           ;BR IF NOT
38
39 011752 012737 011752 001124  MOV   #,$LPERR     ;LOOP ON ERROR ADDRESS
40 011760 012706 001100          MOV   #STACK,SP    ;RESET STACK POINT
41 011764 112737 000173 005542  MOV   #RDHD,FMTDPB+2 ;RELOAD THE READ HEAD AND DATA COMMAND
42 011772 004037 031214          3$:  JSR   R0,RM05     ;READ THE LAST TRACK
43 011776 005540                FMTDPB
44 012000 000774                BR    3$           ;BR IF QUEUE FAIL
45 012002 005737 005556          4$:  TST   FMTDPB+16  ;ALL DONE
46 012006 001775                BEQ   4$           ;BR IF NOT
47 012010 100007                BPL   6$           ;BR IF NO ERROR
60 012012 022737 000003 001342  CMP   #3,RETRY     ;EXCEEDED RETRY LIMIT ?
61 012020 103452                BLO   9$           ;BR IF YES
62 012022 005237 001342          INC   RETRY        ;INCREMENT RETRY COUNT
63 012026 000761                BR    3$           ;TRY AGAIN
64
65 012030 005037 001176          6$:  CLR   $ESCAPE
66 012034 005037 001342          CLR   RETRY
67 012040 005737 005556          TST   FMTDPB+16  ;ANY ERROR ?
68 012044 100440                BMI   9$           ;BR IF YES
69 012046 022737 151466 046234  CMP   #151466,BUFP ;ON THE LAST CYLINDER ?

```



```

70 012054 001034          BNE      9$          :BR IF NOT
71 012056 013700 001324   MOV     LSTRK,R0      :GET LAST TRACK
72 012062 000300          SWAB    R0           :MOVE TRACK TO HI BYTE AND SECTOR TO LO BYTE
73 012064 020037 046236   CMP     R0,BUFP+2    :ON LAST TRACK AND SECTOR 0
74 012070 001026          BNE     9$          :BR IF NOT
75 012072 005737 046240   TST    BUFP+4        :SERIAL NUMBER SHOULD NOT 0
76 012076 001003          BNE     7$          :BR IF NOT 0
77 012100 005737 046242   TST    BUFP+6        :SECOND WORD OF SERIAL NUMBER
78 012104 001420          BEQ     9$          :BR IF ZERO (CORRUPT)
79 012106 005737 046244   TST    BUFP+10       :THE THIRD WORD MUST BE 0
80 012112 001015          BNE     9$          :BR IF NOT (CORRUPT)
81 012114 022737 177777 046246  CMP     #-1,BUFP+12  :AN ALIGNMENT PACK ?
82 012122 001002          BNE     8$          :BR IF NOT
83 012124 000137 016430   JMP     REJCT1       :REJECT FOR BEING AN ALIGNMENT PACK
84 012130 005737 046246   TST    BUFP+12       :IS DISK A DATA PACK ?
85 012134 001004          BNE     9$          :BR IF NO (CORRUPT)
86 012136 012737 000400 001404  MOV     #400,BSFAVL  :INDICATOR BAD SECTOR FILE IS AVAILABLE
87 012144 000400          BR      10$         :FILE OK..
88 012146          9$:
89 012146          10$:
90
91          :THIS CODE INITIALIZES THE MFG16, MFG18, USTAB AND USSAV TABLES
92
93 012146 012706 001100   TABINT: MOV     #STACK,SP      :INITIAL STACK POINT
94 012152 005046          CLR     -(SP)         :TERMINATOR
95 012154 012746 004430   MOV     #USSAV,-(SP)   :USSAV TABLE ADDRESS
96 012160 012746 002420   MOV     #MFG18,-(SP)  :MFG 18 BIT FILE ADDRESS
97 012164 012746 001414   MOV     #MFG16,-(SP)  :MFG 16 BIT FILE ADDRESS
98 012170 012703 003424   MOV     #USTAB,R3     :USER BAD SECTOR FILE ADDRESS
99 012174 012704 000004   MOV     #4,R4         :SET 2 SERIAL NUMBER WORDS AND NEXT 2 WORDS
100 012200 005023 2$:    CLR     (R3)+         :TO ALL 0'S
101 012202 005304          DEC     R4           :DONE YET ?
102 012204 001375          BNE     2$          :BR IF NO
103
104 012206 012704 000374   MOV     #252,R4       :GET NUMBER OF ENTRIES LEFT TO FILL
105 012212 012723 177777 3$:    MOV     #-1,(R3)+     :SET REST OF LOCATIONS TO -1
106 012216 005304          DEC     R4           :DONE YET ?
107 012220 001374          BNE     3$          :BR IF NO
108 012222 012603          MOV     (SP)+,R3     :GET NEXT TABLE ON STACK
109 012224 001363          BNE     1$          :BR IF NOT TERMINATOR
110
111          :THIS CODE LOADS TABLE MFG16, MFG18, USSAV FROM PACK BAD SECTOR FILE
112          :INFORMATION (IF BAD SECTOR FILE IS AVAILABLE)
113
114 012226 005737 001404   TABLD: TST     BSFAVL     :IS BAD SECTOR FILE AVAILABLE ?
115 012232 003451          BLE     6$          :BR IF NO
116
117          :POINT TO FIRST SERIAL NUMBER OF 1ST
118 012234 012702 046240   MOV     #BUFP+4,R2    :16 BIT 'MFG' BLOCK OF BAD SECTOR FILE
119 012240 012703 001414   MOV     #MFG16,R3     :GET ADDRESS OF MFG16 TABLE
120 012244 012704 000400   MOV     #256,R4       :WORD COUNT
121 012250 012223 1$:    MOV     (R2)+,(R3)+  :LOAD THE MFG16 TABLE
122 012252 005304          DEC     R4           :DONE YET ?
123 012254 001375          BNE     1$          :BR IF NO
124
125          :POINT TO FIRST SERIAL NUMBER OF 1ST
126 012256 012702 047244   MOV     #BUFP+4+516,R2 :18 BIT 'MFG' BLOCK OF BAD SECTOR FILE

```


127	012262	012703	002420		MOV	#MFG18,R3	:GET ADDRESS OF MFG18 TABLE
128	012266	012704	000400		MOV	#256.,R4	:WORD COUNT
129	012272	012223		2\$:	MOV	(R2)+,(R3)+	:LOAD THE MFG18 TABLE
130	012274	005304			DEC	R4	:DONE YET ?
131	012276	001375			BNE	2\$:BR IF NO
132							
133							:POINT TO FIRST SERIAL NUMBER OF 1ST
134							:18 BIT 'USR' BLOCK OF BAD SECTOR FILE IN
135	012300	012702	061314		MOV	#BUFF+4+<11.*516.>,R2	:SECTOR 11.
136	012304	005737	001356		TST	SEC30	:FORMAT IN 16 BIT MODE
137	012310	100402			BMI	3\$:BR IF IT IS
138							
139							:POINT TO FIRST SERIAL NUMBER OF 1ST
140							:16 BIT 'USR' BLOCK OF BAD SECTOR FILE IN
141	012312	012702	060310		MOV	#BUFF+4+<10.*516.>,R2	:SECTOR 10.
142	012316	012703	004430	3\$:	MOV	#USSAV,R3	:GET ADDRESS OF USSAV TABLE
143	012322	012704	000400		MOV	#256.,R4	:WORD COUNT
144	012326	012223		4\$:	MOV	(R2)+,(R3)+	:LOAD THE USSAV TABLE
145	012330	005304			DEC	R4	:DONE YET ?
146	012332	001375			BNE	4\$:BR IF NO
147							
148	012334	062702	000004		ADD	#4,R2	:ACCESSES THE USTAB BUFFER ADD
149	012340	012703	003424		MOV	#USTAB,R3	:GET ADDRESS OF USTAB TABLE
150	012344	012704	000400		MOV	#256.,R4	:WORD COUNT
151	012350	012223		5\$:	MOV	(R2)+,(R3)+	:LOAD THE USTAB TABLE
152	012352	005304			DEC	R4	:DONE YET ?
153	012354	001375			BNE	5\$:BR IF NO
154	012356			6\$:			
155							

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

012356	005737	001404	
012362	003123		
012364	105737	001150	
012370	001116		
012372	004737	024414	
012376	104401	037533	
012402	012702	000012	
012406	005037	001414	
012412	005037	001416	
012416	104411		
012420	012601		
012422	105711		
012424	001471		
012426	121127	000060	
012432	103522		
012434	121127	000067	
012440	101117		
012442	013746	001414	
012446	042737	100000	001414
012454	006137	001414	
012460	006137	001414	
012464	006137	001414	
012470	006137	001414	
012474	006137	001414	
012500	042737	177770	001414
012506	006337	001416	
012512	100472		
012514	006337	001416	
012520	100467		
012522	006337	001416	
012526	100464		
012530	063737	001414	001416
012536	012637	001414	
012542	006337	001414	
012546	006337	001414	
012552	006337	001414	
012556	042737	100000	001414
012564	111103		
012566	042703	177770	
012572	060337	001414	
012576	005201		
012600	005302		
012602	001307		
012604	105711		
012606	001034		
012610	005737	001414	
012614	001006		
012616	005737	001416	
012622	001426		
012624	000402		
012626	005237	001414	

.SBTTL ENTER OCTAL SERIAL NUMBER

:THIS CODE ASK OPERATOR TO ENTER A SERIAL NUMBER FOR UNFORMATTED OR
 :INITIALIZED PACK

SERNL:	TST	B\$FAVL	:IS BAD SECTOR FILE AVAILABLE ?
	BGT	5\$:BR IF YES
	TSTB	\$AUTOB	:RUNNING UNDER AUTO MODE ?
	BNE	4\$:BR IF YES
1\$:	JSR	PC,\$TKINT	:INITIALIZE THE KEYBOARD
	TYPE	,MSG1	:TYPE 'ENTER SERIAL NUMBER (OCTAL): '
	MOV	#10.,R2	:MAXMUM 10 OCTAL DIGITS ALLOWED
	CLR	MFG16	:CLEAR THE LSW OF THE SERIAL NUMBER
	CLR	MFG16+2	:CLEAR THE MSW OF THE SERIAL NUMBER
	RDLIN		:READ IN THE STRING
	MOV	(SP)+,R1	:ADDRESS OF THE READ IN STRING
2\$:	TSTB	(R1)	:TERMINATOR OF THE INPUT STRING
	BEQ	3\$:BR IF IT IS
	CMPB	(R1),#0	:LESS THAN ASCIZ 0 ?
	BLO	6\$:ENTER AGAIN IF IT IS
	CMPB	(R1),#7	:HIGH THAN ASCIZ 7 ?
	BHI	6\$:ENTER AGAIN IF SO
	MOV	MFG16,-(SP)	:SAVE THE LSW
	BIC	#BIT15,MFG16	:CLEAR THE SIGN BIT OF LSW
	ROL	MFG16	:ROTATE THE LSW FIVE TIMES
	ROL	MFG16	:TO MOVE THE MOST SIGN DIGIT
	ROL	MFG16	:TO BIT 0 TO BIT 2
	ROL	MFG16	
	ROL	MFG16	
	BIC	#177770,MFG16	:LEFT ON THE MS DIGIT OF THE LSW
	ASL	MFG16+2	:SHIFT THE MSW LEFT ONE OCTAL DIGIT
	BMI	6\$:ENTER AGAIN IF SIGN BIT SET
	ASL	MFG16+2	
	BMI	6\$	
	ASL	MFG16+2	
	BMI	6\$	
	ADD	MFG16,MFG16+2	:APPENDING THE DIGITS INTO MSW
	MOV	(SP)+,MFG16	:RESTORE THE LSW
	ASL	MFG16	:SHIFT THE LSW LEFT ONE OCTAL DIGIT
	ASL	MFG16	
	ASL	MFG16	
	BIC	#BIT15,MFG16	:CLEAR THE SIGN BIT
	MOVB	(R1),R3	:LOAD THE CURRENT READ IN DIGITS
	BIC	#177770,R3	:LEFT ONLY ONE OCTAL DIGIT
	ADD	R3,MFG16	:APPEND THE READ IN CHARACTER TO LSW
	INC	R1	:ADJUST THE READ IN STRING ADDRESS
	DEC	R2	:DECREMENT ONE DIGIT COUNT
	BNE	2\$:BR IF NOT DONE
	TSTB	(R1)	:IF 10 DIGITS HAVE BEEN ENTERED
	BNE	6\$:BR IF NOT <CR> TERMINATOR
3\$:	TST	MFG16	:FIRST SERIAL NUMBER ZERO ?
	BNE	5\$:BR IF NO
	TST	MFG16+2	:SECOND SERIAL NUMBER ZERO ?
	BEQ	6\$:BR IF YES
	BR	5\$	
4\$:	INC	MFG16	:ASSUME S/N 1, IF THE SERIAL NUMBER NEEDS TO


```

1          .SBTTL  UTILITY ROUTINE
2
3          ;THIS CODING TO PROVIDE UTILITY ROUTINE ENTRIES
4
5 012706  004737  024414  CALIN:  JSR      PC,$TKINT      ;INITIALIZE THE KEYBOARD
6 012712  105737  001150      TSTB     $AUTOB      ;AUTO BYTE SET ?
7 012716  001057              BNE      5$          ;IF SO, BRANCH
8 012720  005737  001270      TST      $CDW1       ;ALLOW ANY UTILITY ROUTINE ?
9 012724  001454              BEQ      5$          ;BR IF NOT
10
11 012726  005227  177777      1$:    INC      #-1      ;FIRST TIME THRU ?
12 012732  001002              BNE      3$          ;BR IF NO
13 012734  104401  037740      2$:    TYPE     ,MSGBLK   ;ENTER MESSAGE BLOCK
14 012740  104401  040521      3$:    TYPE     ,MUTLTY   ;TYPE UTILITY MESSAGE
15 012744  104410              RDCHR                    ;READ IN THE ROUTINE NUMBER
16 012746  012637  001174      MOV     (SP)+,$TMPO    ;ADDRESS OF INPUT BUFFER
17 012752  022737  000060  001174  CMP     #'0,$TMPO      ;INPUT NUMBER LESS THAN 0 ?
18 012760  101022              BHI      4$          ;BR IF SO
19 012762  122737  000064  001174  CMPB   #'4,$TMPO      ;INPUT NUMBER LARGER THAN 4 ?
20 012770  103416              BLO      4$          ;BR IF SO
21 012772  104401  001174      TYPE   ,$TMPO         ;TYPE CHARACTER
22 012776  104401  001205      TYPE   ,$CRLF         ;CR-LF
23 013002  013701  001174      MOV     $TMPO,R1      ;GET NUMBER
24 013006  042701  177760      BIC     #177760,R1    ;LEFT OF THREE BITS
25 013012  006301              ASL     R1             ;WORD INDEX
26 013014  004771  006052      JSR     PC,@TABCAL(R1);CALL THE DESIRED UTILITY ROUTINE
27 013020  104401  001205      TYPE   ,$CRLF         ;CR-LF
28 013024  000745              BR      3$           ;LOOP BACK TO THE SAME POINT
29
30 013026  104401  001205      4$:    TYPE   ,$CRLF   ;CR-LF
31 013032  022737  000065  001174  CMP     #'5,$TMPO     ;TYPE WHOLE MESSAGE AGAIN ?
32 013040  001735              BEQ     2$           ;BR IF YES
33 013042  005037  001362      CLR     DS,CYL        ;CLEAR ALL PARAMETERS
34 013046  005037  001364      CLR     DS,TRK
35 013052  005037  001344      CLR     BADSEC
36 013056

```



```

1          .SBTTL  SETUP WRITE BUFFER
2
3          ;THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH
4          ;SECTOR IMAGE. ALSO, RETURN HERE WHEN LOOPING ON DRIVE BEING TESTED
5          ;WITH SW0=1 (SET).
6
7 013056 104401 036573      ST3:  TYPE  ,MUNIT      ;TYPE 'DRIVE'
8 013062 013746 001222      MOV   DRIVE,-(SP)  ;:SAVE DRIVE FOR TYPEOUT
                               ;:TYPE DRIVE NUMBER
                               ;:GO TYPE--OCTAL ASCII
                               ;:TYPE 2 DIGIT(S)
                               ;:SUPPRESS LEADING ZEROS
                               ;TYPE ' '
                               ;TYPE 1 BLANK
                               ;TYPE PACK S/N IN OCTAL
                               ;CR-LF
                               ;VECTOR FOR CONTROL-O
                               ;RESET STACK POINTER
                               ;PRIORITY = 0
                               ;DON'T ALLOW UPDATE OF 'BSF' AT END OF PASS
                               ;'FORMAT' OR 'VERIFY HEADER' MODE ?
                               ;BR IF 'VERIFY HEADER ONLY' MODE
                               ;ALLOW UPDATE OF 'BSF' AT END OF PASS
                               ;TYPE 'STARTING FORMAT'
                               ;BEGIN FORMATTING
                               ;BEGIN VERIFYING HEADERS ONLY
                               1$:  JMP   VFYHDR
                               2$:  MOV   WC,R3      ;SET UP COUNTER
                               3$:  MOV   #BUFP,R0    ;SET UP MEMORY POINTER
                               4$:  MOV   PATA,R1     ;SET UP PATTERN IN R1
                               MOV   PATB,R2     ;SET UP PATTERN IN R2
                               MOV   R1,(R0)+    ;MOV 1ST PAT INTO MEM
                               MOV   R2,(R0)+    ;MOV 2ND PAT INTO MEM
                               DEC   R3         ;KEEP COUNT
                               DEC   R3         ;KEEP COUNT
                               BNE   4$        ;DO IT AGAIN IF R3 NOT 0
                               9 013066 104403      TYPOS
                               013070 002        .BYTE 2
                               013071 000        .BYTE 0
                               10 013072 104401 036602  TYPE ,SCOMMA
                               11 013076 104401 036622  TYPE ,BLNKS1
                               12 013102 004737 022622  JSR  PC,TYPACK
                               13 013106 104401 001205  TYPE ,SCRLF
                               14 013112 012737 017642 000060 MOV  #0,ENTR,@#TKVEC
                               15 013120 012706 001100  MOV  #STACK,SP
                               16 013124 005037 177776  CLR  PS
                               17 013130 005037 001406  CLR  UPDBSF
                               18 013134 005737 001320  TST  MODE
                               19 013140 001006        BNE  1$
                               20 013142 012737 177777 001406 MOV  #-1,UPDBSF
                               21 013150 104401 037315  TYPE ,MSFOR
                               22 013154 000402        BR   2$
                               23 013156 000137 015110  1$:  JMP   VFYHDR
                               24 013162 013703 001350  2$:  MOV   WC,R3      ;SET UP COUNTER
                               25 013166 012700 046234  3$:  MOV   #BUFP,R0    ;SET UP MEMORY POINTER
                               26 013172 013701 001336  4$:  MOV   PATA,R1     ;SET UP PATTERN IN R1
                               27 013176 013702 001340  MOV   PATB,R2     ;SET UP PATTERN IN R2
                               28 013202 010120  MOV   R1,(R0)+    ;MOV 1ST PAT INTO MEM
                               29 013204 010220  MOV   R2,(R0)+    ;MOV 2ND PAT INTO MEM
                               30 013206 005303  DEC   R3         ;KEEP COUNT
                               31 013210 005303  DEC   R3         ;KEEP COUNT
                               32 013212 001373  BNE   4$        ;DO IT AGAIN IF R3 NOT 0

```

```
1 .SBTTL WRITE HEADERS
2
3 ;SET UP TO WRITE HEADERS AND DATA FOR FORMATTING
4
5 013214 004737 016714 WRTRK: JSR PC,SETDPB ;SET DRIVE PARAMETER TABLE
6 013220 004737 017120 JSR PC,SETHDR ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
7
8 013224 112737 000020 005541 MOVB #20,FMTDPB+1 ;LOAD FMT16 BIT
9 013232 005737 001356 TST SEC30 ;18 BIT MODE ?
10 013236 001002 BNE 1$ ;BR IF NOT
11 013240 105037 005541 CLRFB FMTDPB+1 ;CLEAR THE FMT16 BIT
12 013244 112737 000143 005542 1$: MOVB #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
13 013252 004037 031214 2$: JSR R0,RM05 ;START THE COMMAND
14 013256 005540 FMTDPB ;DPB ADDRESS
15 013260 000774 BR 2$ ;QUEUE FULL RETURN
16 013262 005737 005556 3$: TST FMTDPB+16 ;LOOK FOR DONE
17 013266 001775 BEQ 3$ ;LOOP UNTIL FINISHED
18
19 ;SET UP 'SECCU' TABLE WITH BAD SECTORS ON CURRENT TRACK
20
21 013270 023727 005552 001466 WRTRKX: CMP FMTDPB+12,#822. ;LAST CYLINDER ?
22 013276 103406 BLO 1$ ;BR IF NO
23 013300 123737 005551 001324 CMPB FMTDPB+11,LSTRK ;LAST TRACK OF LAST CYLINDER ?
24 013306 103402 BLO 1$ ;BR IF NO
25 013310 000137 015076 JMP WRTRKZ ;DON'T DESTROY LAST TRACK ON CYL 822.
26
27 013314 012703 005434 1$: MOV #SECCU,R3 ;INITILIZE THE 'SECCU' TABLE
28 013320 012704 000040 MOV #32.,R4 ;TOTAL OF 32. ENTRIES
29 013324 012723 177777 2$: MOV #-1,(R3)+ ;LOAD ALL LOCATIONS TO -1
30 013330 005304 DEC R4 ;DONE YET ?
31 013332 001374 BNE 2$ ;BR IF NO
32
33 ;LOAD THE 'MFG' AND 'USR' BAD SECTOR FILES INTO THE 'SECCU' TABLE FOR
34 ;THE CURRENT TRACK TO BE FORMATTED
35
36 013334 005737 001404 TST BSFAVL ;IS BAD SECTOR FILE AVAILABLE ?
37 013340 003454 BLE 9$ ;BR IF NO
38 013342 005046 CLR -(SP) ;CLEAR UP THE DUMMY RETURN ADDRESS
39 013344 012746 003434 MOV #USTAB+8.,-(SP) ;ADDRESS OF 'USR' BAD SECTOR TABLE
40 013350 012702 001424 MOV #MFG16+8.,R2 ;SETUP ADDRESS OF 'MFG16' BAD SECTOR TABLE
41 013354 005737 001356 TST SEC30 ;IS IT 16. BIT MODE ?
42 013360 100402 BMI 3$ ;BR IF YES
43 013362 012702 002430 MOV #MFG18+8.,R2 ;SETUP ADDRESS OF 'MFG18' BAD SECTOR TABLE
44 013366 010201 3$: MOV R2,R1 ;STARTING ADDRESS OF BAD SECTOR TABLE
45 013370 012703 005434 MOV #SECCU,R3 ;STARTING ADDRESS OF 'SECCU' TABLE
46 013374 012704 005534 MOV #SECCU+64.,R4 ;ENDING ADDRESS OF TABLE (32. ENTRIES)
47 013400 062701 000770 4$: ADD #252.*2,R1 ;INDEX ADDRESS TO END OF TABLE (126. ENTRIES)
48 013404 023712 005552 5$: CMP FMTDPB+12,(R2) ;BAD SECTOR ON THIS CYLINDER ?
49 013410 001007 BNE 6$ ;BR IF NO
50 013412 123762 005551 000003 CMPB FMTDPB+11,3(R2) ;BAD SECTOR ON THE SAME TRACK ?
51 013420 001003 BNE 6$ ;BR IF NO
52 013422 116205 000002 MOVB 2(R2),R5 ;YES--SAME CYL & TRK SO SAVE BAD SECTOR ADRS
53 013426 010523 MOV R5,(R3)+ ;PUT BAD SECTOR IN 'SECCU' TABLE
54 013430 062702 000004 6$: ADD #4,R2 ;ADVANCE TO NEXT SET OF BAD SECTORS ENTRIES
55 013434 020201 CMP R2,R1 ;AT END OF BAD SECTOR TABLE ?
56 013436 002012 BGE 8$ ;BR IF YES
57 013440 020304 CMP R3,R4 ;AT END OF 'SECCU' TABLE ?
```



```

58 013442 002001          BGE      7$          :YES--SHOULD NOT BE THAT MANY ENTRIES FOR
59                          :ONE TRACK
60 013444 000757          BR       5$          :GET NEXT ENTRY
61
62 013446 004737 017522    7$:      JSR      PC,RESTR  :IMPROPER MFG INFORMATION
63 013452 012737 177777    001404  MOV      #-1,BSFAVL  :INDICATE BAD SECTOR FILE IS NOT AVAILABLE
64 013460 000137 012146    JMP      TABINT      :AND START AGAIN
65
66 013464 011601          8$:      MOV      (SP),R1    :GET 'USR' BAD SECTOR TABLE
67 013466 012602          MOV      (SP)+,R2    :STARTING ADDRESS OF 'USR' BAD SECTOR TABLE
68 013470 001343          BNE      4$          :BR IF NOT DUMMY (TERMINATOR)
69 013472 000240          9$:      NOP          :DONE
70
71                          :THIS CODE SORTS THE BAD SECTOR TABLE 'SECCU' IN ASCENDING ORDER
72                          :AND DELETES THE DUPLICATED ONES
73
74 013474 012702 005434    SORT1:  MOV      #SECCU,R2  :TOP OF THE TABLE
75 013500 012703 005532    MOV      #SECCU+62.,R3 :BOTTOM OF THE TABLE
76 013504 012704 000037    MOV      #31.,R4      :TOTAL 32 ELEMENT,SORT 31 TIMES
77 013510 022712 177777    CMP      #-1,(R2)     :IS THE TABLE EMPTY ?
78 013514 001451          BEQ      6$          :BR IF IT IS, QUICK EXIT
79 013516 021262 000002    1$:      CMP      (R2),2(R2) :N TH ELEMENT : N+1 TH ELEMENT
80 013522 103430          BLO      4$          :BR IF SMALLER
81 013524 001406          BEQ      2$          :DELET N+1 TH ELEMENT, IF SAME
82                          :OTHERWISE, N > N+1
83 013526 011246          MOV      (R2),-(SP)   :STORE N ELEMENT
84 013530 016212 000002    MOV      2(R2),(R2)   :SWITCH N+1 ELEMENT TO N ELEMENT
85 013534 012662 000002    MOV      (SP)+,2(R2) :SWITCH N ELEMENT TO N+1 ELEMENT
86 013540 000421          BR       4$          :ADJUST TOP POINTER
87 013542 022712 177777    2$:      CMP      #-1,(R2)   :TERMINATOR ?
88 013546 001416          BEQ      4$          :BR IF IT IS
89 013550 010246          MOV      R2,-(SP)    :SAVE TOP POINTER
90 013552 062702 000002    ADD      #2,R2       :DELET 2(R2)
91 013556 016222 000002    3$:      MOV      2(R2),(R2)+ :
92 013562 022702 005532    CMP      #SECCU+62.,R2 :BOTTOM OF THE TABLE ?
93 013566 101373          BHI      3$          :BR IF NOT
94 013570 012737 177777    005532  MOV      #-1,SECCU+62. :LOAD -1 TO TABLE BOTTOM
95 013576 012602          MOV      (SP)+,R2    :RESTORE THE TOP POINTER
96 013600 005304          DEC      R4          :DECREMENT ONE ELEMENT COUNT
97 013602 001416          BEQ      6$          :BR IF ALL DONE
98 013604 062702 000002    4$:      ADD      #2,R2     :INCREMENT THE TOP POINTER
99 013610 020302          CMP      R3,R2      :REACH THE BOTTOM ?
100 013612 001341          BNE      1$         :BR IF NOT
101 013614 005304          5$:      DEC      R4        :DECREMENT ONE SORT COUNT
102 013616 001410          BEQ      6$         :BR IF ALL DONE
103 013620 012702 005434    MOV      #SECCU,R2   :RESET TOP POINTER
104 013624 162703 000002    SUB      #2,R3       :UPDATE BOTTOM POINTER
105 013630 022703 005434    CMP      #SECCU,R3   :EXIT IF TOP = BOTTOM
106 013634 001401          BEQ      6$         :
107 013636 000727          BR       1$         :
108 013640 000240          6$:      NOP          :SORT THE REST ELEMENTS
109 013642 005037 005536    CLR      NEXT        :DONE
110                          :FLAG,START OF A NEW TRACK
111 013646 005037 001316    WRTRK1: CLR      SOFSW   :CLEAR ERROR COUNTER
112 013652 005037 001342    CLR      RETRY      :ZERO THE RETRY COUNTER
113 013656 105037 005550    CLRB    FMTDPB+10  :RESTORE SECTOR
114 013662 013737 001352    005544  MOV      MWC,FMTDPB+4 :RESTORE WC

```

```

115 013670 012737 046234 005546      MOV      #BUFF,FMTDPB+6 ;RESTORE BA
116
117      ;THIS CODE TO SET UP WORD COUNT AND BUFFER ADDRESS
118
119 013676 113737 005536 005550 WRTRKA: MOVB   NEXT,FMTDPB+10 ;STARTING SECTOR
120 013704 012701 005434           MOV      #SECCU,R1      ;R1 POINTS TO THE BAD SECTOR TABLE
121 013710 022711 177777           1$:     CMP      #-1,(R1) ;END OF 'SECCU' TABLE ?
122 013714 001512           BEQ      WRTRKD        ;BR IF YES
123 013716 032711 100000           BIT      #BIT15,(R1)   ;THE BAD SECTOR HAS ALREADY ACCESSED ?
124 013722 001403           BEQ      2$           ;BR IF NOT
125 013724 062701 000002           ADD      #2,R1        ;ADJUST THE POINTER
126 013730 000767           BR       1$           ;LOOPING BACK
127 013732 123711 005550           2$:     CMPB   FMTDPB+10,(R1) ;BAD SECTOR = STARTING SECTOR ?
128 013736 001416           BEQ      WRTRKB        ;BR IF IT IS
129 013740 103402           BLO     3$           ;BR IF START SEC < BAD SECTOR
130 013742 000137 013270           JMP      WRTRKX        ;SOMETHING IS WRONG, TRY SORTING AGAIN ON
131                                     ;THE SAME TRACK
132 013746 111102           3$:     MOVB   (R1),R2    ;UPDATE NEXT STARTING ADDRESS
133 013750 023702 001360           CMP      MAXSEC,R2    ;NEXT BAD SECTOR IS THE LAST SECTOR?
134 013754 001463           BEQ      WRTRKF        ;BR IF LAST SECTOR
135 013756 103471           BLO     WRTRKD        ;BR IF LESS
136 013760 005202           INC      R2           ;NEXT = BAD + 1
137 013762 010237 005536           MOV      R2,NEXT     ;NEXT STARTING ADDRESS
138 013766 052721 100000           BIS      #BIT15,(R1)+ ;MARK THE ACCESSED SECTOR
139 013772 000432           BR       WRTRKC       ;TO SET UP WORD COUNT
140
141 013774 105237 005550 WRTRKB: INCB   FMTDPB+10 ;INCREMENT THE STARTING SECTOR
142 014000 052721 100000           BIS      #BIT15,(R1)+ ;MARK THE ACCESSED SECTOR
143 014004 123711 005550           CMPB   FMTDPB+10,(R1) ;STILL ON ANOTHER BAD SECTOR ?
144 014010 001771           BEQ      WRTRKB        ;LOOP AGAIN
145 014012 123737 005550 001360           CMPB   FMTDPB+10,MAXSEC ;IF START SECTOR > MAXSEC
146 014020 101402           BLOS   1$           ;IMPROPER MFG INFORMATION
147 014022 000137 015010           JMP      HDRSET       ;EXIT
148
149 014026 022711 177777           1$:     CMP      #-1,(R1) ;TO SEE IF NEXT BAD SECTOR IN TABLE
150 014032 001443           BEQ      WRTRKD        ;BR IF NONE
151 014034 111102           MOVB   (R1),R2        ;UPDATE THE NEXT
152 014036 023702 001360           CMP      MAXSEC,R2    ;BAD SECTOR IS THE LAST SECTOR ?
153 014042 001430           BEQ      WRTRKF        ;BR IF LESS
154 014044 103436           BLO     WRTRKD        ;BAD SEC +1
155 014046 005202           INC      R2           ;NEXT STARTING ADDRESS
156 014050 010237 005536           MOV      R2,NEXT     ;NEXT STARTING ADDRESS
157 014054 052711 100000           BIS      #BIT15,(R1) ;MARK THE ACCESSED SECTOR
158
159 014060 013702 005536 WRTRKC: MOV      NEXT,R2 ;CALCULATE THE WORD COUNT
160 014064 162702 000002           SUB      #2,R2        ;ENDSECTOR = NEXT - 2
161 014070 006302           ASL     R2           ;WORD INDEX
162 014072 016203 006164           MOV      WCTBL(R2),R3 ;WORD CTR FOR SECTOR 0 TO ENDING SECTOR
163 014076 113702 005550           MOVB   FMTDPB+10,R2  ;STARTING SECTOR
164 014102 006302           ASL     R2           ;WORD INDEX
165 014104 166203 006164           SUB      WCTBL(R2),R3 ;WORD COUNT FOR SECTOR 0 TO STARTING SECTOR
166 014110 062703 000402           ADD      #258.,R3    ;ADJUST ONE SECTOR
167 014114 005403           NEG     R3           ;GET THE NEAGTIVE WORD COUNT
168 014116 010337 005544           MOV      R3,FMTDPB+4 ;LOAD THE WORD CTR INTO DPB
169 014122 000432           BR       WRTRKE       ;LOAD THE STARTING ADDRESS
170
171 014124 012737 177777 005536 WRTRKF: MOV      #-1,NEXT ;ALL DONE

```



```
172 014132 052711 100000          BIS    #BIT15,(R1)    ;MASK THE SECTOR
173 014136 005302                DEC    R2             ;PATCH FOR THE LAST SECTOR
174 014140 000405                BR     WRTRKH        ;PATCH
175
176 014142 012737 177777 005536 WRTRKD: MOV    #-1,NEXT    ;NO MORE BAD SECTOR
177 014150 013702 001360                MOV    MAXSEC,R2    ;LAST SECTOR
178
179 014154 006302                WRTRKH: ASL   R2     ;WORD INDEX
180 014156 016203 006164                MOV    WCTBL(R2),R3 ;WORD CTR FOR THE ENDING SECTOR
181 014162 113702 005550                MOV    FMTDPB+10,R2 ;STARTING ADDRESS
182 014166 006302                ASL   R2             ;WORD INDEX
183 014170 166203 006164                SUB    WCTBL(R2),R3 ;WORD CTR FOR THE STARTING SECTOR
184 014174 062703 000402                ADD    #258.,R3     ;ADJUST ONE SECTOR
185 014200 005403                NEG    R3            ;NEGATIVE WORD COUNT
186 014202 010337 005544                MOV    R3,FMTDPB+4  ;LOAD THE WORD COUNT INTO DPB
187 014206 000240                NOP                    ;DONE
188
189 014210 113702 005550                WRTRKE: MOV    FMTDPB+10,R2 ;STARTING ADDRESS
190 014214 006302                ASL   R2             ;LOCATE THE BUFFER ADDRESS
191 014216 016237 006064 005546                MOV    ADRTBL(R2),FMTDPB+6 ;BUFFER ADDRESS
192
193 014224 112737 000163 005542 WRTRK2: MOV    #WRTHD,FMTDPB+2 ;SET WRITE HEADER & DATA COMMAND IN TBL
194 014232 012737 014232 001124                MOV    #,$LPERR     ;SET UP LOOP ON ERROR ADDRESS
195 014240 012706 001100                MOV    #STACK,SP    ;LOAD STACK POINT
196 014244 004037 031214                1$:   JSR    R0,RM05  ;GO FORMAT A TRACK
197 014250 005540                FMTDPB                ;ADRS OF PARAMETERS - TBL
198 014252 000774                BR     1$            ;WAIT FOR QUEUE IF FULL
199 014254 005737 005556                2$:   TST    FMTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
200 014260 001775                BEQ   2$            ;BR IF NOT DONE
201 014262 100024                BPL   4$            ;BR IF NO ERROR
202 014264 012737 014312 001176                MOV    #3$, $ESCAPE ;ESCAPE TO 3$ ON ERROR
203 014272 004737 016550                JSR    PC,ERINDX    ;SEE WHICH ERROR
204 014276 104010                EMT   10            ;DRIVE OFFLINE
205 014300 104011                EMT   11            ;PERSISTENT DRIVE UNSAFE ERROR
206 014302 104012                EMT   12            ;UNCORRECTABLE MASSBUS PARITY ERROR
207 014304 104013                EMT   13            ;SOFTWARE TIMEOUT
208 014306 104014                EMT   14            ;DRIVE UNSAFE ERROR
209 014310 104015                EMT   15            ;DRIVE/CONTROLLER ERROR DURING WRITE
210
211 014312 004737 016646                3$:   JSR    PC,LOP.CK ;LOOP ON THE ERROR ?
212 014316 022737 000003 001342                CMP    #3,RETRY     ;ERROR RETRY LIMIT ?
213 014324 001403                BEQ   4$            ;BR IF REACHED
214 014326 005237 001342                INC    RETRY        ;COUNT THE ERROR
215 014332 000744                BR     1$            ;TRY AGAIN
216 014334 005037 001176                4$:   CLR    $ESCAPE  ;CLEAR ERROR ESCAPE ADDRESS
217 014340 005037 001342                CLR    RETRY        ;CLEAR THE RETRY COUNTER
218
```



```

1          .SBTTL CHECK TRACK JUST WRITTEN
2
3          ;SET UP TO CHECK THE TRACK JUST WRITTEN, WITH A WRITE CHECK HEADER
4          ;AND DATA COMMAND
5
6 014344 112737 000153 005542 CKTRK: MOVB #WCKHD,FMTDPB+2 ;SET WRITE CHECK HEADER & DATA COMMAND IN TBL
7 014352 012737 014352 001124   MOV #,$LPERR ;SETUP LOOP ON ERROR ADDRESS
   014360 012706 001100   MOV #STACK,SP ;LOAD STACK POINTER
8
9 014364 004037 031214   1$: JSR R0,RM05 ;GO CHECK THE TRACK JUST FORMATTED
10 014370 005540   FMTDPB ;ADRS OF PARAMETERS - TBL
11 014372 000774   BR 1$ ;WAIT FOR QUEUE IF FULL
12 014374 005737 005556   2$: TST FMTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
13 014400 001775   BEQ 2$ ;BR IF NOT DONE
14 014402 100132   BPL 9$ ;BR IF NO ERROR
15
16 014404 113737 005566 001344   MOVB RM.REG+RMDA,BADSEC ;GET THE SECTOR ADDRESS
17 014412 001005   BNE 3$ ;BR IF NOT SECTOR 0
18 014414 013737 001360 001344   MOV MAXSEC,BADSEC ;RESTORE TO LAST SECTOR +1
19 014422 005237 001344   INC BADSEC ;INCREMENT TO LAST SECTOR +1
20 014426 005337 001344   3$: DEC BADSEC ;ADJUST TO BAD SECTOR
21 014432 012737 014736 001176   MOV #11$, $ESCAPE ;ESCAPE TO 11$ ON ERROR
22 014440 004737 016550   JSR PC,ERINDX ;SEE WHICH ERROR
23 014444 104010   EMT 10 ;DRIVE OFFLINE
24 014446 104011   EMT 11 ;PERSISTENT DRIVE UNSAFE ERROR
25 014450 104012   EMT 12 ;UNCORRECTABLE MASSBUS PARITY ERROR
26 014452 104013   EMT 13 ;SOFTWARE TIMEOUT
27 014454 104014   EMT 14 ;DRIVE UNSAFE ERROR
28 014456 000240   NOP ;FILLER FOR 'ERINDX' ROUTINE
29
30 014460 032737 040000 005570   BIT #WCE,RM.REG+RMCS2 ;WRITE CHECK ERROR ?
31 014466 001005   BNE 4$ ;BR IF YES
32 014470 033727 005574   BIT RM.REG+RMER1,(PC)+ ;CHECK FOR DATA ERRORS
33 014474 130620   .WORD DCK!OPI!DTE!HCRC!HCE!FER
34 014476 001001   BNE 4$ ;BR IF YES
35 014500 104016   EMT 16 ;CONTROLLER/DRIVE ERROR DURING WRITE CHECK
36
37 014502 005737 001316   4$: TST SOFSW ;RETRYING THE SECTOR ?
38 014506 001413   BEQ 5$ ;BR IF NO
39 014510 012737 014762 001176   MOV #12$, $ESCAPE ;ESCAPE TO 12$ ON ERROR
40 014516 004737 020146   JSR PC,RECORD ;RECORD THE BAD SECTOR IN USR TABLE
41 014522 012737 100000 001410   MOV #MF,HDRBIT ;INDICATE 'USR' BAD SECTOR
42 014530 004737 020754   JSR PC,RESET ;RESET THE HEADER BITS AND WRITE HEADER
43 014534 104017   EMT 17 ;RETRIES NOT SUCCESSFUL-SECTOR NOT ACCEPTABLE
44
45 014536 005037 001176   5$: CLR $ESCAPE ;RESET ESCAPE ADDRESS
46 014542 032737 040000 005572   BIT #ERR,RM.REG+RMDS ;DATA ERROR ?
47 014550 001002   BNE 6$ ;BR IF YES
48 014552 104024   EMT 24 ;WRITE CHECK ERROR
49 014554 000401   BR 7$ ;CONTINUE
50 014556   6$: EMT 20 ;DATA ERROR DURING WRITE CHECK
   014556 104020
51
52 014560 013701 001344   7$: MOV BADSEC,R1 ;FAILING SECTOR ADDRESS
53 014564 113737 001344 005550   MOVB BADSEC,FMTDPB+10 ;SECTOR ADDRESS
54 014572 006301   ASL R1 ;SETUP INDEX TO ADDRESS WORDS
55 014574 016137 006064 005546   MOV ADRTBL(R1),FMTDPB+6 ;BUFFER ADDRESS FOR FAILING SECTOR

```


56	014602	013701	001360		MOV	MAXSEC,R1	:R1 ENDING SECTOR #
57	014606	005737	005536		TST	NEXT	:WHOLE TRACK DONE ?
58	014612	100404			BMI	8\$:BR IF IT IS
59	014614	013701	005536		MOV	NEXT,R1	:LOAD CURRENT ENDING SECTOR
60	014620	162701	000002		SUB	#2,R1	:ADJUST TWO SECTORS
61	014624	006301		8\$:	ASL	R1	:WORD INDEX
62	014626	016102	006164		MOV	WCTBL(R1),R2	:WORDCTR FOR ENDING SECTOR
63	014632	113701	005550		MOV	FMTDPB+10,R1	:STARTING ADDRESS
64	014636	006301			ASL	R1	:WORD COUNT
65	014640	166102	006164		SUB	WCTBL(R1),R2	:WORD CTR FOR STARTING SECTOR
66	014644	005402			NEG	R2	
67	014646	010237	001346		MOV	R2,SAVWC	:SAVE THE WORD COUNT
68	014652	012737	177376	005544	MOV	#-258.,FMTDPB+4	:WORD COUNT FOR 1 SECTOR
69	014660	005237	001316		INC	SOF SW	:INDICATE THAT A RETRY IS IN PROGRESS
70	014664	000137	014224		JMP	WRTRK2	:REFORMAT ERROR SECTOR
71							
72	014670	005737	001316		TST	SOF SW	:RETRY IN PROGRESS ?
73	014674	001432		9\$:	BEQ	12\$:BR IF NOT
74	014676	022737	000002	001316	CMP	#2,SOF SW	:SEE IF LAST RETRY
75	014704	001403			BEQ	10\$:BR IF IT IS
76	014706	005237	001316		INC	SOF SW	:INCREMENT RETRY COUNT
77	014712	000624			BR	1\$:READ AGAIN
78							
79	014714	123737	001360	005550	10\$:	CMPB	MAXSEC,FMTDPB+10 ;SEE IF LAST SECTOR ON THE TRACK
80	014722	001417			BEQ	12\$:BR IF IT IS
81	014724	004737	017072		JSR	PC,SCAWC	:SETUP TO CHECK REMAINING SECTORS ON THE TRACK
82	014730	005037	001316		CLR	SOF SW	:CLEAR RETRY COUNTER
83	014734	000613			BR	1\$:FINISH CHECKING THE TRACK
84							
85	014736	004737	016646		11\$:	JSR	PC,LOP.CK ;CHECK FOR LOOP ON ERROR
86	014742	022737	000003	001342	CMP	#3,RETRY	:ERROR RETRY REACHED ?
87	014750	001404			BEQ	12\$:BR IF IT IS
88	014752	005237	001342		INC	RETRY	:COUNT THE RETRY
89	014756	000137	014364		JMP	1\$:DO THE WRITE CHECK AGAIN
90							
91	014762	005037	001176		12\$:	CLR	\$ESCAPE ;CLEAR THE ERROR RETURN ESCAPE ADDRESS
92	014766	005037	001342		CLR	RETRY	:CLEAR THE RETRY COUNTER
93	014772	005037	001316		CLR	SOF SW	:CLEAR THE SOFTWARE RETRY COUNT
94	014776	005737	005536		TST	NEXT	:WHOLE TRACK DONE ?
95	015002	100402			BMI	13\$:BR IF IT IS
96	015004	000137	013676		JMP	WRTRKA	
97	015010			13\$:			


```

1
2
3
4
5
6
7 015110 005037 001342 VFYHDR: CLR RETRY ;CLEAR RETRY COUNTER
8 015114 005737 001320 TST MODE ;IS IT 'VERIFY HEADER ONLY' MODE ?
9 015120 001003 BNE 1$ ;BR IF YES, ELSE
10 015122 104401 037337 TYPE ,MSQVER ;TYPE 'STARTING QUICK HEADER VERIFY'
11 015126 000402 BR 2$
12 015130 104401 037376 1$: TYPE ,MSVER ;TYPE 'STARTING HEADER VERIFY'
13 015134 004737 016714 2$: JSR PC,SETDPB ;SET DRIVE PARAMETER TABLE
14 015140 004737 017120 JSR PC,SETHDR ;SET HEADERS IN CORE
15 015144 012700 046234 3$: MOV #BUFV,R0 ;GET STARTING ADDRESS OF EXPECTED HEADERS
16 015150 012701 106434 MOV #BUFV,R1 ;GET STARTING ADDRESS OF RECIEVED HEADERS
17 015154 010137 005546 MOV R1,FMTDPB+6 ;SET UP BUFFER ADDRESS
18 015160 013737 001352 005544 MOV MWC,FMTDPB+4 ;SETUP WORD FOR VERIFY ONLY MODE
19 015166 005037 001322 CLR SNGSEC ;SETUP MULTI-SECTOR READ FLAG
20 015172 005737 001320 TST MODE ;IS IT 'VERIFY HEADER ONLY' MODE ?
21 015176 001005 BNE 5$ ;BR IF YES
22 015200 012737 177776 005544 4$: MOV #-2,FMTDPB+4 ;SETUP WORD COUNT FOR QUICK VERIFY
23 015206 005237 001322 INC SNGSEC ;SETUP SINGLE SECTOR READ
24
25 015212 012737 015212 001124 5$: MOV #,$LPERR ;SETUP LOOP ON ERROR ADDRESS
26 015220 112737 000020 005541 6$: MOVB #20,FMTDPB+1 ;LOAD 'FMT' BIT FOR 16 BIT FORMAT
27 015226 023727 005552 001466 CMP FMTDPB+12,#822. ;IS IT LAST CYLINDER ?
28 015234 103404 BLO 7$ ;BR IF NO
29 015236 123737 005551 001324 CMPB FMTDPB+11,LSTRK ;LAST TRACK OF LAST CYLINDER (BSF) ?
30 015244 001405 BEQ 8$ ;BR IF YES
31 015246 005737 001356 7$: TST SEC30 ;IS IT 16 BIT MODE ?
32 015252 001002 BNE 8$ ;BR IF YES
33 015254 105037 005541 CLRB FMTDPB+1 ;LOAD 'FMT' BIT FOR 18 BIT FORMAT
34 015260 112737 000143 005542 8$: MOVB #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
35 015266 004037 031214 9$: JSR R0,RM05 ;START THE COMMAND
36 015272 005540 FMTDPB ;DPB ADDRESS
37 015274 000774 BR 9$ ;QUEUE FULL RETURN
38 015276 005737 005556 10$: TST FMTDPB+16 ;LOOK FOR DONE
39 015302 001775 BEQ 10$ ;LOOP UNTIL FINISHED
40
41 015304 112737 000173 005542 MOVB #RDHD,FMTDPB+2 ;SET UP READ HEADER & DATA COMMAND
42 015312 004037 031214 JSR R0,RM05 ;GO READ HEADER
43 015316 005540 FMTDPB ;ADRS OF PARAMETER TBL
44 015320 000737 11$: BR 6$ ;WAIT IF QUEUE IS FULL
45 015322 005737 005556 12$: TST FMTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
46 015326 001775 BEQ 12$ ;BR IF NOT DONE
47 015330 100046 BPL 15$ ;BR IF NOT ERROR
48
49 015332 012737 015562 001176 MOV #18$, $ESCAPE ;:ESCAPE TO 18$ ON ERROR
50 015340 004737 016550 JSR PC,ERINDX ;SEE WHICH ERROR
51 015344 104010 EMT 10 ;DRIVE OFFLINE
52 015346 104011 EMT 11 ;PERSISTENT DRIVE UNSAFE ERROR
53 015350 104012 EMT 12 ;UNCORRECTABLE MASSBUS PARITY ERROR
54 015352 104013 EMT 13 ;SOFTWARE TIMEOUT
55 015354 104014 EMT 14 ;DRIVE UNSAFE ERROR
56 015356 000240 NOP ;FILLER FOR 'ERINDX' ROUTINE
57

```


58	015360	032737	040000	005572		BIT	#ERR,RM.REG+12	:ANY DRIVE ERRORS ?
59	015366	001427				BEQ	15\$:BR IF NO
60	015370	005737	001322			TST	SNGSEC	:ALREADY READING SINGLE SECTORS ?
61	015374	001701				BEQ	4\$:BR IF NO
62	015376	005237	001342			INC	RETRY	:INCREMENT RETRY COUNT
63	015402	022737	000003	001342		CMP	#3,RETRY	:TRIED 3 TIMES ?
64	015410	101273				BHI	4\$:BR IF NO
65	015412	022737	000200	005574		CMP	#HCE,RM.REG+14	:IS IT A HEADER COMPARE ERROR ?
66	015420	001404				BEQ	13\$:BR IF YES
67	015422	032737	000400	005574		BIT	#HCRC,RM.REG+14	:IS IT A HEADER CRC ERROR ?
68	015430	001401				BEQ	14\$:BR IF NO
69	015432				13\$:			
	015432	104022				EMT	22	: 'HCE' OR 'HCRC' ERROR VERIFYING HEADERS
70								
71	015434	022737	100000	005622	14\$:	CMP	#BSE,RM.REG+42	:BAD SECTOR ERROR ?
72	015442	001453				BEQ	19\$:BR IF YES
73	015444	104021				EMT	21	:CONTROLLER/DRIVE ERROR VERIFYING HEADERS
74								
75	015446	005037	001342		15\$:	CLR	RETRY	:CLEAR RETRY COUNT
76	015452	012737	015562	001176		MOV	#18\$, \$ESCAPE	:ESCAPE TO 18\$ ON ERROR
77	015460	011037	046230			MOV	(R0),RBUF+4	:GET EXPECTED CYLINDER HEADER
78	015464	023727	005552	001466		CMP	FMTDPB+12,#822.	:IS IT LAST CYLINDER ?
79	015472	103407				BLO	16\$:BR IF NO
80	015474	123737	005551	001324		CMPB	FMTDPB+11,LSTRK	:LAST TRACK OF LAST CYLINDER (BSF) ?
81	015502	001003				BNE	16\$:BR IF NO
82	015504	052737	010000	046230		BIS	#FMT,RBUF+4	:BAD SECTOR FILE IS ALWAYS 16 BIT MODE
83	015512	052711	140000		16\$:	BIS	#MF!UF,(R1)	:SET MFG AND USR GOOD SECTOR BITS
84	015516	011137	046224			MOV	(R1),RBUF	:GET RECIEVED CYLINDER HEADER
85	015522	023737	046224	046230		CMP	RBUF,RBUF+4	:SEE IF CYL READ EQUALS CYL EXPECTED
86	015530	001401				BEQ	17\$:BR IF CYL CORRECT
87	015532	104023				EMT	23	:CYLINDER FIELD IN HEADER IS NOT CORRECT
88								
89	015534	016037	000002	046232	17\$:	MOV	2(R0),RBUF+6	:GET EXPECTED TRACK/SECTOR 'EADER
90	015542	016137	000002	046226		MOV	2(R1),RBUF+2	:GET RECIEVED TRACK/SECTOR HEADER
91	015550	023737	046226	046232		CMP	RBUF+2,RBUF+6	:IS TRACK AND SECTOR CORRECT ?
92	015556	001405				BEQ	19\$:BR IF YES
93	015560	104007				EMT	7	:TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT
94								
95	015562	004737	016646		18\$:	JSR	PC,LOP.CK	:CHECK FOR LOOP ON ERROR
96	015566	000137	016510			JMP	REJCTS	:FAILED DURING HEADER CHECK, REJECTED
97								
98	015572	005737	001320		19\$:	TST	MODE	:IS IT 'FORMAT' MODE ?
99	015576	001416				BEQ	20\$:BR IF YES
100	015600	123737	005550	001360		CMPB	FMTDPB+10,MAXSEC	:DONE WITH ALL SECTORS ?
101	015606	001412				BEQ	20\$:BR IF YES
102	015610	105237	005550			INCB	FMTDPB+10	:UPDATE SECTOR ADDRESS
103	015614	062700	001004			ADD	#516.,R0	:ADJUST POINTER TO NEXT EXPECTED HEADER
104	015620	005737	001322			TST	SNGSEC	:READING ONE SECTOR AT A TIME ?
105	015624	001235				BNE	11\$:BR IF YES
106	015626	062701	001004			ADD	#516.,R1	:ADJUST POINTER TO NEXT RECIEVED HEADER
107	015632	000705				BR	15\$:GO COMPARE NEXT SECTOR
108								
109	015634	105037	005550		20\$:	CLRB	FMTDPB+10	:START AT SECTOR 0
110	015640	105237	005551			INCB	FMTDPB+11	:INCREMENT TRACK NUMBER
111	015644	123737	005551	001332		CMPB	FMTDPB+11,MAXTRK	:WAS IT LAST TRACK ?
112	015652	101003				BHI	21\$:BR IF YES
113	015654	004737	017320			JSR	PC,UPDTRK	:UPDATE TRACK ADDRESS IN BUFFER

114	015660	000425				BR	24\$:GO READ NEXT TRACK
115									
116	015662	113737	001334	005551	21\$:	MOVB	MINTRK,FMTDPB+11		:GET STARTING TRACK ADDRESS
117	015670	005737	001376			TST	WRAP		:SEEKING FROM HI TO LO CYLINDERS ?
118	015674	001407				BEQ	22\$:BR IF NO
119	015676	005337	005552			DEC	FMTDPB+12		:DECREMENT CYLINDER ADDRESS
120	015702	023737	005552	001326		CMP	FMTDPB+12,MAXCYL		:DONE WITH VERIFY ?
121	015710	002413				BLT	25\$:BR IF YES
122	015712	000406				BR	23\$		
123	015714	005237	005552		22\$:	INC	FMTDPB+12		:INCREMENT CYLINDER ADDRESS BEING CHECKED
124	015720	023737	005552	001326		CMP	FMTDPB+12,MAXCYL		:DONE WITH VERIFY ?
125	015726	101004				BHI	25\$:BR IF YES
126	015730	004737	017206		23\$:	JSR	PC,UPDCYL		:UPDATE CYLINDER ADDRESS IN BUFFER
127	015734	000137	015144		24\$:	JMP	3\$:GO READ NEXT CYLINDER
128	015740				25\$:				

.SBTTL END OF PASS ROUTINE

```

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

```

015740 015740 005737 001406 $EOP:
015744 001535 1$: TST UPDBSF ;UPDATE LAST TRACK ?
015746 005037 001342 BEQ 8$ ;BR IF NO
015752 004537 020250 CLR RETRY ;CLEAR RETRY COUNT
015756 003434 JSR R5,SORT2 ;SORT THE USTAB IN ASCENDING FORM
015760 004737 020410 USTAB+8. ;TABLE ENTRY POINT
015764 112737 000020 005541 JSR PC,SETBSF ;SETUP THE BUFFER FOR THE BAD SECTOR FILE
015772 112737 000143 005542 MOVB #20,FMTDPB+1 ;16 BIT MODE
MOV #SETFMT,FMTDPB+2 ;SET FORMAT COMMAND

016000 004037 031214 2$: JSR R0,RM05 ;CALL THE DRIVER
016004 005540 FMTDPB
016006 000774 BR 2$ ;LOOP IF QUEQU FAILS
016010 005737 005556 3$: TST FMTDPB+16 ;ALL DONE ?
016014 001775 BEQ 3$ ;BR IF NOT

016016 012737 001466 005552 MOV #822.,FMTDPB+12 ;CYLINDER ADDRESS
016024 113737 001324 005551 MOVB LSTRK,FMTDPB+11 ;TRACK NUMBER
016032 105037 005550 CLRB FMTDPB+10 ;SECTOR 0
016036 012737 157700 005544 MOV #-<258.*32.>,FMTDPB+4 ;WORD COUNT
016044 012737 046234 005546 MOV #BUFP,FMTDPB+6 ;BUFFER ADDRESS
016052 112737 000163 005542 MOVB #WRTHD,FMTDPB+2 ;WRITE HEAD AND DATA COMMAND

016060 004037 031214 4$: JSR R0,RM05 ;CALL THE DRIVER
016064 005540 FMTDPB
016066 000774 BR 4$ ;LOOP IF QUEUE FAILS
016070 005737 005556 5$: TST FMTDPB+16 ;ALL DONE ?
016074 001775 BEQ 5$ ;BR IF NOT
016076 100060 BPL 8$ ;EXIT IF NO ERROR

016100 012737 016126 001176 MOV #6$, $ESCAPE ;ERROR ESCAPE ADDRESS
016106 004737 016550 JSR PC,ERINDX ;ERROR INDICATOR RT.
016112 104010 EMT 10 ;DRIVE OFFLINE
016114 104011 EMT 11 ;PERSISTENT DRIVE UNSAFE ERROR
016116 104012 EMT 12 ;UNCORRECTABLE MASSBUS PARITY ERROR
016120 104013 EMT 13 ;SOFTWARE TIMEOUT
016122 104014 EMT 14 ;DRIVE UNSAFE ERROR
016124 104015 EMT 15 ;DRIVE/CONTROLLER ERROR DURING WRITE

016126 005237 001342 6$: INC RETRY ;INCREMENT RETRY COUNT
016132 022737 000003 001342 CMP #3,RETRY ;OVER 3 TIMES ?
016140 101347 BHI 4$ ;BR IF IT IS
016142 000137 016464 JMP REJCT3 ;UNABLE TO WRITE THE BAD SECTOR TRACK

016146 122737 000037 005550 CMPB #31.,FMTDPB+10 ;LAST SECTOR ?
016154 003431 BLE 8$ ;BR IF ALL SET
016156 113737 005566 001344 MOVB RM.REG+RMDA,BADSEC ;FOUND THE BAD SECTOR
016164 001003 BNE 7$ ;BR IF NOT 0
016166 012737 000037 001344 MOV #31.,BADSEC ;THE LAST SECTOR IS A BAD SECTOR
016174 113737 001344 005550 7$: MOVB BADSEC,FMTDPB+10 ;LOAD THE NEW STARTING SECTOR
  
```



```

016202 113701 001344      MOVB   BADSEC,R1      ;LOAD THE WORD CTR AND BUFFER
016206 006301              ASL    R1              ;WORD INDEX
016210 016137 006064 005546  MOV    ADRTBL(R1),FMTDPB+6 ;BUFFER ADDRESS
016216 012737 157700 005544  MOV    #-<258.*32.>,FMTDPB+4 ;WORD COUNT
016224 066137 006164 005544  ADD    WCTBL(R1),FMTDPB+4 ;WORD COUNT
016232 005037 001342      CLR    RETRY          ;RESET THE RETRY FLAG
016236 000710      BR    4$             ;BR BACK

016240 104401 037427      8$:    TYPE    ,MSCMPT      ;TYPE 'COMPLETED'
016244 005737 001126      TST    $ERTTL        ;ANY ERRORS DETECTED ?
016250 001405      BEQ    9$            ;BR IF NO
016252 104401 037442      TYPE    ,NUMERR      ;TOTAL ERRORS
016256 013746 001126      MOV    $ERTTL,-(SP)  ;;SAVE $ERTTL FOR TYPEOUT
016262 104405      TYPDS          ;;GO TYPE--DECIMAL ASCII WITH SIGN
016264 005037 001126      9$:    CLR    $ERTTL    ;CLEAR THE ERROR ACCUMULATOR
016270 005037 001406      CLR    UPDBSF      ;DON'T ALLOW UPDATE OF 'BSF' AT END OF PASS
016274 104401 042423      TYPE    ,STARS      ;TYPE '*****'
016300 005737 001400      TST    DEVMP       ;ALL ASSIGNED DEVICE DONE ?
016304 001030      BNE    DONE        ;TO FORMAT NEXT DRIVE
016306 013737 001402 001400  MOV    DEVMP+2,DEVMP ;RESTORE DEVICE MAP FOR AUTO MODE
016314 005237 001216      INC    $PASS       ;;INCREMENT THE PASS NUMBER
016320 042737 100000 001216  BIC    #100000,$PASS ;DON'T ALLOW A NEG. NUMBER
016326 005327      DEC    (PC)+      ;;LOOP?
016330 000001      $EOPCT: .WORD    1
016332 003013      BGT    $DOAGN      ;;YES
016334 012737      MOV    (PC)+,@(PC)+ ;;;RESTORE COUNTER
016336 000001      $ENDCT: .WORD    1
016340 016330      $EOPCT
016342 013700 000042      $GET42: MOV    @#42,R0 ;;;GET MONITOR ADDRESS
016346 001405      BEQ    $DOAGN      ;;;BRANCH IF NO MONITOR
016350 000005      RESET          ;;;CLEAR THE WORLD
016352 004710      $ENDAD: JSR    PC,(R0) ;;;GO TO MONITOR
016354 000240      NOP            ;;;SAVE ROOM
016356 000240      NOP            ;;;FOR
016360 000240      NOP            ;;;ACT11
016362      $DOAGN:
016362 000137      JMP    @(PC)+      ;;;RETURN
016364 016366      $RTNAD: .WORD    DONE

2
3 016366 105737 001150      DONE:  TSTB   $AUTOB    ;RUNNING IN AUTO MODE ?
4 016372 001406      BEQ    1$          ;BR IF NO
5 016374 004737 017360      JSR    PC,$TKINT  ;START THE CLOCK
6 016400 004737 024414      JSR    PC,$TKINT  ;INITIALIZE THE KEYBOARD
7 016404 000137 010446      JMP    ST1        ;RETURN TO AUTO MODE START
8
9 016410 032777 000001 162536 1$:    BIT    #SW0,@SWR   ;IS SW0=1 (SET) ?
10 016416 001002      BNE    2$          ;BR IF YES
11 016420 000137 011124      JMP    ASK2        ;ASK FOR NEXT DRIVE
12 016424 000137 013056      2$:    JMP    ST3        ;GO--LOOP ON SELECTED DRIVE

```

```

1          .SBTTL REJECT MESSAGE TABLE
2
3 016430 104401 001205 REJCT1: TYPE , $CRLF           ;CR-LF
4 016434 104401 037704   TYPE ,MSG4           ;TYPE 'DISK IS AN ALIGNMENT PACK'
5 016440 000435   BR REJEND
6
7 016442 104401 001205 REJCT2: TYPE , $CRLF           ;CR-LF
8 016446 012737 016534 001176 MOV #REJEND,$ESCAPE ;:ESCAPE TO REJEND ON ERROR
9 016454 113737 005551 001364 MOV#B FMTDPB+11,DS.TRK ;TRACK NUMBER FOR ERROR LOG
:0 016462 104025   EMT 25 ;REPORT ILLEGAL BAD SECTOR
11
12 016464 104401 001205 REJCT3: TYPE , $CRLF           ;CR-LF
13 016470 104401 041563   TYPE ,MSG17          ;TYPE 'UNABLE TO WRITE BAD SECTOR FILE TRACK'
14 016474 000417   BR REJEND
15
16 016476 104401 001205 REJCT4: TYPE , $CRLF           ;CR-LF
17 016502 104401 041630   TYPE ,MSG18          ;TYPE 'FATAL ERROR DETECTED'
18 016506 000412   BR REJEND
19
20 016510 104401 001205 REJCT5: TYPE , $CRLF           ;CR-LF
21 016514 104401 041665   TYPE ,MSG19          ;TYPE 'FAILURE DURING HEADER VERIFY'
22 016520 000405   BR REJEND
23
24 016522 104401 001205 REJCT6: TYPE , $CRLF           ;CR-LF
25 016526 104401 037643   TYPE ,MSG3           ;TYPE 'OVER 126. BAD SECTORS DETECTED'
26 016532 000400   BR REJEND
27
28 016534 005037 001406 REJEND: CLR UPDBSF          ;DON'T ALLOW BSF UPDATE AT END OF PASS
29 016540 104401 042226   TYPE ,ABORT          ;TYPE 'OPERATION ABORTED'
30 016544 000137 015740   JMP $EOP           ;EXIT
31

```



```

1          .SBTTL SUPPORT SUBROUTINES
2
3          ;THIS ROUTINE DETERMINES THE ERROR TYPE
4
5 016550 010046 ERINDX: MOV R0,-(SP) ;SAVE R0
6 016552 010146 MOV R1,-(SP) ;SAVE R1
7 016554 005001 CLR R1 ;CLEAR R1
8 016556 012700 005556 MOV #FMTDPB+16,R0 ;GET DRIVER STATUS WORD
9 016562 032710 BIT (PC)+,(R0) ;CHECK ERROR TYPE
10 016564 060006 .WORD BIT14!BIT13!BIT2!BIT1 ;DRIVE OFFLINE
11 016566 001021 BNE 5$ ;BR IF OFFLINE
12 016570 032710 BIT (PC)+,(R0) ;CHECK ERROR TYPE
13 016572 010000 .WORD BIT12 ;DRIVE PERSISTENTLY UNSAFE
14 016574 001015 BNE 4$ ;BR IF UNSAFE
15 016576 032710 BIT (PC)+,(R0) ;CHECK ERROR TYPE
16 016600 006000 .WORD BIT11!BIT10 ;UNCORRECTABLE MASSBUS PARITY ERROR ?
17 016602 001011 BNE 3$ ;BR IF PARITY ERROR
18 016604 032710 BIT (PC)+,(R0) ;CHECK ERROR TYPE
19 016606 001400 .WORD BIT09!BIT08 ;SOFTWARE TIMEOUT ?
20 016610 001005 BNE 2$ ;BR IF YES
21 016612 032710 BIT (PC)+,(R0) ;CHECK ERROR TYPE
22 016614 000020 .WORD BIT04 ;DRIVE UNSAFE ERROR ?
23 016616 001001 BNE 1$ ;BR IF YES
24 016620 005201 INC R1 ;INDEX=12, ERROR='USER DEFINED'
25 016622 005201 1$: INC R1 ;INDEX=10, ERROR=14
26 016624 005201 2$: INC R1 ;INDEX= 6, ERROR=13
27 016626 005201 3$: INC R1 ;INDEX= 4, ERROR=12
28 016630 005201 4$: INC R1 ;INDEX= 2, ERROR=11
29 016632 006301 5$: ASL R1 ;INDEX= 0, ERROR=10
30 016634 060166 000004 ADD R1,4(SP) ;DEVELOP THE RETURN ADDRESS
31 016640 012601 MOV (SP)+,R1 ;RESTORE R1
32 016642 012600 MOV (SP)+,R0 ;RESTORE R0
33 016644 000207 RTS PC ;RETURN
34
35          ;ROUTINE TO CHECK FOR LOOP ON ERROR
36
37 016646 032777 001000 162300 LOP.CK: BIT #SW09,@SWR ;LOOP ON ERROR ?
38 016654 001402 BEQ 1$ ;BR IF NOT
39 016656 000177 162242 JMP @SLPERR ;GO TO THE LOOP ON ERROR ADDRESS
40
41 016662 005037 001176 1$: CLR $ESCAPE ;CLEAR THE ERROR ESCAPE ADDRESS
42 016666 033727 005556 BIT FMTDPB+16,(PC)+ ;CHECK FOR 'FATAL' ERROR
43 016672 072002 .WORD BIT14!BIT13!BIT12!BIT10!BIT01 ;'FATAL' ERROR BITS
44 016674 001004 BNE 2$ ;BR IF YES
45 016676 032737 004000 005574 BIT #WLE,RM.REG+RMER1 ;WRITE LOCK ERROR ?
46 016704 001402 BEQ 3$ ;BR IF NOT
47 016706 000137 016476 2$: JMP REJCT4 ;TERMINATE THE FORMAT
48
49 016712 000207 3$: RTS PC ;RETURN
50
51          ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
52
53 016714 113737 001222 005540 SETDPB: MOVB DRIVE,FMTDPB ;SET UP DRIVE #
54 016722 105037 005543 CLRFB FMTDPB+3 ;CLEAR HIGH ORDER ADRS BITS
55 016726 013737 001352 005544 MOV MWC,FMTDPB+4 ;LOAD UP WORD COUNT
56 016734 012737 046234 005546 MOV #BUFP,FMTDPB+6 ;LOAD UP CURRENT ADRS
57 016742 105037 005550 CLRFB FMTDPB+10 ;SET SECTOR TO ZERO

```

```
58 016746 113737 001334 005551      MOVB   MINTRK,FMTDPB+11 ;SET UP STARTING TRK ADRS
59 016754 013737 001330 005552      MOV    MINCYL,FMTDPB+12 ;SET UP STARTING CYL
60 016762 005037 005556      CLR    FMTDPB+16        ;CLEAR R/M STATUS
61 016766 000207                RTS     PC                ;RETURN FROM SETUP
62
63 ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
64 ;IT IS ENTERED AFTER EVERY TRK OPERATION
65
66 016770                TRKTST:
67 016770 105237 005551      INCB   FMTDPB+11        ;INCREMENT TRACK NUMBER
68 016774 123737 005551 001332      CMPB   FMTDPB+11,MAXTRK ;LAST TRACK TO BE FORMATTED ON THIS CYLINDER ?
69 017002 101003                BHI    1$                ;BR IF YES
70 017004 004737 017320      JSR    PC,UPDTRK        ;UPDATE TRACK ADDRESS IN BUFFER
71 017010 000425                BR     4$                ;EXIT
72
73 017012 113737 001334 005551 1$:      MOVB   MINTRK,FMTDPB+11 ;GET STARTING TRACK ADDRESS
74 017020 005737 001376      TST    WRAP              ;SEEKING FROM HI TO LO CYLINDER ?
75 017024 001407                BEQ    2$                ;BR IF NO
76 017026 005337 005552      DEC    FMTDPB+12        ;DECREMENT CYLINDER ADDRESS
77 017032 023737 005552 001326      CMP    FMTDPB+12,MAXCYL ;DONE WITH FORMATTING ?
78 017040 002413                BLT    5$                ;BR IF YES
79 017042 000406                BR     3$                ;BR IF YES
80 017044 005237 005552      INC    FMTDPB+12        ;INCREMENT CYLINDER ADDRESS
81 017050 023737 005552 001326      CMP    FMTDPB+12,MAXCYL ;DONE WITH FORMATTING ?
82 017056 101004                BHI    5$                ;BR IF YES
83 017060 004737 017206      JSR    PC,UPDCYL        ;UPDATE CYLINDER ADDRESS IN BUFFER
84
85 017064 062716 000002      4$:      ADD    #2,(SP)          ;ADD TWO TO RETURN ADRS (NOT DONE)
86 017070 000207      5$:      RTS     PC                ;RETURN, DONE FORMAT !!
87
88 ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
89 ;AFTER A WRITE CHECK ERROR
90
91 017072 013737 001346 005544  SCAWC:  MOV    SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
92 017100 062737 001004 005546      ADD    #516,FMTDPB+6    ;SET UP BA FOR REMAINING SECTORS
93 017106 005237 005550      INC    FMTDPB+10        ;ADVANCE TO NEXT SECTOR IN TBL
94 017112 005037 001316      CLR    SOFSW            ;RESET RETRY COUNTER
95 017116 000207                RTS     PC                ;RETURN TO COMPLETE TRK FORMAT
96
97
98 ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
99 ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
100
101 017120 013701 001360      SETHDR: MOV    MAXSEC,R1    ;SET UP SECTOR COUNT
102 017124 012737 000001 001214      MOV    #1,$TESTN        ;GARBAGE CODE FOR APT
103 017132 012700 046234      MOV    #BUFP,R0         ;SET UP HEADER POINTER IN R0
104 017136 013702 001330      MOV    MINCYL,R2        ;PUT STARTING CYL # IN R2
105 017142 052702 140000      BIS    #MF!UF,R2        ;SET MFG AND USR GOOD BITS (DEC STANDARD 144)
106 017146 013703 001334      MOV    MINTRK,R3        ;PUT STARTING TRK # IN R3
107 017152 000303                SWAB   R3                ;JUSTIFY TRACK ADRS
108 017154 005737 001356      TST    SEC30            ;SEE IF 30 OR 32 SECTOR MODE
109 017160 001402                BEQ    1$                ;BR IF 30 SECTOR MODE
110 017162 052702 010000      BIS    #FMT,R2          ;SET THE 32 SECTOR FORMAT BIT
111 017166 010220      1$:      MOV    R2,(R0)+          ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
112 017170 010320                MOV    R3,(R0)+          ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
113 017172 062700 001000      ADD    #512,R0          ;SET UP FOR NEXT HEADER
114 017176 005203                INC    R3                ;UPDATE SECTOR ADRS FOR NEXT HEADER
```



```

115 017200 005301          DEC      R1          ;MAINTAIN COUNT OF SECTORS
116 017202 002371          BGE     1$          ;BR IF NOT LAST SECTOR
117 017204 000207          RTS      PC          ;EXIT - HEADERS ARE LOADED INTO CORE
118
119                          ;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS
120                          ;AND FILLS DATA BUFFER FIELD IN CORE
121
122 017206          UPDCYL:
    017206 010046          MOV     R0,-(SP)      ;;PUSH R0 ON STACK
    017210 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
    017212 010246          MOV     R2,-(SP)      ;;PUSH R2 ON STACK
    017214 010346          MOV     R3,-(SP)      ;;PUSH R3 ON STACK
123 017216 013701 001360          MOV     MAXSEC,R1    ;SET UP SECTOR COUNT
124 017222 012700 046234          MOV     #BUFP,R0    ;SET UP HEADER POINTER IN R0
125 017226 013703 001334          MOV     MINTRK,R3   ;GET STARTING TRACK
126 017232 000303          SWAB    R3          ;STARTING SECTOR = 0
127 017234 005737 001376          1$:   TST     WRAP        ;SEEKING FROM HI TO LO CYLINDERS ?
128 017240 001402          BEQ     2$          ;BR IF NO
129 017242 162710 000002          SUB     #2,(R0)     ;DECREMENT CYLINDER NUMBER
130 017246 005220          2$:   INC     (R0)+      ;INCREMENT FOR NEXT CYLINDER
131 017250 010320          MOV     R3,(R0)+   ;WRITE TRACK AND SECTOR HEADER
132 017252 012702 001000          MOV     #512.,R2   ;DATA FIELD LENGTH
133 017256 013720 001336          3$:   MOV     PATA,(R0)+ ;FILL BUFFER
134 017262 013720 001340          MOV     PATB,(R0)+ ;FILL BUFFER
135 017266 162702 000004          SUB     #4,R2      ;END OF DATA FIELD ?
136 017272 001371          BNE     3$        ;NO
137 017274 005203          INC     R3        ;INCREMENT TO NEXT SECTOR
138 017276 005301          DEC     R1        ;COUNT SECTORS
139 017300 002355          BGE     1$        ;BR IF NOT LAST SECTOR
140 017302 012603          MOV     (SP)+,R3   ;;POP STACK INTO R3
    017304 012602          MOV     (SP)+,R2   ;;POP STACK INTO R2
    017306 012601          MOV     (SP)+,R1   ;;POP STACK INTO R1
    017310 012600          MOV     (SP)+,R0   ;;POP STACK INTO R0
141 017312 005237 001214          INC     $TESTN    ;INCREMENT TST NUMBER FOR EACH CYLINDER CHANGE
142
143 017316 000207          RTS      PC          ;GARBAGE CODE FOR APT
144
145                          ;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE
146
147 017320          UPDTRK:
    017320 010046          MOV     R0,-(SP)      ;;PUSH R0 ON STACK
    017322 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
148 017324 013701 001360          MOV     MAXSEC,R1    ;SET UP SECTOR COUNT
149 017330 012700 046234          MOV     #BUFP,R0    ;SET UP HEADER POINTER IN R0
150 017334 005720          TST     (R0)+      ;POINT HEADER POINTER TO TRK - SEC ADRS
151 017336 062710 000400          1$:   ADD     #400,(R0)   ;INDEX TRK ADRS
152 017342 062700 001004          ADD     #516.,R0    ;SET UP FOR NEXT HEADER
153 017346 005301          DEC     R1        ;COUNT SECTORS
154 017350 002372          BGE     1$        ;BR IF NOT LAST SECTOR
155 017352 012601          MOV     (SP)+,R1   ;;POP STACK INTO R1
    017354 012600          MOV     (SP)+,R0   ;;POP STACK INTO R0
156 017356 000207          RTS      PC          ;EXIT
157
158                          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
159
160 017360 012737 017436 000004 ST.CLK: MOV     #STCLK1,@#ERRVEC ;SET UP VECTOR FOR P CLK
161 017366 005037 000006          CLR     @#ERRVEC+2 ;NEW PSW

```

```
162 017372 005777 161702          TST    @SLKCSR          :CHECK FOR KW11-P
163 017376 013746 001304          MOV    $LPVEC,-(SP)    :VECTOR ADDRESS
164 017402 012776 017616 000000    MOV    #CLOCK,@(SP)   :SET UP KW11-P VECTOR
165 017410 062716 000002          ADD    #2,(SP)        :POINT TO PSW
166 017414 012736 000300          MOV    #PR6,@(SP)+    :PSW - PRI 6
167 017420 012777 177777 161654    MOV    #-1,@SLKCSB    :LOAD COUNTER BUFFER
168 017426 012777 000135 161644    MOV    #135,@SLKCSR   :SET CLK - CNT UP
169 017434 000426                    BR     STCLK3
170 017436 062706 000004          STCLK1: ADD   #4,SP    :RESTORE THE STACK POINTER
171 017442 012737 017506 000004    MOV    #STCLK2,@#ERRVEC :CHANGE ERROR VECTOR
172 017450 005777 161634          TST    @SLKS          :LOOK FOR KW11-L
173 017454 013746 001312          MOV    $LLVEC,-(SP)   :KW11-L VECTOR ADDRESS
174 017460 012776 017616 000000    MOV    #CLOCK,@(SP)   :SET UP KW11-L VECTOR
175 017466 062716 000002          ADD    #2,(SP)        :INCREMENT VECTOR ADDRESS
176 017472 012736 000300          MOV    #PR6,@(SP)+    :PSW - PRI 6
177 017476 012777 000100 161604    MOV    #100,@SLKS     :SET KW11-L INTERRUPT ENABLE
178 017504 000402                    BR     STCLK3
179 017506 062706 000004          STCLK2: ADD   #4,SP    :RESTORE THE STACK POINTER
180 017512 012737 000006 000004    STCLK3: MOV    #6,@#ERRVEC :RESTORE THE ERROR VECTOR
181 017520 000207          RTS     PC
182
183          ;THIS CODE PRINT THE ABORT MESSAGE AND LET O.P. RESTART
184
185 017522 104401 037572          RESTRT: TYPE    ,MSG2    :TYPE 'IMPROPER MFG INFORMATION'
186 017526 105737 001150          TSTB   $AUTOB        :RUN UNDER APT ?
187 017532 001000                    BNE    1$            :BR IF SO
188 017534 004737 024414          1$:   JSR    PC,$TKINT  :ENABLE TO READ
189 017540 104401 042033          TYPE    ,MSINIT     :TYPE 'INITIALIZE BAD SECTOR FILE (L) N ?'
190 017544 104411                    RDLIN                    :READ THE ENTRY
191 017546 012600                    MOV    (SP)+,RO      :SAVE ADDRESS OF RESPONSE
192 017550 105710                    TSTB   (RO)          :WAS RESPONSE A CARRIAGE RETURN (DEFAULT 'N')?
193 017552 001414                    BEQ    3$            :BR IF YES
194 017554 105760 000001          TSTB   1(RO)        :WAS IT TERMINATED WITH CARRIAGE RETURN ?
195 017560 001006                    BNE    2$            :BR IF NO
196 017562 122710 000131          CMPB   #'Y',(RO)    :WAS IT A 'Y' RESPONSE ?
197 017566 001412                    BEQ    4$            :BR IF YES
198 017570 122710 000116          CMPB   #'N',(RO)    :WAS IT A 'N' RESPONSE ?
199 017574 001403                    BEQ    3$            :BR IF YES
200 017576 104401 037174          2$:   TYPE    ,BADENT  :TYPE BAD ENTRY MESSAGE
201 017602 000754                    BR     1$            :TRY AGAIN
202 017604 104401 001205          3$:   TYPE    ,$CRLF   :CR-LF
203 017610 000137 016534          JMP    REJEND        :JUMP TO REJECT END
204 017614 000207          4$:   RTS     PC      :EXIT
205
206          ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
207
208 017616 012746 000020          CLOCK: MOV    #16,-(SP) :PUT MILLISECONDS ON THE STACK
209 017622 004737 034476          JSR    PC,RMTMR     :GO REPORT TIME
210 017626 000002          RTI                    :RETURN AND CONTINUE
211
212          ;ROUTINE TO INTERCEPT 'CONTROL C'
213
214 017630 005737 030400          C.ENTR: TST    TRNSWT  :ALL ACTIVITY STOPPED ?
215 017634 001375                    BNE    C.ENTR        :BR IF NOT
216 017636 000137 007034          JMP    TST1          :START PROGRAM AT BEGINNING
217
218          ;ROUTINE TO INTERCEPT 'CONTROL O' TYPED WHILE PROGRAM IS FORMATTING
```



```

219      ;THE DISK PACK. THE CURRENT DISK ADDRESS IS TYPED TO THE TERMINAL
220      ;DURING THIS ROUTINE IN THE FOLLOWING FORMAT.
221      ;
222      ;PRESENT ADDRESS IS: CXXX TXXX      ;WHERE 'C' IS THE CYLINDER ADDRESS
223      ;                                   ;AND 'T' IS THE TRACK ADDRESS.(IN DECIMAL)
224
225 017642 117746 161314      O.ENTR: MOVB   @STKB,-(SP)      ;READ FROM TTY BUFFER
226 017646 042716 177600      BIC     #^C177,(SP)      ;STRIP PARITY
227 017652 021627 000003      CMP     (SP),#3         ;CONTROL-C ?
228 017656 001014              BNE     2$              ;BR IF NO
229 017660 104401 025621      TYPE   ,SCNTLC         ;TYPE A CONTROL-C (^C)
230 017664 005726              TST     (SP)+          ;RESTORE STACK
231 017666 105737 001150      TSTB   $AUTOB          ;RUNNING IN AUTO MODE ?
232 017672 001404              BEQ     1$              ;BR IF NO
233 017674 005037 001400      CLR    DEVMP           ;CLEAR DEVICE MAPS
234 017700 005037 001402      CLR    DEVMP+2
235 017704 000137 016534      1$:    JMP     REJEND      ;JUMP TO CONTROL C REJECT
236
237 017710 022627 000017      2$:    CMP     (SP)+,#17   ;CONTROL-O ?
238 017714 001406              BEQ     3$              ;YES
239 017716 005777 161240      TST    @STKB           ;CLEAR DONE BIT
240 017722 012777 000100 161230  MOV     #100,@STKS      ;ENABLE TTY INTERRUPT
241 017730 000002              RTI
242 017732
243
244 017732 005037 177776      TYPADR: CLR    PS        ;SET PROCESSOR TO PRIORITY 0
245 017736 012737 017630 001372  MOV     #C.ENTR,CNTLC   ;CHANGE 'CONTROL C' RETURN ADDRESS
246 017744 104401 037506      TYPE   ,ADDRIS        ;'PRESENT ADDRESS IS: '
247 017750 104401 036613      TYPE   ,C              ;'C'
248 017754 013746 005552      MOV     FMTDPB+12,-(SP) ;PUT THE CYLINDER ADDRESS ON THE STACK
249 017760 004737 027100      JSR    PC,$SB2D        ;CONVERT IT TO DECIMAL
250 017764 004737 026670      JSR    PC,$SUPRL       ;TYPE DECIMAL NUMBER, LEFT JUSTIFY
251 017770 104401 036621      TYPE   ,BLNKS2        ;TYPE 2 SPACES
252 017774 104401 036615      TYPE   ,T              ;'T'
253 020000 005046              CLR    -(SP)           ;CLEAR THE STACK
254 020002 113716 005551      MOVB   FMTDPB+11,(SP)  ;PUT THE TRACK ADDRESS ON THE STACK
255 020006 004737 027100      JSR    PC,$SB2D        ;CONVERT IT TO DECIMAL
256 020012 004737 026670      JSR    PC,$SUPRL       ;TYPE DECIMAL NUMBER, LEFT JUSTIFY
257 020016 104401 001205      TYPE   ,$CRLF          ;CR-LF
258 020022 012737 017642 000060  MOV     #O.ENTR,@#TKVEC ;RESTORE ENTRANCE FOR CONTROL-O
259 020030 000002              RTI                    ;RETURN
260
261      ;PARAMETER ENTRY ROUTINE
262      ;CALL
263      ;      JSR    R4,PARENT      ;THE CALLING SEQ
264      ;      TABLE      ;PARAMETERS TABLE
265      ;
266
267 020032 010346      PARENT: MOV     R3,-(SP)      ;SAVE R3
268 020034 011403      MOV     (R4),R3         ;PARAMETER TABLE ADDRESS
269 020036 012337 020046  1$:    MOV     (R3)+,2$        ;ADDRESS OF PARAMETER NAME
270 020042 001435      BEQ     7$              ;BR IF AT END OF TABLE
271 020044 104401      TYPE   ,                ;TYPE THE PARAMETER NAME
272 020046 000000      2$:    .WORD  0            ;ADDRESS OF PARAMETER NAME TEXT
273 020050 012302      MOV     (R3)+,R2        ;MAXIMUM PARAMETER VALUE
274 020052 012305      MOV     (R3)+,R5        ;ADDRESS OF PARAMETER
275 020054 011546      MOV     (R5),-(SP)      ;CURRENT VALUE OF PARAMETER

```



```
276 020056 104405          TYPDS          :TYPE THE CURRENT VALUE OF THE PARAMETER
277 020060 104401 036607  TYPE      ,SLASH  : /
278 020064 104411          RDLIN         :READ THE KEYBOARD
279 020066 012601          MOV      (SP)+,R1  :INPUT ASCII STRING ADDRESS
280 020070 004537 023172  JSR      R5,CK.DIG :CHECK THE DIGIT(S)
    020074 020036          1$           :CARRIAGE RETURN ONLY ENTERED
    020076 020136          7$           :PERIOD ONLY ENTERED
    020100 020114          4$           :ILLEGAL INPUT
    020102 020110          3$           :TERMINATED WITH A CARRIAGE RETURN
    020104 020114          4$           :TERMINATED WITH A ''
    020106 020126          5$           :TERMINATED WITH A ''
281 020110 010215          3$: MOV      R2,(R5)  :MOVE NEW VALUE TO PARAMETER LOCATION
282 020112 000751          BR       1$         :GET MORE PARAMETERS
283 020114 104401 037174  4$: TYPE      ,BADENT. :'BAD ENTRY'
284 020120 162703 000006  SUB      #6,R3      :DECREMENT THE TABLE POINTER
285 020124 000744          BR       1$         :TRY AGAIN
286 020126 010215          5$: MOV      R2,(R5)  :NEW VALUE
287 020130 000402          BR       7$         :EXIT
288          :6$: MOV      #TABLE,R3  :RELOAD THE PARAMETER TABLE ADDRESS
289 020132 011403          6$: MOV      (R4),R3  :RELOAD THE PARAMETER TABLE ADDRESS
290 020134 000740          BR       1$         :TRY AGAIN
291 020136 012603          7$: MOV      (SP)+,R3  :RESTORE R3
292 020140 062704 000002  ADD      #2,R4      ;ADJUST THE RETURN ADDRESS
293 020144 000204          RTS      R4        :RETURN FROM THE R4
294
295          :THIS ROUTINE LOADS THE BAD SECTOR INTO 'USTAB' TABLE
296          :CALL
297          :      JSR PC,RECORD
298          :
299          :ALL REGISTER USED ARE DESTORIED
300
301 020146 012701 003434  RECCRD: MOV      #USTAB+8.,R1  :TABLE ENTRY IN R1
302 020152 022711 177777  1$:  CMP      #-1,(R1)  :AN OPENING IN THE TABLE
303 020156 001421          BEQ      3$         :BR IF IT IS
304 020160 023711 005552  CMP      FMTDPB+12,(R1) :CYLINDER NUMBER MATCHES ?
305 020164 001010          BNE     2$         :BR IF NOT
306 020166 123761 005551 000003  CMPB    FMTDPB+11,3(R1) :TRK NUMBER MATCHES ?
307 020174 001004          BNE     2$         :BR IF NOT
308 020176 123761 005550 000002  CMPB    FMTDPB+10,2(R1) :SECTOR NUMBER MATCHES ?
309 020204 001416          BEQ     4$         :BR TO EXIT,IF THE SPOT HAS RECORDED
310 020206 062701 000004  2$:  ADD      #4,R1      :INCREMENT 4 BYTES
311 020212 022701 004424  CMP      #USTAB+512.,R1 :END OF TABLE ?
312 020216 101355          BHI     1$         :BR IF NOT
313 020220 000411          BR      5$         :THROW AWAY THE PACK
314 020222 013711 005552  3$:  MOV      FMTDPB+12,(R1) :LOAD THE CYLINDER #
315 020226 113761 005551 000003  MOVB    FMTDPB+11,3(R1) :LOAD THE TRK NUMBER
316 020234 113761 001344 000002  MOVB    BADSEC,2(R1)   :LOAD THE SECTOR NUMBER
317 020242 000207          RTS      PC        :EXIT
318 020244 000137 016522  5$:  JMP      REJCT6     :JUMP TO REJECT TOO MANY BAD SECTORS
319
320          :THIS ROUTINE WILL SORT A BAD SECTOR TABLE IN ASCENDING ORDER
321          :CALL
322          :      JSR      R5,SORT2
323          :      TABLE+8.  :ADDRESS OF TABLE TO BE SORTED
324
325 020250 011501          SORT2: MOV      (R5),R1  :TOP POINTER
326 020252 012702 000175  MOV      #125.,R2   :TOTAL 126 ELEMENTS
```


327	020256	010103			MOV	R1,R3	:LOCATE THE BOTTOM POINTER
328	020260	062703	000764		ADD	#500,R3	:254-4) x 2
329	020264	022711	177777		CMP	#-1,(R1)	:THE TABLE IS EMPTY ?
330	020270	001444			BEQ	4\$:QUICK EXIT
331	020272	021161	000004	1\$:	CMP	(R1),4(R1)	:COMPARE THE CYLINDER NUMBER
332	020276	101013			BHI	2\$:SWITCH ELEMENTS
333	020300	103426			BLO	3\$:INCREMENT POINTER
334	020302	126161	000003	000007	CMPB	3(R1),7(R1)	:COMPARE THE TRACK NUMBER
335	020310	101006			BHI	2\$:SWITCH ELEMENTS
336	020312	103421			BLO	3\$:INCREMENT POINTER
337	020314	126161	000002	000006	CMPB	2(R1),6(R1)	:COMPARE THE SECTOR NUMBER
338	020322	101001			BHI	2\$:SWITCH ELEMENTS
339	020324	000414			BR	3\$:INCREMENT POINTER
340	020326	011146		2\$:	MOV	(R1),-(SP)	:SAVE THE CYLINDER NUMBER
341	020330	016146	000002		MOV	2(R1),-(SP)	:SAVE THE TRK/SEC NUMBERS
342	020334	016111	000004		MOV	4(R1),(R1)	:SWITCH THE N+4 BLOCK TO N BLOCK
343	020340	016161	000006	000002	MOV	6(R1),2(R1)	:SWITCH THE N+6 BLOCK TO N+2 BLOCK
344	020346	012661	000006		MOV	(SP)+,6(R1)	:LOAD THE N+6 BLOCK
345	020352	012661	000004		MOV	(SP)+,4(R1)	:LOAD THE N+4 BLOCK
346	020356	062701	000004	3\$:	ADD	#4,R1	:UPDATE THE TOP POINTER
347	020362	020103			CMP	R1,R3	:REACH THE BOTTOM ?
348	020364	001342			BNE	1\$:BR IF NO
349	020366	005302			DEC	R2	:DONE ENTIRE SORT ?
350	020370	001404			BEQ	4\$:BR IF YES
351	020372	162703	000004		SUB	#4,R3	:ADJUST THE BOTTOM POINTER
352	020376	011501			MOV	(R5),R1	:RESET THE TOP POINTER
353	020400	000734			BR	1\$:DO NEXT SORT
354							
355	020402	062705	000002	4\$:	ADD	#2,R5	:ADJUST RETURN ADDRESS
356	020406	000205			RTS	R5	:EXIT
357							

.SBTTL SETUP BAD SECTOR BUFFER

: THIS ROUTINE SETS UP THE BUFFER TO RE-WRITE THE LAST TRACK OF
 : CYLINDER 822. (BAD SECTOR FILE)

: SECTORS 0, 2, 4, 6, 8. 16 BIT MFG
 : SECTORS 1, 3, 5, 7, 9. 18 BIT MFG
 : SECTORS 10, 12, 14, 16,...30. 16 BIT USER
 : SECTORS 11, 13, 15, 17,...31. 18 BIT USER

1
2
3
4
5
6
7
8
9
10
11
12 020410 012700 046234
13 020414 012701 020100
14 020420 012720 177777
15 020424 005301
16 020426 001374
17 020430 013746 001214
020434 013746 001330
020440 013746 001334
020444 013746 0C1360
020450 013746 001356
18 020454 012737 000037 001360
19 020462 012737 177777 001356
20 020470 012737 001466 001330
21 020476 013737 001324 001334
22 020504 004737 017120
23 020510 012637 001356
020514 012637 001360
020520 012637 001334
020524 012637 001330
020530 012637 001214
24
25 020534 012701 001414
26 020540 012702 046240
27 020544 012703 000040
28 020550 011112
29 020552 016162 000002 000002
30 020560 005062 000004
31 020564 005062 000006
32 020570 062702 001004
33 020574 005303
34 020576 001364
35
36 020600 005046
37 020602 005046
38 020604 005046
39 020606 012746 001414
40 020612 012746 046240
41 020616 012746 000005
42 020622 012746 002420
43 020626 012746 047244
44 020632 012746 000005
45 020636 012746 003424
46 020642 012746 060310
47 020646 005737 001356
48 020652 100402
49 020654 012716 061314

SFTBSF: MOV #BUFP,R0 ;BUFFER ADDRESS
 MOV #<258.*32.>,R1 ;TOTAL WORD COUNT
 1\$: MOV #-1,(R0)+ ;SET ALL LOCATIONS TO -1
 DEC R1 ;DONE YET ?
 BNE 1\$;BR IF NO
 MOV \$TESTN,-(SP) ;:PUSH \$TESTN ON STACK
 MOV MINCYL,-(SP) ;:PUSH MINCYL ON STACK
 MOV MINTRK,-(SP) ;:PUSH MINTRK ON STACK
 MOV MAXSEC,-(SP) ;:PUSH MAXSEC ON STACK
 MOV SEC30,-(SP) ;:PUSH SEC30 ON STACK
 MOV #31,MAXSEC ;SET MAX 31 SECTORS
 MOV #-1,SEC30 ;ALWAYS IN 16 BIT MODE
 MOV #822,MINCYL ;LOAD LAST CYLINDER
 MOV LSTRK,MINTRK ;LAST TRACK
 JSR PC,SETHDR ;SET HEADER FIELD IN THE BUFFER
 MOV (SP)+,SEC30 ;:POP STACK INTO SEC30
 MOV (SP)+,MAXSEC ;:POP STACK INTO MAXSEC
 MOV (SP)+,MINTRK ;:POP STACK INTO MINTRK
 MOV (SP)+,MINCYL ;:POP STACK INTO MINCYL
 MOV (SP)+,\$TESTN ;:POP STACK INTO \$TESTN
 MOV #MFG16,R1 ;LOAD FIRST 4 WORDS TO EVERY SECTOR
 MOV #BUFP+4,R2 ;BUFFER LOCATION + 4
 MOV #32,R3 ;TOTAL 32 SECTORS
 2\$: MOV (R1),(R2) ;STORE 1ST OCTAL WORD (MSB OF S/N)
 MOV 2(R1),2(R2) ;STORE 2ND OCTAL WORD (LSB OF S/N)
 CLR 4(R2) ;STORE 3RD WORD ALL 0'S
 CLR 6(R2) ;STORE 4TH WORD ALL 0'S
 ADD #516.,R2 ;ADVANCE TO NEXT SECTOR
 DEC R3 ;DECREMENT ONE SECTOR COUNT
 BNE 2\$;BR IF NO
 CLR -(SP) ;LOAD TABLES INTO APPROPRIATE SECTORS
 CLR -(SP) ;DUMMY PAIRS FOR TERMINATORS
 CLR -(SP)
 MOV #MFG16,-(SP) ;TABLE ADDRESS
 MOV #BUFP+4,-(SP) ;BUFFER ADDRESS
 MOV #5,-(SP) ;NUMBER OF SECTORS TO LOAD
 MOV #MFG18,-(SP) ;TABLE ADDRESS
 MOV #BUFP+4+516.,-(SP) ;BUFFER ADDRESS
 MOV #5,-(SP) ;NUMBER OF SECTORS TO LOAD
 MOV #USTAB,-(SP) ;TABLE ADDRESS
 MOV #BUFP+4+<516.*10.>,-(SP) ;BUFFER ADDRESS
 TST SEC30 ;IS IT 16 BIT FORMAT MODE ?
 BMI 3\$;BR IF YES
 MOV #BUFP+4+<516.*11.>,(SP) ;MFG 18 BIT BUFFER ADDRESS


```

50 020660 012746 000013      3$:  MOV      #11.,-(SP)      ;NUMBER OF SECTORS TO LOAD
51
52 020664 012701 004430      MOV      #USSAV,R1        ;TABLE ADDRESS IN R1
53 020670 012702 061314      MOV      #BUFP+4+<516.*11.>,R2 ;BUFFER ADDRESS IN R2
54 020674 005737 001356      TST      SEC30            ;IS IT 16 BIT FORMAT MODE ?
55 020700 100402                BMI      4$              ;BR IF YES
56 020702 012702 060310      MOV      #BUFP+4+<516.*10.>,R2 ;BUFFER ADDRESS FOR 18 BIT MODE
57 020706 012703 000013      4$:  MOV      #11.,R3        ;SECTOR COUNT IN R3
58 020712 010105                MOV      R1,R5            ;TEMPORARY STORAGE
59 020714 012704 000400      5$:  MOV      #256.,R4        ;WORD COUNT = DATA FIELD OF ONE SECTOR
60 020720 012122 6$:  MOV      (R1)+,(R2)+      ;LOAD TABLE INTO SECTOR DATA FIELD
61 020722 005304                DEC      R4                ;ALL DONE ?
62 020724 001375                BNE     6$                ;BR IF NOT
63 020726 010501                MOV     R5,R1            ;RELOAD THE TABLE ADDRESS
64 020730 062702 001010      ADD     #516.+4,R2        ;SKIP ONE SECTOR
65 020734 005303                DEC     R3                ;DECREMENT ONE SECTOR COUNT
66 020736 001366                BNE     5$                ;BR IF NOT DONE
67 020740 012603                MOV     (SP)+,R3         ;NEXT NUMBER OF SECTORS TO LOAD
68 020742 012602                MOV     (SP)+,R2         ;NEXT BUFFER ADDRESS
69 020744 012601                MOV     (SP)+,R1         ;NEXT TABLE ADDRESS
70 020746 010105                MOV     R1,R5            ;IS IT DUMMY TERMINATOR ?
71 020750 001361                BNE     5$                ;BR IF NO
72 020752 000207                RTS      PC                ;EXIT
73
74                                ;THIS ROUTINE WILL SET OR RESET BIT15 AND BIT14 OF THE FIRST HEADER
75                                ;WORD. IF THE HEADER IS FLAGGED AS A 'MFG' BAD SECTOR, BIT15(MF BIT)
76                                ;WILL BE RESET(0) AND BIT14(UF BIT) WILL BE SET(1). IF THE HEADER IS
77                                ;BEING FLAGGED AS A 'USR' BAD SECTOR, BIT15(MF BIT) WILL BE SET(1)
78                                ;AND BIT14(UF BIT) WILL BE RESET(0).
79                                ;CALL
80                                ;
81                                JSR      PC,RESET
82 020754 013737 005552 046224  RESET:  MOV      FMTDPB+12,RBUF ;GET CYLINDER ADDRESS.
83 020762 001006                BNE     1$                ;BR IF NOT CYL 0
84 020764 122737 000001 005551  CMPB    #1,FMTDPB+11      ;IS THIS CYL 0, TRK 0 OR 1 ?
85 020772 103402                BLO     1$                ;BR IF NO
86 020774 000137 016442                JMP     REJCT2            ;OTHERWISE, ILLEGAL BAD SECTORS
87
88 021000 113737 005551 046227  1$:  MOVB    FMTDPB+11,RBUF+3   ;TRACK ADDRESS
89 021006 113737 001344 046226  MOVB    BADSEC,RBUF+2     ;SECTOR ADDRESS
90 021014 053737 001410 046224  BIS     HDRBIT,RBUF        ;SET MFG BIT, RESET USR BIT
91 021022 113737 001222 005630  MOVB    DRIVE,FMTWRT      ;SETUP THE DPB FOR OPERATION
92 021030 012737 177776 005634  MOV     #-2,FMTWRT+4      ;WORD COUNT =2
93 021036 112737 000163 005632  MOVB    #WRTHD,FMTWRT+2   ;WRITE HEAD AND DATA COMMAND
94 021044 012737 046224 005636  MOV     #RBUF,FMTWRT+6    ;BUFFER ADDRESS
95 021052 113737 001344 005640  MOVB    BADSEC,FMTWRT+10  ;SECTOR ADDRESS
96 021060 113737 005551 005641  MOVB    FMTDPB+11,FMTWRT+11 ;TRACK ADDRESS
97 021066 013737 005552 005642  MOV     FMTDPB+12,FMTWRT+12 ;CYLINDER ADDRESS
98 021074 052737 010000 046224  BIS     #FMT,RBUF         ;ASSUME 16 BIT MODE
99 021102 023727 005552 001466  CMP     FMTDPB+12,#822.   ;IS IT LAST CYLINDER ?
100 021110 001004                BNE     2$                ;BR IF NO
101 021112 123737 005551 001324  CMPB    FMTDPB+11,LSTRK   ;IS IT LAST TRACK (BSF) ?
102 021120 001406                BEQ     3$                ;BR IF YES
103 021122 005737 001356      2$:  TST      SEC30            ;IN 16 BIT MODE ?
104 021126 100403                BMI     3$                ;BR IF IT IS
105 021130 042737 010000 046224  BIC     #FMT,RBUF         ;RESET THE FMT BIT
106

```


107	021136	004037	031214	3\$:	JSR	RO,RM05	:CALL THE DRIVER
108	021142	005630			FMTWRT		:DPB ADDRESS
109	021144	000774			BR	3\$:BR IF QUEUE FAILS
110	021146	005737	005646	4\$:	TST	FMTWRT+16	:DONE ?
111	021152	001775			BEQ	4\$:BR IF NOT
112	021154	000240			NOP		:IGNORE ANY ERRORS
113	021156	000207			RTS	PC	:EXIT


```

1
2
3
4
5
6
7
8 021160 010446
9 021162 012746 000176
10 021166 011504
11 021170 022714 177777
12 021174 001420
13 021176 023714 001362
14 021202 001010
15 021204 123764 001364 000003
16 021212 001004
17 021214 123764 001344 000002
18 021222 001415
19 021224 005316
20 021226 001413
21 021230 062704 000004
22 021234 000755
23 021236 013714 001362
24 021242 113764 001364 000003
25 021250 113764 001344 000002
26 021256 062706 000002
27 021262 012604
28 021264 062705 000002
29 021270 000205

:THE ROUTINE INSERT:
:INSERT A BAD SECTOR ADDRESS INTO ONE OF THE MFG16, MFG18, USTAB AND
:USSAV TABLES
:CALL
:      JSR      R5,INSERT
:      TABLE+8.

INSERT: MOV      R4,-(SP)      :SAVE THE R4
        MOV      #126,-(SP)  :ENTRIES OF THE TABLE
        MOV      (R5),R4     :TABLE ADDRESS + 8 BYTES
1$:    CMP      #-1,(R4)     :AN OPEN ENTRY ?
        BEQ      3$          :BR IF SO
        CMP      DS.CYL,(R4) :CHECK IF THE BAD SECTOR HAS RECORDED
        BNE      2$          :BR IF NOT
        CMPB     DS.TRK,3(R4) :CYL AND TRK MATCH ?
        BNE      2$          :BR IF NOT
        CMPB     BADSEC,2(R4) :CYL ,TRK AND SEC MATCH ?
        BEQ      4$          :BR IF SO
2$:    DEC      (SP)         :DECREMENT THE ENTRY COUNT
        BEQ      4$          :BR IF EXHAUST
        ADD      #4,R4       :LOCATE THE NEXT ENTRY
        BR       1$         :BR BACK
3$:    MOV      DS.CYL,(R4)  :LOAD INTO TABLE
        MOVB     DS.TRK,3(R4) :LOAD THE TRACK NUMBER
        MOVB     BADSEC,2(R4) :LOAD THE SECTOR NUMBER
4$:    ADD      #2,SP        :CLEAR UP THE STACK
        MOV      (SP)+,R4    :RESTORE R4
        ADD      #2,R5       :ASJUST THE RETURN ADDRESS
        RTS      R5         :EXIT
  
```

```

1      .SBTTL  UTILITY FUNCTION ROUTINES
2
3      ;ROUTINE TO ENTER 'MFG' DEFINED BAD SECTORS IN THE 16 BIT TABLE AND
4      ;THE 18 BIT TABLE
5
6      021272 005737 001320  FUNCT1: TST      MODE      ;IS IT 'FORMAT' MODE ?
7      021276 001403          BEQ      1$          ;BR IF YES
8      021300 104401 042255          TYPE     ,NOUPDT     ;TYPE 'NO UPDATE OF FILE IN VERIFY MODE'
9      021304 000207          RTS      PC
10
11     021306 104401 040564  1$:      TYPE     ,MSG5      ;TYPE 'ENTER MFG 16 BIT BAD SECTOR ADDRESS'
12     021312 104401 001205          TYPE     ,$CRLF      ;CR-LF
13     021316 004577 022702  2$:      JSR      R5,MANTER ;ENTER THE VALUE FROM THE KEYBOARD
14     021322 005702          TABLE2
15
16     021324 005737 001362          TST      DS,CYL      ;WAS A VALUE ENTERED ?
17     021330 100404          BMI      3$          ;BR IF NO
18     021332 004537 021160          JSR      R5,INSERT   ;INSERT BAD SECTOR INTO THE 'MFG16' TABLE
19     021336 001424          MFG16+8.
20     021340 000766          BR       2$          ;NEXT ENTRY
21     021342 004537 020250  3$:      JSR      R5,SORT2  ;SORT THE TABLE WHEN DONE WITH INPUTS
22     021346 001424          MFG16+8.
23
24     021350 104401 040642          TYPE     ,MSG6      ;TYPE 'ENTER MFG 18 BIT BAD SECTOR ADDRESS'
25     021354 104401 001205          TYPE     ,$CRLF      ;CR-LF
26     021360 004537 022702  4$:      JSR      R5,MANTER ;INPUT VALUES FROM THE KEYBOARD
27     021364 005720          TABLE3
28
29     021366 005737 001362          TST      DS,CYL      ;UPDATED VALUES ?
30     021372 100404          BMI      5$          ;BR IF NOT
31     021374 004537 021160          JSR      R5,INSERT   ;INSERT BAD SECTOR INTO THE 'MFG18' TABLE
32     021400 002430          MFG18+8.
33     021402 000766          BR       4$          ;NEXT ENTRY
34     021404 004537 020250  5$:      JSR      R5,SORT2  ;SORT THE TABLE WHEN DONE WITH INPUTS
35     021410 002430          MFG18+8.
36     021412 000207          RTS      PC          ;EXIT
37
38     ;ROUTINE TO ENTER 'USR' DEFINED BAD SECTORS IN THE 16 BIT TABLE OR
39     ;THE 18 BIT TABLE
40
41     021414 005737 001320  FUNCT2: TST      MODE      ;IS IT 'FORMAT' MODE ?
42     021420 001403          BEQ      1$          ;BR IF YES
43     021422 104401 042255          TYPE     ,NOUPDT     ;TYPE 'NO UPDATE OF FILE IN VERIFY MODE'
44     021426 000207          RTS      PC
45
46     021430 012737 003434 021546  1$:      MOV      #USTAB+8.,4$ ;SET UP 'USR' TABLES FOR 16 BIT MODE
47     021436 012737 003434 021556          MOV      #USTAB+8.,6$
48     021444 012737 004440 021610          MOV      #USSAV+8.,8$
49     021452 012737 004440 021620          MOV      #USSAV+8.,10$
50     021460 005737 001356          TST      SEC30      ;IS IT 18 BIT MODE ?
51     021464 100414          BMI      2$          ;BR IS NO
52     021466 012737 003434 021610          MOV      #USTAB+8.,8$ ;SET UP 'USR' TABLES FOR 18 BIT MODE
53     021474 012737 003434 021620          MOV      #USTAB+8.,10$
54     021502 012737 004440 021546          MOV      #USSAV+8.,4$
55     021510 012737 004440 021556          MOV      #USSAV+8.,6$
56
57     021516 104401 040720  2$:      TYPE     ,MSG7      ;TYPE 'ENTER USR 16 BIT BAD SECTOR ADDRESS'

```



```

58 021522 104401 001205
59 021526 004537 022702
60 021532 005702
61
62 021534 005737 001362
63 021540 100404
64 021542 004537 021160
65 021546 000000
66 021550 000766
67 021552 004537 020250
68 021556 000000
69
70 021560 104401 040776
71 021564 104401 001205
72 021570 004537 022702
73 021574 005720
74
75 021576 005737 001362
76 021602 100404
77 021604 004537 021160
78 021610 000000
79 021612 000766
80 021614 004537 020250
81 021620 000000
82 021622 000207
83
84
85
86
87 021624 104401 001205
88 021630 004737 022622
89 021634 104401 001205
90 021640 010146
91 021642 012701 001424
92 021646 104401 041054
93 021652 004737 022512
94
95 021656 104401 041140
96 021662 012701 002430
97 021666 004737 022512
98 021672 012601
99 021674 000207
100
101
102
103
104 021676 104401 001205
105 021702 004737 022622
106 021706 104401 001205
107 021712 010146
108 021714 010246
109 021716 012701 003434
110 021722 012702 004440
111 021726 005737 001356
112 021732 100404
113 021734 012702 003434
114 021740 012701 004440

```

```

3$: TYPE $CRLF ;CR-LF
JSR R5,MANTER ;ENTER BAD SECTOR FROM THE KEYBOARD
TABLE2

TST DS,CYL ;UPDATED INPUT VALUE ?
BMI 5$ ;BR IF NO
JSR R5,INSERT ;INSERT THE BAD SECTOR INTO TABLE
4$: .WORD 0 ;TABLE ADDRESS
BR 3$ ;NEXT ENTRY
5$: JSR R5,SORT2 ;SORT THE 'USR' 16 BIT TABLE
6$: .WORD 0 ;TABLE ADDRESS

TYPE ,MSG8 ;TYPE 'ENTER USR 18 BIT BAD SECTOR ADDRESS'
TYPE $CRLF ;CR-LF
7$: JSR R5,MANTER ;ENTER BAD SECTOR FORM KEYBOARD
TABLE3

TST DS,CYL ;UPDATED INPUT VALUE ?
BMI 9$ ;BR IF NOT
JSR R5,INSERT ;INSERT THE BAD SECTOR INTO TABLE
8$: .WORD 0 ;
BR 7$ ;BR TO ENTER NEXT SECTOR
9$: JSR R5,SORT2 ;SORT THE TABLE
10$: .WORD 0 ;TABLE ADDRESS
RTS PC ;EXIT FORM THE TABCAL CALLING

;THIS ROUTINE WILL TYPE THE CONTENTS OF THE 'MFG' BAD SECTOR FILES
;FOR 16 BIT MODE AND 18 BIT MODE
FUNCT3: TYPE $CRLF ;CR-LF
JSR PC,TYPACK ;TYPE PACK SERIAL NUMBER
TYPE $CRLF ;CR-LF
MOV R1,-(SP) ;SAVE R1
MOV #MFG16+8.,R1 ;ADDRESS OF FIRST BAD SECTOR ENTRY IN TABLE
TYPE ,MSG9 ;TYPE 'CONTENTS OF 16 BIT MFG BAD SECTOR FILE'
JSR PC,TYLIST ;TYPE BAD SECTOR LIST IN DECIMAL

TYPE ,MSG10 ;TYPE 'CONTENTS OF 18 BIT MFG BAD SECTOR FILE'
MOV #MFG18+8.,R1 ;ADDRESS OF FIRST BAD SECTOR ENTRY IN TABLE
JSR PC,TYLIST ;TYPE BAD SECTOR LIST IN DECIMAL
MOV (SP)+,R1 ;RESTORE THE R1
RTS PC ;EXIT FROM THE CALLING OF TABCAL

;THIS ROUTINE WILL TYPE THE CONTENTS OF THE 'USR' BAD SECTOR FILES
;FOR 16 BIT MODE AND 18 BIT MODE
FUNCT4: TYPE $CRLF ;CR-LF
JSR PC,TYPACK ;TYPE PACK SERIAL NUMBER
TYPE $CRLF ;CR-LF
MOV R1,-(SP) ;SAVE R1
MOV R2,-(SP) ;SAVE R2
MOV #USTAB+8.,R1 ;SET UP 'USR' TABLES FOR 16 BIT MODE
MOV #USSAV+8.,R2
TST SEC30 ;IS IT 16 BIT MODE ?
BMI 1$ ;BR IF YES
MOV #USTAB+8.,R2 ;SET UP 'USR' TABLES FOR 18 BIT MODE
MOV #USSAV+8.,R1 ;

```

```
115 021744 104401 041224      1$:      TYPE      ,MSG11      ;TYPE 'CONTENTS OF 16 BIT USR BAD SECTOR FILE'
116 021750 004737 022512      JSR      PC,TYLIST      ;TYPE BAD SECTOR LIST IN DECIMAL
117
118 021754 104401 041310      TYPE      ,MSG12      ;TYPE 'CONTENTS OF 18 BIT USR BAD SECTOR FILE'
119 021760 010201      MOV      R2,R1      ;GET ADDRESS TO R1
120 021762 004737 022512      JSR      PC,TYLIST      ;TYPE BAD SECTOR LIST IN DECIMAL
121 021766 012602      MOV      (SP)+,R2      ;RESTORE R2
122 021770 012601      MOV      (SP)+,R1      ;RESTORE R1
123 021772 000207      RTS      PC      ;EXIT FROM THE CALLING OF TABCAL
124
125      ;THIS ROUTINE IS USED TO INITIALIZE THE 'MFG' BAD SECTOR FILE (16 AND 18 BIT)
126
127 021774 005737 001320      FUNCT5: TST      MODE      ;IS IT 'FORMAT' MODE ?
128 022000 001403      BEQ      1$      ;BR IF YES
129 022002 104401 042314      TYPE      ,NOINIT      ;TYPE 'NO INITIALIZATION IN VERIFY MODE'
130 022006 000207      RTS      PC
131
132 022010 104401 042143      1$:      TYPE      ,INITMF      ;TYPE 'INITIALIZING MFG BAD SECTOR FILES'
133 022014 010146      MOV      R1,-(SP)      ;SAVE R1
134 022016 010246      MOV      R2,-(SP)      ;SAVE R2
135 022020 005037 001420      CLR      MFG16+4.      ;CLEAR 3RD WORD (ALL 0'S)
136 022024 005037 001422      CLR      MFG16+6.      ;CLEAR 4TH WORD (ALL 0'S)
137 022030 012701 001424      MOV      #MFG16+8.,R1      ;ADDRESS OF FIRST BAD SECTOR ENTRY
138 022034 012702 000374      MOV      #252.,R2      ;NUMBER OF WORDS LEFT
139 022040 012721 177777      2$:      MOV      #-1,(R1)+      ;RESET REST OF FILE TO -1
140 022044 005302      DEC      R2      ;DONE YET ?
141 022046 001374      BNE      2$      ;BR IF NO
142
143 022050 005037 002424      CLR      MFG18+4.      ;CLEAR 3RD WORD (ALL 0'S)
144 022054 005037 002426      CLR      MFG18+6.      ;CLEAR 4TH WORD (ALL 0'S)
145 022060 012701 002430      MOV      #MFG18+8.,R1      ;ADDRESS OF FIRST BAD SECTOR ENTRY
146 022064 012702 000374      MOV      #252.,R2      ;NUMBER OF WORDS LEFT
147 022070 012721 177777      3$:      MOV      #-1,(R1)+      ;RESET REST OF FILE TO -1
148 022074 005302      DEC      R2      ;DONE YET ?
149 022076 001374      BNE      3$      ;BR IF NO
150 022100 012602      MOV      (SP)+,R2      ;RESTORE R2
151 022102 012601      MOV      (SP)+,R1      ;RESTORE R1
152 022104 104401 041502      TYPE      ,MSG14      ;TYPE 'DONE !'
153 022110 000207      RTS      PC      ;EXIT FROM THE TABCAL CALLING
154
155      ;THIS ROUTINE IS USED TO INITIALIZE THE 'USR' BAD SECTOR FILE (16 AND 18 BIT)
156
157 022112 005737 001320      FUNCT6: TST      MODE      ;IS IT 'FORMAT' MODE ?
158 022116 001403      BEQ      1$      ;BR IF YES
159 022120 104401 042314      TYPE      ,NOINIT      ;TYPE 'NO INITIALIZATION IN VERIFY MODE'
160 022124 000207      RTS      PC
161
162 022126 104401 042100      1$:      TYPE      ,INITUS      ;TYPE 'INITIALIZING USR BAD SECTOR FILES'
163 022132 010146      MOV      R1,-(SP)      ;SAVE R1
164 022134 010246      MOV      R2,-(SP)      ;SAVE R2
165 022136 005037 003430      CLR      USTAB+4.      ;CLEAR 3RD WORD (ALL 0'S)
166 022142 005037 003432      CLR      USTAB+6.      ;CLEAR 4TH WORD (ALL 0'S)
167 022146 012701 003434      MOV      #USTAB+8.,R1      ;ADDRESS OF FIRST BAD SECTOR ENTRY
168 022152 012702 000374      MOV      #252.,R2      ;NUMBER OF WORDS LEFT
169 022156 012721 177777      2$:      MOV      #-1,(R1)+      ;RESET REST OF FILE TO -1
170 022162 005302      DEC      R2      ;DONE YET ?
171 022164 001374      BNE      2$      ;BR IF NO
```



```

172
173 022166 005037 004434          CLR    USSAV+4.      ;CLEAR 3RD WORD (ALL 0'S)
174 022172 005037 004436          CLR    USSAV+6.      ;CLEAR 4TH WORD (ALL 0'S)
175 022176 012701 004440          MOV    #USSAV+8.,R1  ;ADDRESS OF FIRST BAD SECTOR ENTRY
176 022202 012702 000374          MOV    #252.,R2     ;NUMBER OF WORDS LEFT
177 022206 012721 177777          3$:   MOV    #-1,(R1)+ ;RESET REST OF FILE TO -1
178 022212 005302                   DEC    R2           ;DONE YET ?
179 022214 001374                   BNE    3$          ;BR IF NO
180 022216 012602                   MOV    (SP)+,R2    ;RESTORE R2
181 022220 012601                   MOV    (SP)+,R1    ;RESTORE R1
182 022222 104401 041502          TYPE   ,MESG14     ;TYPE 'DONE !'
183 022226 000207                   RTS    PC          ;EXIT FORM THE TABCAL CALLING
184
185                                     ;THIS ROUTINE IS USED TO PRINT A SECTOR OF THE LAST TRACK (BAD SECTOR FILE)
186                                     ;CYL=822., (RM05-TRK=18. OR RM02/3-TRK=4)
187
188 022230 104401 041374          FUNCT7: TYPE   ,MESG13 ;TYPE 'PRINT A SECTOR OF LAST TRACK (OCTAL)'
189 022234 104401 041450          1$:   TYPE   ,MSG13A  ;TYPE 'ENTER SECTOR (DECIMAL): '
190 022240 104411                   RDLIN              ;READ THE SECTOR NUMBER
191 022242 012601                   MOV    (SP)+,R1    ;ADDRESS OF READ-IN BUFFER
192 022244 012702 000037          MOV    #31.,R2     ;UPPER LIMIT OF INPUT
193 022250 004537 023172          JSR    R5,CK.DIG  ;CHECK THE DIGIT(S)
194 022254 022342                   6$              ;CARRIAGE RETURN ONLY ENTERED
195 022256 022334                   5$              ;PERIOD ONLY ENTERED
196 022260 021334                   5$              ;ILLEGAL INPUT
197 022262 022270                   2$              ;TERMINATED WITH A CARRIAGE RETURN
198 022264 022334                   5$              ;TERMINATED WITH A '...'
199 022266 022334                   5$              ;TERMINATED WITH A '...'
200 022270 006302          2$:   ASL    R2           ;FOUND THE WORD INDEX
201 022272 016201 006064          MOV    ADRTBL(R2),R1 ;ADDRESS OF THE BUFFER
202 022276 012702 000402          MOV    #258.,R2    ;WC FOR WHOLE SECTOR INCLUDING HEADER
203 022302 104401 001205          3$:   TYPE   ,$CRLF    ;CR-LF
204 022306 012705 000010          MOV    #8.,R5      ;8 WORDS ON A LINE
205 022312 012146          4$:   MOV    (R1)+,-(SP) ;CONTENTS OF THE BUFFER
206 022314 104402                   TYPOC            ;TYPE OCTAL WORD
207 022316 104401 036621          TYPE   ,BLNKS2    ;TYPE 2 SPACES
208 022322 005302                   DEC    R2         ;DECREMENT WORD CTR
209 022324 001406                   BEQ    6$         ;BR IF FINISH
210 022326 005305                   DEC    R5         ;START ANOTHER LINE ?
211 022330 001370                   BNE    4$        ;BR IF NOT
212 022332 000763                   BR     3$        ;BR IF SO
213 022334 104401 037174          5$:   TYPE   ,BADENT  ;TYPE BAD ENTRY
214 022340 000735                   BR     1$        ;TRY AGAIN
215 022342 104401 001205          6$:   TYPE   , $CRLF  ;CR-LF
216 022346 000207                   RTS    PC        ;EXIT FROM TABCAL CALLING
217
218                                     ;ROUTINE TO ENTER 'MFG' DEFINED BAD SECTORS IN THE 16 BIT TABLE AND
219                                     ;THE 18 BIT TABLE. THE SECTOR ADDRESS IS ENTERED AS A BYTE LOCATION
220                                     ;FROM INDEX ON THE DISK
221
222 022350 104401 041513          FUNCT8: TYPE   ,MESG15 ;TYPE 'ENTER MFG BAD SECTOR ADRS (DECIMAL)'
223 022354 004537 022702          1$:   JSR    R5,MANTER ;ENTER THE VALUE FROM KEYBOARD
224 022360 005736                   TABLE4
225
226 022362 005737 001362          TST    DS.CYL     ;WAS ADDRESS UPDATED ?
227 022366 100442                   BMI    6$         ;BR IF NO
228 022370 010446                   MOV    R4,-(SP)   ;SAVE R4

```

```
222 022372 013746 001344      MOV    BADSEC,-(SP)      :SAVE 'BADSEC' FOR 18 BIT MODE
223 022376 005004              CLR    R4                :TABLE INDEX STARTS FROM 0
224 022400 023764 001344 006264 2$:  CMP    BADSEC,BYTE16(R4) :IS 16 BIT LOCATION FOUND ?
225 022406 101402              BLOS  3$                 :BR IF YES
226 022410 005724              TST   (R4)+              :TRY NEXT SECTOR
227 022412 000772              BR    2$                 :LOOPING
228 022414 006204              3$:  ASR    R4                :SECTOR ADDRESS
229 022416 010437 001344      MOV    R4,BADSEC        :SAVE 16 BIT SECTOR ADDRESS
230 022422 004537 021160      JSR   R5,INSERT        :INSERT BAD SECTOR INTO 'MFG16' TABLE
231 022426 001424              MFG16+8.
232 022430 012637 001344      MOV    (SP)+,BADSEC     :GET LOCATION OF SECTOR FOR 18 BIT MODE
233 022434 005004              CLR    R4                :TABLE INDEX STARTS FROM 0
234 022436 023764 001344 006364 4$:  CMP    BADSEC,BYTE18(R4) :IS 18 BIT LOCATION FOUND ?
235 022444 101403              BLOS  5$                 :BR IF YES
236 022446 062704 000002      ADD   #2,R4             :TRY NEXT SECTOR IF NOT FOUND
237 022452 000771              BR    4$                 :LOOPIN BACK
238 022454 006204              5$:  ASR    R4                :SECTOR ADDRESS
239 022456 010437 001344      MOV    R4,BADSEC        :SAVE 18 BIT SECTOR ADDRESS
240 022462 004537 021160      JSR   R5,INSERT        :INSERT BAD SECTOR INTO 'MFG18' TABLE
241 022466 002430              MFG18+8.
242 022470 012604              MOV    (SP)+,R4         :RESTORE R4
243 022472 000730              BR    1$                 :NEXT ENTRY
244 022474 004537 020250      6$:  JSR   R5,SORT2          :SORT 'MFG16' TABLE
245 022500 001424              MFG16+8.
246 022502 004537 020250      JSR   R5,SORT2          :SORT 'MFG18' TABLE
247 022506 002430              MFG18+8.
248 022510 000207              RTS    PC                :RETURN
```


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

: THIS ROUTINE IS USED TO TYPE THE BAD SECTOR FILE ENTRY IN DECIMAL.
: UPON RETURN, R1 WILL BE POINTING TO THE NEXT BAD SECTOR (CYLINDER)
: ENTRY.

: CALL
: MOV #NUMADR,R1 : ADDRESS OF FIRST TABLE ENTRY
: JSR PC,TYLIST : CALL TYPE LIST ROUTINE

: THE LIST WILL BE TYPED IN THE FOLLOWING FORMAT:

: CONTENTS OF 'FILENAME' THE BAD SECTOR FILE (DECIMAL)
: CYL,TRK,SEC

13 022512 010146
14 022514 104401 001205
15 022520 005711
16 022522 100432
17 022524 011146
18 022526 004737 027100
19 022532 004737 026670
20 022536 104401 036602
21 022542 105046
22 022544 116116 000003
23 022550 004737 027100
24 022554 004737 026670
25 022560 104401 036602
26 022564 105046
27 022566 116116 000002
28 022572 004737 027100
29 022576 004737 026670
30 022602 062701 000004
31 022606 000742
33 022610 020126
34 022612 001002
35 022614 104401 042206
36 022620 000207

TYLIST: MOV R1,-(SP) : SAVE ADDRESS OF FIRST TABLE ENTRY
1\$: TYPE ,\$CRLF : CR-LF
 TST (R1) : IS IT TERMINATOR ?
 BMI 2\$: BR IF YES
 MOV (R1),-(SP) : PUT CYLINDER NUMBER ON STACK
 JSR PC,\$SB2D : CONVERT IT TO DECIMAL
 JSR PC,\$SUPRL : TYPE DECIMAL NUMBER, LEFT JUSTIFY
 TYPE ,\$COMMA : TYPE ','
 CLRB -(SP) : CLEAR STACK LOCATION
 MOVB 3(R1),(SP) : PUT TRACK NUMBER ON STACK
 JSR PC,\$SB2D : CONVERT IT TO DECIMAL
 JSR PC,\$SUPRL : TYPE DECIMAL NUMBER, LEFT JUSTIFY
 TYPE ,\$COMMA : TYPE ','
 CLRB -(SP) : CLEAR STACK LOCATION
 MOVB 2(R1),(SP) : PUT SECTOR NUMBER ON STACK
 JSR PC,\$SB2D : CONVERT IT TO DECIMAL
 JSR PC,\$SUPRL : TYPE DECIMAL NUMBER, LEFT JUSTIFY
 ADD #4,R1 : ADJUST R1 TO NEXT BAD SECTOR ENTRY
 BR 1\$: NEXT ENTRY

2\$: CMP R1,(SP)+ : WERE THERE ANY ENTRIES ?
 BNE 3\$: BR IF YES
 TYPE ,\$NOENTR : TYPE '* NO ENTRIES *'
3\$: RTS PC : RETURN

1
2
3
4
5
6
7
8
9 022622 104401 037473
10 022626 013746 003426
11 022632 001414
12 022634 004737 027330
13 022640 004737 026670
14 022644 013746 003424
15 022650 004737 027330
16 022654 004537 026770
17 022660 000005
18 022662 000406
19 022664 013746 003424
20 022670 004737 027330
21 022674 004737 026670
22 022700 000207

```
.SBTTL TYPE PACK SERIAL NUMBER IN OCTAL
;ROUTINE TO TYPE THE PACK SERIAL NUMBER IN OCTAL
;CALL:
;   JSR     PC,TYPACK      ;CALL ROUTINE
;   OR
;   JSR     PC,TYPCK      ;CALL ROUTINE(WITH NO HEADER MESSAGE)
TYPACK: TYPE      ,PACKSN      ;TYPE 'PACK S/N:'
TYPCK:  MOV      USTAB+2,-(SP) ;GET HI NUMBER (MSB)
        BEQ      1$           ;BR IF ZERO
        JSR      PC,$SB20     ;CONVERT TO OCTAL NUMBER
        JSR      PC,$SUPRL    ;AND TYPE IT LEFT JUSTIFIED
        MOV      USTAB,-(SP)  ;GET LOW NUMBER (LSB)
        JSR      PC,$SB20     ;CONVERT TO OCTAL NUMBER
        JSR      R5,FILLZ     ;TYPE WITH PRECEEDING ZEROS
        .WORD    5            ;5 DIGITS
        BR       2$
1$:     MOV      USTAB,-(SP)  ;GET LOW NUMBER (LSB)
        JSR      PC,$SB20     ;CONVERT TO OCTAL NUMBER
        JSR      PC,$SUPRL    ;AND TYPE IT LEFT JUSTIFIED
2$:     RTS      PC           ;RETURN
```


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

.SBTTL ENTER BAD SECTOR ROUTINE

:ROUTINE TO ENTER BAD SECTOR INFORMATION MANUALLY

:CALL:

: JSR R5,MANTER :CALL ROUTINE
 : TABLE :ADDRESS OF TABLE

MANTER:

```

1$:  MOV R4, -(SP)      ;;PUSH R4 ON STACK
      MOV (R5), R4     ;;GET TABLE POINTER
      MOV (R4)+, 2$    ;;GET ADDRESS OF MESSAGE
      BEQ 3$           ;;BR IF NO MESSAGE
      TYPE
2$:  .WORD 0           ;;ADDRESS OF MESSAGE GOES HERE
3$:  MOV #-1, @2(R4)   ;;INITIALIZE VALUE
      RDLIN .
      MOV (SP)+, R1    ;;ADDRESS OF FIRST ASCII CHARACTER
      MOV (R4)+, R2    ;;UPPER LIMIT OF INPUT
      JSR R5, CK.DIG  ;;CHECK THE DIGIT(S)
      8$              ;;CARRIAGE RETURN ONLY ENTERED
      7$              ;;PERIOD ONLY ENTERED
      7$              ;;ILLEGAL INPUT
      7$              ;;TERMINATED WITH A CARRIAGE RETURN
      4$              ;;TERMINATED WITH A '...'
      7$              ;;TERMINATED WITH A '...'
4$:  MOV R2, @2(R4)+  ;;CYLINDER ADDRESS
      MOV (R4)+, R2    ;;UPPER LIMIT OF INPUT
      JSR R5, CK.DIG  ;;CHECK THE DIGIT(S)
      7$              ;;CARRIAGE RETURN ONLY ENTERED
      7$              ;;PERIOD ONLY ENTERED
      7$              ;;ILLEGAL INPUT
      7$              ;;TERMINATED WITH A CARRIAGE RETURN
      5$              ;;TERMINATED WITH A '...'
      7$              ;;TERMINATED WITH A '...'
5$:  MOV R2, @2(R4)+  ;;TRACK ADDRESS
      MOV (R4)+, R2    ;;UPPER LIMIT OF INPUT
      JSR R5, CK.DIG  ;;CHECK THE DIGIT(S)
      7$              ;;CARRIAGE RETURN ONLY ENTERED
      7$              ;;PERIOD ONLY ENTERED
      7$              ;;ILLEGAL INPUT
      6$              ;;TERMINATED WITH A CARRIAGE RETURN
      7$              ;;TERMINATED WITH A '...'
      7$              ;;TERMINATED WITH A '...'
6$:  MOV R2, @2(R4)+  ;;SECTOR OR BYTE ADDRESS
      BR 8$           ;;EXIT
7$:  TYPE ,BADENT    ;;MESSAGE BAD ENTRY
      BR 1$          ;;ENTER SECTOR ADDRESS AGAIN
8$:  TST (R5)+       ;;ADJUST RETURN ADDRESS
      MOV (SP)+, R4  ;;POP STACK INTO R4
      RTS R5         ;;EXIT
  
```

```

1
2
3
4
5
6
7
8
9
10 023044 121127 000060
11 023050 103407
12 023052 121127 000067
13 023056 101004
14 023060 111102
15 023062 042702 177770
16 023066 005725
17 023070 000205
18
19
20
21
22
23
24
25
26
27
28 023072 121127 000060
29 023076 103407
30 023100 121127 000071
31 023104 101004
32 023106 111102
33 023110 042702 000060
34 023114 005725
35 023116 000205
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50 023120 105711
51 023122 001417
52 023124 121127 000054
53 023130 001413
54 023132 121127 000056
55 023136 001407
56 023140 004537 023072
57 023144 000410

;THIS ROUTINE IS USED TO CHECK IF AN
;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
;CALL
;      MOV      #ADR,R1      ;ADDRESS OF ASCII CHARACTER
;      JSR      R5,CK.OCT    ;CHECK THE CHARACTER
;      RETURN1  ;CHARACTER IS NOT BETWEEN 0-7
;      RETURN2  ;CHARACTER IS IN R2 AS A
;              ;OCTAL DIGIT
;
CK.OCT: CMPB    (R1),#'0      ;LESS THAN ZERO?
        BLO     1$           ;YES
        CMPB    (R1),#'7      ;GREATER THAN SEVEN?
        BHI     1$           ;YES
        MOVB    (R1),R2       ;GET THE CHARACTER
        BIC     #'C7,R2       ;STRIP AWAY THE ASCII
        TST     (R5)+         ;ADJUST FOR RETURN
1$:     RTS      R5           ;RETURN

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

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

```



```

58 023146 004537 023044      JSR    R5,CK.OCT      :OCTAL ?
59 023152 005725              TST    (R5)+          :DIGIT BETWEEN 8-9
60 023154 005725              TST    (R5)+          :DIGIT BETWEEN 0-7
61 023156 005725      1$:   TST    (R5)+          :PERIOD
62 023160 005725      2$:   TST    (R5)+          :COMMA
63 023162 005725      3$:   TST    (R5)+          :CARRIAGE RETURN
64 023164 005201              INC    R1              :MOVE POINTER TO NEXT CHARACTER
65 023166 011505      4$:   MOV    (R5),R5      :UNKNOWN CHARACTER
66 023170 000205              RTS     R5              :RETURN
67
68                          ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
69                          ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
70                          ;CALL
71                          ;
72                          ;
73                          ;
74                          ;
75                          ;
76                          ;
77                          ;
78                          ;
79                          ;
80
81 023172 010446      CK.DIG: MOV    R4,-(SP)      :SAVE R4
82 023174 010346              MOV    R3,-(SP)      :SAVE R3
83 023176 010246              MOV    R2,-(SP)      :SAVE THE MAX. SIZE ON THE STACK
84 023200 005002              CLR    R2              :START WITH 0
85 023202 005003              CLR    R3
86 023204 005004              CLR    R4
87 023206 004537 023120      JSR    R5,CK.CHR      :CHECK ONE CHARACTER
      023212 023306      6$:   :ILLEGAL CHARACTER
      023214 023314      9$:   :CARRIAGE RETURN
      023216 023306      6$:   :
      023220 023310      7$:   :
      023222 023226      1$:   :DIGIT 0-7
      023224 023226      1$:   :DIGIT 8-9
88 023226 062705 000004      1$:   ADD    #4,R5          :STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
89 023232 006303      2$:   ASL    R3              :INPUT NUMBER *2
90 023234 010346              MOV    R3,-(SP)      :SAVE *2
91 023236 006303              ASL    R3              :*4
92 023240 006303              ASL    R3              :*8
93 023242 062603              ADD    (SP)+,R3      :(*2)+(*8) = *10
94 023244 060203              ADD    R2,R3          :UPDATE THE INPUT NUMBER
95 023246 004537 023120      JSR    R5,CK.CHR      :CHECK ONE CHARACTER
      023252 023312      8$:   :ILLEGAL CHARACTER
      023254 023276      5$:   :CARRIAGE RETURN
      023256 023274      4$:   :
      023260 023266      3$:   :
      023262 023232      2$:   :DIGIT 0-7
      023264 023232      2$:   :DIGIT 8-9
96 023266 105711      3$:   TSTB   (R1)          :DOES A 'CR' FOLLOW THE 'PERIOD'
97 023270 001010              BNE   8$              :BR IF NOT
98 023272 005724              TST   (R4)+          :INCREMENT THE RETURN
99 023274 005724      4$:   TST   (R4)+          :INCREMENT THE RETURN
100 023276 005724      5$:   TST   (R4)+          :INCREMENT THE RETURN
101 023300 020316              CMP   R3,(SP)        :CHECK THE MAGNITUDE OF THE NUMBER
102 023302 003004              BGT   9$              :BR IF ENTERED NUMBER TOO LARGE

```


103 023304 000402
104 023306 005725
105 023310 005725
106 023312 060405
107 023314 010302
108 023316 005726
109 023320 012603
110 023322 012604
111 023324 011505
112 023326 000205
113

6\$: BR 8\$
7\$: TST (R5)+
8\$: TST (R5)+
9\$: ADD R4,R5
MOV R3,R2
TST (SP)+
MOV (SP)+,R3
MOV (SP)+,R4
MOV (R5),R5
RTS R5

:BYPASS INCREMENT
:INCREMENT RETURN PAST INVALID RETURN
:INCREMENT RETURN
:SETUP RETURN POINTER
:ENTERED VALUE
:CLEAN MAX. SIZE OFF OF STACK
:RESTORE R3
:RESTORE R4
:GET RETURN ADDRESS
:RETURN

.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 TYPERR ON ERROR
:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
:*SW15=1      HALT ON ERROR
:*SW13=1      INHIBIT ERROR TYPEOUTS
:*SW10=1      BELL ON ERROR
:*SW09=1      LOOP ON ERROR
:*CALL
:*          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
  
```

```

023330 105037 023756 $ERROR: CLR B IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
023334 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
023336 010137 001366 MOV R1,DDRIVE          ;;DRIVE ADDRESS IF DRIVE ERROR CALL
023342 010337 001370 MOV R3,ATTN           ;;ATTENTION REGISTER CONTENTS
023346 013737 005552 001362 MOV FMTDPB+12,DS.CYL  ;;CURRENT CYLINDER ADDRESS
023354 113737 005551 001364 MOV FMTDPB+11,DS.TRK  ;;REQUESTED TRACK ADDRESS
023362 005737 005564          TST RM.REG+RMBA          ;;NON-ZERO BUFFER ADDRESS ?
023366 001406          BEQ 7$          ;;BR IF NO BUFFER ADDRESS
023370 013746 005564          MOV RM.REG+RMBA,-(SP)  ;;BUFFER ADDRESS
023374 162716 000002          SUB #2,(SP)          ;;DECREMENT THE ADDRESS
023400 013637 001140          MOV @$(SP)+,$GDDAT      ;;GET THE BUFFER WORD WHICH DIDN'T COMPARE
023404 105237 001117 7$: INCB $ERFLG          ;;SET THE ERROR FLAG
023410 001775          BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
023412 013777 001116 155536 MOV $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
023420 032777 002000 155526 BIT #BIT10,@SWR      ;;BELL ON ERROR?
023426 001402          BEQ 1$          ;;NO - SKIP
023430 104401 001200          TYPE $BELL          ;;RING BELL
023434 005237 001126 1$: INC $ERTTL          ;;COUNT THE NUMBER OF ERRORS
023440 011637 001132          MOV (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
023444 162737 000002 001132 SUB #2,$ERRPC
023452 117737 155454 001130 MOV B @$(ERRPC,$ITEMB) ;;STRIP AND SAVE THE ERROR ITEM CODE
023460 032777 001000 155466 BIT #BIT09,@SWR      ;;SEE IF LOOP ON ERROR IS SET
023466 001060          BNE 1004$          ;;BRANCH AROUND ROUTINE IF SO
023470 122737 000177 001130 CMPB #177,$ITEMB     ;;SEE IF THIS IS THE POWER FAIL CALL
023476 001454          BEQ 1004$          ;;BRANCH AROUND ROUTINE IF IT IS
023500 105737 023756          TSTB IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
023504 001047          BNE 1003$          ;;BRANCH IF SO
023506 022737 177777 023754 CMP #-1,CPSAVE       ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
023514 001445          BEQ 1004$          ;;BRANCH IF SO
023516 013746 000004          MOV ERRVEC,-(SP)      ;;SAVE CONTENTS OF ERROR VECTOR
023522 012737 023540 000004 MOV #1000$,ERRVEC    ;;SETUP 'TRAP' RETURN ADDRESS
023530 013737 177766 023754 MOV 177766,CPSAVE    ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
023536 000406          BR 1001$
023540 012737 177777 023754 1000$: MOV #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
023546 012716 023554          MOV #1001$,(SP)      ;;SETUP RETURN ADDRESS
023552 000002          RTI
023554 012637 000004 1001$: MOV (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

023560 022737 177777 023754 1002$: CMP #-1,CPSAVE      ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
023566 001420          BEQ 1004$          ;;BRANCH IF SO
023570 032737 000001 023754 BIT #BIT00,CPSAVE    ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
023576 001414          BEQ 1004$          ;;BRANCH IF OK
023600 042737 000001 177766 BIC #BIT00,177766   ;;CLEAR THE BIT FOUND SET
  
```

```

023606 113737 001130 023756      MOVB  $ITEMB,IBSAVE  ::MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
023614 112737 000177 001130      MOVB  #177,$ITEMB   ::SET $ITEMB TO SPECIAL POWER FAIL POINTER
023622 000402                BR    1004$         ::BRANCH OVER IBSAVE CLEARING

023624 105037 023756                1003$: CLRB  IBSAVE      ::CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
023630                1004$:
023630 032777 020000 155316      BIT   #BIT13,@SWR   ::SKIP TYPEOUT IF SET
023636 001004                BNE   20$           ::SKIP TYPEOUTS
023640 004737 023760                JSR   PC,TYPERR    ::GO TO USER ERROR ROUTINE
023644 104401 001205                TYPE  ,SCLRF
023650                20$:
023650 122737 000001 001230      CMPB  #APTENV,$ENV  ::RUNNING IN APT MODE
023656 001007                BNE   2$           ::NO,SKIP APT ERROR REPORT
023660 113737 001130 023672      MOVB  $ITEMB,21$    ::SET ITEM NUMBER AS ERROR NUMBER
023666 004737 027522                JSR   PC,$ATY4     ::REPORT FATAL ERROR TO APT
023672 000                21$:  .BYTE  0
023673 000                .BYTE  0
023674 000777                22$:  BR    22$         ::APT ERROR LOOP
023676 105737 023756                2$:   TSTB  IBSAVE     ::SEE IF IBSAVE IS LOADED
023702 001005                BNE   3$           ::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
023704 005777 155244                TST   @SWR         ::HALT ON ERROR
023710 100002                BPL   3$           ::SKIP IF CONTINUE
023712 000000                HALT                    ::HALT ON ERROR!
023714 104407                CKSWR
023716                3$:
023716 032777 001000 155230      BIT   #BIT09,@SWR  ::LOOP ON ERROR SWITCH SET?
023724 001402                BEQ   4$           ::BR IF NO
023726 013716 001124                MOV   $LPERR,(SP)  ::FUDGE RETURN FOR LOOPING
023732 005737 001176                4$:  TST   $ESCAPE     ::CHECK FOR AN ESCAPE ADDRESS
023736 001402                BEQ   5$           ::BR IF NONE
023740 013716 001176                MOV   $ESCAPE,(SP) ::FUDGE RETURN ADDRESS FOR ESCAPE
023744                5$:
023744 105737 023756                TSTB  IBSAVE     ::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
023750 001215                BNE   7$           ::BRANCH BACK TO CALL ORIGINAL ERROR
023752 000002                RTI                    ::RETURN
023754 000000                CPSAVE: .WORD  0    ::LOCATION TO SAVE CPU ERROR REG CONTENTS
023756 000000                IBSAVE: .WORD  0    ::LOCATION TO SAVE ITEM BYTE

```


: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.

2							
3							
4							
5							
6	023760	104412					
7	023762	005000					
8	023764	113700	001130				
9	023770	122700	000177				
10	023774	001006					
14	023776	013737	001214	024374			
16	024004	012700	024234				
17	024010	000407					
18	024012	010001					
19	024014	005300					
20	024016	006300					
21	024020	006300					
22	024022	006300					
23	024024	062700	006460				
24	024030	012037	024044				
25	024034	001404					
26	024036	104401	001205				
27	024042	104401					
28	024044	000000					
29	024046	012037	024062				
30	024052	001404					
31	024054	104401	001205				
32	024060	104401					
33	024062	000000					
34	024064	012001					
35	024066	001460					
36	024070	005005					
37	024072	012000					
38	024074	012002					
39	024076	001451					
40	024100	005105					
41	024102	104401	001205				
42	024106	112003					
43	024110	112004					
44	024112	006004					
45	024114	103403					
46	024116	013146					
	024120	104402					
47	024122	000402					
48	024124						
	024124	013146					
	024126	104405					
49	024130	005303					
50	024132	001403					
51	024134	104401	036621				
52	024140	000764					
53	024142	005302					
54	024144	003431					
55	024146	104401	001205				
56	024152	005760	000002				
57	024156	001404					
58	024160	005105					

```

TYPERR: SAVREG
          CLR      R0
          MOV      $ITEMB,R0
          CMP      #177,R0
          BNE     1$
          MOV      $TESTN,PFTSTN
          MOV      #PFECB,R0
          BR      3$
1$:      MOV      R0,R1
          DEC      R0
          ASL      R0
          ASL      R0
          ASL      R0
2$:      ADD      #ERRTB,R0
3$:      MOV      (R0)+,4$
          BEQ     5$
          TYPE    ,SCLRF
          TYPE
4$:      .WORD    0
5$:      MOV      (R0)+,6$
          BEQ     7$
          TYPE    ,SCLRF
          TYPE
6$:      .WORD    0
7$:      MOV      (R0)+,R1
          BEQ     17$
          CLR      R5
          MOV      (R0)+,R0
          MOV      (R0)+,R2
          BEQ     16$
          COM     R5
          TYPE    ,SCLRF
8$:      MOV      (R0)+,R3
          MOV      (R0)+,R4
9$:      ROR      R4
          BCS     10$
          MOV     @R1+,-(SP)
          TYPOC
          BR      11$
10$:     MOV      @R1+,-(SP)
          TYPDS
11$:     DEC      R3
          BEQ     12$
          TYPE    ,BLNKS2
          BR      9$
12$:     DEC      R2
          BLE     17$
          TYPE    ,SCLRF
          TST     2(R0)
          BEQ     13$
          COM     R5
          ;SAVE R0-R5
          ;CLEAR R0 FOR ERROR NUMBER
          ;ERROR NUMBER
          ;SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
          ;BRANCH IF NOT
          ;GET TEST NUMBER
          ;MOVE POWER FAIL ERROR CALL TABLE TO R0
          ;AND COPY IT INTO R1
          ;FORM INDEX FOR ERROR TABLE
          ;FORM ADDRESS
          ;GET ERROR MESSAGE (EM) POINTER
          ;BR IF THERE ISN'T ONE
          ;"CARRIAGE RETURN - LINE FEED
          ;"EM" POINTER GOES HERE
          ;PICK UP DATA HEADER (DH) POINTER
          ;BR IF NONE
          ;CARRIAGE RETURN-LINE FEED
          ;"DH" POINTER GOES HERE
          ;PICKUP DATA TABLE (DT) POINTER
          ;BR IF NONE
          ;SET INDENT SWITCH
          ;DATA FORMAT (DF) POINTER
          ;NUMBER OF DH'S TO TYPE
          ;BR IF DH NUMBER IS 0
          ;NO INDENT
          ;CARRIAGE RETURN-LINE FEED
          ;NUMBER OF DATA WORDS TO TYPE
          ;AND HOW TO TYPE THEM
          ;OCTAL OR DECIMAL?
          ;DECIMAL
          ;:SAVE @R1+ FOR TYPEOUT
          ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
          ;:SAVE @R1+ FOR TYPEOUT
          ;:GO TYPE--DECIMAL ASCII WITH SIGN
          ;MORE NUMBERS TO TYPE?
          ;NO
          ;TYPE 2 SPACES
          ;LOOP
          ;MORE DH'S?
          ;NO
          ;YES--START A NEW LINE
          ;ONLY A 'DH' IN THIS REQUEST ?
          ;BR IF YES - BYPASS THE INDENT
          ;INDENT?

```

59	024162	001002				BNE	13\$:NO
60	024164	104401	036621			TYPE	.BLNKS2		:TYPE 2 SPACES
61	024170	012037	024176		13\$:	MOV	(R0)+,14\$:GET NEXT DH
62	024174	104401				TYPE			:AND TYPE IT
63	024176	000000			14\$:	.WORD	0		:DH POINTER GOES HERE
64	024200	005710				TST	(R0)		:TYPE A 'DT' ?
65	024202	001003				BNE	15\$:BR IF A 'DT'
66	024204	062700	000004			ADD	#4,R0		:INCREMENT THE 'DF' POINTER
67	024210	000754				BR	12\$:SEE IF END OF 'DF' BLOCK
68	024212	104401	001205		15\$:	TYPE	,\$CRLF		:CARRIAGE RETURN-LINE FEED
69	024216	005705				TST	R5		:INDENT?
70	024220	001332				BNE	8\$:NO
71	024222	104401	036621		16\$:	TYPE	.BLNKS2		:TYPE 2 SPACES
72	024226	000727				BR	8\$:LOOP
73	024230	104413			17\$:	RESREG			:RESTORE R0-R5
74	024232	000207				RTS	PC		:RETURN
75	024234	024244	024326	024360	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4			:WORDS DEFINING TABLES BELOW
76	024244	120	117	127	PFECH1:	.ASCIZ	?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?		
77	024326	124	105	123	PFECH2:	.ASCIZ	?TESTNO ERR PC CPUERRREG?		
78						.EVEN			
79	024360	024374	001132	023754	PFECH3:	.WORD	PFTSTN,\$ERRPC,CPSAVE,0		
80	024370	000001			PFECH4:	.WORD	1		:NUMBER OF DATA HEADERS
81	024372	003				.BYTE	3		:NUMBER OF WORDS IN DATA TABLE
82	024373	000				.BYTE	0		:ALL 3 NUMBERS ARE OCTAL
83	024374	000000			PFTSTN:	.WORD	0		:CONTAINS TEST NUMBER FOR PF BIT ERROR

1

.SBTTL TTY INPUT ROUTINE

024376 000000
024400 000000
024402 000000
024404 024413

```

*****
ENABL  LSB
$TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
$TKQIN: .WORD 0          ;;INPUT POINTER
$TKQOUT: .WORD 0         ;;OUTPUT POINTER
$TKQSRT: .BLKB 7         ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN
    
```

```

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

```

;*CALL:
;*      JSR      PC,$TKINT
;*      RETURN
    
```

024414 005037 024376
024420 012737 024404 024400
024426 013737 024400 024402
024434 012737 024464 000060
024442 012737 000200 000062
024450 005777 154506
024454 012777 000100 154476
024462 000207

```

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

```

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

024464 117746 154472
024470 042716 177600
024474 021627 000021
024500 001002
024502 005726
024504 000002
024506
024506 021627 000003
024512 001007
024514 104401 025621
024520 004737 024414
024524 005726
024526 000177 154640
024532 021627 000007
024536 001004
024540 022737 000176 001154
024546 001500

024550
024550 022737 000007 024376
024556 001004
024560 104401 001200

```

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

```

024564 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
024566 000451          BR       5$          ;;EXIT
024570 021627 000023  3$:    CMP      (SP),#23  ;;IS IT A CONTROL-S?
024574 001021          BNE     32$          ;;BRANCH IF NO
024576 005077 154356  CLR     @STKS       ;;DISABLE TTY KEYBOARD INTERRUPTS
024602 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
024604 105777 154350  31$:  TSTB   @STKS       ;;WAIT FOR A CHAR
024610 100375          BPL     31$          ;;LOOP UNTIL ITS THERE
024612 117746 154344  MOVB   @STKB,-(SP)  ;;GET THE CHARACTER
024616 042716 177600  BIC    #^C177,(SP) ;;MAKE IT 7-BIT ASCII
024622 022627 000021  CMP     (SP)+,#21  ;;IS IT A CONTROL-Q?
024626 001366          BNE     31$          ;;BRANCH IF NO
024630 012777 000100 154322 MOV     #100,@STKS ;;REENABLE TTY KEYBOARD INTERRUPTS
024636 000002          RTI                    ;;RETURN
024640 005237 024376  32$:  INC     $TKCNT      ;;COUNT THIS CHARACTER
024644 021627 000140  CMP     (SP),#140  ;;IS IT UPPER CASE?
024650 002405          BLT     4$          ;;BRANCH IF YES
024652 021627 000175  CMP     (SP),#175  ;;IS IT A SPECIAL CHAR?
024656 003002          BGT     4$          ;;BRANCH IF YES
024660 042716 000040  BIC    #40,(SP)    ;;MAKE IT UPPER CASE
024664 112677 177510  4$:    MOVB   (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
024670 005237 024400  INC     $TKQIN     ;;UPDATE THE POINTER
024674 023727 024400 024413 CMP     $TKQIN,$TKQEND ;;GO OFF THE END?
024702 001003          BNE     5$          ;;BRANCH IF NO
024704 012737 024404 024400 MOV     #$TKQSRST,$TKQIN ;;RESET THE POINTER
024712 000002          5$:    RTI                    ;;RETURN

```

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

```

024714 022737 000176 001154 $CKSWR: CMP     #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED
024722 001124          BNE     15$          ;;EXIT IF NOT
024724 105777 154230  TSTB   @STKS       ;;IS A CHAR WAITING?
024730 100121          BPL     15$          ;;IF NOT, EXIT
024732 117746 154224  MOVB   @STKB,-(SP) ;;YES
024736 042716 177600  BIC    #^C177,(SP) ;;MAKE IT 7-BIT ASCII
024742 021627 000007  CMP     (SP),#7    ;;IS IT A CONTROL-G?
024746 001300          BNE     2$          ;;IF NOT, PUT IT IN THE TTY QUEUE
                                ;;AND EXIT

```

;;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
 ;;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
 ;;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.

```

024750 123727 001150 000001 6$:    CMPB   $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
024756 001674          BEQ     2$          ;;BRANCH IF YES
024760 005726          TST      (SP)+      ;;CLEAR CONTROL-G OFF STACK
024762 004737 024414  JSR    PC,$TKINT   ;;FLUSH THE TTY INPUT QUEUE
024766 005077 154166  CLR     @STKS       ;;DISABLE TTY KEYBOARD INTERRUPTS
024772 112737 000001 001151 MOVB   #1,$INTAG   ;;SET INTERRUPT MODE INDICATOR

025000 104401 025633          TYPE   , $CNTLG    ;;ECHO THE CONTROL-G (^G)
025004 104401 025640  $GTSWR: TYPE   , $MSWR  ;;TYPE CURRENT CONTENTS
025010 013746 000176  MOV     SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
025014 104402          TYPOC  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)

```



```

025016 104401 025651          TYPE      .SMNEW          ::PROMPT FOR NEW SWR
025022 005046          19$: CLR      -(SP)          ::CLEAR COUNTER
025024 005046          CLR      -(SP)          ::THE NEW SWR
025026 105777 154126      7$: TSTB   @$TKS          ::CHAR THERE?
025032 100375          BPL      7$             ::IF NOT TRY AGAIN

025034 117746 154122          MOVB   @$TKB,-(SP)      ::PICK UP CHAR
025040 042716 177600          BIC    #^C177,(SP)    ::MAKE IT 7-BIT ASCII

025044 021627 000003          CMP    (SP),#3        ::IS IT A CONTROL-C?
025050 001015          BNE    9$             ::BRANCH IF NOT
025052 104401 025621          TYPE   .SCNTLC        ::YES, ECHO CONTROL-C (^C)
025056 062706 000006          ADD    #6,SP          ::CLEAN UP STACK
025062 123727 001151 000001  CMPB   $INTAG,#1      ::REENABLE TTY KEYBOARD INTERRUPTS?
025070 001003          BNE    8$             ::BRANCH IF NO
025072 012777 000100 154060  MOV    #100,@$TKS     ::ALLOW TTY KEYBOARD INTERRUPTS
025100 000177 154266      8$: JMP    @CNTLC      ::CONTROL-C RESTART

025104 021627 000025      9$: CMP    (SP),#25     ::IS IT A CONTROL-U?
025110 001005          BNE    10$            ::BRANCH IF NOT
025112 104401 025626          TYPE   .SCNTLU        ::YES, ECHO CONTROL-U (^U)
025116 062706 000006      20$: ADD    #6,SP          ::IGNORE PREVIOUS INPUT
025122 000737          BR     19$            ::LET'S TRY IT AGAIN

025124 021627 000015      10$: CMP    (SP),#15     ::IS IT A <CR>?
025130 001022          BNE    16$            ::BRANCH IF NO
025132 005766 000004          TST   4(SP)          ::YES, IS IT THE FIRST CHAR?
025136 001403          BEQ   11$            ::BRANCH IF YES
025140 016677 000002 154006  MOV    2(SP),@SWR     ::SAVE NEW SWR
025146 062706 000006      11$: ADD    #6,SP          ::CLEAN UP STACK
025152 104401 001205      14$: TYPE   .SCRLF        ::ECHO <CR> AND <LF>
025156 123727 001151 000001  CMPB   $INTAG,#1      ::RE-ENABLE TTY KBD INTERRUPTS?
025164 001003          BNE    15$            ::BRANCH IF NOT
025166 012777 000100 153764  MOV    #100,@$TKS     ::RE-ENABLE TTY KBD INTERRUPTS
025174 000002          RTI                    ::RETURN
025176 004737 026074      16$: JSR    PC,$TYPEC      ::ECHO CHAR
025202 021627 000060          CMP    (SP),#60      ::CHAR < 0?
025206 002420          BLT   18$            ::BRANCH IF YES
025210 021627 000067          CMP    (SP),#67      ::CHAR > 7?
025214 003015          BGT   18$            ::BRANCH IF YES
025216 042726 000060          BIC   #60,(SP)+      ::STRIP-OFF ASCII
025222 005766 000002          TST   2(SP)          ::IS THIS THE FIRST CHAR
025226 001403          BEQ   17$            ::BRANCH IF YES
025230 006316          ASL   (SP)           ::NO, SHIFT PRESENT
025232 006316          ASL   (SP)           ::CHAR OVER TO MAKE
025234 006316          ASL   (SP)           ::ROOM FOR NEW ONE.
025236 005266 000002      17$: INC    2(SP)          ::KEEP COUNT OF CHAR
025242 056616 177776          BIS   -2(SP),(SP)    ::SET IN NEW CHAR
025246 000667          BR     7$             ::GET THE NEXT ONE
025250 104401 001204      18$: TYPE   $QUES        ::TYPE ?<CR><LF>
025254 000720          BR     20$            ::SIMULATE CONTROL-U
.DSABL  LSB

```

::*****

```

; *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
; *CALL:
; *   RDCHR           ;; GET A CHARACTER FROM THE QUEUE
; *   RETURN HERE    ;; CHARACTER IS ON THE STACK
; *                 ;; WITH PARITY BIT STRIPPED OFF
;
025256 011646          $RDCHR: MOV    (SP),-(SP)      ;; PUSH DOWN THE PC AND
025260 016666 000004 000002  MOV    4(SP),2(SP)    ;; THE PS
025266 005066 000004          CLR    4(SP)         ;; GET READY FOR A CHARACTER
025272 005046          CLR    -(SP)          ;; PUT NEW PS ON STACK
025274 012746 025302        MOV    #64$,-(SP)    ;; PUT NEW PC ON STACK
025300 000002          RTI                    ;; POP NEW PC AND PS
025302
025302 005737 024376 64$:   1$:   TST    $TKCNT          ;; WAIT ON A CHARACTER
025306 001775          BEQ    1$
025310 005337 024376          DEC    $TKCNT          ;; DECREMENT THE COUNTER
025314 117766 177062 000004  MOVB   @$TKQOUT,4(SP)  ;; GET ONE CHARACTER
025322 005237 024402          INC    $TKQOUT        ;; UPDATE THE POINTER
025326 023727 024402 024413  CMP    $TKQOUT,$$TKQEND ;; DID IT GO OFF OF THE END?
025334 001003          BNE    2$                ;; BRANCH IF NO
025336 012737 024404 024402  MOV    $$TKQSRT,$$TKQOUT ;; RESET THE POINTER
025344 000002          2$:   RTI                    ;; RETURN
; *****
; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
; *CALL:
; *   RDLIN           ;; INPUT A STRING FROM THE TTY
; *   RETURN HERE    ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
; *                 ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
;
025346 010346          $RDLIN: MOV    R3, -(SP)        ;; SAVE R3
025350 005046          CLR    -(SP)          ;; CLEAR THE RUBOUT KEY
025352 012703 025602 1$:   MOV    $$TTYIN,R3      ;; GET ADDRESS
025356 022703 025621 2$:   CMP    $$TTYIN+15.,R3    ;; BUFFER FULL?
025362 101456          BLOS   4$                ;; BR IF YES
025364 104410          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
025366 112613          MOVB   (SP)+,(R3)        ;; GET CHARACTER
025370 122713 000177 10$:  CMPB   #177,(R3)        ;; IS IT A RUBOUT
025374 001022          BNE    5$                ;; BR IF NO
025376 005716          TST    (SP)          ;; IS THIS THE FIRST RUBOUT?
025400 001007          BNE    6$                ;; BR IF NO
025402 112737 000134 025600  MOVB   #' \,9$      ;; TYPE A BACK SLASH
025410 104401 025600          TYPE   ,9$
025414 012716 177777          MOV    #-1,(SP)        ;; SET THE RUBOUT KEY
025420 005303 6$:   DEC    R3                ;; BACKUP BY ONE
025422 020327 025602          CMP    R3,$$TTYIN    ;; STACK EMPTY?
025426 103434          BLO    4$                ;; BR IF YES
025430 111337 025600          MOVB   (R3),9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
025434 104401 025600          TYPE   ,9$
025440 000746          BR     2$                ;; GO TYPE
025442 005716 5$:   TST    (SP)          ;; GO READ ANOTHER CHAR.
025444 001406          BEQ    7$                ;; RUBOUT KEY SET?
025446 112737 000134 025600  MOVB   #' \,9$      ;; BR IF NO
025454 104401 025600          TYPE   ,9$      ;; TYPE A BACK SLASH
025460 005016          CLR    (SP)          ;; CLEAR THE RUBOUT KEY
025462 122713 000025 7$:   CMPB   #25,(R3)        ;; IS CHARACTER A CTRL U?
025466 001003          BNE    8$                ;; BR IF NO

```


025470	104401	025626			TYPE	,\$CNTLU	::TYPE A CONTROL 'U'	
025474	000726				BR	1\$::GO START OVER	
025476	122713	000022	8\$:		CMPB	#22,(R3)	::IS CHARACTER A '^R'?	
025502	001011				BNE	3\$::BRANCH IF NO	
025504	105013				CLRB	(R3)	::CLEAR THE CHARACTER	
025506	104401	001205			TYPE	,\$SCLF	::TYPE A 'CR' & 'LF'	
025512	104401	025602			TYPE	,\$TTYIN	::TYPE THE INPUT STRING	
025516	000717				BR	2\$::GO PICKUP ANOTHER CHACTER	
025520	104401	001204	4\$:		TYPE	,\$QUES	::TYPE A '?'	
025524	000712				BR	1\$::CLEAR THE BUFFER AND LOOP	
025526	111337	025600	3\$:		MOVB	(R3),9\$::ECHO THE CHARACTER	
025532	104401	025600			TYPE	,9\$		
025536	122723	000015			CMPB	#15,(R3)+	::CHECK FOR RETURN	
025542	001305				BNE	2\$::LOOP IF NOT RETURN	
025544	105063	177777			CLRB	-1(R3)	::CLEAR RETURN (THE 15)	
025550	104401	001206			TYPE	,\$LF	::TYPE A LINE FEED	
025554	005726				TST	(SP)+	::CLEAN RUBOUT KEY FROM THE STACK	
025556	012603				MOV	(SP)+,R3	::RESTORE R3	
025560	011646				MOV	(SP)-,(SP)	::ADJUST THE STACK AND PUT ADDRESS OF THE	
025562	016666	000004	000002		MOV	4(SP),2(SP)	::FIRST ASCII CHARACTER ON IT	
025570	012766	025602	000004		MOV	#\$TTYIN,4(SP)		
025576	000002				RTI		::RETURN	
025600	000			9\$:	.BYTE	0	::STORAGE FOR ASCII CHAR. TO TYPE	
025601	000				.BYTE	0	::TERMINATOR	
025602					.\$TTYIN:	.BLKB	15	::RESERVE 15 BYTES FOR TTY INPUT
025621	136	103	015		.\$CNTLC:	.ASCIZ	/^C/<15><12>	::CONTROL 'C'
025626	136	125	015		.\$CNTLU:	.ASCIZ	/^U/<15><12>	::CONTROL 'U'
025633	136	107	015		.\$CNTLG:	.ASCIZ	/^G/<15><12>	::CONTROL 'G'
025640	015	012	123		.\$MSWR:	.ASCIZ	<15><12>/SWR = /	
025651	040	040	116		.\$MNEW:	.ASCIZ	/ NEW = /	

.SBITL TYPE ROUTINE

```

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

```

```

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

```

025662	105737	001173	\$TYPE:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
025666	100002			BPL	1\$:: BR IF YES
025670	000000			HALT		:: HALT HERE IF NO TERMINAL
025672	000430			BR	3\$:: LEAVE
025674	010046		1\$:	MOV	RO,-(SP)	:: SAVE RO
025676	017600	000002		MOV	@2(SP),RO	:: GET ADDRESS OF ASCIZ STRING
025702	122737	000001 001230		CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
025710	001011			BNE	62\$:: NO,GO CHECK FOR APT CONSOLE
025712	132737	000100 001231		BITB	#APTSPOOL,\$ENVM	:: SPOOL MESSAGE TO APT
025720	001405			BEQ	62\$:: NO,GO CHECK FOR CONSOLE
025722	010037	025732		MOV	RO,61\$:: SETUP MESSAGE ADDRESS FOR APT
025726	004737	027512		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
025732	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
025734	132737	000040 001231	62\$:	BITB	#APTCSUP,\$ENVM	:: APT CONSOLE SUPPRESSED
025742	001003			BNE	60\$:: YES,SKIP TYPE OUT
025744	112046		2\$:	MOVB	(RO)+,-(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK
025746	001005			BNE	4\$:: BR IF IT ISN'T THE TERMINATOR
025750	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
025752	012600		60\$:	MOV	(SP)+,RO	:: RESTORE RO
025754	062716	000002	3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
025760	000002			RTI		:: RETURN
025762	122716	000011	4\$:	CMPB	#HT,(SP)	:: BRANCH IF <HT>
025766	001430			BEQ	8\$	
025770	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
025774	001006			BNE	5\$	
025776	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
026000	104401			TYPE		:: TYPE A CR AND LF
026002	001205			\$CRLF		
026004	105037	026212		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
026010	000755			BR	2\$:: GET NEXT CHARACTER
026012	004737	026074	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
026016	123726	001172	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
026022	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.
026024	013746	001170		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
026030	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
026034	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK
026036	004737	026074		JSR	PC,\$TYPEC	:: GO TYPE A NULL
026042	105337	026212		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT
026046	000770			BR	7\$:: LOOP

:HORIZONTAL TAB PROCESSOR

026050	112716	000040		8\$:	MOVB	#* (SP)	::REPLACE TAB WITH SPACE
026054	004737	026074		9\$:	JSR	PC,\$TYPEC	::TYPE A SPACE
026060	132737	000007	026212		BITB	#7,\$CHARCNT	::BRANCH IF NOT AT
026066	001372				BNE	9\$::TAB STOP
026070	005726				TST	(SP)+	::POP SPACE OFF STACK
026072	000724				BR	2\$::GET NEXT CHARACTER
026074				\$TYPEC:			
026074	105777	153060			TSTB	@\$TKS	::CHAR IN KYBD BUFFER?
026100	100022				BPL	10\$::BR IF NOT
026102	017746	153054			MOV	@\$TKB,-(SP)	::GET CHAR
026106	042716	177600			BIC	#177600,(SP)	::STRIP EXTRANEIOUS BITS
026112	122716	000023			CMPB	#\$XOFF,(SP)	::WAS CHAR XOFF
026116	001012				BNE	102\$::BR IF NOT
026120				101\$:			
026120	105777	153034			TSTB	@\$TKS	::WAIT FOR CHAR
026124	100375				BPL	101\$	
026126	117716	153030			MOVB	@\$TKB,(SP)	::GET CHAR
026132	042716	177600			BIC	#177600,(SP)	::STRIP IT
026136	122716	000021			CMPB	#\$XON,(SP)	::WAS IT XON?
026142	001366				BNE	101\$::BR IF NOT
026144				102\$:			
026144	005726				TST	(SP)+	::FIX STACK
026146				10\$:			
026146	105777	153012			TSTB	@\$TPS	::WAIT UNTIL PRINTER IS READY
026152	100375				BPL	10\$	
026154	116677	000002	153004		MOVB	2(SP),@\$TPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
026162	122766	000015	000002		CMPB	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?
026170	001003				BNE	1\$::BRANCH IF NO
026172	105037	026212			CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT
026176	000406				BR	\$TYPEX	::EXIT
026200	122766	000012	000002	1\$:	CMPB	#LF,2(SP)	::IS CHARACTER A LINE FEED?
026206	001402				BEQ	\$TYPEX	::BRANCH IF YES
026210	105227				INCB	(PC)+	::COUNT THE CHARACTER
026212	000000			\$CHARCNT:	.WORD	0	::CHARACTER COUNT STORAGE
026214	000207			\$TYPEX:	RTS	PC	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

```

```

*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPON   ;;CALL FOR TYPEOUT

```

```

*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
*   TYPOC   ;;CALL FOR TYPEOUT

```

026216	017646	000000		\$TYPOS:	MOV	@(SP),-(SP)	;;PICKUP THE MODE
026222	116637	000001	026441		MOV	1(SP), \$OFILL	;;LOAD ZERO FILL SWITCH
026230	112637	026443			MOV	(SP)+, \$OMODE+1	;;NUMBER OF DIGITS TO TYPE
026234	062716	000002			ADD	#2,(SP)	;;ADJUST RETURN ADDRESS
026240	000406				BR	\$TYPON	
026242	112737	000001	026441	\$TYPOC:	MOV	#1, \$OFILL	;;SET THE ZERO FILL SWITCH
026250	112737	000006	026443		MOV	#6, \$OMODE+1	;;SET FOR SIX(6) DIGITS
026256	112737	000005	026440	\$TYPON:	MOV	#5, \$OCNT	;;SET THE ITERATION COUNT
026264	010346				MOV	R3, -(SP)	;;SAVE R3
026266	010446				MOV	R4, -(SP)	;;SAVE R4
026270	010546				MOV	R5, -(SP)	;;SAVE R5
026272	113704	026443			MOV	\$OMODE+1, R4	;;GET THE NUMBER OF DIGITS TO TYPE
026276	005404				NEG	R4	
026300	062704	000006			ADD	#6, R4	;;SUBTRACT IT FOR MAX. ALLOWED
026304	110437	026442			MOV	R4, \$OMODE	;;SAVE IT FOR USE
026310	113704	026441			MOV	\$OFILL, R4	;;GET THE ZERO FILL SWITCH
026314	016605	000012			MOV	12(SP), R5	;;PICKUP THE INPUT NUMBER
026320	005003				CLR	R3	;;CLEAR THE OUTPUT WORD
026322	006105			1\$:	ROL	R5	;;ROTATE MSB INTO 'C'
026324	000404				BR	3\$;;GO DO MSB
026326	006105			2\$:	ROL	R5	;;FORM THIS DIGIT
026330	006105				ROL	R5	
026332	006105				ROL	R5	
026334	010503				MOV	R5, R3	
026336	006103			3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
026340	105337	026442			DECB	\$OMODE	;;TYPE THIS DIGIT?
026344	100016				BPL	7\$;;BR IF NO
026346	042703	177770			BIC	#177770, R3	;;GET RID OF JUNK
026352	001002				BNE	4\$;;TEST FOR 0
026354	005704				TST	R4	;;SUPPRESS THIS 0?
026356	001403				BEQ	5\$;;BR IF YES
026360	005204			4\$:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S


```

026362 052703 000060
026366 052703 000040
026372 110337 026436
026376 104401 026436
026402 105337 026440
026406 003347
026410 002402
026412 005204
026414 000744
026416 012605
026420 012604
026422 012603
026424 016666 000002 000004
026432 012616
026434 000002
026436 000
026437 000
026440 000
026441 000
026442 000000
  
```

```

5$: BIS #'0,R3
    BIS #' ,R3
    MOVB R3,8$
    TYPE 8$
7$: DECB $OCNT
    BGT 2$
    BLT 6$
    INC R4
    BR 2$
6$: MOV (SP)+,R5
    MOV (SP)+,R4
    MOV (SP)+,R3
    MOV 2(SP),4(SP)
    MOV (SP)+,(SP)
    RTI
8$: .BYTE 0
    .BYTE 0
    $OCNT: .BYTE 0
    $OFILL: .BYTE 0
    $OMODE: .WORD 0
  
```

```

::MAKE THIS DIGIT ASCII
::MAKE ASCII IF NOT ALREADY
::SAVE FOR TYPING
::GO TYPE THIS DIGIT
::COUNT BY 1
::BR IF MORE TO DO
::BR IF DONE
::INSURE LAST DIGIT ISN'T A BLANK
::GO DO THE LAST DIGIT
::RESTORE R5
::RESTORE R4
::RESTORE R3
::SET THE STACK FOR RETURNING
::RETURN
::STORAGE FOR ASCII DIGIT
::TERMINATOR FOR TYPE ROUTINE
::OCTAL DIGIT COUNTER
::ZERO FILL SWITCH
::NUMBER OF DIGITS TO TYPE
  
```


.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

```

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

```

```

026444      010046      $TYPDS:      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
026444      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
026450      010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
026452      010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
026454      010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
026456      012746      020200      MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
026462      016605      000020      MOV      20(SP),R5      ;;GET THE INPUT NUMBER
026466      100004      BPL      1$           ;;BR IF INPUT IS POS.
026470      005405      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
026472      112766      000055      000001      MOVB     #'-,1(SP)      ;;MAKE THE ASCII NUMBER NEG.
026500      005000      1$:      CLR      R0           ;;ZERO THE CONSTANTS INDEX
026502      012703      026660      MOV      #$DBLK,R3      ;;SETUP THE OUTPUT POINTER
026506      112723      000040      MOVB     #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
026512      005002      2$:      CLR      R2           ;;CLEAR THE BCD NUMBER
026514      016001      026650      MOV      $DTBL(R0),R1    ;;GET THE CONSTANT
026520      160105      3$:      SUB      R1,R5         ;;FORM THIS BCD DIGIT
026522      002402      BLT      4$           ;;BR IF DONE
026524      005202      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
026526      000774      BR       3$
026530      060105      4$:      ADD      R1,R5         ;;ADD BACK THE CONSTANT
026532      005702      TST      R2           ;;CHECK IF BCD DIGIT=0
026534      001002      BNE      5$           ;;FALL THROUGH IF 0
026536      105716      TSTB     (SP)          ;;STILL DOING LEADING 0'S?
026540      100407      BMI      7$           ;;BR IF YES
026542      106316      5$:      ASLB     (SP)          ;;MSD?
026544      103003      BCC      6$           ;;BR IF NO
026546      116663      000001      177777      MOVB     1(SP),-1(R3)    ;;YES--SET THE SIGN
026554      052702      000060      6$:      BIS      #'0,R2        ;;MAKE THE BCD DIGIT ASCII
026560      052702      000040      7$:      BIS      #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
026564      110223      MOVB     R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
026566      005720      TST      (R0)+        ;;JUST INCREMENTING
026570      020027      000010      CMP      R0,#10        ;;CHECK THE TABLE INDEX
026574      002746      BLT      2$           ;;GO DO THE NEXT DIGIT
026576      003002      BGT      8$           ;;GO TO EXIT
026600      010502      MOV      R5,R2        ;;GET THE LSD
026602      000764      BR       6$           ;;GO CHANGE TO ASCII
026604      105726      8$:      TSTB     (SP)+        ;;WAS THE LSD THE FIRST NON-ZERO?
026606      100003      BPL      9$           ;;BR IF NO
026610      116663      177777      177776      MOVB     -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
026616      105013      9$:      CLRB     (R3)         ;;SET THE TERMINATOR
026620      012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
026622      012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
026624      012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
026626      012601      MOV      (SP)+,R1      ;;POP STACK INTO R1

```


026630	012600		MOV	(SP)+,R0	::POP STACK INTO R0
026632	104401	026660	TYPE	,\$DBLK	::NOW TYPE THE NUMBER
026636	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
026644	012616		MOV	(SP)+,(SP)	
026646	000002		RTI		::RETURN TO USER
026650	023420		\$DTBL:	10000.	
026652	001750			1000.	
026654	000144			100.	
026656	000012			10.	
026660			\$DBLK:	.BLKW 4	

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

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

```

:CALL:
:      MOV      #NUMADR,-(SP)  ;FIRST ADDRESS OF ASCIZ STRING
:      JSR      PC,$SUPRS
:
:      OR
:      MOV      #NUMADR,-(SP)  ;FIRST ADDRESS OF ASCIZ STRING
:      JSR      PC,$SUPRL
:
$SUPRL: MOV      R0,-(SP)        ;SAVE R0
        MOV      4(SP),R0      ;GET POINTER TO MESSAGE
        CLR      $SUPR2
        BR       $SUPR1
:
$SUPRS: MOV      R0,-(SP)        ;SAVE R0
        MOV      4(SP),R0      ;GET POINTER TO MESSAGE
        MOV      R0,$SUPR2     ;GET POINTER FOR TYPING
:
$SUPR1: 1$:      TSTB     (R0)      ;TEST FOR TERMINATOR
        BEQ      2$            ;YES
        CMPB    #'0',(R0)      ;IS THIS A '0' ?
        BNE     3$            ;NO
        MOVB   #40,(R0)+      ;REPLACE IT WITH A 'BLANK'
        BR      1$            ;NEXT CHAR.
        2$:      DEC      R0      ;BACKUP 1
        MOVB   #'0',(R0)      ;MAKE IT '0'
        3$:      TST      $SUPR2   ;LEFT JUSTIFY ?
        BNE     4$            ;NO
        MOV     R0,$SUPR2     ;YES
        4$:      TYPE     .WORD    0
$SUPR2: MOV      (SP)+,R0      ;RESTORE R0
        MOV     (SP)+,(SP)    ;RESTORE STACK
        RTS     PC
  
```

```

12 026670 010046
13 026672 016600 000004
14 026676 005037 026760
15 026702 000405
17 026704 010046
18 026706 016600 000004
19 026712 010037 026760
20 026716
21 026716 105710
22 026720 001406
23 026722 122710 000060
24 026726 001006
25 026730 112720 000040
26 026734 000770
27 026736 005300
28 026740 112710 000060
29 026744 005737 026760
30 026750 001002
31 026752 010037 026760
32 026756 104401
33 026760 000000
34 026762 012600
35 026764 012616
36 026766 000207
  
```



```

1      ;ROUTINE TO REPLACE LEADING ZEROS IN A NUMERIC STRING WITH SPACES
2      ;CALL:
3      :      MOV      #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
4      :      JSR      R5,REPLZ      ;REPLACE PRECEDING ZEROS WITH BLANKS
5      :      .WORD    N              ;'N' IS NUMBER OF DIGITS TO BE TYPED
6      :
7      :      OR
8      :      MOV      #ADR,-(SP)      ;ADDRESS OF NUMBER (IN ASCII)
9      :      JSR      R5,FILLZ      ;TYPE PRECEDING ZEROS
10     :      .WORD    N              ;'N' IS NUMBER OF DIGITS TO BE TYPED
11     026770 005237 027076      FILLZ: INC      FILL0      ;LEAVE ZERO'S
12
13     026774 010046              REPLZ: MOV      R0,-(SP)      ;SAVE R0
14     026776 016600 000004      MOV      4(SP),R0      ;ADDRESS OF NUMBER TO R0
15     027002 005737 027076      TST      FILL0      ;LEAVE PRECEDING ZEROS ?
16     027006 001014              BNE      3$          ;BR IF YES
17     027010 122710 000060      1$:  CMPB     #'0',(R0)  ;BYTE EQUAL TO ASCII '0' ?
18     027014 001004              BNE      2$          ;BR IF NOT
19     027016 112710 000040      MOVB     #40,(R0)    ;REPLACE THE ZERO WITH A SPACE
20     027022 005200              INC      R0          ;INCREMENT THE BYTE ADDRESS
21     027024 000771              BR       1$          ;GO BACK AND LOOK FOR MORE LEADING ZEROS
22     027026 105710              2$:  TSTB     (R0)      ;SEE IF ZERO BYTE TERMINATOR
23     027030 001003              BNE      3$          ;BR IF NOT
24     027032 005300              DEC      R0          ;BACKUP STRING POINTER
25     027034 112710 000060      MOVB     #'0',(R0)  ;PUT A ZERO BACK IN
26     027040 016600 000004      3$:  MOV      4(SP),R0  ;PUT ADDRESS OF FIRST CHARACTER ON STACK
27     027044 105720              4$:  TSTB     (R0)+     ;SEE IF ZERO BYTE TERMINATOR
28     027046 001376              BNE      4$          ;BR IF NOT
29     027050 005300              DEC      R0          ;BACKUP STRING POINTER
30     027052 162500              SUB      (R5)+,R0    ;ADJUST ADDRESS
31     027054 010037 027062      MOV      R0,5$      ;GET ADDRESS FOR TYPEOUT
32     027060 104401              TYPE
33     027062 000000              5$:  .WORD    0        ;ADDRESS OF NUMBER
34     027064 012600              MOV      (SP)+,R0    ;POP STACK INTO R0
35     027066 012616              MOV      (SP)+,(SP)  ;RESTORE STACK
36     027070 005037 027076      CLR      FILL0      ;RESET FILL FLAG
37     027074 000205              RTS      R5          ;RETURN
38
39     027076 000000      FILL0: .WORD    0    ;IF SET, LEAVE PRECEDING ZEROS FOR TYPE
    
```

1

.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

::*****
 ::*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,@#\$SB2D ::CALL
 ::* RETURN ::ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK

027100	016637	000002	027130	\$SB2D:	MOV	2(SP),1\$::SAVE BINARY NUMBER
027106	012746	027130			MOV	#1\$,-(SP)	::SET POINTER
027112	004737	027134			JSR	PC,@#\$DB2D	::CALL DOUBLE LENGTH CONVERT
027116	062716	000005			ADD	#5,(SP)	::ONLY ALLOW FIVE CHARACTERS
027122	012666	000032			MOV	(SP)+,2(SP)	::PICKUP POINTER
027126	000207				RTS	PC	::RETURN
027130	000000	000000		1\$:	.WORD	0,0	

.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 ASCII
    ;; IS ON THE STACK
    
```

```

027134 104412          $DB2D: SAVREG      ;; SAVE REGISTERS
027136 016602 000002  MOV      2(SP), R2      ;; PICKUP THE DATA POINTER
027142 012700 027314  MOV      #$DECVL, R0      ;; GET ADDRESS OF '$DECVL' STRING
027146 010066 000002  MOV      R0, 2(SP)        ;; PUT ADDRESS OF ASCII STRING ON STACK
027152 012201        MOV      (R2)+, R1      ;; PICKUP THE BINARY NUMBER
027154 012202        MOV      (R2)+, R2
027156 012737 000012 027232  MOV      #10., 4$        ;; SET UP TO DO 10 CONVERSIONS
027164 012704 027244  MOV      #$TNPWR, R4      ;; ADDRESS OF TEN POWER
027170 012705 027246  MOV      #$TNPWR+2, R5
027174 005003        1$: CLR      R3                ;; CLEAR PARTIAL
027176 161401        2$: SUB      (R4), R1          ;; SUBTRACT TEN POWER
027200 005602        SBC      R2
027202 161502        SUB      (R5), R2
027204 002402        BLT      3$                ;; BR IF TEN POWER TOO LARGE
027206 005203        INC      R3                ;; ADD 1 TO PARTIAL
027210 000772        BR       2$                ;; LOOP
027212 062401        3$: ADD      (R4)+, R1          ;; RESTORE SUBTRACTED VALUE
027214 005502        ADC      R2
027216 062402        ADD      (R4)+, R2
027220 022525        CMP      (R5)+, (R5)+      ;; MOVE TO NEXT TEN POWER
027222 052703 000060  BIS      #'0, R3          ;; CHANGE PARTIAL TO ASCII
027226 110320        MOVB     R3, (R0)+        ;; SAVE IT
027230 005327        DEC      (PC)+          ;; DONE?
027232 000000        4$: .WORD     0
027234 001357        BNE      1$                ;; BR IF NO
027236 105020        CLRB     (R0)+          ;; TERMINATOR
027240 104413        RESREG
027242 000207        RTS      PC          ;; RETURN
027244 145000        $TNPWR: 145000      ;; 1.0E09
027246 035632        35632
027250 160400        160400      ;; 1.0E08
027252 002765        2765
027254 113200        113200      ;; 1.0E07
027256 000230        230
027260 041100        041100      ;; 1.0E06
027262 000017        17
027264 103240        103240      ;; 1.0E05
027266 000001        1
027270 023420        23420      ;; 1.0E04
027272 000000        0
027274 001750        1750      ;; 1.0E03
027276 000000        0
027300 000144        144      ;; 1.0E02
027302 000000        0
    
```


027304 000012
027306 000000
027310 000001
027312 000000
027314

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

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

.SBTTL SINGLE LENGTH BINARY TO OCTAL ASCII ROUTINE

*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED OCTAL ASCII NUMBER.

*CALL
* MOV NUMBER, -(SP) ::PUT BINARY NUMBER ON THE STACK
* JSR PC, @#\$SB20 ::CALL
* RETURN ::ADDRESS OF 1ST ASCII CHAR. IS ON THE STACK

027330	016637	000002	027360	\$SB20:	MOV	2(SP), 1\$::SAVE THE BINARY NUMBER
027336	012746	027360			MOV	#1\$, -(SP)	::SET POINTER
027342	004737	027364			JSR	PC, @#\$DB20	::CALL DOUBLE LENGTH CONVERT ROUTINE
027346	062716	000005			ADD	#5, (SP)	::ONLY ALLOW SIX CHARACTERS
027352	012666	000002			MOV	(SP)+, 2(SP)	::PICKUP POINTER
027356	000207				RTS	PC	::RETURN
027360	000000	000000		1\$:	.WORD	0,0	

.SBTTL DOUBLE LENGTH BINARY TO OCTAL ASCII CONVERT ROUTINE

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

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

027364	104412		\$DB20:	SAVREG	;;SAVE ALL REGISTERS
027366	016601	000002		MOV 2(SP),R1	;;PICKUP THE POINTER TO LOW WORD
027372	012705	027503		MOV #\$OCTVL+13.,R5	;; POINTER TO DATA TABLE
027376	012704	000014		MOV #12.,R4	;;DO ELEVEN CHARACTERS
027402	012703	177770		MOV #^C7,R3	;;MASK
027406	012100			MOV (R1)+,R0	;;LOWER WORD
027410	012101			MOV (R1)+,R1	;;HIGH WORD
027412	005002			CLR R2	;;TERMINATOR
027414	110245		1\$:	MOV R2,-(R5)	;;PUT CHARACTER IN DATA TABLE
027416	010002			MOV R0,R2	;;GET THIS DIGIT
027420	005304			DEC R4	;;COUNT THIS CHARACTER
027422	003007			BGT 3\$;;BR IF NOT THE LAST DIGIT
027424	001405			BEQ 2\$;;BR IF IT IS THE LAST DIGIT
027426	005205			INC R5	;;ALL DIGITS DONE-ADJUST POINTER FOR FIRST
027430	010566	000002		MOV R5,2(SP)	;;ASCII CHAR. & PUT IT ON THE STACK
027434	104413			RESREG	;;RESTORE ALL REGISTERS
027436	000207			RTS PC	;;RETURN TO USER
027440	006203		2\$:	ASR R3	;;POSITION THE MASK FOR THE LAST DIGIT
027442	006001		3\$:	ROR R1	;;POSITION THE BINARY NUMBER FOR
027444	006000			ROR R0	;; THE NEXT OCTAL DIGIT
027446	006001			ROR R1	
027450	006000			ROR R0	
027452	006001			ROR R1	
027454	006000			ROR R0	
027456	040302			BIC R3,R2	;;MASK OUT ALL JUNK
027460	062702	000060		ADD #'0,R2	;;MAKE THIS CHAR. ASCII
027464	000753			BR 1\$;;GO PUT IT IN THE DATA TABLE
027466			\$OCTVL:	.BLKB 14.	;;RESERVE DATA TABLE

.SBTTL APT COMMUNICATIONS ROUTINE

```

*****
027504 112737 000001 027750 $ATY1: MOVB #1,$FFLG      ::TO REPORT FATAL ERROR
027512 112737 000001 027746 $ATY3: MOVB #1,$MFLG      ::TO TYPE A MESSAGE
027520 000403
027522 112737 000001 027750 $ATY4: MOVB #1,$FFLG      ::TO ONLY REPORT FATAL ERROR
027530 $ATYC:
027530 010046          MOV R0,-(SP)      ::PUSH R0 ON STACK
027532 010146          MOV R1,-(SP)      ::PUSH R1 ON STACK
027534 105737 027746   TSTB $MFLG        ::SHOULD TYPE A MESSAGE?
027540 001450          BEQ 5$              ::IF NOT: BR
027542 122737 000001 001230  CMPB #APTENV,$ENV      ::OPERATING UNDER APT?
027550 001031          BNE 3$              ::IF NOT: BR
027552 132737 000100 001231  BITB #APTPOOL,$ENVM    ::SHOULD SPOOL MESSAGES?
027560 001425          BEQ 3$              ::IF NOT: BR
027562 017600 000004          MOV @4(SP),R0      ::GET MESSAGE ADDR.
027566 062766 000002 000004  ADD #2,4(SP)          ::BUMP RETURN ADDR.
027574 005737 001210 1$: TST $MSGTYPE      ::SEE IF DONE W/ LAST XMISSION?
027600 001375          BNE 1$              ::IF NOT: WAIT
027602 010037 001224          MOV R0,$MSGAD      ::PUT ADDR IN MAILBOX
027606 105720          TSTB (R0)+          ::FIND END OF MESSAGE
027610 001376          BNE 2$              ::
027612 163700 001224          SUB $MSGAD,R0      ::SUB START OF MESSAGE
027616 006200          ASR R0              ::GET MESSAGE LNGTH IN WORDS
027620 010037 001226          MOV R0,$MSGGLT      ::PUT LENGTH IN MAILBOX
027624 012737 000004 001210  MOV #4,$MSGTYPE      ::TELL APT TO TAKE MSG.
027632 000413          BR 5$              ::
027634 017637 000004 027660 3$: MOV @4(SP),4$      ::PUT MSG ADDR IN JSR LINKAGE
027642 062766 000002 000004  ADD #2,4(SP)          ::BUMP RETURN ADDRESS
027650 013746 177776          MOV 177776,-(SP)      ::PUSH 177776 ON STACK
027654 004737 025662          JSR PC,$TYPE      ::CALL TYPE MACRO
027660 000000          4$: .WORD 0
027662          5$:
027662 105737 027750          10$: TSTB $FFLG        ::SHOULD REPORT FATAL ERROR?
027666 001416          BEQ 12$          ::IF NOT: BR
027670 005737 001230          TST $ENV          ::RUNNING UNDER APT?
027674 001413          BEQ 12$          ::IF NOT: BR
027676 005737 001210          11$: TST $MSGTYPE      ::FINISHED LAST MESSAGE?
027702 001375          BNE 11$          ::IF NOT: WAIT
027704 017637 000004 001212  MOV @4(SP),$FATAL    ::GET ERROR #
027712 062766 000002 000004  ADD #2,4(SP)          ::BUMP RETURN ADDR.
027720 005237 001210          INC $MSGTYPE        ::TELL APT TO TAKE ERROR
027724 105037 027750          12$: CLRB $FFLG        ::CLEAR FATAL FLAG
027730 105037 027747          CLRB $LFLG        ::CLEAR LOG FLAG
027734 105037 027746          CLRB $MFLG        ::CLEAR MESSAGE FLAG
027740 012601          MOV (SP)+,R1        ::POP STACK INTO R1
027742 012600          MOV (SP)+,R0        ::POP STACK INTO R0
027744 000207          RTS PC          ::RETURN
027746 000          $MFLG: .BYTE 0      ::MESSG. FLAG
027747 000          $LFLG: .BYTE 0      ::LOG FLAG
027750 000          $FFLG: .BYTE 0      ::FATAL FLAG
          .EVEN
          APTSIZE = 200
          APTENV = 001
          APTSPOOL = 100
          APTCSUP = 040
000200
000001
000100
000040

```

.SBTTL SAVE AND RESTORE R0-R5 ROUTINES

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

027752
027752 010046
027754 010146
027756 010246
027760 010346
027762 010446
027764 010546
027766 016646 000022
027772 016646 000022
027776 016646 000022
030002 016646 000022
030006 000002

\$SAVREG:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI

;*RESTORE R0-R5
;*CALL:
;* RESREG

030010
030010 012666 000022
030014 012666 000022
030020 012666 000022
030024 012666 000022
030030 012605
030032 012604
030034 012603
030036 012602
030040 012601
030042 012600
030044 000002

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

.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.

```

```

030046 010046          $TRAP: MOV     R0,-(SP)      ;;SAVE R0
030050 016600 000002  MOV     2(SP),R0      ;;GET TRAP ADDRESS
030054 005740          TST     -(R0)         ;;BACKUP BY 2
030056 111000          MOVVB  (R0),R0        ;;GET RIGHT BYTE OF TRAP
030060 006300          ASL    R0             ;;POSITION FOR INDEXING
030062 016000 030102  MOV     $TRPAD(R0),R0 ;;INDEX TO TABLE
030066 000200          RTS     R0           ;;GO TO ROUTINE

```

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

```

030070 011646          $TRAP2: MOV    (SP),-(SP)  ;;MOVE THE PC DOWN
030072 016666 000004 000002 MOV    4(SP),2(SP)     ;;MOVE THE PSW DOWN
030100 000002          RTI                    ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

:          ROUTINE
:          -----
030102 030070          $TRPAD: .WORD  $TRAP2
030104 025662          $TYPE  ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
030106 026242          $TYPOC ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
030110 026216          $TYPOS ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
030112 026256          $TYPON ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
030114 026444          $TYPDS ;;CALL=TYPDS    TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

030116 025004          $GTSWR ;;CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING

030120 024714          $CKSWR ;;CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
030122 025256          $RDCHR ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
030124 025346          $RDLIN ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
030126 027752          $SAVREG ;;CALL=SAVREG   TRAP+12(104412) SAVE R0-R5 ROUTINE
030130 030010          $RESREG ;;CALL=RESREG   TRAP+13(104413) RESTORE R0-R5 ROUTINE

```

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
030132 012737 030276 000024 $PWRDN: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST UP
030140 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
030146 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
030150 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
030152 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
030154 010346      MOV    R3,-(SP)      ;;PUSH R3 ON STACK
030156 010446      MOV    R4,-(SP)      ;;PUSH R4 ON STACK
030160 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
030162 017746 150766      MOV    @SWR,-(SP)    ;;PUSH @SWR ON STACK
030166 010637 030302      MOV    SP,$SAVR6    ;;SAVE SP
030172 012737 030204 000024      MOV    #$PWRUP,@#PWRVEC ;;SET UP VECTOR
030200 000000      HALT
030202 000776      BR     -2          ;;HANG UP

*****
:POWER UP ROUTINE
030204 012737 030276 000024 $PWRUP: MOV    #$ILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
030212 013706 030302      MOV    $SAVR6,SP    ;;GET SP
030216 005037 030302      CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
030222 005237 030302 1$:  INC    $SAVR6      ;;WAIT FOR THE INC
030226 001375      BNE    1$          ;;OF WORD
030230 012677 150720      MOV    (SP)+,@SWR   ;;POP STACK INTO @SWR
030234 012605      MOV    (SP)+,R5     ;;POP STACK INTO R5
030236 012604      MOV    (SP)+,R4     ;;POP STACK INTO R4
030240 012603      MOV    (SP)+,R3     ;;POP STACK INTO R3
030242 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
030244 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
030246 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
030250 012737 030132 000024      MOV    #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
030256 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
030264 104401      TYPE                    ;;REPORT THE POWER FAILURE
030266 030304      $PWRMG: .WORD $POWER  ;;POWER FAIL MESSAGE POINTER
030270 012716      MOV    (PC)+,(SP)   ;;RESTART AT ST3
030272 013056      $PWRAD: .WORD ST3   ;;RESTART ADDRESS
030274 000002      RTI
030276 000000      $ILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
030300 000776      BR     -2          ;; BEFORE THE POWER DOWN WAS COMPLETE
030302 000000      $SAVR6: 0          ;;PUT THE SP HERE
2 030304 200 042 120 $POWER: .ASCIZ <CRLF>/'POWER UP'/
3 .EVEN

```


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

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

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

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

::*****

:STORAGE FOR RMDS, RMER1, RMER2, AND RMMR2 ON AN ERROR '2'
:RMERRS = RMDS
:RMERRS+2 = RMER1
:RMERRS+4 = RMER2
:RMERRS+6 = RMMR2

030320 000000 000000 000000 RMERRS: .WORD 0,0,0,0

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

030330	000	DRVACT: .BYTE 0	:DRIVE 0
030331	000	.BYTE 0	:DRIVE 1
030332	000	.BYTE 0	:DRIVE 2
030333	000	.BYTE 0	:DRIVE 3
030334	000	.BYTE 0	:DRIVE 4
030335	000	.BYTE 0	:DRIVE 5
030336	000	.BYTE 0	:DRIVE 6
030337	000	.BYTE 0	:DRIVE 7

:TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
:DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
:DRVSTA>0 IF DRIVE IS ONLINE
:DRVSTA<0 IF DRIVE IS UNSAFE

030340	000	DRVSTA: .BYTE 0	:DRIVE 0
030341	000	.BYTE 0	:DRIVE 1
030342	000	.BYTE 0	:DRIVE 2
030343	000	.BYTE 0	:DRIVE 3
030344	000	.BYTE 0	:DRIVE 4
030345	000	.BYTE 0	:DRIVE 5
030346	000	.BYTE 0	:DRIVE 6
030347	000	.BYTE 0	:DRIVE 7

:TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
:DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
:DRV TYP=7 IF DRIVE IS RM05 *****
:DRV TYP=5 IF DRIVE IS RM02 *****
:DRV TYP=4 IF DRIVE IS RM03

```

50          ;DRV TYP=-1 IF NOT RM05/3/2
51
52 030350    000      DRV TYP: .BYTE 0          ;DRIVE 0
55 030351    000      .BYTE 0          ;DRIVE 1
    030352    000      .BYTE 0          ;DRIVE 2
    030353    000      .BYTE 0          ;DRIVE 3
    030354    000      .BYTE 0          ;DRIVE 4
    030355    000      .BYTE 0          ;DRIVE 5
    030356    000      .BYTE 0          ;DRIVE 6
    030357    000      .BYTE 0          ;DRIVE 7
56
57          ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
58          ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
59          ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
60
61 030360    000      DPINT: .BYTE 0          ;DRIVE 0
64 030361    000      .BYTE 0          ;DRIVE 1
    030362    000      .BYTE 0          ;DRIVE 2
    030363    000      .BYTE 0          ;DRIVE 3
    030364    000      .BYTE 0          ;DRIVE 4
    030365    000      .BYTE 0          ;DRIVE 5
    030366    000      .BYTE 0          ;DRIVE 6
    030367    000      .BYTE 0          ;DRIVE 7
65
66          ;TABLE OF PENDING DUAL PORT REQUESTS
67          ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
68          ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
69
70 030370    000      DPRQS: .BYTE 0          ;DRIVE 0
73 030371    000      .BYTE 0          ;DRIVE 1
    030372    000      .BYTE 0          ;DRIVE 2
    030373    000      .BYTE 0          ;DRIVE 3
    030374    000      .BYTE 0          ;DRIVE 4
    030375    000      .BYTE 0          ;DRIVE 5
    030376    000      .BYTE 0          ;DRIVE 6
    030377    000      .BYTE 0          ;DRIVE 7
74
75          ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
76          ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
77          ;'DPB' OF THE I/O OPERATION.
78
79 030400    000000   TRNSWT: .WORD 0
80
81          ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
82          ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
83          ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
84          ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
85          ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
86
87 030402    000000   SRCHWT: .WORD 0
88
89          ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
90          ;ACTDRV=0 IF DRIVER IS INACTIVE
91          ;ACTDRV>0 IF DRIVER IS ACTIVE
92
93 030404    000      ACTDRV: .BYTE 0
94
    
```



```

95      ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
96      ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
97      ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
98
99 030405      000      ACTSTR: .BYTE      0
100
101      ;UNLOAD FLAG (ULDFLG=8 BYTES)
102      ;ULDFLG=0 IF NO UNLOAD COMMAND
103      ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
104      ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
105
106 030406      000      ULDFLG: .BYTE      0      ;DRIVE 0
109 030407      000      .BYTE      0      ;DRIVE 1
      030410      000      .BYTE      0      ;DRIVE 2
      030411      000      .BYT     0      ;DRIVE 3
      030412      000      .BYT     0      ;DRIVE 4
      030413      000      .BYT     0      ;DRIVE 5
      030414      000      .BYT     0      ;DRIVE 6
      030415      000      .BYT     0      ;DRIVE 7
110
111      ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
112      ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
113      ;OPERATION IS COMPLETED AS PER (DPB+14).
114      ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
115      ;(DPB+14), AFTER AN ERROR.
116
117 030416      000000    SAVEFG: .WORD      0
118
119      ;SEEK FLAG (SEEKFG=1 WORD)
120      ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
121      ;FOR A DATA TRANSFER START A SEARCH COMMAND
122      ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
123      ;DISREGARD THE WINDOW
124
125 030420      177777    SEEKFG: .WORD     -1
126
127      ;TIMEOUT TABLE (TIMER=8 WORDS)
128      ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
129
130 030422      177777    TIMER:   .WORD     -1      ;DRIVE 0
133 030424      177777    .WORD     -1      ;DRIVE 1
      030426      177777    .WORD     -1      ;DRIVE 2
      030430      177777    .WORD     -1      ;DRIVE 3
      030432      177777    .WORD     -1      ;DRIVE 4
      030434      177777    .WORD     -1      ;DRIVE 5
      030436      177777    .WORD     -1      ;DRIVE 6
      030440      177777    .WORD     -1      ;DRIVE 7
134
135      ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
136      ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
137      ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
138
139 030442      177777    DTUW:   .WORD     -1
140
141      ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
142      ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
143      ;ATTENTION BIT
    
```

```

144
145 030444 001          ATABIT: .BYTE 1          ;DRIVE 0
146 030445 002          .BYTE 2          ;DRIVE 1
147 030446 004          .BYTE 4          ;DRIVE 2
148 030447 010          .BYTE 10         ;DRIVE 3
149 030450 020          .BYTE 20         ;DRIVE 4
150 030451 040          .BYTE 40         ;DRIVE 5
151 030452 100          .BYTE 100        ;DRIVE 6
152 030453 200          .BYTE 200        ;DRIVE 7
153
154          ;NUMBER OF 'MASSBUS CONTROL PARITY ERRORS' (MCPE) ALLOWED BEFORE
155          ;CALLING IT FATAL (MCPEMX=1 WORD)
156
157 030454 000003      MCPEMX: .WORD 3
158
159          ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
160          ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
161
162 030456 176700      RMADR: .WORD 176700
163 030460 000254 000240 RMVEC: .WORD 254,5*32.
164
165          ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
166
167 030464 000005      MXWNDW: .WORD 5
168
169          ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
170
171
172
173          000000      RMCS1 = 0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 0)
174          000002      RMWC = 2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
175          000004      RMBA = 4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
176          000006      RMDA = 6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 5)
177          000010      RMCS2 = 10         ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
178          000012      RMDS = 12         ;DRIVE STATUS REGISTER (DRIVE REG 1)
179          000014      RMER1 = 14        ;ERROR REGISTER #1 (DRIVE REG. 2)
180          000016      RMAS = 16        ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 4)
181          000020      RMLA = 20        ;LOOK AHEAD REGISTER (DRIVE REG. 7)
182          000022      RMDB = 22        ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
183          000024      RMMR1 = 24        ;MAINTAINABILITY REGISTER (DRIVE REG. 3)
184          000026      RMDT = 26        ;DRIVE TYPE REGISTER (DRIVE REG. 6)
185          000030      RMSN = 30        ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
186          000032      RMOF = 32        ;OFFSET REGISTER (DRIVE REG. 11)
187          000034      RMDC = 34        ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
188          000036      RMHR = 36        ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
189          000040      RMMR2 = 40        ;MAINTENANCE REGISTER #2
190          000042      RMER2 = 42        ;ERROR REGISTER #2 (DRIVE REG. 15)
191          000044      RMEC1 = 44        ;ECC POSITION REGISTER (DRIVE REG. 16)
192          000046      RMEC2 = 46        ;ECC PATTERN REGISTER (DRIVE REG. 17)
193
194          .SBTTL RH/RM DRIVER INITIALIZATION CODE
195
196
197
198
199
200          ;THIS ROUTINE WILL DETERMINE WHICH RM DRIVES ARE
201          ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
202          ;TO THE PROPER STATE FOR EACH DRIVE.
203          ;NOTE: THIS ROUTINE CALLS DRVINT
204          ;
205          ;CALL
206          ;
207          ; JSR PC,RMINIT
    
```



```

208          :          RETURN
209          :
210          :NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
211          :
212
213 030466 104412          RMINIT: SAVREG          ;SAVE R0 - R5
214 030470 013746 177776          MOV          PS,-(SP)          ;SAVE THE PRESENT PROCESSOR STATUS
215 030474 012737 000240 177776          MOV          #<5*32.>,PS          ;CHANGE THE PRIORITY TO 5
216 030502 004737 036144          JSR          PC,CLRQUE          ;CLEAR ALL REQUEST QUEUES
217 030506 012701 030320          MOV          #R:MERRS,R1          ;FIRST ADDRESS TO BE CLEARED
218 030512 012702 030420          MOV          #SEEKFG,R2          ;LAST ADDRESS TO BE CLEARED
219 030516 005021          1$:          CLR          (R1)+          ;CLEAR
220 030520 020102          CMP          R1,R2          ;ARE WE DONE?
221 030522 103775          BLO          1$          ;BR IF NO
222 030524 012702 030442          MOV          #DTUW,R2          ;LAST ADDRESS
223 030530 012721 177777          2$:          MOV          #-1,(R1)+          ;INITIALIZE
224 030534 020102          CMP          R1,R2          ;DONE?
225 030536 101774          BLOS         2$          ;LOOP IF NO
226 030540 005037 030340          CLR          DRVSTA          ;SET ALL DRIVES TO OFFLINE
227 030544 005037 030342          CLR          DRVSTA+2
228 030550 005037 030344          CLR          DRVSTA+4
229 030554 005037 030346          CLR          DRVSTA+6
230 030560 013703 030460          MOV          RMVEC,R3          ;SETUP THE RH/RM VECTOR
231 030564 012723 033114          MOV          #ISR,(R3)+
232 030570 013713 030462          MOV          RMVEC+2,(R3)
233 030574 013704 030456          MOV          RMADR,R4          ;FIRST ADDRESS OF RH/RM
234 030600 012764 000040 000010          MOV          #CLR,RMCS2(R4)          ;MASSBUS INIT
235 030606 005001          CLR          R1          ;START WITH DRIVE 0
236 030610 004037 030700          3$:          JSR          R0,DRVINT          ;INIT THE DRIVE
237 030614 000401          BR          4$          ;'DVA' NOT SET OR PARITY ERROR
238 030616 000402          BR          5$          ;NORMAL RETURN
239 030620 105061 030340          4$:          CLRB         DRVSTA(R1)          ;SET DRIVE STATUS TO OFFLINE
240 030624 005201          5$:          INC          R1          ;GO TO NEXT DRIVE
241 030626 042701 177770          BIC          #^C7,R1          ;MASK OUT UNUSED BITS
242 030632 001366          BNE          3$          ;BR IF MORE DRIVES TO GO
243 030634 012701 000007          MOV          #7,R1          ;START WITH DRIVE 7
244 030640 005037 177776          CLR          PS          ;CLEAR THE PROCESSOR STATUS
245 030644 105761 030360          6$:          TSTB         DPINT(R1)          ;WAITING FOR DRIVE TO SWITCH PORTS ?
246 030650 001405          BEQ          8$          ;BR NOT WAITING
247 030652 004737 035600          JSR          PC,SET.IE          ;SET INTERRUPT
248 030656 105761 030360          7$:          TSTB         DPINT(R1)          ;DRIVE SWITCHED PORTS ?
249 030662 001375          BNE          7$          ;BR IF NOT
250 030664 005301          8$:          DEC          R1          ;GO TO THE NEXT DRIVE
251 030666 100366          BPL          6$          ;CHECK NEXT DRIVE
252 030670 012637 177776          MOV          (SP)+,PS          ;RESTORE THE PROCESSOR STATUS
253 030674 104413          RESREG
254 030676 000207          RTS          PC          ;RESTORE R0 - R5
255          ;BYE-BYE
256          ;DRIVE INITIALIZATION ROUTINE
257          ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
258          ;AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
259          ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
260          ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
261          ;DRVSTA IS SET TO THE PROPER CONDITION.
262          ;CALL
263          :          MOV          #DRVNUM,R1          ;DRIVE NUMBER TO R1
264          :          MOV          RMADR,R4          ;UNIBUS ADDRESS OF RH/RM (RMCS1)

```

```

265      :      JSR      R0,DRVINT      : CALLED BY A JSR
266      :      RETURN1      : ERROR OCCURRED (PARITY)
267      :      RETURN2      : NORMAL RETURN
268      :
269      :
270 030700 010546      • DRVINT: MOV      R5,-(SP)      : SAVE R5
271 030702 105061 030340      CLR      DRVSTA(R1)      : START DRIVE STATUS AS OFFLINE
272 030706 105061 030350      CLR      DRVTP(R1)      : CLEAR THE DRIVE TYPE INDICATOR
273 030712 105061 030406      CLR      ULDFLG(R1)      : CLEAR THE UNLOAD FLAG
274 030716 010164 000010      MOV      R1,RMCS2(R4)      : SELECT A DRIVE
275 030722 112764 000111 000000      MOV      #111,RMCS1(R4)      : DO A DRIVE (CLEAR COMMAND (& SEIZE DRIVE))
276 030730 032764 010000 000010      BIT      #BIT12,RMCS2(R4)      : NONEXISTENT DRIVE?
277 030736 001403      BEQ      1$      : NO
278 030740 004737 035600      JSR      PC,SET.IE      : GO SET "IE" WITHOUT A "TRE"
279 030744 000520      BR      4$      : LEAVE THIS ROUTINE
280
281 030746 105061 030340      1$: CLR      DRVSTA(R1)      : SET DRIVE STATUS TO OFFLINE
282 030752 032764 004000 000000      BIT      #BIT11,RMCS1(R4)      : SEE IF DRIVE AVAILABLE
283 030760 001512      BEQ      4$      : BR IF DRIVE NOT AVAILABLE
284 030762 004037 035110      JSR      R0,RD.RM      : READ THE DRIVE TYPE REG.
285 030766 000026      RMDT
286 030770 031210      5$
287 030772 012605      MOV      (SP)+,R5      : ERROR RETURN ADDRESS
288 030774 112761 000004 030350      MOV      #4,DRVTP(R1)      : PUT DRIVE TYPE IN R5
289 031002 022705 020024      CMP      #20024,R5      : SET RM03 INDICATOR
290 031006 001431      BEQ      2$      : SINGLE PORT RM03 ?
291 031010 022705 024024      CMP      #24024,R5      : BR IF YES
292 031014 001426      BEQ      2$      : DUAL PORT RM03 ?
293 031016 112761 000005 030350      MOV      #5,DRVTP(R1)      : BR IF YES
294 031024 022705 020025      CMP      #20025,R5      : SET RM02 INDICATOR
295 031030 001420      BEQ      2$      : SINGLE PORT RM02 ?
296 031032 022705 024025      CMP      #24025,R5      : BR IF SO
297 031036 001415      BEQ      2$      : DUAL PORT RM02 ?
298 031040 112761 000007 030350      MOV      #7,DRVTP(R1)      : BR IF SO
299 031046 022705 020027      CMP      #20027,R5      : SET RM05 INDICATOR
300 031052 001407      BEQ      2$      : SINGLE PORT RM05 ?
301 031054 022705 024027      CMP      #24027,R5      : BR IF YES
302 031060 001404      BEQ      2$      : DUAL PORT RM05 ?
303 031062 112761 177777 030350      MOV      #-1,DRVTP(R1)      : BR IF YES
304 031070 000446      BR      4$      : SET II )ICATOR TO 'OTHER'
305      :EXIT
306 031072 012746 000121      2$: MOV      #121,-(SP)      : DO A 'READ-IN PRESET'
307 031076 004037 035270      JSR      R0,WRT.RM
308 031102 000000      RMCS1
309 031104 031210      5$
310 031106 012746 010000      MOV      #BIT12,-(SP)      : SET FMT16=1
311 031112 004037 035270      JSR      R0,WRT.RM
312 031116 000032      RMOF
313 031120 031210      5$
314 031122 004037 035110      JSR      R0,RD.RM      : READ RMDS
315 031126 000012      RMD5
316 031130 031210      5$
317 031132 012605      MOV      (SP)+,R5      : AND SAVE IT IN R5
318 031134 100015      BPL      3$      : BR IF ATA=0
319 031136 116164 030444 000016      MOV      ATABIT(R1),RMAS(R4)      : CLEAR ATTENTION BIT
320 031144 004037 035110      JSR      R0,RD.RM      : FIND OUT WHY ATA=1
321 031150 000014      RMER1
  
```



```

322 031152 031210          5$
323 031154 006126          ROL      (SP)+      :IS IT UNSAFE?
324 031156 100004          BPL      3$          :BR IF NOT
325 031160 112761 177777 030340  MOVB     #-1,DRVSTA(R1) :SET UNSAFE INDICATOR
326 031166 000407          BR       4$          :EXIT
327
328 031170 005105          3$:      COM      R5          :CHECK MOL, DPR, DRY, AND VV
329 031172 042705 167077      BIC      #^C<BIT12!BIT08!BIT07!BIT06>,R5
330 031176 001003          BNE      4$          :BR IF MOL, DPR, DRY, OR VV IS CLEAR
331 031200 112761 000001 030340  MOVB     #1,DRVSTA(R1) :SET DRIVE STATUS TO ONLINE
332 031206 005720          4$:      TST      (R0)+      :STEP OVER THE ERROR RETURN
333 031210 012605          5$:      MOV      (SP)+,R5      :RESTORE R5
334 031212 000200          RTS      R0          :EXIT
335
336          :REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
337
338          :CALL
339
340          :
341          JSR     R0,RM05      :CALL THE RM05 DRIVER
342          PNTADR      :ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
343          RETURN1     :RETURN HERE IF QUEUE IS FULL
344          RETURN2     :RETURN HERE IF REQUEST IS IN QUEUE OR THERE
345          :IS AN ERROR CONDITION
346 031214 013746 177776      RM05:    MOV      PS,-(SP)      :SAVE THE CALLING STATUS
347 031220 013737 030462 177776  MOV      RMVEC+2,PS    :DON'T ALLOW ANY RM INTERRUPTS
348 031226 112737 000001 030404  MOVB     #1,ACTDRV    :SET 'ACTIVE DRIVER' FLAG
349 031234 104412          SAVREG
350 031236 011002          MOV      (R0),R2      :PICKUP THE DRIVE PARAMETER BLOCK POINTER
351 031240 005062 000016      CLR      16(R2)       :CLEAR THE STATUS/ERROR INDICATOR
352 031244 111201          MOVB     (R2),R1      :PICKUP THE DRIVE NUMBER
353 031246 013704 030456      MOV      RMADR,R4     :UNIBUS ADDRESS OF RMCS1
354 031252 105761 030340      TSTB    DRVSTA(R1)   :CHECK DRIVES STATUS
355 031256 003014          BGT      1$          :BR IF ONLINE
356 031260 105761 030406      TSTB    ULDFLG(R1)   :UNLOAD COMMAND IN QUEUE?
357 031264 001036          BNE      3$          :BR IF YES
358 031266 105761 030360      TSTB    DPINT(R1)    :TRYING TO INIT THE DRIVE
359 031272 001042          BNE      5$          :BR IF YES
360 031274 004037 030700      JSR     R0,DRVINT    :GO INIT. THE DRIVE
361 031300 000434          BR       4$          :ERROR RETURN
362 031302 105761 030340      TSTB    DRVSTA(R1)   :IS DRIVE STATUS ONLINE?
363 031306 003445          BLE      6$          :BR IF NOT
364 031310 105761 030370      1$:     TSTB    DPRQS(R1)   :OUTSTANDING PORT REQUEST FOR THE DRIVE ?
365 031314 001031          BNE      5$          :BR IF YES
366 031316 010164 000010      MOV      R1,RMCS2(R4) :SELECT THE DRIVE
367 031322 004037 036242      JSR     R0,DRVQUE    :PUT THIS REQUEST IN QUEUE
368 031326 000460          BR       9$          :QUEUE IS FULL
369 031330 122762 000103 000002  CMPB     #103,2(R2)   :IS THIS REQ. FOR AN UNLOAD?
370 031336 001003          BNE      2$          :BR IF NO
371 031340 112761 177777 030406  MOVB     #-1,ULDFLG(R1) :SET THE 'UNLOAD IN QUEUE' FLAG
372 031346 105761 030330      2$:     TSTB    DRVACT(R1)   :IS THIS DRIVE ACTIVE?
373 031352 001043          BNE      8$          :BR IF YES
374 031354 004737 031506      JSR     PC,OPT        :CALL THE OPTIMIZER
375 031360 000440          BR       8$
376 031362 012762 120000 000016  3$:     MOV      #BIT15!BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
377 031370 000434          BR       8$          :EXIT
378 031372 004737 032564      4$:     JSR     PC,C17      :GO HANDLE THE PARITY ERROR

```

```

379 031376 000431          BR      8$
380 031400 004037 036242 5$: JSR    R0,DRVQUE ;PUT REQUEST IN QUEUE
381 031404 000431          BR      9$ ;QUEUE IS FULL
382 031406 032714 000100  BIT    #BIT06,(R4) ;IE BIT SET ?
383 031412 001023          BNE    8$ ;YES
384 031414 004737 035600  JSR    PC,SET.IE ;SET THE INTERRUPT
385 031420 000420          BR      8$ ;RETURN
386 031422 105761 030340 6$: TSTB  DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
387 031426 002412          BLT    7$ ;BR IF UNSAFE
388 031430 012762 140000 000016 MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
389 031436 105761 030350  TSTB  DRVSTYP(R1) ;SEE IF OFFLINE OR NONEXISTENT
390 031442 001007          BNE    8$ ;BR IF OFFLINE
391 031444 012762 100002 000016 MOV    #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
392 031452 000403          BR      8$ ;GO TO EXIT
393 031454 012762 110000 000016 7$: MOV    #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
394 031462 104413          8$: RESREG ;RESTORE R0 - R5
395 031464 005720          TST    (R0)+ ;SETUP FOR NORMAL RETURN
396 031466 000401          BR      10$ ;FINISH UP, THEN EXIT
397 031470 104413          9$: RESREG ;RESTORE R0 - R5
398 031472 005720          10$: TST   (R0)+ ;CORRECT THE RETURN ADDRESS
399 031474 105037 030404  CLRB  ACTDRV ;CLEAR 'ACTIVE DRIVER' FLAG
400 031500 012637 177776  MOV    (SP)+,PS ;RETURN 'PS' TO USER LEVEL
401 031504 000200          RTS    R0 ;RETURN TO CALLER
402
403 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
404
405 ;CALL
406 ;
407 ;
408 ;
409 031506 104412          OPT: SAVREG ;SAVE R0 - R5
410 031510 013746 177776  MOV    PS,-(SP) ;SAVE PROC. STATUS
411 031514 146137 030444 030402 BICB  ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG
412 031522 105061 030370  CLRB  DPRQS(R1) ;RESET THE PORT REQ FLAG ****
413 031526 004737 036316  JSR    PC,GETREQ ;GET 'DPB' POINTER OF REQUEST
414 031532 005702          TST    R2 ;IS THERE A REQUEST IN QUEUE?
415 031534 001466          BEQ    7$ ;NO--BR TO EXIT
416 031536 010164 000010  MOV    R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
417 031542 012764 000111 000000 MOV    #111,RMCS1(R4) ;CLEAR THE DRIVE
418 031550 032764 004000 000000 BIT    #BIT11,RMCS1(R4) ;DVA SET ?
419 031556 001442          BEQ    5$ ;TO PORT REQUEST, IF NOT
420 031560 105761 030340 1$: TSTB  DRVSTA(R1) ;IS DRIVE ONLINE?
421 031564 003014          BGT    2$ ;YES
422 031566 004737 036340  JSR    PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
423 031572 012762 140000 000016 MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
424 031600 105761 030340  TSTB  DRVSTA(R1) ;IS DRIVE UNSAFE ?
425 031604 100047          BPL    8$ ;BR TO EXIT IF NOT
426 031606 012762 110000 000016 MOV    #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
427 031614 000443          BR      8$ ;BR TO EXIT
428 031616          2$:
429 031616 122762 000150 000002 CMPB  #150,2(R2) ;IS THE REQUEST FOR I/O?
430 031624 002403          BLT    3$ ;YES
431 031626 004737 032150  JSR    PC,C14 ;CALL THE COMMAND INITIATOR
432 031632 000434          BR      8$ ;BR TO EXIT
433 031634 005737 030442 3$: TST   DTUW ;DATA TRANSFER UNDERWAY?
434 031640 002006          BGE    4$ ;YES--GO START A SEARCH
435 031642 005737 030420  TST   SEEKFG ;DO IMPLIED SEEKS?

```



```

444 031646 100003          BPL      4$          ;NO. DO A SEARCH
445 031650 004737 031734  JSR      PC,C11     ;START A DATA TRANSFER
446 031654 000423          BR       8$
447 031656 004737 032042  4$: JSR      PC,C13     ;START A SEARCH
448 031662 000420          BR       8$          ;GO TO THE EXIT
449 031664 112761 177777 030370 5$: MOVB   #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
450 031672 010103          MOV      R1,R3      ;SET UP TO ADDRESS WORDS
451 031674 006303          ASL     R3          ;CONVERT TO WORD INDEX
452 031676 012763 035230 030422  MOV     #15000.,TIMER(R3) ;START 15. SECOND TIMER
453 031704 000402          BR       7$          ;EXIT
454 031706 004737 032564  6$: JSR      PC,C17     ;PROCESS THE PARITY ERROR
455 031712 032714 000100  7$: BIT     #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
456 031716 001002          BNE     8$          ;BR IF SET
457 031720 004737 035600          JSR      PC,SET.IE  ;SET 'IE' WITHOUT A 'TRE'
458 031724 012637 177776  8$: MOV     (SP)+,PS    ;RESTORE PROC. STATUS
459 031730 104413          RESREG
460 031732 000207          RTS     PC          ;RESTORE R0 - R5
461
462          ;COMMAND INITIATOR
463          ;
464          ;CALL
465          ;      MOV     #DRVNUM,R1    ;DRIVE NUMBER
466          ;      MOV     #DPB,R2      ;ADDRESS OF DPB
467          ;      JSR     PC,C1?       ;C1?= C11,C13, OR C14
468          ;      ;WHERE:
469          ;      ;C11=DATA TRANSFER
470          ;      ;C13=SEARCH REQUESTED BY DATA XFER
471          ;      ;C14=NOT DATA TRANSFER
472          ;
473 031734 004737 036340  C11: JSR     PC,POPQUE   ;REMOVE REQUEST FROM 'DRIVES WAIT' QUEUE
474 031740 010237 030400  MOV     R2,TRNSWT   ;PUT REQ. IN TRANSFER WAIT QUEUE
475 031744 010203          MOV     R2,R3      ;DPB ADDRESS TO R3
476 031746 013704 030456  MOV     RMADR,R4    ;RMCS1 ADDRESS
477 031752 010164 000010  MOV     R1,RMCS2(R4) ;SELECT DRIVE
478 031756 062703 000004  ADD     #4,R3      ;DESIRED WORD COUNT
479 031762 062704 000002  ADD     #2,R4      ;RMWC ADDRESS
480 031766 012324          MOV     (R3)+,(R4)+ ;LOAD WORD COUNT
481 031770 012324          MOV     (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
482 031772 012346          MOV     (R3)+,-(SP) ;LOAD SECTOR AND TRACK
483 031774 004037 035270  JSR     R0,WRT.RM   ;CALL THE LOAD(WRITE) ROUTINE
484 032000 000006          RMDA   C17         ;INDEX OF REGISTER TO LOAD
485 032002 032564          CI7
486 032004 012346          MOV     (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
487 032006 004037 035270  JSR     R0,WRT.RM
488 032012 000034          RMDC   C17
489 032014 032564          CI7
490 032016 016246 000002  MOV     2(R2),-(SP) ;LOAD 'COMMAND+GO', 'A17&A16', AND 'PSEL'
491 032022 004037 035270  JSR     R0,WRT.RM
492 032026 000000          RMCS1
493 032030 032564          CI7
494 032032 010137 030442  MOV     R1,DTUW    ;SET 'DATA TRANSFER UNDERWAY'
495 032036 000137 032526  JMP     C15
496
497 032042 013704 030456  C13: MOV     RMADR,R4    ;RMCS1 ADDRESS
498 032046 010164 000010  MOV     R1,RMCS2(R4) ;SELECT DRIVE
499 032052 016246 000012  MOV     12(R2),-(SP) ;DESIRED CYLINDER ADDRESS
500 032056 004037 035270  JSR     R0,WRT.RM

```


501	032062	000034		RMDC		
502	032064	032564		CI7		
503	032066	116203	000010	MOVB	10(R2),R3	:PICKUP SECTOR ADDRESS
504	032072	163703	030464	SUB	MXWINDW,R3	:BACKUP BY MAX. SEARCH FOR I/O WINDOW
505	032076	002002		BGE	1\$	
506	032100	062703	000040	ADD	#32,R3	
507	032104	010346		MOV	R3,-(SP)	:COMBINE THE ADJUSTED SECTOR WITH
508	032106	116266	000011	MOVB	11(R2),1(SP)	:THE DESIRED TRACK
509	032114	004037	035270	JSR	R0,WRT.RM	:LOAD DESIRED TRACK & SECTOR
510	032120	000006		RMDA		
511	032122	032564		CI7		
512	032124	012746	000131	MOV	#131,-(SP)	:START A SEARCH
513	032130	004037	035270	JSR	R0,WRT.RM	
514	032134	000000		RMCS1		
515	032136	032564		CI7		
516	032140	156137	030444	BISB	ATABIT(R1),SRCHWT	:SET "SEARCH WAIT" KEY
517	032146	000567		BR	CI5	
518						
519	032150	013704	030456	MOV	RMADR,R4	:RMCS1 ADDRESS
520	032154	010164	000010	MOV	R1,RMCS2(R4)	:SELECT DRIVE
521	032160	116203	000002	MOVB	2(R2),R3	:PICKUP THE REQUESTED COMMAND
522	032164	122703	000131	CMPB	#131,R3	:IS IT A SEARCH COMMAND?
523	032170	001007		BNE	1\$:BR IF NO
524	032172	016246	000010	MOV	10(R2),-(SP)	:LOAD DESIRED TRACK & SECTOR
525	032176	004037	035270	JSR	R0,WRT.RM	
526	032202	000006		RMDA		
527	032204	032564		CI7		
528	032206	000403		BR	2\$:GO LOAD CYLINDER
529	032210	122703	000105	CMPB	#105,R3	:IS IT A SEEK COMMAND
530	032214	001007		BNE	3\$:BR IF NO
531	032216	016246	000012	MOV	12(R2),-(SP)	:LOAD DESIRED CYLINDER
532	032222	004037	035270	JSR	R0,WRT.RM	
533	032226	000034		RMDC		
534	032230	032564		CI7		
535	032232	000546		BR	CI6	
536	032234	122703	000115	CMPB	#115,R3	:IS IT AN "OFFSET" COMMAND?
537	032240	001013		BNE	4\$:BR IF NO
538	032242	004037	035110	JSR	R0,RD.RM	:MERGE THE OFFSET VALUE INTO RMOF
539	032246	000032		RMOF		:BUT DON'T CHANGE THE UPPER
540	032250	032564		CI7		
541	032252	116216	000001	MOVB	1(R2),(SP)	:BYTE WHEN LOADING THE
542	032256	004037	035270	JSR	R0,WRT.RM	:REGISTER (RMOF)
543	032262	000032		RMOF		
544	032264	032564		CI7		
545	032266	000530		BR	CI6	:GO START THE COMMAND
546	032270	122703	000107	CMPB	#107,R3	:IS IT A "RECALIBRATE" COMMAND?
547	032274	001525		BEQ	CI6	:BR IF YES
548	032276	122703	000117	CMPB	#117,R3	:IS IT A RETURN TO CENTER?
549	032302	001522		BEQ	CI6	:BR IF YES
550	032304	122703	000103	CMPB	#103,R3	:IS IT AN "UNLOAD" COMMAND?
551	032310	001016		BNE	5\$:BR IF NO
552	032312	112761	000001	MOVB	#1,DRVACT(R1)	:SET THE DRIVE ACTIVE INDICATOR
553	032320	105061	030340	CLRB	DRVSTA(R1)	:PUT DRIVE STATUS TO OFFLINE
554	032324	112761	000001	MOVB	#1,ULDFLG(R1)	:SET "UNLOAD IN PROGRESS" FLAG
555	032332	010346		MOV	R3,-(SP)	:START THE "UNLOAD" COMMAND
556	032334	004037	035270	JSR	R0,WRT.RM	
557	032340	000000		RMCS1		


```

624 032606 012746 000111          C17B:  MOV    #111,-(SP)      ;DO A 'DRIVE CLEAR'
625 032612 004037 035270          JSR    RO,WRT.RM
626 032616 000000          RMCS1
627 032620 032664          C18
628 032622 004737 036222          1$:   JSR    PC,EMPTYQ      ;EMPTY THE QUEUE
629 032626 105061 030370          CLR   DPRQS(R1)      ;CLEAR THE PORT REQUEST FLAG
630 032632 105061 030406          CLR   ULDFLG(R1)     ;CLEAR THE UNLOAD IN QUEUE FLAG
631 032636 105061 030330          CLR   DRVACT(R1)     ;DRIVE IS IDLE
632 032642 020237 030400          CMP   R2,TRNSWT      ;IF THIS DRIVE HAD AN I/O REQUEST
633 032646 001005          BNE   2$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
634 032650 005037 030400          CLR   TRNSWT
635 032654 012737 177777 030442 2$:   MOV    #-1,DTUW
636 032662 000207          RTS    PC
640
641 032664 104412          C18:   SAVREG              ;SAVE R0 - R5
642 032666 005001          CLR   R1
643 032670 005003          CLR   R3
644 032672 105761 030330          1$:   TSTB   DRVACT(R1)     ;DRIVE ACTIVE?
645 032676 001003          BNE   2$              ;BR IF IN ACTIVE
646 032700 105761 030370          TSTB   DPRQS(R1)     ;PORT REQUEST
647 032704 001443          BEQ   6$              ;BR IF NOT
648 032706 013702 030400          2$:   MOV    TRNSWT,R2     ;GET THE 'TRANSFER WAIT' QUEUE
649 032712 020137 030442          CMP   R1,DTUW        ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
650 032716 001402          BEQ   3$              ;BR IF YES
651 032720 004737 036316          JSR    PC,GETREQ      ;GET THE DPB POINTER
652 032724 005702          3$:   TST   R2              ;QUEUE ENTRY FOR DRIVE ?
653 032726 001413          BEQ   5$              ;BR IF NOT
654 032730 032764 010000 000010  BIT   #BIT12,RMCS2(R4) ;'NED' SET ?
655 032736 001404          BEQ   4$              ;BR IF NOT
656 032740 012762 100002 000016  MOV   #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
657 032746 000403          BR    5$              ;CONTINUE
658 032750 012762 102000 000016 4$:   MOV   #BIT15!BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
662 032756 012763 177777 030422 5$:   MOV   #-1,TIMER(R3)   ;STOP THE TIMER
663 032764 105061 030330          CLR   DRVACT(R1)     ;SET 'DRIVE ACTIVE' TO IDLE
664 032770 105061 030370          CLR   DPRQS(R1)     ;CLEAR PORT REQUEST FLAG
665 032774 020137 030442          CMP   R1,DTUW        ;IS THIS DRIVE SETUP FOR A TRANSFER
666 033000 001005          BNE   6$              ;BR IF NOT
667 033002 012737 177777 030442  MOV   #-1,DTUW        ;RESET THE INDICATOR
668 033010 005037 030400          CLR   TRNSWT         ;CLEAR THE TRANSFER QUEUE
669 033014 105061 030406          6$:   CLR   ULDFLG(R1)     ;CLEAR UNLOAD FLAG
670 033020 032764 010000 000010  BIT   #BIT12,RMCS2(R4) ;'NED' SET ?
674 033026 005201          INC   R1              ;MOVE TO THE NEXT DRIVE
675 033030 062703 000002          ADD   #2,R3
676 033034 042701 177770          BIC   #^C7,R1
677 033040 001314          BNE   1$              ;BR IF MORE DRIVES
678 033042 012737 177777 030442  MOV   #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY
679 033050 005037 030400          CLR   TRNSWT         ;CLEAR THE 'TRANSFER WAIT' QUEUE
680 033054 004737 036144          JSR    PC,CLRQUE      ;CLEAR ALL OF THE REQUEST QUEUES
681 033060 012764 000040 000010  MOV   #CLR,RMCS2(R4) ;DO A MASSBUS INIT.
682 033066 000406          BR    8$              ;CONTINUE
683 033070 004737 036222          7$:   JSR    PC,EMPTYQ      ;CLEAR THE DRIVE'S QUEUE
684 033074 105061 030340          CLR   DRVSTA(R1)     ;SET DRIVE TO OFFLINE
685 033100 105061 030350          CLR   DRVTP(R1)     ;CLEAR THE DRIVE TYPE INDICATOR
686 033104 004737 035600          8$:   JSR    PC,SET.IE     ;SET 'IE' WITHOUT 'TRE'
687 033110 104413          RESREG              ;RESTORE R0 - R5
688 033112 000207          RTS    PC              ;RETURN
689

```



```

690 ;INTERRUPT SERVICE ROUTINE
691
692 033114 112737 000001 030404 1SR:  MOVB  #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
693 033122 104412 SAVREG ;SAVE R0 - R5
694 033124 013704 030456 MOV  RMADR,R4 ;ADDRESS OF RMCS1
695 033130 013701 030442 MOV  DTUW,R1 ;GET "DATA TRANSFER UNDERWAY" INDICATOR
696 033134 002402 BLT  1$ ;BR IF NO DATA TRANSFER UNDERWAY
697 033136 004737 033156 JSR  PC,TD ;CALL TRANSFER DONE
698 033142 004737 033326 1$: JSR  PC,SC ;CALL SPECIAL CONDITIONS
699 033146 104413 2$: RESREG ;RESTORE R0 - R5
700 033150 105037 030404 CLRB  ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
701 033154 000002 RTI  ;RETURN
702
703 ;TRANSFER DONE ROUTINE
704
705 033156 105061 030330 TD:  CLRB  DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
706 033162 012737 177777 030442 MOV  #-1,DTUW ;NO DATA TRANSFERS UNDERWAY
707 033170 006301 ASL  R1
708 033172 012761 177777 030422 MOV  #-1,TIMER(R1) ;CANCEL TIMEOUT
709 033200 006201 ASR  R1
710 033202 013702 030400 MOV  TRNSWT,R2 ;GET "DPB" ADDRESS FROM THE
711 033206 005037 030400 CLR  TRNSWT ;TRANSFER WAIT QUEUE--CLEAR QUEUE
712 033212 052762 000200 000016 BIS  #BIT07,16(R2) ;SET DONE
713 033220 010164 000010 MOV  R1,RMCS2(R4) ;SELECT THE DRIVE
714 033224 004037 035110 JSR  RO,RD.RM ;TRANSFER ERROR(TRE=1)?
715 033230 000000 RMCS1
716 033232 032564 CI7
717 033234 006126 ROL  (SP)+ ;IS TRE=1 ?
718 033236 100417 BMI  3$ ;BR IF YES
719 033240 005737 030416 TST  SAVEFG ;SAVE THE RH/RM REGISTERS?
720 033244 100002 BPL  1$ ;BR IF NO
721 033246 004737 035462 JSR  PC,SVRH70 ;YES--SAVE THE REGISTERS
726 033252 004737 036316 1$: JSR  PC,GETREQ ;GET DPB POINTER
728 033256 005702 TST  R2 ;ENTRY FOR DRIVE ?
729 033260 001403 BEQ  2$ ;BR IF NOT
730 033262 004737 031506 JSR  PC,OPT ;CALL OPTIMIZER
731 033266 000207 RTS  PC ;RETURN
732 033270 012714 000113 2$: MOV  #113,(R4) ;RELEASE THE DRIVE
733 033274 000207 RTS  PC ;RETURN
734
735 033276 052762 100100 000016 3$: BIS  #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
736 033304 004737 036222 JSR  PC,EMPTYQ ;EMPTY THE "DRIVE'S WAIT" QUEUE
737 033310 004737 035462 JSR  PC,SVRH70 ;SAVE THE RH/RM REGISTERS
738 033314 012714 040111 MOV  #40111,(R4) ;ISSUE A "DRIVE CLEAR"
739 033320 012714 000113 MOV  #113,(R4) ;ISSUE A RELEASE TO THE DRIVE
740 033324 000207 RTS  PC ;RETURN
741
742 ;SPECIAL CONDITION ROUTINE
743
744
745
746 033326 116403 000016 SC:  MOVB  RMAS(R4),R3 ;READ "RMAS"
747 033332 001014 BNE  2$ ;BR IF ANY "ATA" BITS SET
748 033334 004037 035110 JSR  RO,RD.RM ;READ CONTROL AND STATUS REGISTER
749 033340 000000 RMCS1
750 033342 032664 CI8
751 033344 106126 ROLB (SP)+ ;IS "IE"=1?
752 033346 100405 BMI  1$ ;YES, NO DRIVES TO CHECK
753 033350 004037 036406 JSR  RO,ES.SAV ;SAVE THE ADDRESS IN "$ESCAPE"

```


774	033354	104001			EMT	1	:REPORT AN ILLEGAL INTERRUPT
	033356	004737	035600		JSR	PC,SET.IE	:SET INTERRUPT ENABLE
775	033362	000207		1\$:	RTS	PC	:RETURN
776	033364	005046		2\$:	CLR	-(SP)	:PROCESS ALL DRIVES THAT HAVE
777	033366	110316			MOV	R3,(SP)	:AN 'ATA'=1
778	033370	012703	000001		MOV	#1,R3	
779	033374	005001			CLR	R1	
780							
781	033376	030316		3C3:	BIT	R3,(SP)	:ATA=1?
782	033400	001005			BNE	SC5	:YES
783							
784	033402	005201		3C4:	INC	R1	:MOVE TO THE NEXT DRIVE
785	033404	106303			ASLB	R3	
786	033406	001373			BNE	SC3	:BR IF MORE TO CHECK?
787	033410	005726			TST	(SP)+	:CLEAN OFF THE STACK
788	033412	000207			RTS	PC	:RETURN TO USER
789							
790	033414	105761	030360	3C5:	TSTB	DPINT(R1)	:INITIALIZING THE DRIVE ?
791	033420	001402			BEQ	1\$:BR IF NOT
792	033422	000137	034340		JMP	SC13	:PROCESS THE DRIVE
793	033426	105761	030370	1\$:	TSTB	DPRQS(R1)	:PORT REQUEST OUTSTANDING ?
794	033432	001402			BEQ	2\$:BR IF NOT
795	033434	000137	034340		JMP	SC13	:START THE OUTSTANDING COMMAND
796	033440	105761	030340	2\$:	TSTB	DRVSTA(R1)	:CHECK THE DRIVE STATUS
797	033444	003023			BGT	4\$:BR IF ONLINE
798	033446	105761	030406		TSTB	ULDFLG(R1)	:UNLOAD IN PROGRESS?
799	033452	003420			BLE	4\$:BR IF NOT
800	033454	004737	036316		JSR	PC,GETREQ	:GET DPB POINTER
801	033460	004737	035462		JSR	PC,SVRH70	:SAVE THE RH/RM REGISTERS
802	033464	004737	034270		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
803							:ALSO DO A DRIVE INIT (DRVINT)
804	033470	105761	030340		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
805	033474	003414			BLE	5\$:NO
806	033476	032737	040000	030320	BIT	#BIT14,RMERRS	:WAS THERE AN ERROR?
807	033504	001000			BNE	3\$:BR IF ERROR
811	033506	013705	030322	3\$:	MOV	RMERRS+2,R5	:YES -- PICKUP RMER1 AND
812	033512	000504			BR	SC6A	:GO PROCESS THE ERROR
813	033514	105761	030330	4\$:	TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
814	033520	001033			BNE	SC6	:BR IF EITHER
815	033522	004737	034270		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
816							:ALSO DO A DRVINT
817	033526	105761	030360	5\$:	TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
818	033532	001323			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
819	033534	105761	030340		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
820	033540	100412			BMI	6\$:BR IF UNSAFE
821	033542	032737	020000	030324	BIT	#BIT13,RMERRS+4	:ADDRESS PLUG CHANGED ?
822	033550	001013			BNE	7\$:BR IF YES
826	033552	012746	000111		MOV	#111, -(SP)	:DRIVE CLEAR
827	033556	004037	035270		JSR	RO,WRT.RM	:WRITE THE COMMAND INTO RMCS1
828	033562	000000			RMCS1		:REGISTER INDEX
829	033564	034130			SC8		:PARITY EXIT ADDRESS
830	033566	011605		6\$:	MOV	(SP),R5	:PICKUP (RMAS) BEFORE THE ERROR CALL
831	033570	004037	036406		JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
	033574	104002			EMT	2	:REPORT THE UNEXPECTED ATTENTION
832	033576	000701			BR	SC4	:GO CHECK FOR MORE ATA'S
833	033600			7\$:			
	033600	004037	036406		JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'


```

834 033604 104005          EMT      5          :REPORT THE ADDRESS PLUG CHANGE
835 033606 000675          BR       SC4         :CHECK FOR MORE DRIVES
836 033610 006301          SC6:   ASL      R1          :SETUP TO ADDRESS WORDS
837 033612 012761 177777 030422 MOV     #-1,TIMER(R1) :STOP THE TIMER
838 033620 006201          ASR      R1          :RESTORE THE DRIVE ADDRESS
839 033622 004737 036316 JSR     PC,GETREQ    :GET THE DPB POINTER FROM THE QUEUE
840 033626 010164 000010 MOV     R1,RMCS2(R4) :SELECT DRIVE
841 033632 000137 034160 JMP     SC11         :PROCESS THE SEARCH
842 033636 004037 035110 JSR     R0,RD.RM    :READ THE RM'S STATUS REG.
843 033642 000012          RMDS
844 033644 034130          SC8
845 033646 011605          MOV     (SP),R5     :AND PUT IT IN R5
846 033650 006126          ROL     (SP)+       :WAS THERE AN ERROR?
847 033652 100407          BMI     1$         :BR IF ERROR
848 033654 105761 030330 TSTB   DRVACT(R1)  :CHECK DRIVE'S STATE
849 033660 003137          BGT     SC11       :BR IF DRIVE ACTIVE WITH ORDER
850 033662 052762 100210 000016 BIS     #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
851 033670 000470          BR
852 033672 004037 035110 1$:   JSR     R0,RD.RM    :READ ERROR REGISTER #1
853 033676 000014          RMER1
854 033700 034130          SC8
855 033702 012605          MOV     (SP)+,R5   :AND SAVE IT IN R5
856 033704 004737 035462 JSR     PC,SVRH70   :SAVE RH/RM REGISTERS
857 033710 012746 000111 MOV     #111,-(SP)  :ISSUE A DRIVE CLEAR
858 033714 004037 035270 JSR     R0,WRT.RM
859 033720 000000          RMCS1
860 033722 034130          SC8
861
862 033724 006105          SC6A:  ROL     R5          :WAS 'UNSAFE' CONDITION =1?
863 033726 100406          BMI     1$         :BR IF YES
864 033730 005702          TST     R2          :ANYTHING IN QUEUE ?
865 033732 001447          BEQ     SC7         :BR IF NOT
866 033734 052762 100240 000016 BIS     #BIT15!BIT07!BIT05,16(R2) ;INFORM USER OF ERROR
867 033742 000443          BR
868 033744 004037 035110 1$:   JSR     R0,RD.RM    :READ DRIVE STATUS REG. #1
869 033750 000012          RMDS
870 033752 034130          SC8
871 033754 011605          MOV     (SP),R5   :SAVE RMDS IN R5
872 033756 006126          ROL     (SP)+       :'ERR'=1?
873 033760 100011          BPL     2$         :BR IF NO--UNSAFE CLEARED
874 033762 112761 177777 030340 MOVB   #-1,DRVSTA(R1) :DRIVE IS UNSAFE
875 033770 004737 035462 JSR     PC,SVRH70   :SAVE RH/RM REGISTERS
876 033774 052762 110000 000016 BIS     #BIT15!BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
877 034002 000423          BR
878 034004 032705 010000 2$:   BIT     #BIT12,R5    :'MOL' = 1 ?
879 034010 001015          BNE     3$         :BR IF YES
880 034012 112761 177777 030330 MOVB   #-1,DRVACT(R1) :ACTIVE ERROR RECOVER
881 034020 112761 000001 030340 MOVB   #1,DRVSTA(R1) :ONLINE
882 034026 006301          ASL     R1
883 034030 012761 035230 030422 MOV     #15000.,TIMER(R1) ;START 15. SECOND TIMER
884 034036 006201          ASR     R1
885 034040 000137 033402 JMP     SC4
886 034044 052762 100220 000016 3$:  BIS     #BIT15!BIT07!BIT04,16(R2) ;INFORM USER OF ERROR
887
888 034052 105061 030330 SC7:  (LRB   DRVACT(R1)   :DRIVE IS IDLE
892 034056 004737 036340 JSR     PC,POPQUE   :REMOVE THE QUEUE

```

```

893 034062 105761 030406      TSTB  ULDFLG(R1)      ;UNLOAD IN PROGRESS OR QUEUE?
894 034066 003002              BGT   1$              ;BR IF NOT
895 034070 105061 030406      CLRB  ULDFLG(R1)      ;CLEAR UNLOAD FLAG
896 034074 116164 030444 000016 1$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
897 034102 105761 030340      TSTB  DRVSTA(R1)      ;IS THE DRIVE UNSAFE ?
898 034106 100406              BMI   2$              ;BR IF IT IS
902 034110 012746 000111      MOV   #-111,-(SP)     ;DRIVE CLEAR COMMAND
903 034114 004037 035270      JSR   R0,WRT.RM       ;WRITE THE COMMAND INTO RPCS1
904 034120 000000              RMCS1                 ;REGISTER INDEX
905 034122 034130              SC8                    ;PARITY EXIT ADDRESS
906 034124 000137 033402      2$:  JMP   SC4            ;CHECK FOR MORE DRIVES
907
908 034130 105761 030330      SC8:  TSTB  DRVACT(R1)   ;IS DRIVE IDLE?
909 034134 001405              BEQ   1$              ;YES
910 034136 004737 036316      JSR   PC,GETREQ       ;GET DPB POINTER
911 034142 004737 032564      JSR   PC,C17          ;PROCESS THE PARITY ERROR
912 034146 000402              BR    2$              ;CONTINUE
913 034150
917 034150 004737 032606      1$:  JSR   PC,C17B        ;PROCESS THE UNCORRECTABLE PARITY ERROR
918 034154 000137 033402      2$:  JMP   SC4            ;CHECK MORE DRIVES
919
920 034160 105761 030406      SC11: TSTB  ULDFLG(R1)   ;'UNLOAD IN PROGRESS'?
921 034164 003402              BLE   1$              ;BR IF NO
922 034166 105061 030406      CLRB  ULDFLG(R1)      ;CLEAR UNLOAD FLAG
923 034172 105061 030330      1$:  CLRB  DRVACT(R1)     ;SET DRIVE IDLE
924 034176 136137 030444 030402  BITB  ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
925                                     ;AN I/O COMMAND?
926 034204 001012              BNE   2$              ;BR IF YES
927 034206 004737 036340      JSR   PC,POPQUE       ;REMOVE REQUEST FROM QUEUE
928 034212 052762 000200 000016  BIS   #BIT07,16(R2)   ;SET 'DONE' BIT
929 034220 005737 030416      TST   SAVEFG          ;SAVE THE REGISTERS?
930 034224 100002              BPL   2$              ;BR IF NO
931 034226 004737 035462      JSR   PC,SVRH70       ;YES--SAVE ALL OF THE RH/RM REG'S
932 034232 116164 030444 000016  2$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
933 034240 146137 030444 030402  BICB  ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
934 034246 006301              ASL   R1               ;WORD INDEX
935 034250 012761 177777 030422  MOV   #-1,TIMER(R1)   ;STOP CLOCK
936 034256 006201              ASR   R1               ;RESTORE R1
937 034260 004737 031506      JSR   PC,OPT          ;START A REQUEST
938 034264 000137 033402      JMP   SC4            ;CHECK FOR MORE DRIVES
939
940 034270 010164 000010      SC12: MOV   R1,RMCS2(R4)    ;SELECT DRIVE
941 034274 016437 000012 030320  MOV   RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
942 034302 016437 000014 030322  MOV   RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
943 034310 016437 000042 030324  MOV   RMER2(R4),RMERRS+4
944 034316 016437 000040 030326  MOV   RMMR2(R4),RMERRS+6
945 034324 004037 030700      JSR   R0,DRVINT       ;INIT. THE STATE OF THE DRIVE
946 034330 000401              BR    1$              ;TAKE ERROR EXIT
947 034332 000207              RTS   PC               ;RETURN
948 034334 005726      1$:  TST   (SP)+          ;POP PC OFF OF THE STACK
949 034336 000674              BR    SC8              ;PROCESS THE PARITY ERROR
950
951 034340 006301      SC13: ASL   R1               ;SETUP TO ADDRESS WORDS
952 034342 012761 177777 030422  MOV   #-1,TIMER(R1)   ;STOP THE TIMER
953 034350 006201              ASR   R1               ;
954 034352 010164 000010      MOV   R1,RMCS2(R4)    ;SELECT THE DRIVE
955 034356 116164 030444 000016  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT

```



```

956 034364 105761 030360 1$: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
957 034370 001424 BEQ 2$ ;BR IF NOT
958 034372 105061 030360 CLR B DPINT(R1) ;CLEAR THE INIT INDICATOR
959 034376 004037 030700 JSR R0,DRVINT ;GO INIT THE DRIVE
960 034402 000240 NOP ;DUMMY PARITY ERROR RETURN
961 034404 105761 030340 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
962 034410 003014 BGT 2$ ;BR IF YES -- START ORDER
963 034412 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
964 034414 001426 BEQ 3$ ;BR IF NOT
965 034416 004737 036316 JSR PC,GETREQ ;GET DPB ADDRESS
966 034422 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
967 034430 004737 035462 JSR PC,SVRH70 ;SAVE THE REGISTERS
971 034434 004737 036340 JSR PC,POPQUE ;REMOVE THE QUEUE
972 034440 000414 BR 3$
973 034442 032764 004000 000000 2$: BIT #BIT11,RMCS1(R4) ;DVA SET ?
974 034450 001006 BNE 4$ ;SET THEN CALL OPT
975 034452 006301 ASL R1
976 034454 012761 035230 030422 MOV #15000.,TIMER(R1) ;START 15. SECOND TIMER
977 034462 006201 ASR R1
978 034464 000402 BR 3$
979 034466 004737 031506 4$: JSR PC,OPT ;START THE PENDING REQUEST
980 034472 000137 033402 3$: JMP SC4 ;PROCESS OTHER DRIVES
981
982 ;RM TIMER ROUTINE
983 ;CALL
984 ;
985 ; MOV #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
986 ; JSR PC,RMTMR ;CALL RM05 TIME ROUTINE
987 034476 005737 030404 RMTMR: TST ACTDRV ;CHECK 'ACTDRV & ACTSTR'
988 034502 001027 BNE 4$ ;IF NON ZERO EXIT
989 034504 112737 000001 030405 MOV B #1,ACTSTR ;SET 'ACTSTR'
990 034512 104412 SAVREG ;SAVE R0 - R5
991 034514 005001 CLR R1 ;START WITH DRIVE 0
992 034516 005003 CLR R3
993 034520 005763 030422 1$: TST TIMER(R3) ;IS THE TIMER RUNNING?
994 034524 002406 BLT 2$ ;BR IF NO
995 034526 166663 000002 030422 SUB 2(SP),TIMER(R3) ;COUNT THE INTERVAL
996 034534 003002 BGT 2$ ;BR IF NO SOFTWARE TIMEOUT
997 034536 004737 034566 JSR PC,STO ;CALL SOFTWARE TIMEOUT ROUTINE
998 034542 005201 2$: INC R1 ;MOVE TO NEXT DRIVE
999 034544 005723 TST (R3)+
1000 034546 022701 000010 CMP #8.,R1 ;OUT OF DRIVES?
1001 034552 003362 BGT 1$ ;BR IF NO
1002 034554 104413 3$: RESREG ;RESTORE R0 - R5
1003 034556 105037 030405 CLR B ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1004 034562 012616 4$: MOV (SP)+,(SP) ;ADJUST THE STACK
1005 034564 000207 RTS PC ;RETURN
1006
1007 ;SOFTWARE TIMEOUT ROUTINE
1008 ;
1009 ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1010 ; OR GREATER
1011 ;
1012 ;CALL: STO
1013 ; MOV #DRVNUM,R1 ;DRIVE NUMBER
1014 ; JSR PC,STO ;CALL
1015 ; RETURN

```

```

1016
1017 034566 010146          STO:  MOV    R1,-(SP)      ;SAVE R1
1018 034570 010246          MOV    R2,-(SP)      ;SAVE R2
1019 034572 010346          MOV    R3,-(SP)      ;SAVE R3
1020 034574 010446          MOV    R4,-(SP)      ;SAVE R4
1021 034576 013704 030456    MOV    RMADR,R4      ;GET ADDRESS OF 'RMCS1'
1022 034602 010164 000010    MOV    R1,RMCS2(R4)  ;SELECT THE DRIVE
1023 034606 004037 035110    JSR    R0,RD.RM      ;READ 'DRIVE STATUS REG'
1024 034612 000012          RMDS
1028 034614 035076          ST09
1029 034616 105726          TSTB   (SP)+         ;IS 'DRY'=1?
1030 034620 100436          BMI    ST02          ;BR IF YES
1031 034622 105761 030360    ST01: TSTB   DPINT(R1)   ;TRYING TO INTIALIZE THE DRIVE ?
1032 034626 001033          BNE    ST02          ;BR IF YES
1033 034630 105761 030370    TSTB   DPRQS(R1)    ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1034 034634 001030          BNE    ST02          ;BR IF YES
1035 034636 013702 030400    MOV    TRNSWT,R2     ;PICKUP TRANSFER WAIT QUEUE
1036 034642 020137 030442    CMP    R1,DTUW       ;TRANSFER UNDERWAY ON THIS DRIVE?
1037 034646 001404          BEQ    1$           ;BR IF YES
1038 034650 000137 035076    JMP    ST09          ;IF NOT DON'T BOTHER DRIVES
1039 034654 004737 036316    JSR    PC,GETREQ     ;GET DPB ADDRESS
1040 034660 052762 101000 000016 1$:  BIS    #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
1041 034666 004737 035462    JSR    PC,SVRH70     ;SAVE RH/RM REGISTERS
1045 034672 105061 030330    CLRB  DRVACT(R1)    ;DRIVE IS IDLE
1046 034676 105061 030406    CLRB  ULDFLG(R1)    ;CLEAR THE UNLOAD FLAG
1047 034702 005037 030400    CLR   TRNSWT        ;CLEAR DPB ADDRESS
1048 034706 012737 177777 030442  MOV    #-1,DTUW      ;CLEAR THE TRANSFER DRIVE #
1049 034714 000470          BR     ST09          ;DON'T BOTHER OTHER DRIVES
1050
1051 034716 116405 000016    ST02: MOVB   RMAS(R4),R5 ;READ ATTENTION REG
1052 034722 136105 030444    BITB  ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
1053 034726 001007          BNE    ST03          ;YES
1054 034730 105761 030360    TSTB  DPINT(R1)     ;TRYING TO INTIALIZE THE DRIVE ?
1055 034734 001021          BNE    ST06          ;BR IF YES - DRIVE NOT ONLINE
1056 034736 105761 030370    TSTB  DPRQS(R1)    ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1057 034742 001035          BNE    ST07          ;BR IF YES - NO RESPONSE TO REQUEST
1058 034744 000454          BR     ST09          ;OTHER WISE EXIT
1059
1060 034746 105761 030360    ST03: TSTB  DPINT(R1)   ;INITIALIZING THE DRIVE ?
1061 034752 001003          BNE    1$           ;BR IF INIT PENDING
1062 034754 105761 030370    TSTB  DPRQS(R1)    ;PORT REQUEST PENDING ?
1063 034760 001446          BEQ    ST09          ;BR IF NOT
1064 034762 012763 177777 030422 1$:  MOV    #-1,TIMER(R3) ;STOP THE TIMER
1065 034770 000442          BR     ST09          ;EXIT
1066
1067 034772 004737 032664    ST05: JSR    PC,C18      ;GO HANDLE THE PARITY ERROR
1068 034776 000437          BR     ST09
1069
1070 035000 105061 030360    ST06: CLRB  DPINT(R1)   ;CLEAR THE INITIALIZE INDICATOR
1071 035004 105061 030340    CLRB  DRVSTA(R1)    ;SET UNIT OFFLINE
1072 035010 012763 177777 030422  MOV    #-1,TIMER(R3) ;STOP THE TIMER
1073 035016 004737 036316    JSR    PC,GETREQ     ;GET THE DPB ADDRESS
1074 035022 005702          TST   R2            ;REQUEST IN QUEUE ?
1075 035024 001424          BEQ    ST09          ;BR IF NOT
1076 035026 052762 140000 000016  BIS    #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1077 035034 000414          BR     ST08          ;FINISH
1078

```



```

1079 035036 012763 177777 030422 ST07: MOV #-1,TIMER(R3) :STOP THE TIMER
1080 035044 105061 030370 CLR B DPRQS(R1) :CLEAR PORT REQUEST INDICATOR
1081 035050 004737 036316 JSR PC,GETREQ :GET DPB ADDRESS
1082 035054 005702 TST R2 :QUEUE ENTRY FOR DRIVE ?
1083 035056 001407 BEQ ST09 :BR IF NONE
1084 035060 012762 100004 000016 MOV #BIT15:BIT2,16(R2) :INFORM USER OF PORT REQUEST ERROR
1085 035066 004737 036222 ST08: JSR PC,EMPTYQ :CLEAR THE QUEUE FOR THE DRIVE
1086 035072 004737 035462 JSR PC,SVRH70 :SAVE THE REGISTERS
1087 035076 012604 ST09: MOV (SP)+,R4 :RESTORE R4
1088 035100 012603 MOV (SP)+,R3 :RESTORE R3
1089 035102 012602 MOV (SP)+,R2 :RESTORE R2
1090 035104 012601 MOV (SP)+,R1 :RESTORE R1
1091 035106 000207 RTS PC :RETURN
1092
1093 :ROUTINE TO READ A RH/RM REGISTER
1094 :CALL
1095 :
1096 JSR R0,RD.RM :GO READ A REGISTER
1097 INDEX :REG. INDEX FROM BASE
1098 ERRADR :ERROR ADDRESS--PROCESS ERROR STARTING
1099 :AT THIS ADDRESS
1100 :
1101 RETURN :CONTENTS OF REG. IS ON THE STACK
1102 035110 013737 030454 035256 RD.RM: MOV MCPMX,RD.RM2 :MAX. RETRYS ALLOWED
1103 035116 011646 MOV (SP)-,(SP) :SAVE R0 FOR RETURN
1104 035120 013737 030456 035134 MOV RMADR,RD.ADR :FORM THE DESIRED ADDRESS
1105 035126 062037 035134 ADD (R0)+,RD.ADR :USING THE BASE AND THE INDEX
1106 035132 013727 RD.RM1: MOV @ (PC)+,(PC)+ :READ THE DESIRED REGISTER OF THE RM DRIVE
1107 035134 000000 RD.ADR: .WORD 0 :ADDRESS IS FORMED HERE
1108 035136 000000 RD.WRD: .WORD 0 :REG. CONTENTS PUT HERE
1109 035140 013766 035136 000002 MOV RD.WRD,2(SP) :RETURN IT TO THE USER
1110 035146 013746 030456 MOV RMADR,-(SP) :PUT THE ADDRESS ON THE STACK
1111 035152 062716 000010 ADD #RMCS2,(SP) :FORM THE ADDRESS OF RMCS2
1112 035156 032736 010000 BIT #BIT12,@(SP)+ :CHECK THE 'NED' BIT
1113 035162 001037 BNE RD.RM3 :BR IF DRIVE NON-EXISTENT
1114 035164 017746 173266 MOV @RMADR,-(SP) :READ RMCS1
1115 035170 032716 020000 BIT #BIT13,(SP) :DID MCPE SET?
1116 035174 001002 BNE 1$ :BR IF YES
1117 035176 022620 CMP (SP)+,(R0)+ :ADJUST FOR RETURN
1118 035200 000432 BR RD.RM4 :EXIT
1119 035202 1$:
035202 004037 036406 JSR R0,ES.SAV :SAVE THE ADDRESS IN '$ESCAPE'
035206 104003 EMT 3 :REPORT 'MCPE' ERROR
1120 035210 005737 030442 TST DTUW :DATA TRANSFER UNDERWAY?
1121 035214 100405 BMI 2$ :NO
1122 035216 032716 040000 BIT #BIT14,(SP) :'"TRE" = 1 ?
1123 035222 001402 BEQ 2$ :NO
1124 035224 005726 TST (SP)+ :YES--CLEAN OFF THE STACK AND
1125 035226 000415 BR RD.RM3 :TAKE THE FATAL ERROR EXIT
1126 035230 052716 040000 2$: BIS #BIT14,(SP) :CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1127 035234 000316 SWAB (SP) :POSITION BEFORE WRITING
1128 035236 013737 030456 035252 MOV RMADR,3$ :FORM ADDRESS OF HIGH BYTE
1129 035244 005237 035252 INC 3$
1130 035250 112637 MOV B (SP)+,@(PC)+ :WRITE THE HIGH BYTE OF RMCS1
1131 035252 000000 3$: .WORD 0 :ADDRESS STORAGE
1132 035254 005327 DEC (PC)+ :EXCEEDED MAX. RETRYS
1133 035256 000003 RD.RM2: .WORD 3

```

```

1134 035260 002324
1135 035262 011000
1136 035264 012616
1137 035266 000200
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148 035270 013737 030454 035446 WRT.RM: MOV MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
1149 035276 016637 000002 035356 MOV 2(SP),WRT.WD ;SAVE THE WORD TO WRITE
1150 035304 012616 MOV (SP)+,(SP) ;ADJUST THE STACK
1151 035306 012037 035360 MOV (R0)+,WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
1152 035312 001015 BNE 1$ ;BR IF NOT RMCS1
1153 035314 122737 000150 035356 CMPB #150,WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
1154 035322 002411 BLT 1$ ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1155 035324 004037 035110 JSR R0,RD.RM ;NO---COMBINE A16&A17, & PSEL WITH
1156 035330 000000 RMCS1 ;THE COMMAND BEFORE SENDING IT TO
1157 035332 035452 WRT.R3 ;THE RH/RM
1158 035334 000316 SWAB (SP)
1159 035336 042716 177770 BIC #^C7,(SP)
1160 035342 112637 035357 MOVB (SP)+,WRT.WD+1
1161 035346 063737 030456 035360 1$: ADD RMADR,WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
1162 035354 012737 WRT.R1: MOV (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
1163 035356 000000 WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
1164 035360 000000 WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
1165 035362 013746 030456 MOV RMADR,-(SP) ;PUT THE ADDRESS ON THE STACK
1166 035366 062716 000010 ADD #RMCS2,(SP) ;FORM THE ADDRESS OF RMCS2
1167 035372 032736 010000 BIT #BIT12,@(SP)+ ;CHECK THE 'NED' BIT
1168 035376 001025 BNE WRT.R3 ;BR IF DRIVE NON-EXISTENT
1169 035400 004037 035110 JSR R0,RD.RM ;CHECK FOR PARITY ERROR ON WRITE
1170 035404 000014 RMER1
1171 035406 035452 WRT.R3
1172 035410 032726 000010 BIT #BIT03,(SP)+
1173 035414 001420 BEQ WRT.R4 ;BR IF 'PAR=0'
1174 035416 016037 177776 035430 MOV -2(R0),1$ ;PICKUP THE INDEX
1175 035424 004037 035110 JSR R0,RD.RM ;READ THE REG.
1176 035430 000000 1$: .WORD 0 ;REG. INDEX
1177 035432 035452 WRT.R3 ;RETURN TO THIS ADDRESS ON ERROR
1178 035434 004037 036406 JSR R0,ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
1179 035440 104004 EMT 4 ;REPORT THE PARITY ON WRITE ERROR
1180 035442 005726 TST (SP)+ ;CLEAR OFF THE STACK
1181 035444 005327 DEC (PC)+ ;DECREMENT THE ERROR COUNT
1182 035446 000003 WRT.R2: .WORD 3 ;RETRY COUNTER
1183 035452 011000 WRT.R3: BGE WRT.R1 ;TRY AGAIN IF NOT FINISHED
1184 035454 000401 WRT.R3: MOV (R0),R0 ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1185 035456 005720 WRT.R4: BR WRT.R5 ;EXIT
1186 035460 000200 WRT.R4: TST (R0)+ ;ADJUST FOR ERROR FREE EXIT
1187 WRT.R5: RTS R0
1188 ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1189 ;

```



```

1190      ;CALL
1191      ;
1192      ;      MOV      #DPBNUM,R2      ;DPB POINTER TO R2
1193      ;      JSR      PC,SVRH70      ;SAVE THE DRIVES REG'S
1194 035462 104412      SVRH70: SAVREG      ;SAVE R0 - R5
1195 035464 005702      TST      R2      ;QUEUE ENTRY FOR THE DRIVE ?
1196 035466 001442      BEQ      6$      ;BR IF NONE
1197 035470 013704 030456      MOV      RMADR,R4
1198 035474 111264 000010      MOV      (R2),RMCS2(R4) ;SELECT DRIVE
1199 035500 016203 000014      MOV      14,(R2),R3      ;GET THE ERROR TABLE POINTER
1200 035504 001433      BEQ      6$      ;EXIT IF NO ADDRESS
1201 035506 005037 035542      CLR      3$      ;COUNTER & POINTER
1202 035512 023727 035542 000022 1$:      CMP      3$,#RMDB      ;REACHED THE BUFFER REGISTER ?
1203 035520 001006      BNE      2$      ;BR IF NOT
1204 035522 032764 000200 000010      BIT      #BIT07,RMCS2(R4) ;'OR' SET ?
1205 035530 001002      BNE      2$      ;BR IF SET
1206 035532 005023      CLR      (R3)+      ;STORE RMDB AS ZEROES
1207 035534 000405      BR      4$      ;CONTINUE
1208 035536 004037 035110      2$:      JSR      R0,RD.RM      ;READ THE SELECTED REGISTER
1209 035542 000000      3$:      .WORD      0      ;REGISTER INDEX
1210 035544 035570      5$:      MOV      (SP)+,(R3)+      ;ERROR RETURN ADDRESS
1211 035546 012623      MOV      (SP)+,(R3)+      ;STORE THE REGISTER CONTENTS
1212 035550 023727 035542 000046 4$:      CMP      3$,#RMEC2      ;REACHED THE END ?
1213 035556 001406      BEQ      6$      ;BR IF YES
1214 035560 062737 000002 035542      ADD      #2,3$      ;INCREMENT THE REGISTER INDEX
1215 035566 000751      BR      1$      ;CONTINUE READING THE REGISTERS
1216 035570 004737 032564      5$:      JSR      PC,C17      ;PROCESS THE UNCORRECTABLE PARITY ERROR
1227 035574 104413      6$:      RESREG      ;RESTORE R0 - R5
1229 035576 000207      RTS      PC      ;RETURN
1230
1231      ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
1232      ;CALL
1233      ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
1234      ;      JSR      PC,SET.IE      ;SET "IE"
1235      ;      RETURN
1236
1237 035600 010446      SET.IE: MOV      R4,-(SP)      ;SAVE R4
1238 035602 013704 030456      MOV      RMADR,R4      ;PICKUP ADDRESS OF RMCS1
1239 035606 010164 000010      MOV      R1,RMCS2(R4)      ;SELECT DRIVE
1240 035612 011446      MOV      (R4),-(SP)      ;READ RMCS1
1241 035614 052716 040000      BIS      #BIT14,(SP)      ;SET THE "TRE" BIT OF THE WORD READ
1242 035620 000316      SWAB      (SP)      ;ADJUST FOR DATO
1243 035622 112714 000100      MOV      #BIT06,(R4)      ;SET "IE"
1244 035626 032764 010000 000010      BIT      #BIT12,RMCS2(R4) ;IS "NED"=1?
1245 035634 001002      BNE      1$      ;YES--CLEAR "TRE"
1246 035636 005726      TST      (SP)+      ;CLEAN OFF THE STACK
1247 035640 000402      BR      2$
1248 035642 112664 000001      1$:      MOV      (SP)+,1(R4)      ;CLEAR "TRE"
1249 035646 012604      2$:      MOV      (SP)+,R4      ;RESTORE R4
1250 035650 000207      RTS      PC      ;RETURN TO CALLER
1251
1252      ;QUEUE COUNT
1253
1254 035652 000      QCNT:  .BYTE      0      ;DRIVE 0
1257 035653 000      .BYTE      0      ;DRIVE 1
1258 035654 000      .BYTE      0      ;DRIVE 2
1259 035655 000      .BYTE      0      ;DRIVE 3

```

```

035656 000 .BYTE 0 :DRIVE 4
035657 000 .BYTE 0 :DRIVE 5
035660 000 .BYTE 0 :DRIVE 6
035661 000 .BYTE 0 :DRIVE 7

1258
1259 ;QUEUE INPUT POINTERS
1260
1261 035662 035744 QINPT: .WORD QDRV0 :DRIVE 0
1264 035664 035764 .WORD QDRV1 :DRIVE 1
035666 036004 .WORD QDRV2 :DRIVE 2
035670 036024 .WORD QDRV3 :DRIVE 3
035672 036044 .WORD QDRV4 :DRIVE 4
035674 036064 .WORD QDRV5 :DRIVE 5
035676 036104 .WORD QDRV6 :DRIVE 6
035700 036124 .WORD QDRV7 :DRIVE 7

1265
1266 ;QUEUE OUTPUT POINTERS
1267
1268 035702 035744 QOUTPT: .WORD QDRV0 :DRIVE 0
1271 035704 035764 .WORD QDRV1 :DRIVE 1
035706 036004 .WORD QDRV2 :DRIVE 2
035710 036024 .WORD QDRV3 :DRIVE 3
035712 036044 .WORD QDRV4 :DRIVE 4
035714 036064 .WORD QDRV5 :DRIVE 5
035716 036104 .WORD QDRV6 :DRIVE 6
035720 036124 .WORD QDRV7 :DRIVE 7

1272
1273 035722 035744 QSTART: .WORD QDRV0 :DRIVE 0 START ADDRESS
1274 035724 035764 QSTOP: .WORD QDRV1 :DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1275 035726 036004 .WORD QDRV2 :STOP DRIVE 1--START DRIVE 2
1276 035730 036024 .WORD QDRV3 :STOP DRIVE 2--START DRIVE 3
1277 035732 036044 .WORD QDRV4 :STOP DRIVE 3--START DRIVE 4
1278 035734 036064 .WORD QDRV5 :STOP DRIVE 4--START DRIVE 5
1279 035736 036104 .WORD QDRV6 :STOP DRIVE 5--START DRIVE 6
1280 035740 036124 .WORD QDRV7 :STOP DRIVE 6--START DRIVE 7
1281 035742 036144 .WORD QTERM :STOP DRIVE 7
1282
1283 ;DRIVE REQUEST QUEUES
1284
1287 035744 QDRV0: .BLKW 10
035764 QDRV1: .BLKW 10
036004 QDRV2: .BLKW 10
036024 QDRV3: .BLKW 10
036044 QDRV4: .BLKW 10
036064 QDRV5: .BLKW 10
036104 QDRV6: .BLKW 10
036124 QDRV7: .BLKW 10
1288 036144 QTERM=.
1289
1290 ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1291
1292 ;CALL
1293 ; JSR PC,CLRQUE
1294
1295 036144 104412 CLRQUE: SAVREG :SAVE R0 - R5
1296 036146 012702 035652 MOV #QCNT,R2 :ZERO THE QUEUE COUNTS
1297 036152 005022 CLR (R2)+ :DRIVES 0 & 1

```



```

1298 036154 005022          CLR      (R2)+          ;DRIVES 2 & 3
1299 036156 005022          CLR      (R2)+          ;DRIVES 4 & 5
1300 036160 005022          CLR      (R2)+          ;DRIVES 6 & 7
1301 036162 012703 000010    MOV      #8.,R3          ;MOVE THE STARTING
1302 036166 012701 035722    MOV      #QSTART,R1     ;ADDRESS OF THE QUEUE INTO
1303 036172 012122          1$:     MOV      (R1)+,(R2)+   ;THE QUEUE INPUT POINTER
1304 036174 005303          DEC      R3
1305 036176 001375          BNE     1$
1306 036200 012703 000010    MOV      #8.,R3          ;MOVE THE STARTING ADDRESS
1307 036204 012701 035722    MOV      #QSTART,R1     ;OF THE QUEUE INTO THE
1308 036210 012122          2$:     MOV      (R1)+,(R2)+   ;QUEUE OUTPUT POINTER
1309 036212 005303          DEC      R3
1310 036214 001375          BNE     2$
1311 036216 104413          RESREG
1312 036220 000207          RTS      PC              ;RESTORE R0 - R5
1313
1314          ;EMPTY THE QUEUE SPECIFIED BY R1
1315          ;
1316          ;CALL
1317          ;
1318          ;     MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
1319          ;     JSR      PC,EMPTYQ
1320 036222 105061 035652    EMPTYQ: CLRB      QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
1321 036226 006301          ASL      R1
1322 036230 016161 035662 035702  MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
1323 036236 006201          ASR      R1
1324 036240 000207          RTS      PC
1325
1326          ;ROUTINE TO PUT A REQUEST IN QUEUE
1327          ;
1328          ;CALL
1329          ;
1330          ;     MOV      #DRVNUM,R1          ;DRIVE NUMBER
1331          ;     MOV      #DPB,R2           ;ADDRESS OF PARAMETER BLOCK
1332          ;     JSR      R0,DRVQUE        ;GO PUT REQUEST IN QUEUE
1333          ;     RETURN1                    ;RETURN HERE IF QUEUE IS FULL
1334          ;     RETURN2                    ;RETURN HERE IF REQUEST IS IN QUEUE
1335 036242 122761 000010 035652  DRVQUE: CMPB      #10,QCNT(R1)          ;IS QUEUE FULL?
1336 036250 001421          BEQ     2$              ;BR IF YES-TAKE RETURN1
1337 036252 105261 035652          INCB     QCNT(R1)          ;INCREMENT QUEUE COUNT
1338 036256 006301          ASL      R1
1339 036260 010271 035662          MOV      R2,@QINPT(R1)     ;PUT THIS REQUEST IN QUEUE
1340 036264 062761 000002 035662  ADD      #2,QINPT(R1)      ;UPDATE THE QUEUE POINTER
1341 036272 026161 035662 035724  CMP      QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1342 036300 001003          BNE     1$              ;BR IF NO
1343 036302 016161 035722 035662  MOV      QSTART(R1),QINPT(R1) ;YES--RESET POINTER
1344 036310 006201          1$:     ASR      R1
1345 036312 005720          TST     (R0)+            ;TAKE RETURN 2
1346 036314 000200          2$:     RTS      R0              ;RETURN TO USER
1347
1348          ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1349          ;
1350          ;CALL
1351          ;
1352          ;     MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
1353          ;     JSR      PC,GETREQ        ;GO GET THE REQUEST
1354          ;     RETURN                    ;R2='DPB' ADDRESS OF THE REQUEST
1355          ;     ;R2=0 IF NO REQUEST IN QUEUE

```



```

1355
1356 036316 005002
1357 036320 105761 035652
1358 036324 001404
1359 036326 006301
1360 036330 017102 035702
1361 036334 006201
1362 036336 000207
1363
1364
1365
1366
1367
1368
1369
1370
1371 036340 105361 035652
1372 036344 006301
1373 036346 017102 035702
1374 036352 005071 035702
1375 036356 062761 000002 035702
1376 036364 026161 035702 035724
1377 036372 001003
1378 036374 016161 035722 035702
1379 036402 006201
1380 036404 000207
1381
1383
1384
1385
1386
1387
1388
1389
1390
1391 036406 012037 036422
1392 036412 013746 001176
1393 036416 005037 001176
1394 036422 000000
1395 036424 012637 001176
1396 036430 000200

GETREQ: CLR R2
        TSTB QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
        BEQ 2$ ;NO
1$: ASL R1
   MOV @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
   ASR R1
2$: RTS PC ;RETURN TO USER

;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

POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
        ASL R1
        MOV @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
        CLR @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
        ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
        CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
        BNE 1$ ;NO--BR TO EXIT
        MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1$: ASR R1
   RTS PC ;RETURN TO USER

;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
;REPORTS AN ERROR DIRECTLY.
:CALL
: JSR R0,ES.SAV
: ERROR N ;:THE ERROR CALL
: RETURN ;:THE RETURN IS PAST THE ERROR CALL

ES.SAV: MOV (R0)+,1$ ;GET THE ERROR CALL
        MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN '$ESCAPE'
        CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
1$: .WORD 0 ;THE ERROR CALL IS MOVED HERE
   MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
   RTS R0 ;RETURN

```



```

1          .SBTTL TELETYPE MESSAGES
2
3 036432    040    117    106 UNTOFF: .ASCIZ / OFFLINE/
4 036443    040    117    116 UNTON:  .ASCIZ / ONLINE/
5 036453    040    116    117 NOTRM:  .ASCIZ @ NOT AN RM05/3/2@
6 036474    040    116    117 NOTPRS: .ASCIZ / NOT PRESENT/
7 036511    040    125    116 NOTSAF: .ASCIZ / UNSAFE/
8 036521    040    114    117 LODEV:  .ASCIZ / LOAD DEVICE/
9 036536    122    115    060 $RM02:  .ASCIZ /RM02/
10 036543   122    115    060 $RM03:  .ASCIZ /RM03/
11 036550   122    115    060 $RM05:  .ASCIZ /RM05/
12 036555   200    125    116 UNSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
13 036573   200    104    122 MUNIT:  .ASCIZ <CRLF>/DRIVE/
14 036602   054    000          $COMMA: .ASCIZ /./
15 036604   072    040    000 COLON:  .ASCIZ /:/
16 036607   040    057    040 SLASH:  .ASCIZ @ / @
17 036613   103    000          C:      .ASCIZ /C/
18 036615   124    000          T:      .ASCIZ /T/
19 036617   040          BLNKS4: .ASCII / /
20 036620   040          BLNKS3: .ASCII / /
21 036621   040          BLNKS2: .ASCII / /
22 036622   040    000          BLNKS1: .ASCIZ / /
23 036624   200    105    116 ENTADR: .ASCIZ <CRLF>/ENTER ADDRESS LIMITS:/<CRLF>
24 036654   040    077    104 MOFFLN: .ASCIZ / ?DRIVE OFFLINE/<CRLF>
25 036675   040    077    104 MDRNP:  .ASCIZ / ?DRIVE NOT PRESENT/<CRLF>
26 036722   040    077    104 MER11:  .ASCIZ / ?DRIVE NOT AVAILABLE/<CRLF>
27 036751   040    077    104 MUSDR:  .ASCIZ / ?DRIVE UNSAFE/<CRLF>
28 036771   040    077    104 MLODEV: .ASCIZ / ?DRIVE IS LOAD DEVICE/<CRLF>
29 037021   040    123    105 MSEL:   .ASCIZ / SELECTED/
30 037033   106    117    122 MFORMT: .ASCIZ /FORMAT & HEADER VERIFY/
31 037062   126    105    122 MVERFY: .ASCIZ /VERIFY HEADERS ONLY/
32 037106   054    040    117 M16BIT: .ASCIZ /, OPERATE IN 16. BIT MODE/<CRLF>
33 037141   054    040    117 M18BIT: .ASCIZ /, OPERATE IN 18. BIT MODE/<CRLF>
34 037174   040    077    111 BADENT: .ASCIZ / ?INVALID ENTRY/<CRLF>
35 037215   115    101    130 MADRER: .ASCII /MAX TRACK ADRS MUST BE EQUAL TO OR GREATER/<CRLF>
36 037270   124    110    101      .ASCIZ /THAN MIN TRACK ADRS/<CRLF>
37 037315   200    123    124 MSFOR:  .ASCIZ <CRLF>/STARTING FORMAT/<CRLF>
38 037337   200    123    124 MSQVER: .ASCIZ <CRLF>/STARTING QUICK HEADER VERIFY/<CRLF>
39 037376   200    123    124 MSVER:  .ASCIZ <CRLF>/STARTING HEADER VERIFY/<CRLF>
40 037427   200    103    117 MSCMPT: .ASCIZ <CRLF>/COMPLETED/
41 037442   054    040    124 NUMERR: .ASCIZ /, TOTAL ERRORS DETECTED /
42 037473   120    101    103 PACKSN: .ASCIZ @PACK S/N: @
43 037506   120    122    105 ADDRIS: .ASCIZ /PRESENT ADDRESS IS: /
44 037533   200    105    116 MSG1:   .ASCIZ <CRLF>/ENTER SERIAL NUMBER (OCTAL): /
45 037572   200    077    111 MSG2:   .ASCIZ <CRLF>/?INCORRECT 'MFG' BAD SECTOR INFORMATION/
46 037643   200    077    117 MSG3:   .ASCIZ <CRLF>/?OVER 126. BAD SECTORS DETECTED/
47 037704   200    077    104 MSG4:   .ASCIZ <CRLF>/?DISK IS AN ALIGNMENT PACK/
48 037740   200    123    105 MSGBLK: .ASCII <CRLF>/SELECT ONE OF THE FOLLOWING FUNCTIONS/<CRLF>
49 040007   125    120    104      .ASCII /UPDATE 'USR' BAD SECTORS.....0/<CRLF>
50 040066   120    122    111      .ASCII /PRINT 'MFG' BAD SECTORS.....1/<CRLF>
51 040145   120    122    111      .ASCII /PRINT 'USR' BAD SECTORS.....2/<CRLF>
52 040224   111    116    111      .ASCII /INITIALIZE 'USR' BAD SECTOR FILE.....3/<CRLF>
53 040303   120    122    111      .ASCII /PRINT A SECTOR OF THE LAST TRACK.....4/<CRLF>
54 040362   120    122    111      .ASCII /PRINT THIS MESSAGE.....5/<CRLF>
55 040441   103    117    116      .ASCIZ /CONTINUE.....<CR>/<CRLF>
56 040521   200    105    116 MUTLTY: .ASCIZ <CRLF>/ENTER UTILITY FUNCTION (HELP=5): /
57 040564   200    105    116 MSG5:   .ASCIZ <CRLF>/ENTER 16 BIT 'MFG' BAD SECTOR ADRS (DECIMAL)/

```


58	040642	200	105	116	MESG6:	.ASCIZ	<CRLF>/ENTER 18 BIT 'MFG' BAD SECTOR ADRS (DECIMAL)/
59	040720	200	105	116	MESG7:	.ASCIZ	<CRLF>/ENTER 16 BIT 'USR' BAD SECTOR ADRS (DECIMAL)/
60	040776	200	105	116	MESG8:	.ASCIZ	<CRLF>/ENTER 18 BIT 'USR' BAD SECTOR ADRS (DECIMAL)/
61	041054	200	103	117	MESG9:	.ASCIZ	<CRLF>/CONTENTS OF 16 BIT 'MFG' BAD SECTOR FILE (DECIMAL)/
62	041140	200	103	117	MESG10:	.ASCIZ	<CRLF>/CONTENTS OF 18 BIT 'MFG' BAD SECTOR FILE (DECIMAL)/
63	041224	200	103	117	MESG11:	.ASCIZ	<CRLF>/CONTENTS OF 16 BIT 'USR' BAD SECTOR FILE (DECIMAL)/
64	041310	200	103	117	MESG12:	.ASCIZ	<CRLF>/CONTENTS OF 18 BIT 'USR' BAD SECTOR FILE (DECIMAL)/
65	041374	200	120	122	MESG13:	.ASCIZ	<CRLF>/PRINT A 'SECTOR' OF THE LAST TRACK (OCTAL)/
66	041450	200	105	116	MSG13A:	.ASCIZ	<CRLF>/ENTER SECTOR (DECIMAL): /
67	041502	200	104	117	MESG14:	.ASCIZ	<CRLF>/DONE !/<CRLF>
68	041513	200	105	116	MESG15:	.ASCIZ	<CRLF>/ENTER 'MFG' BAD SECTOR ADRS (DECIMAL)/<CRLF>
69	041563				MESG16:		:NOT USED
70	041563	200	077	125	MESG17:	.ASCIZ	<CRLF>/?UNABLE TO WRITE 'BSF' SUCCESSFULLY/
71	041630	200	077	106	MESG18:	.ASCIZ	<CRLF>/?FATAL DRIVE ERROR DETECTED/
72	041665	200	077	106	MESG19:	.ASCIZ	<CRLF>/?FAILURE DURING HEADER VERIFICATION/
73	041732	104	117	040	MMODE:	.ASCIZ	/DO YOU WANT TO FORMAT (L) Y ? /
74	041771	104	117	040	MSIZE:	.ASCIZ	/DO YOU WANT 16. BIT MODE (L) Y ? /
75	042033	200	111	116	MSINIT:	.ASCIZ	<CRLF>/INITIALIZE BAD SECTOR FILE (L) N ? /
76	042100	200	111	116	INITUS:	.ASCIZ	<CRLF>/INITIALIZE 'USR' BAD SECTOR FILES/
77	042143	200	111	116	INITMF:	.ASCIZ	<CRLF>/INITIALIZE 'MFG' BAD SECTOR FILES/
78	042206	052	040	116	NOENTR:	.ASCIZ	/* NO ENTRIES */<CRLF>
79	042226	055	117	120	ABORT:	.ASCIZ	/-OPERATION ABORTED-/<CRLF><07><07>
80	042255	200	077	116	NOUPDT:	.ASCIZ	<CRLF>/?NO UPDATES IN 'VERIFY' MODE/<CRLF>
81	042314	200	077	116	NOINIT:	.ASCIZ	<CRLF>/?NO INITIALIZATION IN 'VERIFY' MODE/<CRLF>
82	042362	103	110	101	MSPRM:	.ASCIZ	/CHANGE DRIVE PARAMETERS (L) N ? /
83	042423	200	052	052	STARS:	.ASCIZ	<CRLF>/*****/<CRLF>
84					.EVEN		

.SBTTL ERROR MESSAGES

1									
2									
3	042446	122	110	040	EM1:	.ASCIZ	/RH CONTROLLER INTERRUPT OCCURRED (RMAS=0)/		
4	042520	125	116	105	EM2:	.ASCIZ	/UNEXPECTED ATTENTION OCCURRED/		
5	042556	115	101	123	EM3:	.ASCIZ	/MASSBUS PARITY ERROR (MCPE=1)/		
6	042614	115	101	123	EM4:	.ASCIZ	/MASSBUS PARITY ERROR (PAR=1)/		
7	042651	101	104	104	EM5:	.ASCIZ	/ADDRESS PLUG CHANGE BIT SET/		
8	042705	122	110	040	EM6:	.ASCIZ	/RH CONTROLLER DIDN'T RESPOND TO ADDRESSING/		
9	042760	124	122	101	EM7:	.ASCIZ	@TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT@		
10	043034	104	122	111	EM10:	.ASCIZ	/DRIVE OFFLINE/		
11	043052	120	105	122	EM11:	.ASCIZ	/PERSISTENT DRIVE UNSAFE ERROR/		
12	043110	125	116	103	EM12:	.ASCIZ	/UNCORRECTABLE MASSBUS PARITY ERROR/		
13	043153	123	117	106	EM13:	.ASCIZ	/SOFTWARE TIMEOUT/		
14	043174	104	122	111	EM14:	.ASCIZ	/DRIVE UNSAFE ERROR/		
15	043217	103	117	116	EM15:	.ASCIZ	@CONTROLLER/DRIVE ERROR DURING WRITE@		
16	043263	103	117	116	EM16:	.ASCIZ	@CONTROLLER/DRIVE ERROR DURING WRITE CHECK@		
17	043335	122	105	124	EM17:	.ASCIZ	@RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@		
18	043413	104	101	124	EM20:	.ASCIZ	@DATA ERROR DURING WRITE CHECK@		
19	043451	103	117	116	EM21:	.ASCIZ	@CONTROLLER/DRIVE ERROR VERIFYING HEADERS@		
20	043522	047	110	103	EM22:	.ASCIZ	@'HCE' OR 'HCRC' ERROR VERIFYING HEADERS@		
21	043572	103	131	114	EM23:	.ASCIZ	@CYLINDER FIELD IN HEADER IS NOT CORRECT@		
22	043642	127	122	111	EM24:	.ASCIZ	@WRITE CHECK ERROR@		
23	043664	111	114	114	EM25:	.ASCIZ	/ILLEGAL BAD SECTOR, PACK NOT ACCEPTABLE/		

```
1          .SBTTL DATA HEADERS
2
3 043734    122    115    101 DH1:  .ASCIZ /RMAS  RMCS1  RMER1  RMER2  RMDS  RMDC  RMDA/
4 044002    104    122    111 DH2:  .ASCIZ /DRIVE RMDS  RMER1  RMER2  RMAS  RMCS1/
5 044054    104    122    111 DH3:  .ASCIZ /DRIVE  REG ADR  DATA  RMCS1  RMER1  RMER2/
6 044125    104    122    111 DH4:  .ASCIZ /DRIVE  REG ADR   GOOD   BAD   RMCS1  RMER1  RMER2/
7 044210    122    110    040 DH6:  .ASCIZ /RH ADDRESS/
8 044223    104    122    111 DH10: .ASCIZ /DRIVE  ERR PC  RMCS1  RMCS2  RMDS  RMER1  RMER2/
9 044310    122    115    105 DH10A: .ASCIZ /RMEC1  RMEC2  RMWC  RMBA  RMDA  RMAS  RMLA/
10 044374   122    115    104 DH10B: .ASCIZ /RMDB  RMMR1  RMDT  RMSN  RMOF  CYLINDER  TRACK/
11 044452    104    122    111 DH17: .ASCIZ /DRIVE  ERR PC  CYLINDER  TRACK  SECTOR/
12 044521    104    122    111 DH20: .ASCIZ /DRIVE  ERR PC  CYLINDER  TRACK  SECTOR/
13 044570    122    115    103 DH20A: .ASCIZ /RMCS1  RMCS2  RMDS  RMER1  RMER2  RMMR2  RMEC1  RMEC2/
14 044665    122    115    127 DH20B: .ASCIZ /RMWC  RMBA  RMDA  RMAS  RMLA  RMDB  RMMR1  RMDT/
15 044761    122    115    123 DH20C: .ASCIZ /RMSN  RMOF  CYLINDER  TRACK/
16 045015    104    122    111 DH23: .ASCIZ /DRIVE  ERR PC  EXPCTD  RECVD/
17 045054    040    040    040 DH24: .ASCII  /
18 045142    104    122    111          .ASCIZ /DRIVE  ERR PC  CYLNDR  TRACK  SECTOR  MEMORY  DISK/<CR><LF>
19 045227    104    122    111          .ASCIZ /DRIVE  CYLNDR  TRACK  SECTOR  /          DATA  DATA/
20
```


.SBTTL DATA TABLES

Line	Address	Offset	Label	Field	Description
1					
2					
3	045270	001370	005560	005574	DT1: .WORD ATTN, RM.REG, RM.REG+14, RM.REG+42, RM.REG+12, RM.REG+34, RM.REG+6
4	045306	001366	030320	030322	DT2: .WORD DDRIVE, RMERRS, RMERRS+2, RMERRS+4, ATTN, RMCS1
5	045322	001366	035134	035136	DT3: .WORD DDRIVE, RD.ADR, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
6	045336	001366	035360	035356	DT4: .WORD DDRIVE, WRT.AD, WRT.WD, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
7	045354	001274			DT6: .WORD \$RMADR
8	045356	001222	001132	046232	DT7: .WORD DRIVE, \$ERRPC, RBUF+6, RBUF+2
9	045366	001222	001132	005560	DT10: .WORD DRIVE, \$ERRPC, RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42
10	045404	005624	005626	005562	.WORD RM.REG+44, RM.REG+46, RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20
11	045422	005602	005604	005606	.WORD RM.REG+22, RM.REG+24, RM.REG+26, RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
12	045440	001222	001132	001362	DT17: .WORD DRIVE, \$ERRPC, DS.CYL, DS.TRK, BADSEC
13	045452	001222	001132	001362	DT20: .WORD DRIVE, \$ERRPC, DS.CYL, DS.TRK, BADSEC
14	045464	005560	005570	005572	.WORD RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42, RM.REG+40, RM.REG+44, RM.REG+46
15	045504	005562	005564	005566	.WORD RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20, RM.REG+22, RM.REG+24, RM.REG+26
16	045524	005610	005612	001362	.WORD RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
17	045534	001222	001132	046230	DT23: .WORD DRIVE, \$ERRPC, RBUF+4, RBUF
18	045544	001222	001132	001362	DT24: .WORD DRIVE, \$ERRPC, DS.CYL, DS.TRK, BADSEC, \$GDDAT, RM.REG+22
19	045562	005560	005570	005572	.WORD RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42, RM.REG+40, RM.REG+44, RM.REG+46
20	045602	005562	005564	005566	.WORD RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20, RM.REG+22, RM.REG+24, RM.REG+26
21	045622	005610	005612	001362	.WORD RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
22	045632	001222	005552	001364	DT25: .WORD DRIVE, FMTDPB+12, DS.TRK, BADSEC

1			
2			
3	045642	000001	
4	045644	007	000
5	045646	000001	
6	045650	006	000
7	045652	000001	
8	045654	006	000
9	045656	000001	
10	045660	007	000
11	045662	000001	
12	045664	001	000
13	045666	000003	
14	045670	007	000
15	045672	044310	
16	045674	007	000
17	045676	044374	
18	045700	007	140
19	045702	000001	
20	045704	005	034
21	045706	000004	
22	045710	005	034
23	045712	044570	
24	045714	010	000
25	045716	044665	
26	045720	010	000
27	045722	044761	
28	045724	004	014
29	045726	000001	
30	045730	005	000
31	045732	000004	
32	045734	007	034
33	045736	044570	
34	045740	010	000
35	045742	044665	
36	045744	010	000
37	045746	044761	
38	045750	004	014
39	045752	000001	
40	045754	004	000

.SBTTL DATA FORMAT TABLES

DF1:	.WORD	1
	.BYTE	7,0
DF2:	.WORD	1
	.BYTE	6,0
DF3:	.WORD	1
	.BYTE	6,0
DF4:	.WORD	1
	.BYTE	7,0
DF6:	.WORD	1
	.BYTE	1,0
DF10:	.WORD	3
	.BYTE	7,0
	.WORD	DH10A
	.BYTE	7,0
	.WORD	DH10B
	.BYTE	7,140
DF17:	.WORD	1
	.BYTE	5,34
DF20:	.WORD	4
	.BYTE	5,34
	.WORD	DH20A
	.BYTE	8,0
	.WORD	DH20B
	.BYTE	8,0
	.WORD	DH20C
	.BYTE	4,14
DF23:	.WORD	1
	.BYTE	5,0
DF24:	.WORD	4
	.BYTE	7,34
	.WORD	DH20A
	.BYTE	8,0
	.WORD	DH20B
	.BYTE	8,0
	.WORD	DH20C
	.BYTE	4,14
DF25:	.WORD	1
	.BYTE	4,0


```

1
2
3
4
5
6
7
8
9
10
11
12
13 045756 005737 001374
14 045762 001436
15 045764 005037 001374
16 045770 104401 001205
17 045774 012700 001274
18 046000 104401 046076
19 046004 011046
20 046006 104402
21 046010 104401 036621
22 046014 104411
23 046016 012601
24 046020 004537 046114
25 046024 000763
26 046026 012700 001276
27 046032 104401 046105
28 046036 011046
29 046040 104402
30 046042 104401 036621
31 046046 104411
32 046050 012601
33 046052 004537 046114
34 046056 000763
35 046060 012700 001274
36 046064 012701 030456
37 046070 012021
38 046072 012021
39 046074 000207
40
41 046076 122 115 103 MRMCS1: .ASCIZ @RMCS1=@
42 046105 122 115 126 MRMVEC: .ASCIZ @RMVEC=@

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH/RM
;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
;OF THE RH/RM IS SETUP FOR THE PROPER ADDRESS.
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
;REQUIRED.
;NOTE: THIS ROUTINE DESTROYS R0-R4
;CALL
;
; JSR PC,BUSADR
; RETURN
;
BUSADR: TST CHGADR ;INPUT FROM TTY REQUESTED?
        BEQ 3$ ;NO
        CLR CHGADR ;DISABLE CHANGE OF RH/RM ADDRESS
        TYPE , $CRLF ;CR-LF
1$: MOV # $RMADR,R0 ;FIRST ADDRESS
    TYPE ,MRMCS1 ;'RMCS1='
    MOV (R0),-(SP) ;PRESENT RMCS1 ADDRESS
    TYPOC ;TYPE IT
    TYPE ,BLNKS2 ;TYPE 2 SPACES
    RDLIN ;GET THE ENTRY
    MOV (SP)+,R1 ;ADDRESS OF ASCII TEXT
    JSR R5,CK.NUM ;ENTER AND STORE NEW ADDRESS
    BR 1$ ;ERROR RET
2$: MOV # $RMVEC,R0 ;CHECK VERC
    TYPE ,MRMVEC ;'RMVEC='
    MOV (R0),-(SP) ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
    TYPOC ;TYPE IT
    TYPE ,BLNKS2 ;TYPE 2 SPACES
    RDLIN ;READ THE ENTRY
    MOV (SP)+,R1 ;ASCII TEXT ADDRESS
    JSR R5,CK.NUM ;ENTER AND STORE NEW ADDRESS
    BR 2$ ;ERROR RET
3$: MOV # $RMADR,R0 ;FIRST ADDRESS OF NEW PARAMETERS
    MOV # $RMADR,R1 ;FIRST ADDRESS OF WHERE TO PUT THEM
    MOV (R0)+,(R1)+ ;BUS ADDRESS
    MOV (R0)+,(R1)+ ;VECTOR ADDRESS
    RTS PC ;RETURN
    
```

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

.SBTTL CK.NUM - CHECK NUMBER (OCTAL)
:THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL CHARACTERS
:AND FORMS AN OCTAL NUMBER IN R2
:CALL:
: MOV #ADR,R1 :ADDRESS OF ASCII STRING
: R0=ADDRESS TO STORE THE NUMBER
: JSR R5,CK.NUM
: RET1 ERROR RETURN
: RET2 NORMAL RET

046114 010246
046116 010346
046120 010446
046122 012703 000006
046126 005002
046130 112104
046132 001424
046134 120427 000060
046140 103425
046142 120427 000067
046146 101022
046150 006302
046152 103420
046154 006302
046156 103416
046160 006302
046162 103414
046164 042704 177770
046170 060402
046172 005303
046174 001401
046176 000754
046200 112104
046202 001004
046204 005702
046206 001401
046210 010210
046212 005725
046214 012604
046216 012603
046220 012602
046222 000205

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


```
1      .SBTTL  READ/WRITE BUFFERS
2
3      ;BUFFER STARTS HERE
4
5 046224 000000 000000 000000 RBUF:  .WORD  0,0,0,0      ;BUFFER FOR HEADER CHECK
6
7 046234      BUFP:  .BLKW  <258.*32.>      ;FORMAT AND CHECK BUFFER STARTS HERE AND
8                                     ;WILL OVERLAY ALL REMAINING CODE IN PROGRAM
9 106434      BUFV:  .BLKW  <258.*32.>      ;BUFFER FOR VERIFY ONLY MODE
10
11      .END    200
```


ABASE = 176700	ASK3A 011514	BUFP 046234	DH23 045015	FM11 043052
ABORT 042226	ASWREG= 000000	BUFV 106434	DH24 045054	FM12 043110
ACDW1 = 000000	ATA = 100000	BUSADR 045756	DH25 045227	FM13 043153
ACDW2 = 000000	ATABIT 030444	BYTE16 006264	DH3 044054	FM14 043174
ACK = 000123	AATESTN= 000000	BYTE18 006364	DH4 044125	FM15 043217
ACPUOP= 000000	ATTN 001370	C 036613	DH6 044210	FM16 043263
ACTDRV 030404	ATO = 000001	CALIN 012706	DISPLA 001156	FM17 043335
ACTSTR 030405	AT1 = 000002	CHGADR 001374	DISPRE 000174	EM2 042520
ADDRIS 037506	AT2 = 000004	CI1 031734	DLT = 100000	EM20 043413
ADDW0 = 000000	AT3 = 000010	CI3 032042	DMD = 000001	EM21 043451
ADDW1 = 000000	AT4 = 000020	CI4 032150	DONE 016366	EM22 043522
ADDW10= 000000	AT5 = 000040	CI5 032526	DPINT 030360	EM23 043572
ADDW11= 000000	AT6 = 000100	CI6 032550	DPR = 000400	EM24 043642
ADDW12= 000000	AT7 = 000200	CI7 032564	DPRQS 030370	EM25 043664
ADDW13= 000000	AUNIT = 000000	CI7B 032606	DRIVE = 001222	EM3 042556
ADDW14= 000000	AUSWR = 000000	CI8 032664	DRQ = 004000	EM4 042614
ADDW15= 000000	AVECT1= 000254	CKSWR = 104407	DRVACT 030330	EM5 042651
ADDW2 = 000000	AVECT2= 000000	CKTRK 014344	DRVCLR= 000111	EM6 042705
ADDW3 = 000000	A16 = 000400	CK.CHR 023120	DRVINT 030700	EM7 042760
ADDW4 = 000000	A17 = 001000	CK.DEC 023072	DRVQUE 036242	ENTADR 036624
ADDW5 = 000000	BADENT 037174	CK.DIG 023172	DRVSTA 030340	ERINDX 016550
ADDW6 = 000000	BADSEC 001344	CK.NUM 046114	DRVSTYP 030350	ERR = 040000
ADDW7 = 000000	BADTMO 006730	CK.OCT 023044	DRY = 000200	ERROR = 104000
ADDW8 = 000000	BAI = 000010	CLOCK 017616	DSWR = 177570	ERRVEC= 000004
ADDW9 = 000000	BIT0 = 000001	CLR = 000040	DS.CYL 001362	ES.SAV 036406
ADEVCT= 000000	BIT00 = 000001	CLRQUE 036144	DS.TRK 001364	FER = 000020
ADEVM = 000000	BIT01 = 000002	CNTLC 001372	DTE = 010000	FILLZ 026770
ADRTBL 006064	BIT02 = 000004	COLON 036604	DTSY = 000200	FILLO 027076
AENV = 000000	BIT03 = 000010	CPSAVE 023754	DTUW 030442	FMT = 010000
AENVM = 000000	BIT04 = 000020	CR = 000015	DT00 = 000001	FMTDPB 005540
AFATAL= 000000	BIT05 = 000040	CRLF = 000200	DT01 = 000002	FMTWRT 005630
AMADR1= 000000	BIT06 = 000100	C.ENTR 017630	DT02 = 000004	FMT16 = 010000
AMADR2= 000000	BIT07 = 000200	DCK = 100000	DT03 = 000010	FUNCT1 021272
AMADR3= 000000	BIT08 = 000400	DDISP = 177570	DT04 = 000020	FUNCT2 021414
AMADR4= 000000	BIT09 = 001000	DDRIVE 001366	DT05 = 000040	FUNCT3 021624
AMAMS1= 000000	BIT1 = 000002	DEVMP 001400	DT06 = 000100	FUNCT4 021676
AMAMS2= 000000	BIT10 = 002000	DF1 045642	DT07 = 000200	FUNCT5 021774
AMAMS3= 000000	BIT11 = 004000	DF10 045666	DT08 = 000400	FUNCT6 022112
AMAMS4= 000000	BIT12 = 010000	DF17 045702	DT1 045270	FUNCT7 022230
AMSGAD= 000000	BIT13 = 020000	DF2 045646	DT10 045366	FUNCT8 022350
AMSGLG= 000000	BIT14 = 040000	DF20 045706	DT17 045440	F1 = 000002
AMSGTY= 000000	BIT15 = 100000	DF23 045726	DT2 045306	F2 = 000004
AMTYP1= 000000	BIT2 = 000004	DF24 045732	DT20 045452	F3 = 000010
AMTYP2= 000000	BIT3 = 000010	DF25 045752	DT23 045534	F4 = 000020
AMTYP3= 000000	BIT4 = 000020	DF3 045652	DT24 045544	F5 = 000040
AMTYP4= 000000	BIT5 = 000040	DF4 045656	DT25 045632	GETREG= 000141
AOE = 001000	BIT6 = 000100	DF6 045662	DT3 045322	GETREQ 036316
APASS = 000000	BIT7 = 000200	DH1 043734	DT4 045336	GO = 000001
APRIOR= 000000	BIT8 = 000400	DH10 044223	DT6 045354	GTSWR = 104406
APTCSU= 000040	BIT9 = 001000	DH10A 044310	DT7 045356	HCE = 000200
APTENV= 000001	BLNKS1 036622	DH10B 044374	DVA = 004000	HCI = 002000
APTSIZ= 000200	BLNKS2 036621	DH17 044452	ECH = 000100	HCRC = 000400
APTSPO= 000100	BLNKS3 036620	DH2 044002	ECI = 004000	HDRBIT 001410
ASK0 010714	BLNKS4 036617	DH20 044521	EMPTYQ 036222	HDRSET 015010
ASK1 011000	BPTVEC= 000014	DH20A 044570	EMTVEC= 000030	HEDERR 001354
ASK2 011124	BSE = 100000	DH20B 044665	EM1 042446	HT = 000011
ASK3 011444	BSFAVL 001404	DH20C 044761	EM10 043034	IAE = 002000

IBSAVE 023756
IE = 000100
ILF = 000001
ILR = 000002
INITMF 042143
INITUS 042100
INSERT 021160
IOTVEC= 000020
IR = 000100
ISR 033114
LF = 000012
LODEV 036521
LOP.CK 016646
LST = 002000
LSTRK 001324
MADRER 037215
MANTER 022702
MAXCYL 001326
MAXSEC 001360
MAXTRK 001332
MCLK = 000002
MCPE = 020000
MCPEMX 030454
MDRNP 036675
MER11 036722
MSG1 037533
MSG10 041140
MSG11 041224
MSG12 041310
MSG13 041374
MSG14 041502
MSG15 041513
MSG16 041563
MSG17 041563
MSG18 041630
MSG19 041665
MSG2 037572
MSG3 037643
MSG4 037704
MSG5 040564
MSG6 040642
MSG7 040720
MSG8 040776
MSG9 041054
MF = 100000
MFG16 001414
MFG18 002420
MFORMT 037033
MINCYL 001330
MINTRK 001334
MINX = 000004
MLODEV 036771
MMODE 041732
MODE 001320
MOFLN 036654
MOH = 020000
MOL = 010000

MPE = 000400
MRD = 000020
MRMCS1 046076
MRMVEC 046105
MSCMPT 037427
MSELD 037021
MSFOR 037315
MSGBLK 037740
MSG13A 041450
MSINIT 042033
MSIZE 041771
MSPRM 042362
MSQVER 037337
MSTCK = 000010
MSVER 037376
MUNIT 036573
MUSDR 036751
MUTLTY 040521
MVERFY 037062
MwC 001352
MwR = 000040
MXF = 001000
MXWWDW 030464
M16BIT 037106
M18BIT 037141
NBA = 100000
NED = 010000
NEM = 004000
NEXT 005536
NOENTR 042206
NOINIT 042314
NOTPRS 036474
NOTRM 036453
NOTSAF 036511
NOUPDT 042255
NUMERR 037442
OFDIR = 000001
OFFSET= 000115
OPI = 020000
OPT 031506
OR = 000200
O.ENTR 017642
PACKSN 037473
PAR = 000010
PARENT 020032
PAR1 005754
PAR2 005764
PAR3 005774
PAR4 006004
PAR5 006014
PAR6 006032
PAT = 000020
PATA 001336
PATB 001340
PFEC H 024234
PFEC H1 024244
PFEC H2 024326

PFEC H3 024360
PFEC H4 024370
PFTSTN 024374
PGE = 002000
PGM = 001000
PIP = 020000
PIRQ = 177772
PIRQVE= 000240
POPQUE 036340
PRO = 000000
PR1 = 000040
PR2 = 000100
PR3 = 000140
PR4 = 000200
PR5 = 000240
PR6 = 000300
PR7 = 000340
PS = 177776
PSEL = 002000
PSW = 177776
PWRVEC= 000024
QCNT 035652
QDRV0 035744
QDRV1 035764
QDRV2 036004
QDRV3 036024
QDRV4 036044
QDRV5 036064
QDRV6 036104
QDRV7 036124
QINPT 035662
QOUTPT 035702
QSTART 035722
QSTOP 035724
QTERM = 036144
RBUF 046224
RDCHR = 104410
RDDAT = 000171
RDHD = 000173
RDLIN = 104411
RDY = 000200
RD.ADR 035134
RD.RM 035110
RD.RM1 035132
RD.RM2 035256
RD.RM3 035262
RD.RM4 035266
RD.WRD 035136
READIN= 000121
RECAL = 000107
RECORD 020146
REJCT1 016430
REJCT2 016442
REJCT3 016464
REJCT4 016476
REJCT5 016510
REJCT6 016522

REJEND 016534
RELSE = 000113
REPLZ 026774
RESET 020754
RESREG= 104413
RESTRT 017522
RESVEC= 000010
RETBAD 011644
RETRY 001342
RMADR 030456
RMAS = 000016
RMBA = 000004
RMCS1 = 000000
RMCS2 = 000010
RMDA = 000006
RMDB = 000022
RMDC = 000034
RMDS = 000012
RMDT = 000026
RMEC1 = 000044
RMEC2 = 000046
RMERRS 030320
RMER1 = 000014
RMER2 = 000042
RMHR = 000036
RMINIT 030466
RMLA = 000020
RMMR1 = 000024
RMMR2 = 000040
RMOF = 000032
RMR = 000004
RMSN = 000030
RMTMR 034476
RMVEC 030460
RMWC = 000002
RM.REG 005560
RM05 031214
RTC = 000117
R6 = %000006
R7 = %000007
SAVEFG 030416
SAVREG= 104412
SAVWC 001346
SC 033326
SCAWC 017072
SCOPE = 000004
SC01 = 000100
SC02 = 000200
SC04 = 000400
SC10 = 001000
SC11 034160
SC12 034270
SC13 034340
SC20 = 002000
SC3 033376
SC4 033402
SC5 033414

SC6 033610
SC6A 033724
SC7 034052
SC8 034130
SEARCH= 000131
SECCU 005434
SEC30 001356
SEEK = 000105
SEEKFG 030420
SELDRV= 000145
SERNL 012356
SETBSF 020410
SETDPB 016714
SETFMT= 000143
SETHDR 017120
SETVEC 010100
SET.IE 035600
SKI = 040000
SLASH 036607
SNGSEC 001322
SOF SW 001316
SORT1 013474
SORT2 020250
SRCHWT 030402
STACK = 001100
STARS 042423
START 007012
START1 007020
STCLK1 017436
STCLK2 017506
STCLK3 017512
STKLMT= 177774
STO 034566
STO1 034622
STO2 034716
STO3 034746
STO5 034772
STO6 035000
STO7 035036
STO8 035066
STO9 035076
ST.CLK 017360
ST1 010446
ST2 011566
ST3 013056
SVRH70 035462
SWR 001154
SWREG 000176
SW0 = 000001
SW00 = 000001
SW01 = 000002
SW02 = 000004
SW03 = 000010
SW04 = 000020
SW05 = 000040
SW06 = 000100
SW07 = 000200

SW08 = 000400
 SW09 = 001000
 SW1 = 000002
 SW10 = 002000
 SW11 = 004000
 SW12 = 010000
 SW13 = 020000
 SW14 = 040000
 SW15 = 100000
 SW2 = 000004
 SW3 = 000010
 SW4 = 000020
 SW5 = 000040
 SW6 = 000100
 SW7 = 000200
 SW8 = 000400
 SW9 = 001000
 T = 036615
 TABCAL 006052
 TABINT 012146
 TABLD 012226
 TABLE 005650
 TABLE2 005702
 TABLE3 005720
 TABLE4 005736
 TAB.XY= 001114
 TAP = 040000
 TBITVE= 000014
 TD 033156
 TIMER 030422
 TKVEC = 000060
 TPVEC = 000064
 TRAPVE= 000034
 TRE = 040000
 TRKTST 016770
 TRK1 = 004000
 TRK10 = 040000
 TRK2 = 010000
 TRK20 = 100000
 TRK4 = 020000
 TRNSWT 030400
 TRTVEC= 000014
 TST1 007034
 TYLIST 022512
 TYPACK 022622
 TYPADR 017732
 TYPCK 022626
 TYPDS = 104405
 TYPE = 104401
 TYPERR 023760
 TYPOC = 104402
 TYPON = 104404
 TYPOS = 104403
 UF = 040000

ULDFLG 030406
 UNS = 040000
 UNSTAT 036555
 UNTOFF 036432
 UNTON 036443
 UPDBSF 001406
 UPDCYL 017206
 UPDTRK 017320
 UPE = 020000
 USSAV 004430
 USTAB 003424
 US1 = 000001
 US2 = 000002
 US4 = 000004
 VFYHDR 015110
 VV = 000100
 WC = 001350
 WCE = 040000
 WCF = 000040
 WCKD = 000151
 WCKHD = 000153
 WCTBL 006164
 WLE = 004000
 WRAP = 001376
 WRL = 004000
 WRTDAT= 000161
 WRTHD = 000163
 WRTRK 013214
 WRTRKA 013676
 WRTRKB 013774
 WRTRKC 014060
 WRTRKD 014142
 WRTRKE 014210
 WRTRKF 014124
 WRTRKH 014154
 WRTRKX 013270
 WRTRKZ 015076
 WRTRK1 013646
 WRTRK2 014224
 WRT.AD 035360
 WRT.RM 035270
 WRT.R1 035354
 WRT.R2 035446
 WRT.R3 035452
 WRT.R4 035456
 WRT.R5 035460
 WRT.WD 035356
 XSIZE 010406
 XXDP 001412
 \$APTHD 001100
 \$ATYC 027530
 \$ATY1 027504
 \$ATY3 027512
 \$ATY4 027522

\$AUTOB 001150
 \$BASE 001264
 \$BDADR 001136
 \$BDDAT 001142
 \$BELL 001200
 \$CDW1 001270
 \$CDW2 001272
 \$CHARC 026212
 \$CKSWR 024714
 \$CMTAG 001114
 \$CM3 = 000000
 \$CM4 = 000001
 \$CNTLC 025621
 \$CNTLG 025633
 \$CNTLU 025626
 \$COMMA 036602
 \$CPUOP 001236
 \$CRLF 001205
 \$DBLK 026660
 \$DB2D 027134
 \$DB20 027364
 \$DECVL 027314
 \$DEVCT 001220
 \$DEVM 001266
 \$DOAGN 016362
 \$DTBL 026650
 \$ENDAD 016352
 \$ENDCT 016336
 \$ENV 001230
 \$ENVM 001231
 \$EOP 015740
 \$EOPCT 016330
 \$ERFLG 001117
 \$ERMAX 001131
 \$ERROR 023330
 \$ERRPC 001132
 \$ERRTB 006460
 \$ERTTL 001126
 \$ESCAP 001176
 \$ETABL 001230
 \$ETEND 001274
 \$FATAL 001212
 \$FFLG 027750
 \$FILLC 001172
 \$FILLS 001171
 \$GDADR 001134
 \$GDDAT 001140
 \$GET42 016342
 \$GTSWR 025004
 \$HD = 000000
 \$HIBTS 001100
 \$ICNT 001120
 \$ILLUP 030276
 \$INTAG 001151

\$ITEMB 001130
 \$LF 001206
 \$LFLG 027747
 \$LKCSB 001302
 \$LKCSR 001300
 \$LKS 001310
 \$LLVEC 001312
 \$LPADR 001122
 \$LPERR 001124
 \$LPVEC 001304
 \$MADR1 001242
 \$MADR2 001246
 \$MADR3 001252
 \$MADR4 001256
 \$MAIL 001210
 \$MAMS1 001240
 \$MAMS2 001244
 \$MAMS3 001250
 \$MAMS4 001254
 \$MBADR 001102
 \$MFLG 027746
 \$MNEW 025651
 \$MSGAD 001224
 \$MSGLG 001226
 \$MSGTY 001210
 \$MSWR 025640
 \$MTYP1 001241
 \$MTYP2 001245
 \$MTYP3 001251
 \$MTYP4 001255
 \$NULL 001170
 \$NWTST= 000000
 \$OCNT 026440
 \$OCTVL 027466
 \$OMODE 026442
 \$PASS 001216
 \$PASTM 001106
 \$POWER 030304
 \$PWRAD 030272
 \$PWRDN 030132
 \$PWRMG 030266
 \$PWRUP 030204
 \$QUES 001204
 \$RDCHR 025256
 \$RDLIN 025346
 \$RDSZ = 000017
 \$RESRE 030010
 \$RMADR 001274
 \$RMVEC 001276
 \$RMO2 036536
 \$RMO3 036543
 \$RMO5 036550
 \$RTNAD 016364

\$SAVRE 027752
 \$SAVR6 030302
 \$SB2D 027100
 \$SB20 027330
 \$SETUP= 000116
 \$STUP = 177777
 \$SUPRL 026670
 \$SUPRS 026704
 \$SUPR1 026716
 \$SUPR2 026760
 \$SVPC = 000210
 \$SWR = 123000
 \$SWREG 001232
 \$TESTN 001214
 \$TKB 001162
 \$TKCNT 024376
 \$TKINT 024414
 \$TKQEN= 024413
 \$TKQIN 024400
 \$TKQOU 024402
 \$TKQSR 024404
 \$TKS 001160
 \$TKSRV 024464
 \$TMPO 001174
 \$TN = 000002
 \$TNPWR 027244
 \$TPB 001166
 \$TPFLG 001173
 \$TPS 001164
 \$TRAP 030046
 \$TRAP2 030070
 \$TRP = 000014
 \$TRPAD 030102
 \$TSTM 001104
 \$TSTNM 001116
 \$TTYIN 025602
 \$TYPDS 026444
 \$TYPE 025662
 \$TYPEC 026074
 \$TYPEX 026214
 \$TYPOC 026242
 \$TYPON 026256
 \$TYPOS 026216
 \$UNIT 001222
 \$UNITM 001110
 \$USWR 001234
 \$VECT1 001260
 \$VECT2 001262
 \$XOFF = 000023
 \$XON = 000021
 \$\$GET4= 000000
 \$OFILL 026441
 \$.X = 001100

. ABS. 146634 000
 000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 53504 WORDS (209 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 77 PAGES
CZRMLB.BIC,CZRMLB/C=CZRMLB.DOC,CZRMLB.SYSMAC/M

\$FILLC	6-0#	32-1	32-1	32-1					
\$FILLS	6-0#	32-1	32-1						
\$GDADR	6-0#								
\$GDDAT	6-0#	29-1*	49-18						
\$GET42	10-20	19-1#							
\$GTSWR	31-1#	43-1	43-1						
\$HD	4-62	4-62	4-62						
\$HIBTS	5-8#								
\$ICNT	6-0#								
\$ILLUP	44-1	44-1	44-1#						
\$INTAG	6-0#	31-1	31-1	31-1	31-1	31-1*			
\$ITEMB	6-0#	29-1	29-1	29-1	29-1	29-1	29-1*	29-1*	30-8
\$LF	6-0#	29-1	29-1	31-1	31-1	31-1	32-1	32-1	
\$LFLG	41-1#	41-1*							
\$LKCSB	7-0#	21-167*							
\$LKCSR	7-0#	21-162	21-168*						
\$LKS	7-0#	21-172	21-177*						
\$LLVEC	7-0#	21-173							
\$LPADR	6-0#								
\$LPERR	6-0#	11-39*	15-194*	16-7*	18-25*	21-39	29-1		
\$LPVEC	7-0#	21-163							
\$MADR1	6-0#								
\$MADR2	6-0#								
\$MADR3	6-0#								
\$MADR4	6-0#								
\$MAIL	5-8	5-8	6-0#	9-21	9-26	9-31	29-1	32-1	
\$MAMS1	6-0#								
\$MAMS2	6-0#								
\$MAMS3	6-0#								
\$MAMS4	6-0#								
\$MBADR	5-8#								
\$MFLG	41-1	41-1#	41-1*	41-1*					
\$MNEW	31-1	31-1#							
\$MSGAD	6-0#	41-1	41-1*						
\$MSGLG	6-0#	41-1*							
\$MSGTY	6-0#	41-1	41-1	41-1*	41-1*				
\$MSWR	31-1	31-1#							
\$MTYP1	6-0#								
\$MTYP2	6-0#								
\$MTYP3	6-0#								
\$MTYP4	6-0#								
\$NULL	6-0#	32-1	32-1	32-1					
\$NWTST	9-21#								
\$OCNT	33-1#	33-1*	33-1*						
\$OCTVL	40-1	40-1#							
\$OMODE	33-1	33-1#	33-1*	33-1*	33-1*	33-1*	33-1*		
\$PASS	6-0#	9-26*	19-1	19-1	19-1*	19-1*			
\$PASTM	5-8#								
\$POWER	44-1	44-2#							
\$PWRAD	44-1#								
\$PWRDN	9-26	44-1	44-1#						
\$PWRMG	44-1#								
\$PWRUP	44-1	44-1#							
\$QUES	6-0#	29-1	29-1	31-1	31-1	31-1	31-1	32-1	32-1
\$R2A	43-1								
\$RDCHR	31-1#	43-1	43-1						
\$RDDEC	43-1								

\$TYPE	32-1#	41-1	43-1	43-1		
\$TYPEC	31-1	32-1	32-1	32-1	32-1#	
\$TYPEX	32-1	32-1	32-1#			
\$TYPOC	33-1#	43-1	43-1			
\$TYPON	33-1	33-1#	43-1			
\$TYPOS	33-1#	43-1				
\$UNIT	6-0#	7-0				
\$UNITM	5-8#					
\$USWR	6-0#					
\$VECT1	6-0#	9-78	9-80			
\$VECT2	6-0#					
\$XOFF	32-1	32-1				
\$XON	31-1	32-1	32-1			
.\$ASTA	41-1	41-1				
.\$X	5-8	5-8#				
A16	4-76#					
A17	4-77#					
ABASE	4-263#	6-0	6-0			
ABORT	20-29	46-79#				
ACDW1	6-0	6-0				
ACDW2	6-0	6-0				
ACK	4-251#					
ACPUOP	6-0	6-0				
ACTDRV	45-93#	45-348*	45-399*	45-692*	45-700*	45-987
ACTSTR	45-99#	45-989*	45-:03*			
ADDRIS	21-246	46-43#				
ADDW0	6-0					
ADDW1	6-0					
ADDW10	6-0					
ADDW11	6-0					
ADDW12	6-0					
ADDW13	6-0					
ADDW14	6-0					
ADDW15	6-0					
ADDW2	6-0					
ADDW3	6-0					
ADDW4	6-0					
ADDW5	6-0					
ADDW6	6-0					
ADDW7	6-0					
ADDW8	6-0					
ADDW9	6-0					
ADEVCT	6-0	6-0				
ADEVN	6-0	6-0				
ADRTBL	7-0#	15-191	16-55	19-1	24-194	
AENV	6-0	6-0				
AENVN	6-0	6-0				
AFATAL	6-0	6-0				
AMADR1	6-0	6-0				
AMADR2	6-0	6-0				
AMADR3	6-0	6-0				
AMADR4	6-0	6-0				
AMAMS1	6-0	6-0				
AMAMS2	6-0	6-0				
AMAMS3	6-0	6-0				
AMAMS4	6-0	6-0				
AMSGAD	6-0	6-0				

BIT13	4-68#	21-10	21-43	29-1	45-376	45-821	45-:15							
BIT14	4-68#	17-11	17-13	21-10	21-43	45-388	45-423	45-806	45-966	45-:76	45-:22	45-:26	45-<41	
BIT15	4-68#	12-25	12-43	15-123	15-138	15-142	15-157	15-172	45-376	45-388	45-391	45-393	45-423	45-426
	45-619	45-656	45-658	45-735	45-850	45-866	45-876	45-886	45-966	45-:40	45-:76	45-:84		
BIT2	4-68#	21-10	45-:84											
BIT3	4-68#													
BIT4	4-68#													
BIT5	4-68#													
BIT6	4-68#													
BIT7	4-68#	10-7												
BIT8	4-68#													
BIT9	4-68#													
BLNKS1	14-10	46-22#												
BLNKS2	9-134	21-251	24-200	30-51	30-60	30-71	46-21#	51-21	51-30					
BLNKS3	46-20#													
BLNKS4	9-104	46-19#												
BPTVEC	4-68#													
BSE	4-228#	18-71												
BSFAVL	7-0#	11-24*	11-86*	11-114	12-6	15-36	15-63*							
BUFP	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0	7-0
	7-0	7-0	7-0	7-0	11-29	11-69	11-73	11-75	11-77	11-79	11-81	11-84	11-118	11-126
	11-135	11-141	14-25	15-115	18-15	19-1	21-56	21-103	21-124	21-149	22-12	22-26	22-40	22-43
	22-46	22-49	22-53	22-56	53-7#									
BUFV	18-16	53-9#												
BUSADR	9-72	51-13#												
BYTE16	7-0#	24-224												
BYTE18	7-0#	24-234												
C	21-247	46-17#												
C.ENTR	10-117	21-214#	21-215	21-245										
CALIN	13-5#													
CHGADR	7-0#	9-18*	51-13	51-15*										
CI1	45-445	45-473#												
CI3	45-447	45-497#												
CI4	45-439	45-519#												
CI5	45-495	45-517	45-597#	45-607										
CI6	45-535	45-545	45-547	45-549	45-603#									
CI7	45-378	45-454	45-485	45-489	45-493	45-502	45-511	45-515	45-527	45-534	45-540	45-544	45-558	45-564
	45-568	45-578	45-589	45-606	45-609#	45-716	45-911	45-<16						
CI7B	45-624#	45-917												
CI8	45-627	45-641#	45-770	45-:67										
CK.CHR	10-124	28-50#	28-87	28-95										
CK.DEC	28-28#	28-56												
CK.DIG	21-280	24-192	27-17	27-20	27-23	28-81#								
CK.NUM	51-24	51-33	52-13#											
CK.OCT	28-10#	28-58												
CKSWR	29-1	29-1	43-1#											
CKTRK	16-6#													
CLOCK	21-164	21-174	21-208#											
CLR	4-96#	9-93	10-140	45-234	45-681									
CLRQUE	45-216	45-680	45-<95#											
CNTLC	7-0#	9-67*	10-117*	21-245*	31-1	31-1								
COLON	10-121	46-15#												
CPSAVE	29-1	29-1	29-1	29-1	29-1	29-1#	29-1*	29-1*	30-79					
CR	4-68#	32-1	32-1	48-17										
CRLF	4-68#	9-6	9-31	9-31	9-45	9-56	9-62	9-63	32-1	32-1	44-2	46-12	46-13	46-23
	46-23	46-24	46-25	46-26	46-27	46-28	46-32	46-33	46-34	46-35	46-36	46-37	46-37	46-38

	46-38	46-39	46-39	46-40	46-44	46-45	46-46	46-47	46-48	46-48	46-49	46-50	46-51	46-52
	46-53	46-54	46-55	46-56	46-57	46-58	46-59	46-60	46-61	46-62	46-63	46-64	46-65	46-66
	46-67	46-67	46-68	46-68	46-70	46-71	46-72	46-75	46-76	46-77	46-78	46-79	46-80	46-80
	46-81	46-81	46-83	46-83										
DCK	4-153#	16-33												
DDISP	4-68#	6-0	9-26											
DDRIVE	7-0#	29-1*	49-4	49-5	49-6									
DEVMP	7-0#	9-99*	9-132*	9-133	9-133*	9-151	9-151*	10-12*	10-13*	10-18	10-40	10-48*	19-1	19-1
	19-1*	21-233*	21-234*											
DF1	8-7	50-3#												
DF10	8-56	8-63	8-70	8-77	8-84	8-91	8-98	8-119	8-126	50-13#				
DF17	8-105	50-19#												
DF2	8-14	50-5#												
DF20	8-112	50-21#												
DF23	50-29#													
DF24	8-140	50-31#												
DF25	8-49	8-133	8-147	50-39#										
DF3	8-21	50-7#												
DF4	8-28	50-9#												
DF6	8-42	50-11#												
DH1	8-5	48-3#												
DH10	8-54	8-61	8-68	8-75	8-82	8-89	8-96	8-117	8-124	48-8#				
DH10A	48-9#	50-15												
DH10B	48-10#	50-17												
DH17	8-103	48-11#												
DH2	8-12	48-4#												
DH20	8-110	48-12#												
DH20A	48-13#	50-23	50-33											
DH20B	48-14#	50-25	50-35											
DH20C	48-15#	50-27	50-37											
DH23	8-47	8-131	48-16#											
DH24	8-138	48-17#												
DH25	8-145	48-19#												
DH3	8-19	48-5#												
DH4	8-26	48-6#												
DH6	8-40	48-7#												
DISPLA	6-0#	9-26*	9-26*	29-1*										
DISPRE	5-1#	9-26												
DLT	4-106#													
DMD	4-157#													
DONE	19-1	19-1	19-3#											
DPINT	45-61#	45-245	45-248	45-358	45-790	45-817	45-956	45-958*	45-:31	45-:54	45-:60	45-:70*		
DPR	4-127#													
DPRQS	45-70#	45-364	45-412*	45-449*	45-629*	45-646	45-664*	45-793	45-:33	45-:56	45-:62	45-:80*		
DRIVE	7-0#	10-47*	10-129*	10-158	11-25	14-8	21-53	22-91	49-8	49-9	49-12	49-13	49-17	49-18
	49-22													
DRQ	4-190#													
DRVACT	45-30#	45-372	45-552*	45-600*	45-631*	45-644	45-663*	45-705*	45-813	45-848	45-880*	45-888*	45-908	45-923*
	45-:45*													
DRVCLR	4-246#													
DRVINT	45-236	45-270#	45-360	45-945	45-959									
DRVQUE	45-367	45-380	45-:35#											
DRVSTA	9-105	10-49	10-144	45-40#	45-226*	45-227*	45-228*	45-229*	45-239*	45-271*	45-281*	45-325*	45-331*	45-354
	45-362	45-386	45-420	45-424	45-553*	45-684*	45-796	45-804	45-819	45-874*	45-881*	45-897	45-961	45-:71*
DRVTYP	9-109	9-136	10-57	10-150	10-163	45-52#	45-272*	45-288*	45-293*	45-298*	45-303*	45-389	45-685*	
DRY	4-126#													
DS.CYL	7-0	7-0	7-0	7-0#	13-33*	23-13	23-23	24-16	24-29	24-62	24-75	24-219	29-1*	49-11

SLASH	4-68#													
STARS	4-68#	5-5	5-8	5-8	5-8	6-0	6-0	6-0	9-21	19-1	29-1	31-1	31-1	31-1
	31-1	31-1	32-1	33-1	34-1	35-4	37-1	38-1	39-1	40-1	41-1	42-1	43-1	44-1
	44-1	45-15												
SWRSU	4-68#	9-26	9-26#											
TRMTRP	43-1#													
TYPBIN	4-68#													
TYPDEC	4-68#	19-1	30-48											
TYPNAM	4-68#	9-31												
TYPNUM	4-68#													
TYPPCS	4-68#	9-103	14-8											
TYPQCT	4-68#	30-46	31-1											
TYPXT	4-68#	9-6	9-45	9-56	9-62	9-63								