

RM05/3/2

RM05/3/2 FORMATTER
CZRMLAO

AH-F919A-MC
FICHE 1 OF 1

JUN 1980
COPYRIGHT © 1980
MADE IN USA



The main body of the document is a large, dense grid of data. The grid is composed of approximately 15 columns and 25 rows of small, illegible text or symbols. The text is too faint to be transcribed accurately, but it appears to be organized in a structured format, possibly a table or a list of entries. There are some faint markings and artifacts on the page, such as a small '5' in the lower-left quadrant and some smudges on the left side.

.REM 8

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

IDENTIFICATION

PRODUCT CODE: AC-F918A-MC
PRODUCT NAME: CZRMLAO RM05/3/2 FORMATTER
DATE CREATED: APRIL 1980
MAINTAINER: CX DIAGNOSTIC ENGINEERING
AUTHOR: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1980 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 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
 - 4.4 OTHER UNIBUS ADDRESSES
- 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 CHECK OPERATION
 - 8.3 POSITIONER VERIFICATION
 - 8.4 HEADER VERIFICATION
- 9. BAD SPOT FILE
- 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.

IN THE CHECK OPERATION, THE PROGRAM REPEATS THE FORMAT OPERATION THREE TIMES WITH DATA PATTERN BEING ROTATED ONE BIT AT EACH PASS.

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 PROCESSOR
8K 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

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

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

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

6. MEMORY TYPES 1-4 AND MAX MEMORY ADDRESSES
NOT USED

7. INTERRUPT VECTOR 1:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 254

8. BUS PRIORITY 1:
NOT USED.

9. INTERRUPT VECTOR 2:
NOT USED

10. BUS PRIORITY 2:
NOT USED

11. BASE ADDRESS:
USED WHEN ENVIRONMENT MODE BIT 7 = 1;DEFAULT = 176700

12. DEVICE MAP:
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

THE PROGRAM IS STARTED FROM LOCATION 200 IF THE ADDRESS OF THE
RH11 OR RH70 WILL NOT BE CHANGED FROM ITS DEFAULT VALUE OF 176700.

THE PROGRAM IS STARTED FROM LOCATION 204 IF THE ADDRESS OF
THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE.
NOTE THAT STARTING ADDRESS 204 IS VALID ONLY ONCE AFTER THE
PROGRAM IS LOADED.
(SEE SECTION 4.3)

4.2 OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3).
2. LOAD THE STARTING ADDRESS - 200 OR 204

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

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>.
4. THE PROGRAM WILL THEN TYPE THE FOLLOWING MESSAGE:
'PROGRAM MODE (F, C OR V):'
ENTER THE APPROPRIATE CODE: 'F' FOR FORMAT & QUICK VERIFY, 'C' FOR CHECK OR 'V' FOR VERIFY OPERATION. IF A CARRIAGE RETURN IS ENTERED IN RESPONSE TO THE REQUEST, THE PROGRAM WILL ASSUME 'FORMAT & QUICK VERIFY'.
5. THE PROGRAM WILL THEN ASK FOR THE FORMATTING MODE:
'OPERATE IN 32 SECTOR (16 BIT) MODE (Y OR N)'
ENTER THE APPROPRIATE CHARACTER: 'Y' FOR 16 BIT MODE OR 'N' FOR 18 BIT MODE. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THIS REQUEST, THE PROGRAM WILL ASSUME 16 BIT MODE.
6. THE PROGRAM WILL THEN ASK FOR A DRIVE:
'DRIVE: '
ENTER THE ADDRESS OF THE DRIVE TO BE FORMATTED. A 'PERIOD' OR 'CARRIAGE RETURN' ENTRY WILL SELECT DRIVE 0. IF THE DRIVE SELECTED IS NOT AVAILABLE, THE PROGRAM WILL TYPE AN ERROR MESSAGE AND RETURN TO THE DRIVE ADDRESS REQUEST.
7. AFTER THE OPERATOR HAS SELECTED A DRIVE, 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 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.
8. THE PROGRAM WILL THEN ASK FOR THE DATA PATTERN:
(IN CASE OF FORMAT MODE)
'SELECT DATA PATTERN
(0) ZERO'S

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

(1) ONE'S
(2) WORST CASE
ENTER (0, 1 OR 2):'

ENTER THE CODE FOR THE REQUIRED PATTERN. 'CARRIAGE RETURN' OR
'PERIOD' ENTRIES WILL CAUSE THE PROGRAM TO USE THE 'WORST CASE'
PATTERN.

THE 'WORST CASE' PATTERN IS THE FOLLOWING OCTAL SEQUENCE REPEATED
THROUGH THE DATA AREA OF THE SECTOR:

066666

NOTE: IN THE CHECK OPERATION, THE ABOVE MENTIONED MESSAGE IS
BYPASSED. THE DATA PATTERN IS INITIATED TO 133331 AND IS
ROTATED ONE BIT TO THE RIGHT AT EACH PASS OF THE CHECK
OPERATION. AFTER THE CHECK OPERATION IS DONE, THE DATA
PATTERN WILL BE RESTORED TO 066666.

9. THE PROGRAM WILL THEN TYPE:

'STARTING FORMAT ON DRIVE N'

OR

'STARTING CHECK ON DRIVE N'

OR

'STARTING VERIFY ON DRIVE N'

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.

4.5 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.
IF THE RH11 - RH70 IS NOT AT THE DEFAULT ADDRESS, THE PROGRAM MUST
BE STARTED FROM 204 INITIALLY AS THE PROGRAM GOES THROUGH THE
ADDRESS CHANGE ROUTINE AT INITIAL START ONLY. ENTER THE RH11/RH70
ADDRESS IN RESPONSE TO THE REQUEST FROM THE PROGRAM.

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

4.4 OTHER UNIBUS ADDRESSES

LOC	TAG	CONTENTS	FUNCTION
---	---	-----	-----
1160	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1162	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1164	\$TPS	177564	TTY PRINTER STATUS REGISTER
1166	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1276	\$LKCSR	172540	KW11-P CONTROL REGISTER
1300	\$LKCSB	172542	KW11-P COUNTER REGISTER
1302	\$LPVEC	104	KW11-P VECTOR ADDRESS
1304	\$LKS	177546	KW11-L CONTROL REGISTER
1306	\$LKV	100	;ADDRESS OF KW11-L VECTOR

5. SWITCH REGISTER SETTINGS

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...DON'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

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RM 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 SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED., 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

6. ERROR MESSAGES

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

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

400 FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.
401
402
403 19. 'TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT' - THE TRACK AND
404 SECTOR FIELD FROM A HEADER READ DURING THE VERIFICATION IS NOT
405 CORRECT.
406
407 20. 'WRITE CHECK ERROR' - THE RH REPORTED A WRITE CHECK ERROR AND
408 NO DRIVE ERRORS WERE SET.
409

410
411 7. MISCELLANEOUS
412 -----
413

414 7.1 FORMAT TIME
415

416 IT TAKES APPROX. 10 MINUTES TO FORMAT AN ENTIRE RM03/2 DISK PACK
417 IN 'FORMAT' MODE. THE 'CHECK' MODE TAKES APPROX. 30 MINUTES FOR
418 AN ENTIRE RM03/2 DISK PACK.
419

420 IT TAKES APPROX. 30 MINUTES TO FORMAT AN ENTIRE RM05 DISK PACK
421 IN 'FORMAT' MODE. THE 'CHECK' MODE TAKES APPROX. 90 MINUTES FOR
422 AN ENTIRE RM05 DISK PACK.
423

424 THE 'VERIFY' ONLY MODE TAKES APPROX. 7 MINUTES FOR AN RM03/2 AND
425 APPROX. 20 MINUTES FOR RM05.
426

427
428 7.2 HALTING THE PROGRAM
429

430 THE OPERATOR SHOULD NOT HALT THE PROGRAM VIA THE CONSOLE DURING A
431 FORMAT OPERATION. HALTING THE PROGRAM MAY LEAVE A SECTOR INCORRECTLY
432 FORMATTED. TO TERMINATE THE FORMAT, TYPE A 'CONTROL C'.
433

434 7.3 SURFACE VERIFICATION
435

436 THE FORMATTER PROGRAM IS NOT INTENDED TO BE USED TO PERFORM DISK
437 PACK VERIFICATION. IF THE PROGRAM REPORTS A SECTOR AS BEING 'NOT
438 ACCEPTABLE', THIS MAY IN FACT INDICATE A BAD SURFACE; HOWEVER,
439 SECTORS WHICH 'PASSED' MAY OR MAY NOT BE GOOD.
440

441
442 8. PROGRAM DESCRIPTION
443 -----
444

445 8.1 FORMAT OPERATION
446

447 THE PROGRAM FORMATS THE PACK ONE TRACK AT A TIME BETWEEN THE LIMITS
448 SPECIFIED BY THE OPERATOR.
449

450 THE FORMAT OPERATION CONSISTS OF A WRITE HEADER AND DATA COMMAND FOR
451 THE ENTIRE TRACK FOLLOWED BY A WRITE CHECK HEADER AND DATA COMMAND.
452 IF AN ERROR OCCURS DURING THE WRITE, THE PROGRAM WILL RETRY THE
453 WRITE BEFORE CONTINUING. IF A DATA RELATED ERROR IS DETECTED
454 DURING THE WRITE CHECK (A DATA ERROR IS DEFINED AS ONE OF
455 THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE',
456

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

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 CHECK OPERATION

THE CHECK OPERATION IS IDENTICAL TO THE THE FORMAT OPERATION DESCRIBED IN SECTION 8.1, EXCEPT THAT IN EACH CHECK OPERATION THE FORMAT OPERATION IS REPEATED THREE TIMES WITH DATA PATTERN BEING ROTATED ONE BIT POSITION AT EACH PASS.

8.3 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.

8.4 HEADER VERIFICATION (VERIFY ONLY)

THE HEADER VERIFICATION OPERATION PERFORMS A READ HEADER AND DATA 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

514
515
.16
.17
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

AND ALL THE HEADERS ARE IN GOOD CONDITION.

9. BAD SPOT FILE

SECTORS 0 THRU 31., ON CYLINDER 822., LAST TRACK ARE ALWAYS FORMATTED IN 16 BIT MODE. THEY CONTAIN THE BAD SECTOR ADDRESS ON THE PACK FOR BOTH 30. AND 32. SECTOR OPERATIONS. THE DATA FIELD OF EACH SECTOR OF THE BAD SPOT 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 ZERO.

THE FORTH WORD DEFINES THE PACK IS A DATA PACK ,IF IT IS 0.

THE FOLLOWING 252 WORDS DEFINE THE BAD SECTOR ADDRESS;TWO WORDS ARE USED TO DEFINE A SIGNLE BAD SECTOR, THE FIRST WORD SPECIFIES THE CYLINDER ADDRESS, THE SECOND WORD SPEDIFIES THE TRACK (HIGH BYTE) AND SECTOR (LOW BATE) ADDRESSES. ALL UNUSED LOCATIONS ARE RECORDED WITH ONES.

THE SECTORS ON THE LAST TRACK ARE FURTHER DIVIDED INTO FOUR GROUPS AS FOL_OWS:

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

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 ROUTINES BY ENTERING THE SELECTED FUNCTION NUMBER.

KEYIN U.S.R. BAD SECTORS.....0
PRINT M.F.G. BAD SECTORS.....1

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

PRINT U.S.R. BAD SECTORS..... 2
INITIALIZE U.S.R. BAD SECTOR FILE.....3
PRINT A SECTOR OF THE LAST TRACK.....4
PRINT THIS MESSAGE.....5
EXIT.....<CR>
ENTER UTILITY FUNCTION:

THE FIRST ROUTINE ALLOWS THE USER DEFINED BAD SECTORS TO BE ENTERED MANUALLY BY SPECIFYING THE CYLINDER ADDRESS, TRACK ADDRESS AND SECTOR ADDRESS IN DECIMAL VALUE. TO EXIT FROM THE ROUTINE JUST ENTER A PERIOD(.) FOLLOWED BY A <CR>.

EXAMPLE:
CYLINDER -1 / 200
TRACK -1 / 4
SECTOR -1 / 23
CYLINDER -1 / .<CR>

THE SECOND AND THIRD ROUTINES TYPE THE MANUFACTURE AND USER DEFINED BAD SECTORS ON THE CONSOLE IN OCTAL NUMBER. THE FIRST TWO WORDS ON THE LISTING ARE THE SERIAL NUMBERS FOLLOWED BY TWO WORDS OF ALL 0'S.

THE FOURTH ROUTINE RESETS ALL USER DEFINED BAD SECTOR ADDRESSES TO -1.

THE FIFTH ROUTINE ALLOWS A WHOLE SECTOR OF THE BAD SPOT FILE RETRIEVED FROM THE PACK BEING FORMATTED TO BE TYPED ON THE CONSOLE.

TO EXIT THE ROUTINE 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

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

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

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

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

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

CALL:

JSR	RO, RM03	;MAKE THE CALL
PNTDPB		;ADDRESS OF DPB*
RETURN1		;RETURN IF QUEUE IS FULL
RETURN2		;RETURN IF REQUEST IS IN
		;QUEUE OR THERE IS AN
		;ERROR CONDITION

*DPB (DATA PARAMETER BLOCK)

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

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
-----	-------------	--------------------------

685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741

```

                :CLOCK TICKS
JSR    PC,RMTMR :CALL THE TIMER ROUTINE
    
```

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

10.4.1 EXAMPLE - WRITE 1000. WORDS

```

1$:    JSR    R0,RM03    :CALL THE DRIVER
        FMTDPB          :DPB ADDRESS
2$:    BR     1$         :WAIT FOR QUEUE IF FULL
        TST    FMTDPB+16 :WAI* FOR COMMAND TO COMPLETE
        BEQ    2$
        BMI    ERROR1    :ERROR OCCURRED
        .
        .
    
```

```

FMTDPB: .BYTE 5          :DRIVE #5
        .BYTE 0
        .BYTE 161       :WRITE COMMAND
        .BYTE 0
        .WORD -1000.    :WORD COUNT
        .WORD WRTBUF    :BUFFER ADDRESS
        .BYTE 3         :SECTOR
        .BYTE 5         :TRACK
        .WORD 400       :CYLINDER
        .WORD ERRTB5    :ERROR TABLE
        .WORD 0         :STATUS/ERROR INDICATOR
    
```

ALTERNATE DPB SETUP

```

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

10.5 RH/RM REGISTERS

MNEMONIC	INDEX
-----	-----
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMD5	12
RMER1	14
RMA5	16
RMLA	20

742	RMDB	22
743	RMMR1	24
744	RMDT	26
745	RMSN	30
746	RMOF	32
747	RMDC	34
748	RMHR	36
749	RMMR2	40
750	RMER2	42
751	RMEC1	44
752	RMEC2	46

10.6 COMMANDS PERFORMED BY THE DRIVER

	COMMAND	CODE	COMMAND TYPE
	-----	----	-----
757	NO OPERATION	101	N
758	UNLOAD	103	N
759	SEEK	105	P
760	RECALIRATE	107	P
761	DRIVE CLEAR	111	N
762	RELEASE	113	N
763	OFFSET	115	P
764	RETURN TO CENTER	117	P
765	READIN PRESET	121	N
766	PACK ACKNOWLEDGE	123	N
767	SEARCH	131	P
768	GET REGISTER(S)	141	S
769	SET FORMAT	143	S
770	SELECT DRIVE	145	S
771	WRITE CHECK DATA	151	D
772	WRT CHK HDR & DATA	153	D
773	WRITE DATA	161	D
774	WRITE HEADER & DATA	163	D
775	READ DATA	171	D
776	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 14-10 SPECIFIES TYPE DONE (BIT07=1); BITS 06-03 SPECIFIES TYPE
14(1)	USER MADE A REQUEST FOR A FUNCTION TO BE

799		PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
800		
801	13(1)	USER MADE A REQUEST FOR A FUNCTION
802		TO BE PERFORMED ON A DRIVE THAT HAS AN
803		UNLOAD REQUEST IN QUEUE.
804		
805	12(2)	PERSISTENT UNSAFE CONDITION EXIST.
806		
807	11(2)	UNCORRECTABLE PARITY ERROR OCCURRED
808		
809	10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS
810		PERFORMED, ALL QUEUES WERE EMPTIED, AND
811		ALL DRVACT'S SET TO THE IDLE STATE
812		
813	9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
814		
815	8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
816		
817	7	DONE
818		
819	6(2)	ERROR OCCURRED DURING AN I/O OPERATION
820		
821	5(2)	ERROR OCCURRED DURING AN OPERATION
822		OTHER THAN I/O.
823		
824	4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
825		
826	3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN
827		AUTOMATIC "RECALIBRATE" SEQUENCE
828		
829	2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED
830		THE DRIVE BUT THE OPPOSITE PORT DID NOT
831		RELEASE THE DRIVE WITHIN 20 SECONDS.
832		
833	1	NON-EXISTENT DRIVE REQUESTED. USER MADE
834		A REQUEST FOR A NON-EXISTENT DRIVE.
835		
836	(1) ->	REQUEST WASN'T PUT IN QUEUE. (RH/RM
837		REGISTERS WERE NOT SAVED)
838		
839	(2) >	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER
840		ISSUED A "DRIVE CLEAR" TO THE DRIVE.
841		NOTE: ALL RH/RM REGISTERS ARE SAVED
842		AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
843		
844	(3) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE
845		DRIVER ISSUED A MASSBUS INIT. ALL
846		RH/RM REGISTERS FOR THE DRIVE WERE
847		SAVED AS PER DPB+14 BEFORE THE INIT.
848		
849	(4) >	A "RECALIBRATE" SHOULD BE ISSUED
850		BEFORE ANY OTHER COMMAND.
851		
852	10.8	ERROR CALLS MADE BY THE DRIVER.
853		
854		THERE ARE A FEW ERRORS THAT CAN OCCUR THAT CAN NOT BE INDICATED IN A DPB.
855		

856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894

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

N	TYPE	DATA AVAILABLE
-	----	-----
1	RH70 INTERRUPT OCCURRED (RMAS=0)	*R4= RMCS1'S ADDRESS
2	UNEXPECTED ATTENTION OCCURRED	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2
3	MASSBUS PARITY ERROR (MCPE=1)	RD.ADR= ADDRESS OF REG. READ RD.WRD= WORD READ
4	MASSBUS PARITY ERROR (PAR=1)	WRT.AD= ADDRESS OF REG. WRITTEN WRT.WD= WORD WRITTEN RD.WRD= WORD READ BACK
5	ADDRESS PLUG CHANGE BIT SET ('OPE' ERROR)	R1= DRIVE NUMBER R3= ATA BIT *R4= RMCS1'S ADDRESS R5= (RMAS) RMERRS =RMDS RMERRS+2=RMER1 RMERRS+4=RMER2 RMERRS+6=RMMR2

* THIS IS THE ACTUAL UNIBUS ADDRESS (176700)

z

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      BITS=BIT05
000020      BIT4=BIT04
000010      BIT3=BIT03
000004      BIT2=BIT02
000002      BIT1=BIT01
000001      BIT0=BIT00

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

79
80      .SBTTL  RH11/RH70 REGISTERS
81
82      ;CONTROL AND STATUS REGISTER 1 (RMCS1)
83
84      000100      IE = 100           ;INTERRUPT ENABLE (BIT #6)
85      000200      RDY = 200         ;READY (BIT #7)
86      000400      A16 = 400         ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
87      001000      A17 = 1000        ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
88      002000      PSEL = 2000       ;PORT SELECT (BIT #10)
89      020000      MCPE = 20000      ;MASSBUSS PARITY ERROR (BIT #13)
90      040000      TRE = 40000       ;TRANSFER ERROR (BIT #14)
91      ;SC = 100000                 ;SPECIAL CONDITION (BIT #15)
92
93      ;WORD COUNT REGISTER (RMWC)
94      ;(EACH BIT IS CALLED BY BIT NUMBER)
95
96      ;BUS ADDRESS REGISTER (RMBA)
97      ;(EACH BIT IS CALLED BY BIT NUMBER)
98
99      ;CONTROL AND STATUS REGISTER 2 (RMCS2)
100
101      000001      US1 = 1           ;UNIT SELECT (BIT #0)
102      000002      US2 = 2           ;UNIT SELECT (BIT #1)
103      000004      US4 = 4           ;UNIT SELECT (BIT #2)
104      000010      BAI = 10          ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
105      000020      PAT = 20          ;MASSBUS PARITY TEST (BIT #4)
106      000040      CLR = 40          ;CLEAR (BIT #5)
107      000100      IR = 100          ;INPUT READY (BIT #6)
108      000200      OR = 200          ;OUTPUT READY (BIT #7)
109      000400      MPE = 400         ;MASS BUS PARITY ERROR (BIT #8)
110      001000      MXF = 1000        ;MISSED TRANSFER ERROR (BIT #9)
111      002000      PGE = 2000       ;PROGRAM ERROR (BIT #10)
112      004000      NEM = 4000        ;NON EXISTENT MEMORY (BIT #11)
113      010000      NED = 10000       ;NON EXISTENT DRIVE (BIT #12)
114      020000      UPE = 20000      ;UNIBUS PARITY ERROR (BIT #13)
115      040000      WCE = 40000       ;WRITE CHECK ERROR (BIT #14)

```

```
116      100000      DLT      = 100000      ;DATA LATE (BIT #15)
117
118      ;DATA BUFFER REGISTER (RMDB)
119      ;(EACH BIT IS CALLED BY BIT NUMBER)
120
121      .SBTTL  RM REGISTERS
122
123      ;CONTROL AND STATUS 1 REGISTER. (#00)
124
125      000001      GO      = 1      ;GO BIT (BIT #0)
126      000002      F1      = 2      ;FUNCTION CODE BIT #1
127      000004      F2      = 4      ;FUNCTION CODE BIT #2
128      000010      F3      = 10     ;FUNCTION CODE BIT #3
129      000020      F4      = 20     ;FUNCTION CODE BIT #4
130      000040      F5      = 40     ;FUNCTION CODE BIT #5
131      004000      DVA      = 4000   ;DEVICE AVAILABLE (BIT #11)
132
133      ;DRIVE STATUS REGISTER (RMDS1) (#01)
134
135      000100      VV      = 100     ;VOLUME VALID (BIT #6)
136      000200      DRY     = 200     ;DRIVE READY (BIT #7)
137      000400      DPR     = 400     ;DRIVE PRESENT (BIT #8)
138      001000      PGM     = 1000    ;PROGRAMABLE (BIT #9)
139      002000      LST     = 2000    ;LAST SECTOR TRANSFERRED (BIT #10)
140      004000      WRL     = 4000    ;WRITE LOCK (BIT #11)
141      010000      MOL     = 10000   ;MEDIUM ON-LINE (BIT #12)
142      020000      PIP     = 20000   ;POSITIONING OPERATION IN PROGRESS (BIT #13)
143      040000      ERR     = 40000   ;COMPOSITE ERROR (BIT #14)
144      100000      ATA     = 100000  ;ATTENTION ACTIVE (BIT #15)
145
146      ;ERROR REGISTER #01 (RMER1) (#02)
147
148      000001      ILF     = 1      ;ILLEGAL FUNCTION (BIT #0)
149      000002      ILR     = 2      ;ILLEGAL REGISTER (BIT #1)
150      000004      RMR     = 4      ;REGISTER MODIFICATION REFUSED (BIT #2)
151      000010      PAR     = 10     ;PARITY ERROR (BIT #3)
152      000020      FER     = 20     ;FORMAT ERROR (BIT #4)
153      000040      WCF     = 40     ;WRITE CLOCK FAIL (BIT #5)
154      000100      ECH     = 100    ;ECC HARD ERROR (BIT #6)
155      000200      HCE     = 200    ;HEADER COMPARE ERROR (BIT #7)
156      000400      HCRC    = 400    ;HEADER CRC ERROR (BIT #8)
157      001000      AOE     = 1000   ;ADDRESS OVERFLOW ERROR (BIT #9)
158      002000      IAE     = 2000   ;INVALID ADDRESS ERROR (BIT #10)
159      004000      WLE     = 4000   ;WRITE LOCK ERROR (BIT #11)
160      010000      DTE     = 10000  ;DRIVE TIMING ERROR (BIT #12)
161      020000      OPI     = 20000  ;OPERATION INCOMPLETE (BIT #13)
162      040000      UNS     = 40000  ;DRIVE UNSAFE (BIT #14)
163      100000      DCK     = 100000 ;DATA CHECK ERROR (BIT 15)
164
165      ;MAINTAINABILITY REGISTER (RMMR1)(#03)
166
167      000001      DMD     = 1      ;DIAGINOSTIC MODE (BIT #0)
168      000002      MCLK    = 2      ;MAINTAINABILITY CLOCK (BIT #1)
169      000004      MINX    = 4      ;MAINTAINABILITY INDEX (BIT #2)
170      000010      MSTCK   = 10     ;MAINTAINABILITY SECTOR CLOCK (BIT #3)
171      000020      MRD     = 20     ;MAINTAINABILITY READ (BIT #4)
172      000040      MWR     = 40     ;MAINTAINABILITY WRITE (BIT #5)
```

```

173      000200      DTSY      = 200      ;MAINTAINABILITY SYNC DETECTED (BIT #7)
174
175      ;ATTENTION SUMMARY PSEUDO-REGISTER (RMAS) (#04)
176
177      000001      AT0       = 1      ;DEVICE 0 (BIT #0)
178      000002      AT1       = 2      ;DEVICE 1 (BIT #1)
179      000004      AT2       = 4      ;DEVICE 2 (BIT #2)
180      000010      AT3       = 10     ;DEVICE 3 (BIT #3)
181      000020      AT4       = 20     ;DEVICE 4 (BIT #4)
182      000040      AT5       = 40     ;DEVICE 5 (BIT #5)
183      000100      AT6       = 100    ;DEVICE 6 (BIT #6)
184      000200      AT7       = 200    ;DEVICE 7 (BIT #7)
185
186      ;DESIRED SECTOR/TRACK ADDRESS REGISTER (RMDA) (#05)
187      ;(EACH BIT IS CALLED BY BIT NUMBER)
188
189      ;DRIVE TYPE REGISTER (RMDT) (#06)
190
191      000001      DT00      = 1      ;DRIVE TYPE NUMBER BIT 1
192      000002      DT01      = 2      ;DRIVE TYPE NUMBER BIT 2
193      000004      DT02      = 4      ;DRIVE TYPE NUMBER BIT 3
194      000010      DT03      = 10     ;DRIVE TYPE NUMBER BIT 4
195      000020      DT04      = 20     ;DRIVE TYPE NUMBER BIT 5
196      000040      DT05      = 40     ;DRIVE TYPE NUMBER BIT 6
197      000100      DT06      = 100    ;DRIVE TYPE NUMBER BIT 7
198      000200      DT07      = 200    ;DRIVE TYPE NUMBER BIT 8
199      000400      DT08      = 400    ;DRIVE TYPE NUMBER BIT 9
200      004000      DRQ       = 4000   ;DRIVE REQUEST REQUIRED (BIT #11)
201      020000      MOH       = 20000  ;MOVING HEAD (BIT #13)
202      040000      TAP       = 40000  ;TAPE DRIVE (BIT #14)
203      100000      NBA       = 100000 ;NOT BLOCK ADDRESSED (BIT #15)
204
205      ;LOOK-AHEAD REGISTER (RMLA) (#07)
206
207      000100      SC01      = 100    ;SECTOR COUNT FIELD 0 (BIT #6)
208      000200      SC02      = 200    ;SECTOR COUNT FIELD 1 (BIT #7)
209      000400      SC04      = 400    ;SECTOR COUNT FIELD 2 (BIT #8)
210      001000      SC10      = 1000   ;SECTOR COUNT FIELD 3 (BIT #9)
211      002000      SC20      = 2000   ;SECTOR COUNT FIELD 4 (BIT #10)
212
213      004000      TRK1      = 4000   ;TRACK FIELD 1 (BIT #11)
214      010000      TRK2      = 10000  ;TRACK FIELD 2 (BIT #12)
215      020000      TRK4      = 20000  ;TRACK FIELD 3 (BIT #13)
216      040000      TRK10     = 40000  ;TRACK FIELD 4 (BIT #14)
217      100000      TRK20     = 100000 ;TRACK FIELD 5 (BIT #15)
218
219      ;OFFSET REGISTER (RMOF) (#11)
220
221      000001      OFDIR     = 1      ;OFFSET DIRECTION
222      002000      HCI       = 2000   ;HEADER COMPARE INHIBIT (BIT #10)
223      004000      ECI       = 4000   ;ERROR CORRECTION CODE INHIBIT (BIT #11)
224      010000      FMT16    = 10000  ;FORMAT BIT (BIT #12)
225
226      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
227      ;(EACH BIT IS CALLED BY BIT NUMBER)
228
229      ;SERIAL NUMBER REGISTER (RMSN) (#14)

```



```

230      ;(EACH IS CALLED BY BIT NUMBER)
231
232      ;RM MAINTENANCE REGISTER #02 (RMMR2) (#15)
233
234      040000      SKI      = 40000      ;SEEK INCOMPLETE (BIT #14)
235
236      ;ERROR REGISTER #02 (RMER2)
237
238      100000      BSE      = 100000      ;BAD SECTOR ERROR (BIT #15)
239
240      ;ECC POSITION REGISTER (RMEC1) (#16)
241      ;(EACH BIT IS CALLED BY BIT NUMBER)
242
243      ;ECC PATTERN REGISTER (RMEC2) (#17)
244      ;(EACH BIT IS CALLED BY BIT NUMBER)
245
246      ;SOME 1ST HEADER WORD DEFINITIONS
247
248      100000      MF      = 100000      ;MANUFACTURES BAD SECTOR FLAG (BIT #15)
249      040000      UF      = 40000      ;USERS BAD SECTOR FLAG (BIT #14)
250      010000      FMT      = 10000      ;FORMAT 16 BIT MODE (BIT #12)
251
252      .SBTTL  RM DRIVER COMMANDS
253
254      000101      RNOP     = 101      ;NO OPERATION
255      000105      SEEK     = 105      ;SEEK
256      000107      RECAL    = 107      ;RECALIBRATE
257      000111      DRVCLR   = 111      ;DRIVE CLEAR
258      000113      RELSE    = 113      ;RELEASE
259      000115      OFFSET   = 115      ;OFFSET
260      000117      RTC      = 117      ;RETURN TO CENTER LINE
261      000121      READIN   = 121      ;READ IN PRESET
262      000123      ACK      = 123      ;PACK ACKNOWLEDGE
263      000131      SEARCH   = 131      ;SEARCH
264      000141      GETREG   = 141      ;GET REGISTERS
265      000143      SETFMT   = 143      ;SET FORMAT (& ECI OR HCI)
266      000145      SELDRV   = 145      ;SELECT DRIVE
267      000151      WCKD     = 151      ;WRITE CHECK DATA
268      000153      WCKHD    = 153      ;WRITE CHECK HEADER & DATA
269      000161      WRDAT    = 161      ;WRITE DATA
270      000163      WRTHD    = 163      ;WRITE HEADER & DATA
271      000171      RDDAT    = 171      ;READ DATA
272      000173      RDHD     = 173      ;READ HEADER & DATA
273
274      176700      ABASE    = 176700
275      000254      AVECT1   = 254
276
277      .SBTTL  TRAP CATCHER
278
279      000000      .=0
      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
      .=174
000174 000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
000176 000000      SWREG:  .WORD 0      ;;SOFTWARE SWITCH REGISTER

```

```

                .SBTTL  STARTING ADDRESS(ES)
000200 000137 007014          JMP    @#START      ;;JUMP TO STARTING ADDRESS OF PROGRAM
280 000204 000137 006776          JMP    @#BEGIN      ;CHANGE THE RH ADDRESS
281
282
283
                .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
284
285      001100      .=1100
286
                .SBTTL  APT PARAMETER BLOCK
                ;*****
                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                ;*****
                .SX=.          ;;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
                .=.SX        ;;RESET LOCATION COUNTER
                ;*****
                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                ;INTERFACE SPEC.
001100      $APTHD:
001100      $HIBTS: .WORD 0          ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001102      $MBADR: .WORD $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001104      $TSTM:  .WORD 200.     ;;RUN TIM OF LONGEST TEST
001106      $PASTM: .WORD 200.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001110      $UNITM: .WORD 200.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDED UNIT
001112      .WORD $ETEND-$MAIL/2  ;;LENGTH MAILBOX-ETABLE(WORDS)
287      TAB.XY=.          ;CMTAGSTARTING ADDRESS
288

```

0

.SBTTL COMMON TAGS

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

001114	001114			SCMTAG: . =TAB.XY	:: START OF COMMON TAGS
001114	000000			.WORD 0	
001116	000			\$TSTNM: .BYTE 0	:: CONTAINS THE TEST NUMBER
001117	000			\$ERFLG: .BYTE 0	:: CONTAINS ERROR FLAG
001120	000000			\$ICNT: .WORD 0	:: CONTAINS SUBTEST ITERATION COUNT
001122	000000			\$LPADR: .WORD 0	:: CONTAINS SCOPE LOOP ADDRESS
001124	000000			\$LPERR: .WORD 0	:: CONTAINS SCOPE RETURN FOR ERRORS
001126	000000			\$ERTTL: .WORD 0	:: CONTAINS TOTAL ERRORS DETECTED
001130	000			\$ITEMB: .BYTE 0	:: CONTAINS ITEM CONTROL BYTE
001131	001			\$ERMAX: .BYTE 1	:: CONTAINS MAX. ERRORS PER TEST
001132	000000			\$ERRPC: .WORD 0	:: CONTAINS PC OF LAST ERROR INSTRUCTION
001134	000000			\$GDADR: .WORD 0	:: CONTAINS ADDRESS OF 'GOOD' DATA
001136	000000			\$BDADR: .WORD 0	:: CONTAINS ADDRESS OF 'BAD' DATA
001140	000000			\$GDDAT: .WORD 0	:: CONTAINS 'GOOD' DATA
001142	000000			\$BDDAT: .WORD 0	:: CONTAINS 'BAD' DATA
001144	000000			.WORD 0	:: RESERVED--NOT TO BE USED
001146	000000			.WORD 0	
001150	000			\$AUTOM: .BYTE 0	:: AUTOMATIC MODE INDICATOR
001151	000			\$INTAG: .BYTE 0	:: INTERRUPT MODE INDICATOR
001152	000000			.WORD 0	
001154	177570			SWR: .WORD DSWR	:: ADDRESS OF SWITCH REGISTER
001156	177570			DISPLAY: .WORD DDISP	:: ADDRESS OF DISPLAY REGISTER
001160	177560			\$TKS: 177560	:: TTY KBD STATUS
001162	177562			\$TKB: 177562	:: TTY KBD BUFFER
001164	177564			\$TPS: 177564	:: TTY PRINTER STATUS REG. ADDRESS
001166	177566			\$TPB: 177566	:: TTY PRINTER BUFFER REG. ADDRESS
001170	000			\$NULL: .BYTE 0	:: CONTAINS NULL CHARACTER FOR FILLS
001171	002			\$FILLS: .BYTE 2	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
001172	012			\$FILLC: .BYTE 12	:: INSERT FILL CHARS. AFTER A 'LINE FEED'
001173	000			\$TPFLG: .BYTE 0	:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001174	000000			\$TMPD: .WORD 0	:: USER DEFINED
001176	000000			\$ESCAPE: 0	:: ESCAPE ON ERROR ADDRESS
001200	207	377	377	\$BELL: .ASCIZ <207><377><377>	:: CODE FOR BELL
001204	077			\$QUES: .ASCII /?/	:: QUESTION MARK
001205	015			\$CRLF: .ASCII <15>	:: CARRIAGE RETURN
001206	012	000		\$LF: .ASCIZ <12>	:: LINE FEED

.SBTTL APT MAILBOX-EIABLE

001210				.EVEN	
001210	000000			\$MAIL: .WORD	:: APT MAILBOX
001212	000000			\$MSGTY: .WORD	:: MESSAGE TYPE CODE
001214	000000			\$FATAL: .WORD	:: FATAL ERROR NUMBER
001216	000000			\$TESTN: .WORD	:: TEST NUMBER
001220	000000			\$PASS: .WORD	:: PASS COUNT
001222	000000			\$DEVCT: .WORD	:: DEVICE COUNT
001224	000000			\$UNIT: .WORD	:: I/O UNIT NUMBER
001226	000000			\$MSGAD: .WORD	:: MESSAGE ADDRESS
001228	000000			\$MSGLG: .WORD	:: MESSAGE LENGTH

```

001230          SETABLE:                ;; APT ENVIRONMENT TABLE
001230          000                      $ENV: .BYTE   AENV      ;; ENVIRONMENT BYTE
001231          000                      $ENVM: .BYTE  AENVM     ;; ENVIRONMENT MODE BITS
001232          000000                   $$WREG: .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
00 240          000                      $MAMS1: .BYTE  AMAMS1   ;; HIGH ADDRESS, M.S. BYTE
00 241          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  AMAMS2   ;; 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#2, BUS 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

```

0

.SBTTL USER DEFINED TAGS

001274	176700	SRMADR: .WORD	176700	:RH/RM UNIBUS ADDRESS
001276	000254	SRMVEC: .WORD	254	:RM INTERRUPT VECTOR
001300	172540	SLKCSR: .WORD	172540	:ADDRESS OF KW11-P CSR
001302	172542	SLKCSB: .WORD	172542	:ADDRESS OF KW11-P COUNTER BUFFER
001304	000104	000106	SLPVEC: .WORD	104,106 :ADDRESS OF KW11-P VECTOR
001310	177546	SLKS: .WORD	177546	:ADDRESS OF KW11-L CONTROL REGISTER
001312	000100	000102	SLLVEC: .WORD	100,102 :ADDRESS OF KW11-L VECTOR
	001222	DRIVE =SUNIT		: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 = -1, CHECK = 0 AND VERIFY ONLY = 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	ENDCYL: .WORD	0	:ENDING CYLINDER
001330	000000	BEGCYL: .WORD	0	:STARTING CYLINDER
001332	000000	ENDTRK: .WORD	0	:ENDING TRACK
001334	000000	BEGTRK: .WORD	0	:STARTING TRACK
001336	000000	PATSEL: .WORD	0	:CONTAINS PATTERN SELECTED
001340	000000	PATA: .WORD	0	:1ST WORD OF PATTERN
001342	000000	PATB: .WORD	0	:2ND WORD OF PATTERN
001344	000000	RETRY: .WORD	0	:MAINTAINS # OF WRITE RETRIES MADE
001346	000000	SAVSEC: .WORD	0	:CONTAINS LAST BAD SECTOR ON FORMAT
001350	000000	SAVWC: .WORD	0	:CONTAINS WC FOR REMAINING SECTORS ON ERROR
001352	000000	WC: .WORD	0	:30 OR 32 SECTOR TRACK SIZE (IN WORDS)
001354	000000	MWR: .WORD	0	:2'S COMPLEMENT OF 'WC'
001356	000000	HEDERR: .WORD	0	:POSITIONING ERROR DURING FORMAT INDICATOR
001360	000000	SEC30: .WORD	0	:30 OR 32 SECTOR MODE INDICATOR :0 = 30 SECTOR MODE :1'S = 32 SECTOR MODE
001362	000000	MAXSEC: .WORD	0	:MAXIMUM SECTOR ADDRESS (FOR EITHER 30 OR 32 SECTOR :FORMAT)
001364	000000	DS.CYL: .WORD	0	:ADDRESS OF CURRENT CYLINDER
001366	000000	DS.TRK: .WORD	0	:ADDRESS OF CURRENT TRACK
001370	000000	DDRIVE: .WORD	0	:DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
001372	000000	ATTN: .WORD	0	:ATTENTION REGISTER IMAGE FOR 'DRIVER' :ERROR MESSAGES
001374	000000	CNTLC: .WORD	0	:ADDRESS OF '^C' RETURN
001376	000001	CHGADR: .WORD	1	: 'CHANGE RH/RM ADDRESS' FLAG = 1
001400	000000	WRAP: .WORD	0	:WRAP AROUND FLAG,FORMAT FROM HIGH TO LOW :CYLINDER NUMBER
001402	000000	DEVMP: .WORD	0	:DEVICE MAP FOR APT
001404	000000	PACK: .WORD	0	:PACK >0, BAD SPOT FILE AVAILABLE FROM PACK
001406	000000	EDIT: .WORD	0	:EDIT THE BAD SPOT FILE ,IF EDIT >0
001410	000000	HEADX: .WORD	0	:HEAD BIT INDICATOR :BIT15:BIT14 GOOD SECTOR :BIT15 ALONE, USER DEFINED BAD SECTOR :BIT14 ALONE, MFG DEFINED 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.
001414		BAD16: .BLKW	256.	:MFG 16 BIT BAD SPOT TABLE
002414	177777	.WORD	-1	


```

002416 177777          .WORD  -1
002420 177777          BAD18: .BLKW 256. ;MFG 18 BIT BAD SPOT TABLE
003420 177777          .WORD  -1
003422 177777          .WORD  -1
003424 177777          USTAB: .BLKW 256. ;USER BAD SPOT TABLE
004424 177777          .WORD  -1
004426 177777          .WORD  -1
004430 177777          USSAV: .BLKW 256. ;USER BAD SPOT TABLE SAVE FROM PACK
005430 177777          .WORD  -1
005432 177777          .WORD  -1
005434 177777          SECCU: .BLKW 32. ;32 WORDS MAX TO STORE BAD SECTORS
005534 177777          .WORD  -1
005536 000000          NEXT:  .WORD  0 ;FLAG, IF NEXT=-1,ALL GOOD SPOT CN
                                     ;THE CURRENT TRACK ARE FORMATTED

```

;DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

```

005540 000          FMTDPB: .BYTE  0 ;DRIVE NUMBER
005541 000          .BYTE  0 ;OFFSET VALUE OR FMT16, ECI, AND HCI
005542 000          .BYTE  0 ;COMMAND
005543 000          .BYTE  0 ;PSEL AND A17 AND A16
005544 000000       .WORD  0 ;WORD COUNT (NEG)
005546 000000       .WORD  0 ;BUFFER ADDRESS
005550 000          .BYTE  0 ;SECTOR ADDRESS
005551 000          .BYTE  0 ;TRACK ADDRESS
005552 000000       .WORD  0 ;CYLINDER ADDRESS
005554 005560       .WORD  RM.REG ;ERROR TABLE POINTER
005556 000000       .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

```

;RM/RM REGISTERS STORED HERE AFTER AN OPERATION

```

005560 000000       RM.REG: .WORD  0 ;RMCS1
005562 000000       .WORD  0 ;RMWC
005564 000000       .WORD  0 ;RMBA
005566 000000       .WORD  0 ;RMDA
005570 000000       .WORD  0 ;RMCS2
005572 000000       .WORD  0 ;RMDS
005574 000000       .WORD  0 ;RMER1
005576 000000       .WORD  0 ;RMAS
005600 000000       .WORD  0 ;RMLA
005602 000000       .WORD  0 ;RMDB
005604 000000       .WORD  0 ;RMMR1
005606 000000       .WORD  0 ;RMDT
005610 000000       .WORD  0 ;RMSN
005612 000000       .WORD  0 ;RMCF
005614 000000       .WORD  0 ;RMCA
005616 000000       .WORD  0 ;RMDC
005620 000000       .WORD  0 ;RMER2
005622 000000       .WORD  0 ;RMMR2

```

```

005624 000000      .WORD 0      ;RMEC1
005626 000000      .WORD 0      ;RMEC2

005630      000      FMTX: .BYTE 0      ;SPECIAL DPB TO WRITE HEADER OF BAD SPOT
005631      000      .BYTE 0
005632      000      .BYTE 0
005633      000      .BYTE 0
005634 177776      .WORD -2      ;ONLY TWO WORDS FOR HEADER
005636 045764      .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 005776 001467 001330 TABLE: PAR1,823.,BEGCYL
005656 006011 000005 001334      PAR2,5.,BEGTRK
005664 006024 001467 001326      PAR3,823.,ENDCYL
005672 006035 000005 001332      PAR4,5.,ENDTRK,0

005702 006046 001467 001364 TABLE2: PAR5,823.,DS.CYL
005710 006057 000005 001366      PAR6,5.,DS.TRK
005716 006070 000040 001346      PAR7,32.,SAVSEC,0

005726 006046 001467 001364 TABLE3: PAR5,823.,DS.CYL
005734 006057 000005 001366      PAR6,5.,DS.TRK
005742 006070 000036 001346      PAR7,30.,SAVSEC,0

005752 006046 001467 001364 TABLE4: PAR5,823.,DS.CYL
005760 006057 000005 001366      PAR6,5,DS.TRK
005766 006101 047301 001346      PAR8,20161.,SAVSEC,0
  
```

:ASCII MESSAGES FOR ADDRESS PARAMETERS

```

005776      123      124      101 PAR1: .ASCIZ @START CYL @
006011      123      124      101 PAR2: .ASCIZ @START TRK @
006024      105      116      104 PAR3: .ASCIZ @END CYL @
006035      105      116      104 PAR4: .ASCIZ @END TRK @

006046      103      131      114 PAR5: .ASCIZ /CYLINDER/
006057      124      122      101 PAR6: .ASCIZ /TRACK /
006070      123      105      103 PAR7: .ASCIZ /SECTOR /
006101      102      131      124 PAR8: .ASCIZ /BYTE POSITION /
                                .EVEN
  
```

```

006120 022116 022414 022502 TABCAL: .WORD FUNCT2,FUNCT3,FUNCT4,FUNCT6,FUNCT7
  
```

:SECTOR BUFFER ADDRESS TABLE

```

006132 045774      ADRTBL: .WORD BUFP      ;ADDRESS OF SECTOR 0
006134 047000      .WORD BUFP+<516.*1.> ;ADDRESS OF SECTOR 1.
006136 050004      .WORD BUFP+<516.*2.> ;ADDRESS OF SECTOR 2.
006140 051010      .WORD BUFP+<516.*3.> ;ADDRESS OF SECTOR 3.
006142 052014      .WORD BUFP+<516.*4.> ;ADDRESS OF SECTOR 4.
  
```

006144	053020	.WORD	BUFP+<516.*5.>	; ADDRESS OF SECTOR 5.
006146	054024	.WORD	BUFP+<516.*6.>	; ADDRESS OF SECTOR 6.
006150	055030	.WORD	BUFP+<516.*7.>	; ADDRESS OF SECTOR 7.
006152	056034	.WORD	BUFP+<516.*8.>	; ADDRESS OF SECTOR 8.
006154	057040	.WORD	BUFP+<516.*9.>	; ADDRESS OF SECTOR 9.
006156	060044	.WORD	BUFP+<516.*10.>	; ADDRESS OF SECTOR 10.
006160	061050	.WORD	BUFP+<516.*11.>	; ADDRESS OF SECTOR 11.
006162	062054	.WORD	BUFP+<516.*12.>	; ADDRESS OF SECTOR 12.
006164	063060	.WORD	BUFP+<516.*13.>	; ADDRESS OF SECTOR 13.
006166	064064	.WORD	BUFP+<516.*14.>	; ADDRESS OF SECTOR 14.
006170	065070	.WORD	BUFP+<516.*15.>	; ADDRESS OF SECTOR 15.
006172	066074	.WORD	BUFP+<516.*16.>	; ADDRESS OF SECTOR 16.
006174	067100	.WORD	BUFP+<516.*17.>	; ADDRESS OF SECTOR 17.
006176	070104	.WORD	BUFP+<516.*18.>	; ADDRESS OF SECTOR 18.
006200	071110	.WORD	BUFP+<516.*19.>	; ADDRESS OF SECTOR 19.
006202	072114	.WORD	BUFP+<516.*20.>	; ADDRESS OF SECTOR 20.
006204	073120	.WORD	BUFP+<516.*21.>	; ADDRESS OF SECTOR 21.
006206	074124	.WORD	BUFP+<516.*22.>	; ADDRESS OF SECTOR 22.
006210	075130	.WORD	BUFP+<516.*23.>	; ADDRESS OF SECTOR 23.
006212	076134	.WORD	BUFP+<516.*24.>	; ADDRESS OF SECTOR 24.
006214	077140	.WORD	BUFP+<516.*25.>	; ADDRESS OF SECTOR 25.
006216	100144	.WORD	BUFP+<516.*26.>	; ADDRESS OF SECTOR 26.
006220	101150	.WORD	BUFP+<516.*27.>	; ADDRESS OF SECTOR 27.
006222	102154	.WORD	BUFP+<516.*28.>	; ADDRESS OF SECTOR 28.
006224	103160	.WORD	BUFP+<516.*29.>	; ADDRESS OF SECTOR 29.
006226	104164	.WORD	BUFP+<516.*30.>	; ADDRESS OF SECTOR 30.
006230	105170	.WORD	BUFP+<516.*31.>	; ADDRESS OF SECTOR 31.

:REMAINING WORD COUNT TABLE

006232	000402	WCTBL: .WORD	258.	; REMAINING WORD COUNT AFTER SECTOR 0
006234	001004	.WORD	258.+<258.*1.>	; REMAINING WORD COUNT AFTER SECTOR 1.
006236	001406	.WORD	258.+<258.*2.>	; REMAINING WORD COUNT AFTER SECTOR 2.
006240	002010	.WORD	258.+<258.*3.>	; REMAINING WORD COUNT AFTER SECTOR 3.
006242	002412	.WORD	258.+<258.*4.>	; REMAINING WORD COUNT AFTER SECTOR 4.
006244	003014	.WORD	258.+<258.*5.>	; REMAINING WORD COUNT AFTER SECTOR 5.
006246	003416	.WORD	258.+<258.*6.>	; REMAINING WORD COUNT AFTER SECTOR 6.
006250	004020	.WORD	258.+<258.*7.>	; REMAINING WORD COUNT AFTER SECTOR 7.
006252	004422	.WORD	258.+<258.*8.>	; REMAINING WORD COUNT AFTER SECTOR 8.
006254	005024	.WORD	258.+<258.*9.>	; REMAINING WORD COUNT AFTER SECTOR 9.
006256	005426	.WORD	258.+<258.*10.>	; REMAINING WORD COUNT AFTER SECTOR 10.
006260	006030	.WORD	258.+<258.*11.>	; REMAINING WORD COUNT AFTER SECTOR 11.
006262	006432	.WORD	258.+<258.*12.>	; REMAINING WORD COUNT AFTER SECTOR 12.
006264	007034	.WORD	258.+<258.*13.>	; REMAINING WORD COUNT AFTER SECTOR 13.
006266	007436	.WORD	258.+<258.*14.>	; REMAINING WORD COUNT AFTER SECTOR 14.
006270	010040	.WORD	258.+<258.*15.>	; REMAINING WORD COUNT AFTER SECTOR 15.
006272	010442	.WORD	258.+<258.*16.>	; REMAINING WORD COUNT AFTER SECTOR 16.
006274	011044	.WORD	258.+<258.*17.>	; REMAINING WORD COUNT AFTER SECTOR 17.
006276	011446	.WORD	258.+<258.*18.>	; REMAINING WORD COUNT AFTER SECTOR 18.
006300	012050	.WORD	258.+<258.*19.>	; REMAINING WORD COUNT AFTER SECTOR 19.
006302	012452	.WORD	258.+<258.*20.>	; REMAINING WORD COUNT AFTER SECTOR 20.
006304	013054	.WORD	258.+<258.*21.>	; REMAINING WORD COUNT AFTER SECTOR 21.
006306	013456	.WORD	258.+<258.*22.>	; REMAINING WORD COUNT AFTER SECTOR 22.
006310	014060	.WORD	258.+<258.*23.>	; REMAINING WORD COUNT AFTER SECTOR 23.
006312	014462	.WORD	258.+<258.*24.>	; REMAINING WORD COUNT AFTER SECTOR 24.
006314	015064	.WORD	258.+<258.*25.>	; REMAINING WORD COUNT AFTER SECTOR 25.
006316	015466	.WORD	258.+<258.*26.>	; REMAINING WORD COUNT AFTER SECTOR 26.

006320	016070	.WORD	258.<258.*27.>	:REMAINING WORD COUNT AFTER SECTOR 27.
006322	016472	.WORD	258.<258.*28.>	:REMAINING WORD COUNT AFTER SECTOR 28.
006324	017074	.WORD	258.<258.*29.>	:REMAINING WORD COUNT AFTER SECTOR 29.
006326	017476	.WORD	258.<258.*30.>	:REMAINING WORD COUNT AFTER SECTOR 30.
006330	020100	.WORD	258.<258.*31.>	:REMAINING WORD COUNT AFTER SECTOR 31.

006332	001166	BYTE16: .WORD	630.	
006334	002354	.WORD	630.<630.*1.>	
006336	003542	.WORD	630.<630.*2.>	
006340	004730	.WORD	630.<630.*3.>	
006342	006116	.WORD	630.<630.*4.>	
006344	007304	.WORD	630.<630.*5.>	
006346	010472	.WORD	630.<630.*6.>	
006350	011660	.WORD	630.<630.*7.>	
006352	013046	.WORD	630.<630.*8.>	
006354	014234	.WORD	630.<630.*9.>	
006356	015422	.WORD	630.<630.*10.>	
006360	016610	.WORD	630.<630.*11.>	
006362	017776	.WORD	630.<630.*12.>	
006364	021164	.WORD	630.<630.*13.>	
006366	022352	.WORD	630.<630.*14.>	
006370	023540	.WORD	630.<630.*15.>	
006372	024726	.WORD	630.<630.*16.>	
006374	026114	.WORD	630.<630.*17.>	
006376	027302	.WORD	630.<630.*18.>	
006400	030470	.WORD	630.<630.*19.>	
006402	031656	.WORD	630.<630.*20.>	
006404	033044	.WORD	630.<630.*21.>	
006406	034232	.WORD	630.<630.*22.>	
006410	035420	.WORD	630.<630.*23.>	
006412	036606	.WORD	630.<630.*24.>	
006414	037774	.WORD	630.<630.*25.>	
006416	041162	.WORD	630.<630.*26.>	
006420	042350	.WORD	630.<630.*27.>	
006422	043536	.WORD	630.<630.*28.>	
006424	044724	.WORD	630.<630.*29.>	
006426	046112	.WORD	630.<630.*30.>	
006430	047300	.WORD	630.<630.*31.>	

006432	001240	BYTE18: .WORD	672.	
006434	002500	.WORD	672.<672.*1.>	
006436	003740	.WORD	672.<672.*2.>	
006440	005200	.WORD	672.<672.*3.>	
006442	006440	.WORD	672.<672.*4.>	
006444	007700	.WORD	672.<672.*5.>	
006446	011140	.WORD	672.<672.*6.>	
006450	012400	.WORD	672.<672.*7.>	
006452	013640	.WORD	672.<672.*8.>	
006454	015100	.WORD	672.<672.*9.>	
006456	016340	.WORD	672.<672.*10.>	
006460	017600	.WORD	672.<672.*11.>	
006462	021040	.WORD	672.<672.*12.>	
006464	022300	.WORD	672.<672.*13.>	
006466	023540	.WORD	672.<672.*14.>	
006470	025000	.WORD	672.<672.*15.>	
006472	026240	.WORD	672.<672.*16.>	
006474	027500	.WORD	672.<672.*17.>	

006476	030740	.WORD	672.+<672.*18.>
006500	032200	.WORD	672.+<672.*19.>
006502	033440	.WORD	672.+<672.*20.>
006504	034700	.WORD	672.+<672.*21.>
006506	036140	.WORD	672.+<672.*22.>
006510	037400	.WORD	672.+<672.*23.>
006512	040640	.WORD	672.+<672.*24.>
006514	042100	.WORD	672.+<672.*25.>
006516	043340	.WORD	672.+<672.*26.>
006520	044600	.WORD	672.+<672.*27.>
006522	046040	.WORD	672.+<672.*28.>
006524	047300	.WORD	672.+<672.*29.>

.EVEN

0

.SBTTL ERROR POINTER TABLE

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

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

1	006526			\$ERRTB:	
2				:ERROR 1	
3					
4	006526	042226	EM1		:RH/RM INTERRUPT OCCURRED (RMAS=0)
5	006530	043472	DH1		
6	006532	045030	DT1		
7	006534	045402	DF1		
8					
9				:ERROR 2	
10					
11	006536	042270	EM2		:UNEXPECTED ATTENTION OCCURRED
12	006540	043540	DH2		
13	006542	045046	DT2		
14	006544	045406	DF2		
15					
16				:ERROR 3	
17					
18	006546	042326	EM3		:MASSBUS PARITY ERROR (MCPE=1)
19	006550	043612	DH3		
20	006552	045062	DT3		
21	006554	045412	DF3		
22					
23				:ERROR 4	
24					
25	006556	042364	EM4		:MASSBUS PARITY ERROR (PAR-1)
26	006560	043663	DH4		
27	006562	045076	DT4		
28	006564	045416	DF4		
29					
30				:ERROR 5	
31					
32	006566	000000	0		:UNUSED
33	006570	000000	0		
34	006572	000000	0		
35	006574	000000	0		
36					
37				:ERROR 6	
38					
39	006576	042455	EM6		:RH/RM DIDN'T RESPOND TO ADDRESSING
40	006600	043746	DH6		
41	006602	045114	DT6		
42	006604	045422	DF6		

43			
44			
45			:ERROR 7
46	006606	042520	EM7
47	006610	044554	DH23
48	006612	045116	DT7
49	006614	045512	DF25
50			
51			:ERROR 10
52			
53	006616	042572	EM10
54	006620	043761	DH10
55	006622	045126	DT10
56	006624	045426	DF10
57			
58			:ERROR 11
59			
60	006626	042610	EM11
61	006630	043761	DH10
62	006632	045126	DT10
63	006634	045426	DF10
64			
65			:ERROR 12
66			
67	006636	042646	EM12
68	006640	043761	DH10
69	006642	045126	DT10
70	006644	045426	DF10
71			
72			:ERROR 13
73			
74	006646	042711	EM13
75	006650	043761	DH10
76	006652	045126	DT10
77	006654	045426	DF10
78			
79			:ERROR 14
80			
81	006656	042732	EM14
82	006660	043761	DH10
83	006662	045126	DT10
84	006664	045426	DF10
85			
86			:ERROR 15
87			
88	006666	042755	EM15
89	006670	043761	DH10
90	006672	045126	DT10
91	006674	045426	DF10
92			
93			:ERROR 16
94			
95	006676	043021	EM16
96	006700	043761	DH10
97	006702	045126	DT10
98	006704	045426	DF10
99			

:TRACK/SECTOR FIELD IN HEADER IS NOT CORRECT

:DRIVE OFFLINE

:PERSISTENT DRIVE UNSAFE ERROR

:UNCORRECTABLE MASSBUS PARITY ERROR

:SOFTWARE TIMEOUT

:DRIVE UNSAFE ERROR

:CONTROLLER/DRIVE ERROR DURING WRITE

:CONTROLLER/DRIVE ERROR DURING WRITE CHECK

100			;ERROR 17	
101				
102	006706	043073	EM17	;RETRIES NOT SUCESSFUL - SECTOR NOT ACCEPTABLE
103	006710	044210	DH17	
104	006712	045200	DT17	
105	006714	045442	DF17	
106				
107			;ERROR 20	
108				
109	006716	043151	EM20	;DATA ERROR DURING WRITE CHECK
110	006720	044257	DH20	
111	006722	045212	DT20	
112	006724	045446	DF20	
113				
114			;ERROR 21	
115				
116	006726	043207	EM21	;CONTROLLER/DRIVE ERROR VERIFYING HEADERS
117	006730	043761	DH10	
118	006732	045126	DT10	
119	006734	045426	DF10	
120				
121			;ERROR 22	
122				
123	006736	043260	EM22	; 'MCF' OR 'HCRC' ERROR VERIFYING HEADERS
124	006740	043761	DH10	
125	006742	045126	DT10	
126	006744	045426	DF10	
127				
128			;ERROR 23	
129				
130	006746	043330	EM23	;CYLINDER FIELD IN HEADER IS NOT CORRECT
131	006750	044554	DH23	
132	006752	045274	DT23	
133	006754	045512	DF25	
134				
135			;ERROR 24	
136				
137	006756	043400	EM24	;WRITE CHECK ERROR
138	006760	044613	DH24	
139	006762	045304	DT24	
140	006764	045472	DF24	
141				
142			;ERROR 25	
143				
144	006766	043422	EM25	;ILLEGAL BAD SECTOR
145	006770	044766	DH25	
146	006772	045372	DT25	
147	006774	045512	DF25	
148				

```
1 .SBTTL PROGRAM START
2
3 006776 012737 000001 001376 BEGIN: MOV #1,CHGADR ;ENABLE CHANGE OF 'RH/RM BUS ADDRESS'
4 007004 012737 177777 001270 MOV #-1,$CDW1 ;ENABLE USE OF UTILITY ROUTINES
5 007012 000402 BR BEGIN2
6
7 007014 005037 001270 START: CLR $CDW1 ;DISABLE USE OF UTILITY ROUTINES
8
9 007020 BEGIN2:
;*****
10 007020 000240 TST1: NOP
11 007022 012737 000001 001214 MOV #1,$TESTN ;:SET TEST NUMBER IN APT MAIL BOX
12 007030 005227 000000 INC #0 ;:TTY LOOP, WAIT FOR INCREMENT
13 007034 001375 BNE .-4 ;:OF WORD
14 007036 000005 RESET ;:CLEAR THE WORLD

.SBTTL INITIALIZE THE COMMON TAGS
;:CLEAR THE COMMON TAGS ($CMTAG) AREA
007040 012706 001114 MOV #CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
007044 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION
007046 022706 001154 CMP #SWR,R6 ;:DONE?
007052 001374 BNE .-6 ;:LOOP BACK IF NO
007054 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER

;:INITIALIZE A FEW VECTORS
007060 012737 023620 000030 MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
007066 012737 000340 000032 MOV #340,@EMTVEC+2 ;:LEVEL 7
007074 012737 027602 000034 MOV #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
007102 012737 000340 000036 MOV #340,@TRAPVEC+2 ;:LEVEL 7
007110 012737 027030 000024 MOV #SPWRDN,@PWRVEC ;:POWER FAILURE VECTOR
007116 012737 000340 000026 MOV #340,@PWRVEC+2 ;:LEVEL 7
007124 005037 001176 CLR $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
007130 112737 000001 001131 MOVB #1,$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.
007136 013746 000004 MOV @ERRVEC,-(SP) ;:SAVE ERROR VECTOR
007142 012737 007176 000004 MOV #64$,@ERRVEC ;:SET UP ERROR VECTOR
007150 012737 177570 001154 MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
007156 012737 177570 001156 MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
007164 022777 177777 171762 CMP #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
007172 001012 BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
;:AND THE HARDWARE SWR IS NOT = -1
007174 000403 BR 65$ ;:BRANCH IF NO TIMEOUT
007176 012716 007204 64$: MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
007202 000002 RTI
007204 012737 000176 001154 65$: MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
007212 012737 000174 001156 MOV #DISPREG,DISPLAY
007220 012637 000004 66$: MOV (SP)+,@ERRVEC ;:RESTORE ERROR VECTOR

007224 005037 001216 CLR $PASS ;:CLEAR PASS COUNT
007230 132737 000200 001231 BITB #APTSIZE,$ENVM ;:TEST USER SIZE UNDER APT
007236 001403 BEQ 67$ ;:YES,USE NON-APT SWITCH
007240 012737 001232 001154 MOV #SSWREG,SWR ;:NO,USE APT SWITCH REGISTER
007246 67$:

15 .SBTTL TYPE PROGRAM NAME
;:TYPE THE NAME OF THE PROGRAM* IF FIRST PASS
007246 005227 177777 INC #-1 ;:FIRST TIME?
007252 001043 BNE 68$ ;:BRANCH IF NO
```

```

007254 104401 007322
007260 005737 000042
007264 001012
007266 123727 001230 000001
007274 001406
007276 023727 001154 000176
007304 001005
007306 104406
007310 000403
007312 112737 000001 001150
007320
007320 000420
007362
16 007362 005227 177777
17 007366 001024
18 007370 104401 007376
007374 000421
007440
22
23
24
25
26 007440 005037 001412
27 007444 122737 000016 000041
28 007452 001160
29 007454 013737 000040 001412
30 007462 122737 000007 001412
31 007470 103002
32 007472 105037 001412
33 007476 005737 000042
34 007502 001425
35 007504 104401 007512
007510 000412
007536
36 007536 005046
37 007540 113716 001412
38 007544 104403
39 007546 001
40 007547 000
41 007550 104401 001205
42 007554 000517
43
44 007556 005227 177777
45 007562 001114
46 007564 104401 007572
007570 000410
007612
47 007612 005046
48 007614 113716 001412
49 007620 104403
50 007622 001
51 007623 000

.SBTTL TYPE ,69$ ;;TYPE ASCIZ STRING
GET VALUE FOR SOFTWARE SWITCH REGISTER
TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
BNE 70$ ;;BRANCH IF YES
CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
BEQ 70$ ;;BRANCH IF YES
CMP SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
BNE 71$ ;;BRANCH IF NO
GTSWR ;;GET SOFT-SWR SETTINGS
BR 71$
MOV# #1,$AUTOB ;;SET AUTO-MODE INDICATOR
BR 68$ ;;GET OVER THE ASCIZ
69$: .ASCIZ <CRLF>@CZRMLAO - RM05/3/2 FORMATTER@<CRLF>
68$:
INC #-1 ;;FIRST TIME THRU ?
BNE 1$ ;;BR IF NO
TYPE ,72$ ;;TYPE ASCIZ STRING
BR 1$ ;;GET OVER THE ASCIZ
72$: .ASCIZ <CRLF>@PROGRAM REQUIRES 20K OF MEMORY@<CRLF>
1$:
; THE FOLLOWING FINDS OUT THE PROGRAM CONTROL MODE:
;PAPER TAPE (MANUAL), ACT11, XXDP CHAIN OR DUMP
CLR XXDP ;;CLEAR 'XXDP' LOAD DEVICE STORAGE
CMPB #16,@#41 ;;LOADED FROM AN RM05/3/2 ?
BNE 4$ ;;BR IF NOT
MOV @#40,XXDP ;;GET DEVICE INDICATOR AND NUMBER
CMPB #7,XXDP ;;IS IT A VALID NUMBER ?
BHS 2$ ;;YES
CLRB XXDP ;;NO, DEFAULT TO DRIVE 0
TST @#42 ;;CHAIN MODE OR ACT11 AUTO ACCEPT ?
BEQ 3$ ;;BR IF NEITHER
TYPE ,74$ ;;TYPE ASCIZ STRING
BR 73$ ;;GET OVER THE ASCIZ
74$: .ASCIZ <CRLF>/NOT TESTING DRIVE /
73$:
CLR -(SP) ;;CLEAR WORD ON STACK
MOV# XXDP,(SP) ;;GET DRIVE ADDRESS
TYPOS ;;TYPE THE ADDRESS
.BYTE 1 ;;ONLY 1 CHARACTER
.BYTE 0 ;;SUPRESS LEADING ZEROS
TYPE ,8CRLF ;;CR-LF
BR 4$ ;;GET NUMBER OF DRIVES
3$: INC #-1 ;;FIRST TIME THRU HERE ?
BNE 4$ ;;NO
TYPE ,76$ ;;TYPE ASCIZ STRING
BR 75$ ;;GET OVER THE ASCIZ
76$: .ASCIZ <CRLF>/TO TEST DRIVE /
75$:
CLR -(SP) ;;CLEAR WORD ON STACK
MOV# XXDP,(SP) ;;GET DRIVE ADDRESS
TYPOS ;;TYPE DRIVE ADDRESS
.BYTE 1 ;;ONLY 1 CHARACTER
.BYTE 0 ;;SUPRESS LEADING ZEROS

```

```

52 007624 104401 007632      TYPE      ,78$      ;;TYPE ASCIZ STRING
   007630 000431              BR          ,77$      ;;GET OVER THE ASCIZ
   007714              ;;78$: .ASCIZ /, HALT PROGRAM, REMOVE RRD P AND REPLACE IT/<CRLF>
53 007714 104401 007722      TYPE      ,79$      ;;TYPE ASCIZ STRING
   007720 000435              BR          ,4$        ;;GET OVER THE ASCIZ
   010014              ;;79$: .ASCIZ /WITH A WORK PACK, CLEAR LOCATION 40 AND RESTART PROGRAM./<CRLF>
57 010014 012737 007014 001374  MOV      #START,CNTLC  ;'CONTROL C' ADDRESS
58 010022 004737 025576      JSR      PC,$TKINT    ;TURN ON THE KEYBOARD INTERRUPT
59
60 010026 105737 001150      TSTB     $AUTOB      ;RUNNING AUTO MODE ?
61 010032 001010      BNE     5$          ;BR IF YES
62 010034 004737 045516      JSR      PC,BUSADR   ;CHECK THE RH/RM ADDRESS
63 010040 013737 001274 030034  MOV      $RMADR,$RADR ;RH/RM ADDRESS
64 010046 013737 001276 030036  MOV      $RMVEC,$RVEC ;RH/RM VECTOR ADDRESS
65 010054              5$:
66 010054 105737 001230      TSTB     $ENV        ;RUN UNDER APT MODE?
67 010060 001422      BEQ     SETVEC      ;NO, DO NOT LOAD FROM E-TABLE
68 010062 105737 001260      TSTB     $VECT1     ;UUT INTERRUPT VECTOR
69 010066 001406      BEQ     6$          ;DO NOT CHANGE IF = 0
70 010070 113737 001260 001276  MOVB     $VECT1,$RMVEC ;NEW VALUE
71 010076 013737 001276 030036  MOV      $RMVEC,$RVEC ;RH/RM VECTOR
72 010104 005737 001264      6$: TST      $BASE      ;UUT BASE REGISTER ADDRESS
73 010110 001406      BEQ     SETVEC      ;DO NOT CHANGE IF = 0
74 010112 013737 001264 001274  MOV      $BASE,$RADR ;NEW BASE ADDRESS
75 010120 013737 001274 030034  MOV      $RMADR,$RADR ;RH/RM ADDRESS
76
77 ;DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE
78 ;PROGRAM WILL USE
79
80 010126 004737 020016      SETVEC: JSR      PC,$T.CLK ;START THE CLOCK
81 010132 004737 030052      JSR      PC,$RMINIT  ;INITIALIZE THE RM DRIVER
82 010136 012737 177777 027774  MOV      #-1,$SAVEFG ;SET THE SAVE REGISTERS FLAG
83 010144 005227 177777      INC     #-1         ;FIRST TIME THRU HERE ?
84 010150 001404      BEQ     1$          ;BR IF NO
85 010152 032777 000004 170774  BIT     #SW02,$SWR   ;TYPEOUT THE DRIVE STATUS TABLE ?
86 010160 001113      BNE     12$         ;BR IF NO
87 010162 012737 000340 177776  1$: MOV     #PR7,$PS    ;DON'T ALLOW RM INTERRUPT
88 010170 005037 001402      CLR     DEVMP       ;CLEAR DEVICE MAP
89 010174 005004      CLR     R4          ;DRIVE TABLE POINTER
90 010176 104401 036354      TYPE    ,UNSTAT     ;TYPE 'UNIT STATUS'
91 010202 104401 001205      2$: TYPE    ,CRLF     ;CR-LF
92 010206 010446      MOV     R4,-(SP)    ;;SAVE R4 FOR TYPEOUT
   010210 104403      TYPOS   ;TYPE DRIVE NUMBER
   010212 002      .BYTE  2      ;;GO TYPE--OCTAL ASCII
   010213 000      .BYTE  0      ;;TYPE 2 DIGIT(S)
93 010214 104401 036423      TYPE    ,BLNKS4    ;;SUPPRES LEADING ZEROS
94 010220 105764 027706      TSTB   DRVSTA(R4)  ;TYPE 4 SPACES
95 010224 100416      BMI    5$          ;CHECK DRIVE'S STATUS
96 010226 001020      BNE    6$          ;BR IF UNSAFE
97
98 010230 105764 027716      TSTB   DRVSTYP(R4) ;BR IF ONLINE
99 010234 001404      BEQ    3$          ;SEE IF OFFLINE OR NONEXISTENT
100 010236 100004      BPL    4$          ;BR IF NONEXISTENT
101 010240 104401 056247      TYPE    ,NOTRM     ;BR IF OFFLINE
                        ;DRIVE NOT AN RM05/3/2

```

```

102 010244 000455          BR      11$          ;CHECK NEXT DRIVE
103
104 010246 104401 036270  3$:    TYPE      ,NOTPRS      ;DRIVE NOT PRESENT
105 010252 000452          BR      11$          ;CHECK NEXT DRIVE
106
107 010254 104401 036226  4$:    TYPE      ,UNTOFF      ;DRIVE OFFLINE
108 010260 000421          BR      8$           ;PRINT DRIVE TYPE
109
110 010262 104401 036305  5$:    TYPE      ,NOTSAF      ;DRIVE UNSAFE
111 010266 000416          BR      8$           ;PRINT DRIVE TYPE
112
113 010270 005737 001412  6$:    TST       ,XXDP        ;LOADED FROM THIS DEVICE ?
114 010274 001406          BEQ      7$           ;BR IF NO
115 010276 123704 001412  CMPB    ,XXDP,R4      ;LOADED FROM THIS DRIVE ?
116 010302 001003          BNE     7$           ;BR IF NO
117 010304 104401 036315  TYPE    ,LODEV        ;DRIVE IS LOAD DEVICE
118 010310 000433          BR      11$
119
120 010312 104401 036237  7$:    TYPE      ,UNTON        ;DRIVE ONLINE
121 010316 156437 030022 001402  BISB    ,ATABIT(R4),DEVMP ;SET DFVILE MAP
122 010324 104401 036425  8$:    TYPE      ,BLNKS2      ;TYPE 2 SPACES
123 010330 005000          CLR     R0
124 010332 116400 027716  MOVB    ,DRVTP(R4),R0   ;GET DRIVE TYPE
125 010336 012737 036342 010376  MOV     ,#SRM03,10$    ;ASSUME ADDRESS OF RM03 MESSAGE
126 010344 122700 000004  CMPB    ,#4,R0         ;IS DEVICE AN RM03 ?
127 010350 001411          BEQ     9$           ;BR IF YES
128 010352 012737 036335 010376  MOV     ,#SRM02,10$    ;ADDRESS OF RM02 MESSAGE
129 010360 122700 000005  CMPB    ,#5,R0         ;IS DEVICE AN RM02 ?
130 010364 001403          BEQ     9$           ;BR IF YES
131 010366 012737 036347 010376  MOV     ,#SRM05,10$    ;ADDRESS OF RM05 MESSAGE
132
133 010374 104401          9$:    TYPE      ;TYPE THE DRIVE TYPE MESSAGE
134 010376 000000          10$:   .WORD    0        ;MESSAGE ADDRESS HERE
135
136 010400 005204          11$:   INC      R4         ;INCREMENT DRIVE NUMBER/TABLE POINTER
137 010402 020427 000010  CMP     ,R4,#8        ;FINISHED ?
138 010406 001275          BNE     2$           ;BR IF NOT
139 010410 104401 001205  12$:   TYPE      ,SCLF        ;CR-LF
140
141 010414 105737 001150  TSTB    ,SAUTOB       ;RUNNING UNDER AUTO MODE ?
142 010420 001566          BEQ     M1          ;BR IF NO
143
144          ;PROGRAM IS RUNNING IN AUTO MODE - SEE IF SIZING IS ALLOWED
145 010422          XSIZE:
146 010422 132737 000200 001231  BITB    ,#BIT7,$ENVM   ;APT AUTO SIZING ALLOWED ?
147 010430 001162          BNE     M1          ;BR IF NO
148
149 010432 005737 001266  TST     ,SDEVMP       ;DEVICE MAP AVAILABLE FROM APT MAIL BOX
150 010436 001406          BEQ     1$           ;BR IF NONE
151 010440 013737 001266 001402  MOV     ,SDEVMP,DEVMP  ;LOAD IT FROM MAIL BOX
152 010446 042737 177400 001402  BIC     ,#177400,DEVMP ;ALLOW MAX 8 DRIVES
153 010454 005737 001402  1$:    TST     ,DEVMP      ;CHECK DEVICE MAP
154 010460 001002          BNE     M5FX        ;BR IF DRIVES ASSIGNED
155 010462 000137 016342  JMP     ,SEOP        ;OTHERWISE,EXIT
156
157 010466 012737 000000 001334  M5FX:  MOV     ,#0,BEGTRK    ;ALWAYS STARTS FROM 0 TRACK
158 010474 012737 000000 001330  MOV     ,#0,BEGCYL    ;ALWAYS STARTS FROM 0 CYLINDER
    
```

```

159 010502 012737 001466 001326      MOV      #822.,ENDCYL      ;LAST CYLINDER
160 010510 005037 001400                CLR      WRAP              ;SEEK FROM LO TO HI CYLINDER ADDRESS
161 010514 005037 001406                CLR      EDIT              ;DON'T ALLOW UPDATE OF BAD SECTOR FILE
162 010520 012737 177777 001320      MOV      #-1,MODE          ;SET FORMAT AND VERIFY MODE
163 010526 005737 001270                TST      $CDW1             ;APT PARAMETER = 0
164 010532 001402                BEQ      .+6               ;YES,DEFAULT TO -1
165 010534 005037 001320                CLR      MODE              ;NO, SET TO CHECK MODE
166 010540 012737 177777 001360      MOV      #-1,SEC30         ;SET 32 SECTOR
167 010546 005737 001272                TST      $CDW2             ;APT PARAMETER =0
168 010552 001012                BNE      1$                ;NO,30 SECTOR
169 010554 012737 020100 001352      MOV      #<258.*32.>,WC     ;COUNT OF WORDS
170 010562 012737 157700 001354      MOV      #-<258.*32.>,MWC   ;WORD COUNT
171 010570 012737 000037 001362      MOV      #31.,MAXSEC       ;MAX 32 SECTOR
172 010576 000413                BR       2$
173
174 010600 012737 017074 001352 1$:    MOV      #<258.*30.>,WC     ;COUNT OF WORDS
175 010606 012737 160704 001354      MOV      #-<258.*30.>,MWC   ;WORD COUNT
176 010614 005037 001360                CLR      SEC30             ;30 SECTOR
177 010620 012737 000035 001362      MOV      #29.,MAXSEC       ;MAX 30 SECTOR
178 010626 005737 001320                TST      MODE              ;CHECK MODE ?
179 010632 001006                BNE      .+16              ;NO
180 010634 012737 133331 001340      MOV      #133331,PATA      ;INITIATE PATTERN
181 010642 012737 133331 001342      MOV      #133331,PATB      ;INITIATE PATTERN
182 010650 012737 000002 001336      MOV      #2,PATSEL         ;WORSE CASE
183 010656 005001                CLR      R1                ;START WITH DRIVE NUMBER 0
184 010660 136137 030022 001402 3$:    BITB    ATABIT(R1),DEVMP   ;IS THIS DRIVE IN DEVICE MAP ?
185 010666 001005                BNE      4$                ;BR IF YES
186 010670 005201                INC      R1                ;INCREMENT DRIVE #
187 010672 020127 000007                CMP      R1,#7             ;ALL DONE WITH MAP ?
188 010676 003770                BLE      5$                ;BR IF NO
189 010700 000434                BR       7$                ;DONE !!
190
191 010702 010137 001222 001402 4$:    MOV      R1,DRIVE          ;LOAD R1 INTO DRIVE
192 010706 146137 030022                BICB    ATABIT(R1),DEVMP   ;CLEAR THE BIT FROM DEVICE MAP
193 010714 105761 027706                TSTB    DRVSTA(R1)         ;IS DRIVE ON LINE ?
194 010720 003757                BLE      3$                ;BR IF NO
195 010722 005737 001412                TST      XXDP              ;LOADED FROM THIS DEVICE ?
196 010726 001403                BEQ      5$                ;BR IF NO
197 010730 123701 001412                CMPB    XXDP,R1            ;LOADED FROM THIS DRIVE ?
198 010734 001751                BEQ      3$                ;BR IF YES
199
200 010736 012702 000004 027716 5$:    MOV      #4,R2             ;SETUP TRACK PARAMETER FOR AN RM03/2
201 010742 122761 000007                CMPB    #7,DRVSTYP(R1)     ;IS DEVICE AN RM05 ?
202 010750 001002                BNE      6$                ;BR IF NO
203 010752 012702 000022                MOV      #18.,R2           ;SETUP TRACK PARAMETER FOR AN RM05
204 010756 010237 001324 6$:    MOV      R2,LSTRK          ;SETUP LAST TRACK AND
205 010762 010237 001332                MOV      R2,ENDTRK         ;END TRACK FOR AUTO MODE.
206 010766 000137 012146                JMP      M5                 ;START TO FORMAT OR CHECK
207
208 010772 000137 016342 7$:    JMP      $EOP              ;END OF PASS FOR ALL DRIVES ON $DEV
209
210
211 ;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
212 ;MODES ARE: 'FORMAT & VERIFY', 'CHECK FORMAT' OR 'VERIFY HEADER FORMAT'
213
214 010776 005037 177776 M1:    CLR      PS
215 011002 005037 001320                CLR      MODE              ;SET MODE TO 'CHECK FORMAT'

```

```

216 011006 104401 036636          TYPE      ,MMODE          ;TYPE 'PROGRAM MODE'
217 011012 104410          RDCHR          ;READ THE KEYBOARD
218 011014 012637 001174          MOV      (SP)+, $TMP0     ;GET CHARACTER INPUT
219 011020 023727 001174 000015    CMP      $TMP0, #CR       ;'CR' ENTERED (DEFAULT)
220 011026 001432          BEQ      3$              ;BR IF YES
221 011030 104401 001174          TYPE      , $TMP0        ;TYPE CHARACTER
222 011034 022737 000103 001174    CMP      #'C', $TMP0      ;'CHECK' ?
223 011042 001413          BEQ      1$              ;BR IF YES
224 011044 022737 000106 001174    CMP      #'F', $TMP0      ;'FORMAT' ?
225 011052 001422          BEQ      4$              ;BR IF YES
226 011054 022737 000126 001174    CMP      #'V', $TMP0      ;'VERIFY' ONLY ?
227 011062 001406          BEQ      2$              ;BR IF YES
228 011064 104401 001204          TYPE      , $QUES        ;NO CORRECT ENTRY
229 011070 000742          BR       M1              ;TRY AGAIN
230 011072 104401 036713          1$:      TYPE      , MCHECK ;TYPE REST OF 'CHECK'
231 011076 000415          BR       5$              ;GET STARTING ADDRESS
232
233 011100 104401 036767          2$:      TYPE      , MVERIFY ;TYPE 'VERIFY'
234 011104 012737 000001 001320    MOV      #1, MODE         ;SET VERIFY ONLY MODE
235 011112 000407          BR       5$
236
237 011114 104401 036413          3$:      TYPE      , F       ;TYPE 'F'
238 011120 104401 036672          4$:      TYPE      , MFORMAT ;TYPE 'FORMAT & VERIFY'
239 011124 012737 177777 001320    MOV      #-1, MODE        ;SET MODE TO FORMAT & VERIFY
240 011132          5$:
241
242          ;FIND OUT IF FORMAT IS TO BE IN 30 OR 32 SECTOR MODE
243
244 011132 104401 001205          M1A:     TYPE      , $CRLF   ;CR-LF
245 011136 104401 037004          1$:      TYPE      , $MSIZE  ;TYPE FORMAT MODE REQUEST
246 011142 104410          RDCHR          ;READ KEYBOARD
247 011144 012637 001174          MOV      (SP)+, $TMP0     ;GET CHARACTER ENTRY
248 011150 023727 001174 000015    CMP      $TMP0, #CR       ;CARRIAGE RETURN ?
249 011156 001415          BEQ      2$              ;BR IF YES
250 011160 104401 001174          TYPE      , $TMP0        ;TYPE CHARACTER
251 011164 022737 000131 001174    CMP      #'Y', $TMP0      ;IS ENTRY A 'Y' ?
252 011172 001411          BEQ      3$              ;BR IF IT IS
253 011174 022737 000116 001174    CMP      #'N', $TMP0      ;IS ENTRY A 'N' ?
254 011202 001424          BEQ      4$              ;BR IF IT IS
255 011204 104401 001204          TYPE      , $QUES        ;TYPE '?'
256 011210 000752          BR       1$              ;TRY AGAIN
257
258 011212 104401 036417          2$:      TYPE      , Y       ;TYPE 'Y'
259 011216 104401 037132          3$:      TYPE      , MSEC32  ;TYPEOUT MODE SELECTED
260 011222 012737 177777 001360    MOV      #-1, SEC30       ;32 SECTOR INDICATOR
261 011230 012737 020100 001352    MOV      #<258.*32.>, WC   ;TRACK SIZE IN 32 SECTOR MODE
262 011236 012737 157700 001354    MOV      #-<258.*32.>, MWC  ;2'S COMPLEMENT WORD COUNT
263 011244 012737 000037 001362    MOV      #31., MAXSEC     ;MAX SECTOR ADDRESS IN 32 SECTOR MODE
264 011252 000415          BR       5$              ;CONTINUE
265
266 011254 104401 037053          4$:      TYPE      , MSEC30  ;TYPE OUT 30 SECTOR MODE SELECTED
267 011260 005037 001360          CLR      SEC30           ;30 SECTOR INDICATOR
268 011264 012737 017074 001352    MOV      #<258.*30.>, WC   ;TRACK SIZE IN 30 SECTOR MODE
269 011272 012737 160704 001354    MOV      #-<258.*30.>, MWC  ;2'S COMPLEMENT WORD COUNT
270 011300 012737 000035 001362    MOV      #29., MAXSEC     ;MAX SECTOR ADDRESS IN 30 SECTOR MODE
271 011306
272

```



```

273 011306 012737 000000 001334 M1B:  MOV    #0,BEGTRK    ;CLEAR STARTING TRACK ADDRESS
274 011314 012737 000000 001330      MOV    #0,BEGCYL    ;CLEAR BEGINNING CYLINDER ADDRESS
275 011322 005037 001400      CLR    WRAP         ;SEEK FROM LO TO HI CYLINDER ADDRESS
276 011326 012737 000002 001336      MOV    #2,PATSEL    ;SETUP FOR WORST CASE PATTERN
277 011334 112737 177777 005540      MOVB   #-1,FMTDPB   ;SETUP INVALID DRIVE NUMBER
278
279      ;GO FIND OUT WHAT DRIVE
280
281 011342 012737 020262 001374 MO:   MOV    #C.ENTR,CNTLC ;CONTROL C ABORT ENTRANCE
282 011350 004737 025576      JSR    PC,$TKINT    ;INITIALIZE THE TTY KEYBOARD
283 011354 005037 001126      CLR    $ERTTL       ;CLEAR THE ERROR ACCUMULATOR
284 011360 104401 036372      TYPE   ,MUNIT       ;ASK FOR DRIVE NUMBER
285 011364 104410      RDCHR              ;READ THE KEYBOARD
286 011366 012637 001174      MOV    (SP)+,$TMP0  ;GET CHARACTER TYPED
287 011372 023727 001174 000015      CMP    $TMP0,#CR    ;CARRIAGE RETURN ?
288 011400 001420      BEQ    1$           ;BR IF YES
289 011402 023727 001174 000056      CMP    $TMP0,#'.    ;PERIOD ?
290 011410 001414      BEQ    1$           ;BR IF YES
291 011412 104401 001174      TYPE   , $TMP0      ;TYPE CHARACTER
292 011416 012701 001174      MOV    # $TMP0,R1   ;GET ADDRESS OF ENTRY
293 011422 004537 023410      JSR    R5,CK.CHR    ;CHECK ONE CHARACTER
      011426 011452      2$                ;ILLEGAL CHARACTER
      011430 011442      1$                ;CARRIAGE RETURN
      011432 011452      2$                ;'.'
      011434 011442      1$                ;'.'
      011436 011460      3$                ;DIGIT 0-7
      011440 011452      2$                ;DIGIT 8-9
294 011442 005002      1$:  CLR    R2           ;SELECT DRIVE ZERO
295 011444 104401 036403      TYPE   ,ZERO        ;TYPE UNIT NUMBER 0
296 011450 000403      BR     3$           ;GO SEE IF DRIVE ZERO IS THERE
297
298 011452 104401 001204      2$:  TYPE   , $QUES    ;TYPE '?'
299 011456 000731      BR     M0           ;ASK FOR DRIVE NUMBER AGAIN
300 011460 010237 001222      3$:  MOV    R2,DRIVE    ;SAVE DRIVE NUMBER
301
302 011464 005737 001412      TST    XXDP         ;LOADED FROM THIS DEVICE ?
303 011470 001406      BEQ    4$           ;BR IF NO
304 011472 123702 001412      CMPB   XXDP,R2     ;WAS THIS THE DRIVE ?
305 011476 001003      BNE    4$           ;BR IF NO
306 011500 104401 036574      TYPE   ,MLODEV     ;DRIVE IS LOAD DEVICE
307 011504 000716      BR     M0
308
309 011506 004737 020016      4$:  JSR    PC,$T.CLK   ;START THE CLOCK
310 011512 004737 030052      JSR    PC,$R.MINIT ;GO SEE WHAT DRIVES ARE AVAILABLE
311 011516 012737 177777 027774      MOV    #-1,SAVEFG  ;SAVE THE REGISTERS
312 011524 012737 177777 027776      MOV    #-1,SEEKFG  ;SET 'NO OPTIMIZATION' FLAG
313 011532 005037 177776      CLR    PS           ;PRIORITY = 0
314 011536 105762 027706      TSTB   DRVSTA(R2)  ;LOOK AT DRIVE STATUS
315 011542 003015      BGT    7$           ;BR IF ONLINE
316 011544 001403      BEQ    5$           ;BR IF DRIVE NOT AVAILABLE
317 011546 104401 036554      TYPE   ,MUSDR      ;'DRIVE UNSAFE'
318 011552 000673      BR     M0           ;GO GET DRIVE NUMBER AGAIN
319
320 011554 105762 027716      5$:  TSTB   DRVTP(R2)   ;A DRIVE PRESENT?
321 011560 001003      BNE    6$           ;BR IF SO
322 011562 104401 036500      TYPE   ,MDRNP      ;TYPE 'DRIVE NOT PRESENT'
323 011566 000665      BR     M0           ;GO GET DRIVE NUMBER AGAIN
    
```

```

324
325 011570 104401 036525      6$:  TYPE      ,MER11      ;'DRIVE NOT AVAILAB_E'
326 011574 000662              BR          MO          ;GO GET DRIVE NUMBER AGAIN
327
328 011576 113737 001222 005540 7$:  MOVB      DRIVE,FMTDPB ;SETUP DRIVE ADDRESS
329 011604 012737 000000 001330      MOV      #0,BEGCYL      ;CLEAR STARTING CYLINDER ADDRESS
330 011612 012737 001466 001326      MOV      #822.,ENDCYL   ;SETUP END CYLINDER
331 011620 012737 000000 001334      MOV      #0,BEGTRK      ;CLEAR STARTING TRACK ADDRESS
332 011626 012700 000004              MOV      #4,RO          ;GET LAST TRACK FOR AN RM03/2
333 011632 122762 000007 027716      CMPB     #7,DRVTYP(R2)  ;IS DEVICE AN RM05 ?
334 011640 001002              BNE      8$            ;BR IF NO
335 011642 012700 000022              MOV      #18.,RO        ;GET LAST TRACK FOR AN RM05
336 011646 010037 001324      8$:  MOV      RO,LSTRK      ;SETUP LAST TRACK AND
337 011652 010037 001332      MOV      RO,ENDTRK     ;END TRACK PARAMETERS
338 011656 005200              INC      RO            ;ADJUST TRACK LIMIT
339 011660 012701 000010      MOV      #10,R1        ;GET TRACK OFFSET IN PARAMETER TABLES
342 011664 010061 005650      MOV      RO,TABLE(R1)  ;SETUP TRACK PARAMETER LIMITS IN 'TABLE'
      011670 010061 005702      MOV      RO,TABLE2(R1) ;SETUP TRACK PARAMETER LIMITS IN 'TABLE2'
      011674 010061 005726      MOV      RO,TABLE3(R1) ;SETUP TRACK PARAMETER LIMITS IN 'TABLE3'
      011700 010061 005752      MOV      RO,TABLE4(R1) ;SETUP TRACK PARAMETER LIMITS IN 'TABLE4'
343 011704 062701 000014      ADD     #14,R1         ;GET OFFSET FOR ENDTRK LIMIT IN 'TABLE'
344 011710 010061 005650      MOV      RO,TABLE(R1)  ;SETUP TRACK PARAMETER LIMIT
345 011714 104401 001205      TYPE     ,$CRLF        ;CR-LF
346
347      ;GET ADDRESS LIMITS
348
349 011720 104401 036430      M2:  TYPE     ,ENTADR    ;'ENTER ADDRESS LIMITS'
350 011724 004437 020454      JSR     R4,PARENT      ;ENTER THE ADDRESS LIMITS
351 011730 005650              TABLE              ;TABLE PARAMETERS
352 011732 005037 001400      CLR     WRAP          ;SEEK FROM LO TO HI CYLINDER ADDRESS
353 011736 023737 001326 001330      CMP     ENDCYL,BEGCYL ;SEE IF ENDING CYL EQ TO GT THAN BEGINNING
354 011744 001402              BEQ     1$            ;BR IF ON THE SAME CYLINDER
355 011746 103410              BLO     3$            ;BR IF LESS
356 011750 000413              BR      M4            ;TO CHECK DRIVE
357
358 011752 023737 001332 001334 1$:  CMP     ENDTRK,BEGTRK ;SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
359 011760 103007              BHIS   M4            ;BR IF YES
360 011762 104401 037230      2$:  TYPE     ,MADRER    ;INVALID ADDRESS ENTERED
361 011766 000754              BR      M2            ;TRY AGAIN
362
363 011770 012737 177777 001400 3$:  MOV     #-1,WRAP      ;ALLOW SEEK FROM HI TO LO CYLINDER ADDRESS
364 011776 000400              BR      M4
365
366      ;GO GET DATA PATTERN FOR FORMAT
367
368 012000 005737 001320      M4:  TST     MODE        ;IS IT CHECK MODE ?
369 012004 003060              BGT     M5            ;NO, VERIFY ONLY MODE
370 012006 001007              BNE     M4A          ;NO, FORMAT & VERIFY MODE
371 012010 012737 133331 001340      MOV     #133331,PATA  ;CHECK MODE USE WORST CASE DATA
372 012016 012737 133331 001342      MOV     #133331,PATB  ;INITIAL THE DATA PATTERN -SHIFT 3 BITS
373 012024 000450              BR      M5            ;START TO FORMAT AND CHECK
374
375 012026 104401 037331      M4A:  TYPE     ,MSELP    ;GO TYPE 'SELECT PATTERN'
376 012032 104410              RDCHR              ;READ THE KEYBOARD
377 012034 012637 001174      MOV     (SP)+,$TMPO   ;CHARCTER ENTRY
378 012040 023727 001174 000015      CMP     $TMPO,#CR    ;CARRIAGE RETURN ?
379 012046 001420              BEQ     1$            ;BR IF YES

```



```

1          .SBTTL  START OF FORMAT
2
3          ;GO TYPE 'STARTING FORMAT ON DRIVE N'
4 012146   MS:
5 012146   012737 020302 000060   MOV    #0,ENTR,@#TKVEC ;VECTOR FOR CONTROL-0
6 012154   005037 177776         CLR    PS                ;PRIORITY = 0
7
8 012160   005037 001406         CLR    EDIT              ;DON'T ALLOW UPDATE OF BAD SECTOR FILE
9 012164   005737 001320         TST    MODE              ;'FORMAT', 'CHECK' OR VERIFY MODE ?
10 012170  003017
11 012172  001403         BGT    3$                ;BR IF 'VERIFY ONLY' MODE
12 012174  104401 037452         BEQ    1$                ;BR IF 'CHECK' MODE
13 012200  000402
14
15 012202  104401 037505         TYPE   ,MSFOR           ;TYPE 'STARTING FORMAT ON DRIVE N'
16 012206  000402
17 012206  013746 001222         BR     2$
18 012212  104403
19 012214    001
20 012215    000
21 012216  104401 001205         MOV    DRIVE,-(SP)      ;;SAVE DRIVE FOR TYPEOUT
22 012222  012737 177777 001406   TYPOS
23 012230
24 012230  005037 001344         .BYTE 1                ;;TYPE DRIVE NUMBER
25 012234  012737 177777 001404   .BYTE 0                ;;GO TYPE--OCTAL ASCII
26 012242  013737 001222 005540   .BYTE 0                ;;TYPE 1 DIGIT(S)
27 012250  012737 001466 005552   TYPE   ,SCRLF          ;;SUPPRESS LEADING ZEROS
28 012256  113737 001324 005551   MOV    #-1,EDIT        ;CR-LF
29 012264  105037 005550         ;UPDATE BAD SECTOR FILE AT END OF PASS
30 012270  012737 045774 005546   3$:
31 012276  012737 157700 005544
32 012304  112737 000020 005541
33 012312  112737 000143 005542
34 012320  004037 030622
35 012324  005540
36 012326  000774
37 012330  005737 005556
38 012334  001775
39
40 012336  012737 012336 001124   ;THIS CODE RETRIEVES THE BAD SPOT FILE FROM THE PACK (IF FORMATTED PACK)
41 012344  012706 001100         ;SET UP DPB BLOCK READ CYL 822., LAST TRACK.
42 012350  112737 000173 005542   RETBAD: CLR    RETRY      ;CLEAR THE RETRY FLAG
43 012356  004037 030622         MOV    #-1,PACK        ;RESET THE MFG AVAILABLE FLAG
44 012362  005540
45 012364  000774
46 012366  005737 005556         MOV    DRIVE,FMTDPB    ;LOAD DRIVE #
47 012372  001775
48 012374  100007
49
50
51
52

```

```

53      :      ERROR 11
54      :      ERROR 12
55      :      ERROR 13
56      :      ERROR 14
57      :      ERROR 21
58      :5$: JSR    PC,LOP.CK      ;CHECK LOOP ON ERROR
59 012376 022737 000003 001344  CMP    #3,RETRY      ;DONT' TRY OVER 3 TIMES
60 012404 103451      :
61 012406 005237 001344  INC    RETRY
62 012412 000761      BR     3$           ;TRY AGAIN
63
64 012414 005037 001176      6$: CLR    $ESCAPE      ;
65 012420 005037 001344      CLR    RETRY
66 012424 005737 005556      TST   FMTDPB+16     ;ANY ERROR ?
67 012430 100437      BMI   8$           ;BR IF ANY
68 012432 022737 151466 045774  CMP    #151466,BUFP ;ON THE LAST CYLINDER ?
69 012440 001033      BNE   8$           ;BR IF NOT
70 012442 013700 001324      MOV   LSTRK,R0      ;GET LAST TRACK
71 012446 000300      SWAB  R0            ;MOVE TRACK TO HI BYTE AND SECTOR TO LO BYTE
72 012450 020037 045776      CMP    R0,BUFP+2    ;ON LAST TRACK AND SECTOR 0
73 012454 001025      BNE   8$           ;BR IF NOT
74 012456 005737 046000      TST   BUFP+4        ;SERIAL NUMBER SHOULD NOT 0
75 012462 001003      BNE   7$           ;BR IF NOT 0
76 012464 005737 046002      TST   BUFP+6        ;SECOND WORD OF SERIAL NUMBER
77 012470 001417      BEQ   8$           ;BR IF ZERO
78 012472 005737 046004      7$: TST   BUFP+10     ;THE THIRD WORD MUST BE 0
79 012476 001014      BNE   8$           ;BR IF NOT
80 012500 022737 177777 046006  CMP    #-1,BUFP+12  ;AN ALIGNMENT PACK ?
81 012506 001002      BNE   .+6          ;BR IF NOT
82 012510 000137 017112      JMP   REJEC1
83 012514 005737 046006      TST   BUFP+12      ;A DATA PACK ?
84 012520 001003      BNE   8$           ;BR IF NOT IDENTIFIED
85 012522 012737 000400 001404  MOV    #400,PACK    ;SET MFG BAD SPOT FILE AVAILABLE FLAG
86 012530 000240      8$: NOP           ;DONE
87
88      ;THIS CODE INITIALIZES THE BAD16,BAD18,USTAB,USSAV TABLES
89
90 012532 012706 001100      TABINT: MOV    #STACK,SP      ;INITIAL STACK POINT
91 012536 005046      CLR    -(SP)        ;TERMINATOR
92 012540 012746 004430      MOV    #USSAV,-(SP)  ;USSAV TABLE ADDRESS
93 012544 012746 002420      MOV    #BAD18,-(SP) ;MFG 18 BIT FILE ADDRESS
94 012550 012746 001414      MOV    #BAD16,-(SP) ;MFG 16 BIT FILE ADDRESS
95 012554 012703 003424      MOV    #USTAB,R3    ;USER BAD SPOT FILE ADDRESS
96 012560 012704 000004      1$: MOV    #4,R4      ;FIRST 4 WORDS SET TO 0 TEMPERARY
97 012564 005023      2$: CLR    (R3)+
98 012566 005304      DEC   R4
99 012570 001375      BNE   2$           ;BR IF NOT DONE
100 012572 012704 000374      MOV    #252,R4      ;SET -1 TO ALL THE REST LOCATIONS
101 012576 012705 177777      MOV    #-1,R5
102 012602 010523      3$: MOV    R5,(R3)+
103 012604 005304      DEC   R4           ;DECREMENT WORD COUNT
104 012606 001375      BNE   3$           ;BR IF NOT DONE
105 012610 012603      MOV    (SP)+,R3     ;NEXT TABLE
106 012612 001362      BNE   1$           ;LOOPING BACK IF NOT TERMINATOR
107
108
109      ;THIS CODE LOADS TABLE BAD16,BAD18,USSAV, FROM PACK (IF FORMATTED PACK)

```

```

110
111 012614 005737 001404      TABLD:  TST      PACK      ;BAD SPOT FILE AVAILABLE FROM PACK ?
112 012620 003451              BLE      6$          ;BR IF NOT AVAILIABLE
113 012622 012702 046000      MOV      #BUFP+4,R2    ;LOAD BAD16 TABLE FIRST
114 012626 012703 001414      MOV      #BAD16,R3     ;TABLE ADDRESS
115 012632 012704 000400      MOV      #256.,R4     ;WORD COUNT
116 012636 012223              1$:      MOV      (R2)+,(R3)+ ;LOAD THE BAD16 TABLE
117 012640 005304              DEC      R4           ;BR IF NOT DONE
118 012642 001375              BNE      1$
119 012644 012702 047004      MOV      #BUFP+520.,R2 ;258 X 2 + 2 X 2 + BUFP
120 012650 012703 002420      MOV      #BAD18,R3     ;TABLE ADDRESS
121 012654 012704 000400      MOV      #256.,R4     ;WORD COUNT
122 012660 012223              2$:      MOV      (R2)+,(R3)+ ;LOAD THE BAD18 TABLE
123 012662 005304              DEC      R4           ;DECREMENT WORD COUNT
124 012664 001375              BNE      2$          ;BR IF NOT DONE
125 012666 012702 061054      MOV      #BUFP+4+<11.*516.>,R2 ;SECTOR 11.
126 012672 005737 001360      TST      SEC30        ;FORMAT IN 16 BIT MODE
127 012676 100402              BMI      3$          ;BR IF IT IS
128 012700 012702 060050      MOV      #BUFP+4+<10.*516.>,R2 ;SECTOR 10.
129 012704 012703 004430      3$:      MOV      #USSAV,R3     ;LOAD THE USSAV TABLE
130 012710 012704 000400      MOV      #256.,R4     ;WORD COUNT
131 012714 012223              4$:      MOV      (R2)+,(R3)+ ;
132 012716 005304              DEC      R4           ;
133 012720 001375              BNE      4$
134 012722 062702 000004      ADD      #4,R2        ;ACCESSES THE USTAB BUFFER ADD
135 012726 012703 003424      MOV      #USTAB,R3     ;LOAD THE USTAB TABLE
136 012732 012704 000400      MOV      #256.,R4     ;WORD COUNT
137 012736 012223              5$:      MOV      (R2)+,(R3)+ ;LOAD THE TABLE
138 012740 005304              DEC      R4           ;DECREMENT THE COUNT
139 012742 001375              BNE      5$          ;BR IF NOT ALL DONE
140 012744 000240              6$:      NOP                ;EXIT
141
142      ;THIS CODE ASK O.P TO ENTER A SERIAL NUMBER FOR UNFORMATTED PACK
143
144 012746 005737 001320      SERNL:  TST      MODE      ;VERIFY ONLY MODE ?
145 012752 003162              BGT      6$          ;BR IF YES
146 012754 005737 001404      TST      PACK      ;BAD SPOT FILE AVAILABLE ?
147 012760 003124              BGT      4$          ;BR IF AVAILABLE
148 012762 105737 001150      TSTB     $AUTOB      ;RUNNING UNDER AUTO MODE ?
149 012766 001144              BNE      5$          ;BR IF YES
150
151 012770 004737 025576      JSR      PC,$KINT     ;INITIALTHE KEYBOARD
152 012774 104401 001205      TYPE     ,$CRLF      ;TYPE CRLF
153 013000 104401 040022      TYPE     ,MSG1       ;'ENTER TWO WORDS FOR SN''
154 013004 104401 040101      1$:      TYPE     ,SN1
155 013010 012702 000012      MOV      #10.,R2     ;MAXMUM 10 OCTAL DIGITS ALLOWED
156 013014 005037 001414      CLR      BAD16       ;CLEAR THE LSW OF THE SERIAL NUMBER
157 013020 005037 001416      CLR      BAD16+2     ;CLEAR THE MSW OF THE SERIAL NUMBER
158 013024 104411              RDLIN
159 013026 012601              MOV      (SP)+,R1    ;ADDRESS OF THE READ IN STRING
160 013030 105711              2$:      TSTB     (R1)        ;TERMINATOR OF THE INPUT STRING
161 013032 001471              BEQ      3$          ;BR IF IT IS
162 013034 121127 000060      CMPB     (R1),#'0    ;LESS THAN ASCIZ 0 ?
163 013040 103761              BLO      1$          ;ENTER AGAIN IF IT IS
164 013042 121127 000067      CMPB     (R1),#'7    ;HIGH THAN ASCIX 7 ?
165 013046 101356              BHI      1$          ;ENTER AGAIN IF SO
166 013050 013746 001414      MOV      BAD16,-(SP) ;SAVE THE LSW

```

```

167 013054 042737 100000 001414      BIC      #BIT15,BAD16      ;CLEAR THE SIGN BIT OF LSW
168 013062 006137 001414      ROL      BAD16           ;ROTATE THE LSW FIVE TIMES
169 013066 006137 001414      ROL      BAD16           ;TO MOVE THE MOST SIGN DIGIT
170 013072 006137 001414      ROL      BAD16           ;TO BIT 0 TO BIT 2
171 013076 006137 001414      ROL      BAD16           ;
172 013102 006137 001414      ROL      BAD16           ;
173 013106 042737 177770 001414      BIC      #177770,BAD16   ;LEFT ON THE MS DIGIT OF THE LSW
174 013114 006337 001416      ASL      BAD16+2        ;SHIFT THE MSW LEFT ONE OCTAL DIGIT
175 013120 100731 001416      BMI      1$             ;ENTER AGAIN IF SIGN BIT SET
176 013122 006337 001416      ASL      BAD16+2        ;
177 013126 100726 001416      BMI      1$             ;
178 013130 006337 001416      ASL      BAD16+2        ;
179 013134 100723 001416      BMI      1$             ;
180 013136 063737 001414 001416      ADD      BAD16,BAD16+2   ;APPENDING THE DIGITS INTO MSW
181 013144 012637 001414      MOV      (SP)+,BAD16    ;RESTORE THE LSW
182 013150 006337 001414      ASL      BAD16           ;SHIFT THE LSW LEFT ONE OCTAL DIGIT
183 013154 006337 001414      ASL      BAD16           ;
184 013160 006337 001414      ASL      BAD16           ;
185 013164 042737 100000 001414      BIC      #BIT15,BAD16   ;CLEAR THE SIGN BIT
186 013172 111103 001414      MOV      (R1),R3        ;LOAD THE CURRENT READ IN DIGITS
187 013174 042703 177770 001414      BIC      #177770,R3     ;LEFT ONLY ONE OCTAL DIGIT
188 013200 060337 001414      ADD      R3,BAD16       ;APPEND THE READ IN CHARACTER TO LSW
189 013204 005201 001414      INC      R1             ;ADJUST THE READ IN STRING ADDRESS
190 013206 005302 001414      DEC      R2             ;DECREMENT ONE DIGIT COUNT
191 013210 001307 001414      BNE      2$             ;BR IF NOT DONE
192 013212 105711 001414      TST      (R1)           ;IF 10 DIGITS HAVE BEEN ENTERED
193 013214 001273 001414      BNE      1$             ;MUST BE FOLLOWED BY <CR>
194 013216 005737 001414 3$:      TST      BAD16          ;SERIAL NUMBER MUST BE NOT 0
195 013222 001003 001414      BNE      4$             ;BR IF NOT 0
196 013224 005737 001416      TST      BAD16+2        ;
197 013230 001665 001416      BEQ      1$             ;ENTER AGAIN IF ZERO
198 013232 013737 001414 002420 4$:      MOV      BAD16,BAD18    ;LOAD ALL TABLE WITH THE SERIAL NUMBER,
199 013240 013737 001414 003424      MCV      BAD16,USTAB
200 013246 013737 001414 004430      MOV      BAD16,USSAV
201 013254 013737 001416 003426      MOV      BAD16+2,USTAB+2
202 013262 013737 001416 004432      MOV      BAD16+2,USSAV+2
203 013270 013737 001416 002422      MOV      BAD16+2,BAD18+2
204 013276 000410 001416      BR       6$             ;EXIT
205
206 013300 005237 004430 5$:      INC      USSAV          ;CODE FOR AUTO MODE APT,ACT,ETC.
207 013304 005237 001414      INC      BAD16
208 013310 005237 002420      INC      BAD18
209 013314 005237 003424      INC      USTAB
210
211 013320 000240 001416 6$:      NOP                    ;DONE
212
213      ;THIS CODING TO PROVIDE UTILITY ROUTINE ENTRIES
214
215 013322 000240 CALIN:  NOP
216 013324 004737 025576      JSR      PC,$TKINT      ;INITIALIZE THE KEYBOARD
217 013330 105737 001150      TST      $AUTOB         ;AUTO BYTE SET ?
218 013334 001052      BNE      4$             ;IF SO, BRANCH
219 013336 005737 001270      TST      $CDW1          ;ALLOW ANY UTILITY ROUTINE ?
220 013342 001447      BEQ      4$             ;BR IF NOT
221
222 013344 104401 040415 1$:      TYPE     ,MSGBLK        ;ENTER MESSAGE BLOCK
223 013350 104401 041204 2$:      TYPE     ,MUTLTY       ;TYPE UTILITY MESSAGE

```



```
1
2 013462
3
4
5
6
7 013462 000240
8 013464 012737 020302 00006C
9 013472 005037 177776
10
11 013476 005737 001320
12 013502 001432
13 013504 002402
14 013506 000137 015550
15
16 013512 005037 001340
17 013516 005037 001342
18 013522 005737 001336
19 013526 001420
20 013530 012737 177777 001340
21 013536 012737 177777 001342
22 013544 022737 000001 001336
23 013552 001406
24 013554 012737 066666 001340
25 013562 012737 066666 001342
26 013570 013703 001352
27 013574 012700 045774
28 013600 013701 001340
29 013604 013702 001342
30 013610 010120
31 013612 010220
32 013614 005303
33 013616 005303
34 013620 001373
35
36
37
38 013622 004737 017352
39 013626 004737 017556
40
41 013632 112737 000020 005541
42 013640 005737 001360
43 013644 001002
44 013646 105037 005541
45 013652 112737 000143 005542
46 013660 004037 030622
47 013664 005540
48 013666 000774
49 013670 005737 005556
50 013674 001775
51
52
53
54 013676 023727 005552 001466
55 013704 103406
56 013706 123737 005551 001324
57 013714 103402
```

ST1:
:*****
:THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH
:SECTOR IMAGE

SELPAT: NOP
MOV #0,ENTR,@#TKVEC ;VECTOR FOR CONTROL-0
CLR PS ;PRIORITY = 0

1\$: TST MODE ;CHECK MODE ?
BEQ 3\$;YES
BLT 2\$;NO, FORMAT & VERIFY
JMP HDREAD ;NO, VERIFY ONLY

2\$: CLR PATA ;CLEAR DATA PATTERN A
CLR PATB ;CLEAR DATA PATTERN B
TST PATSEL ;SEE IF PATTERN OF ONES
BEQ 3\$;BR IF SO
MOV #177777,PATA ;PATA = AB
MOV #177777,PATB ;PATB=CD
CMP #1,PATSEL ;SEE IF PATTERN OF ZEROS
BEQ 3\$;BR IF SO
MOV #066666,PATA ;SET UP WORST CASE
MOV #066666,PATB ;SET UP WORST CASE

3\$: MOV WC,R3 ;SET UP COUNTER
MOV #BUFP,R0 ;SET UP MEMORY POINTER

4\$: MOV PATA,R1 ;SET UP PATTERN IN R1
MOV PATB,R2 ;SET UP PATTERN IN R2

5\$: 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 5\$;DO IT AGAIN IF R3 NOT 0

:THIS CODE SETS UP FOR THE ACTUAL FORMAT

WRTRK: JSR PC,SETTBL ;GO SET UP DRIVER TABLE
JSR PC,SETHDR ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE

MOV #20,FMTDPB+1 ;LOAD FMT16 BIT
TST SEC30 ;18 BIT MODE ?
BNE 1\$;BR IF NOT
CLRB FMTDPB+1 ;CLEAR THE FMT16 BIT

1\$: MOV #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
2\$: JSR R0,RM05 ;START THE COMMAND
FMTDPB ;DPB ADDRESS
BR 2\$;QUEUE FULL RETURN

3\$: TST FMTDPB+16 ;LOOK FOR DONE
BEQ 3\$;LOOP UNTIL FINISHED

:SET UP SECCU TABLE: BAD SPOT ON CURRENT TRACK ,RETRIEVED FROM PACK

WRTRKX: CMP FMTDPB+12,#822. ;LAST CYLINDER ?
BLO 1\$;BR IF NO
CMPB FMTDPB+11,LSTRK ;LAST TRACK OF LAST CYLINDER ?
BLO 1\$;BR IF NO

```

58 013716 000137 015536          JMP      WRTRKZ          ;DON'T DESTROY LAST TRACK ON CYL 822.
59
60 013722 012703 005434          1$:     MOV      #SECCU,R3      ;INITILIZE THE SECCU TABLE
61 013726 012704 000040          MOV      #32.,R4          ;TOTAL 32 WORD MAX
62 013732 012702 177777          MOV      #-1,R2          ;LOAD ALL LOCATION TO -1
63 013736 010223                    2$:     MOV      R2,(R3)+      ;
64 013740 005304                    DEC      R4                ;
65 013742 001375                    BNE     2$                ;BR IF NOT DONE
66
67                                ;LOAD THE BAD SPOT INTO SECCU TABLE IF ANY
68
69 013744 005737 001404          WRTRKY: TST     PACK          ;BAD SPOT FILE AVAILABLE AT ALL
70 013750 003461                    BLE     6$                ;BR IF NOT
71 013752 005046                    CLR     -(SP)             ;CLEAR UP THE DUMMY RETURN ADDRESS
72 013754 012746 003434          MOV      #USTAB+8.,-(SP)   ;USER BAD SECTOR TABLE
73 013760 012702 001414          MOV      #BAD16,R2        ;R2 POINTER OF MFG BAD SPOT TABLE
74 013764 005737 001360          TST     SEC30             ;IF 18 BIT MODE CHANGE POINTER
75 013770 100402                    BMI     1$                ;BR IF 16 BIT MODE
76 013772 012702 002420          MOV      #BAD18,R2        ;
77 013776 010201                    1$:     MOV      R2,R1          ;R1: LAST ADDRESS OF BAD SPOT TABLE
78 014000 062701 000776          ADD     #510.,R1          ;
79 014004 012703 005434          MOV      #SECCU,R3        ;TABLE ADDRESS
80 014010 012704 000076          MOV      #62.,R4          ;LAST ADDRESS OF THE TABLE
81 014014 060304                    ADD     R3,R4             ;
82 014016 062702 000010          ADD     #10,R2            ;BAD SPOT STARTS FROM THE 5TH WORD OF THE FILE
83 014022 023712 005552          2$:     CMP      FMTDPB+12,(R2) ;BAD SPOT ON THIS CYLINDER ?
84 014026 001007                    BNE     3$                ;BR IF NOT
85 014030 123762 005551 000003    CMPB    FMTDPB+11,3(R2)   ;ALSO, ON THE SAME TRK ?
86 014036 001003                    BNE     3$                ;BR IF NOT
87 014040 116205 000002          MOVB    2(R2),R5          ;BAD SECTOR
88 014044 010523                    MOV     R5,(R3)+         ;WORD STORAGE
89 014046 062702 000004          3$:     ADD     #4,R2          ;ADVANCE TO NEXT SET
90 014052 020102                    CMP     R1,R2            ;ALL SPOT ARE CHECKED ?
91 014054 103412                    BLO     5$                ;BR IF IT IS
92 014056 020403                    CMP     R4,R3            ;END OF SECCU TABLE ?
93 014060 103401                    BLO     4$                ;REJECT THE PACK
94 014062 000757                    BR      2$                ;LOOPING BACK
95
96 014064 004737 020160          4$:     JSR     PC,RESTR1     ;IMPROPER MFG INFORMATION
97 014070 012737 177777 001404    MOV     #-1,PACK         ;RESET MFG FILE AVAILABLE FLAG
98 014076 000137 012532          JMP     TABINT           ;AND START AGAIN
99
100 014102 011601                    5$:     MOV     (SP),R1        ;NEXT TABLE
101 014104 062701 000766          ADD     #502.,R1         ;ENDING OF THE TABLE
102 014110 012602                    MOV     (SP)+,R2         ;BEGINNING OF THE TABLE
103 014112 001343                    BNE     2$                ;BR IF NOT DUMMY ADD
104 014114 000240                    6$:     NOP                    ;DONE
105
106                                ;THIS CODE SORTS THE BAD SECTOR TABLE "SEC.11" IN ASCENDING ORDER
107                                ;AND DELETES THE DUPLICATED ONES
108
109 014116 012702 005434          SORT1:  MOV     #SECCU,R2      ;TOP OF THE TABLE
110 014122 012703 005532          MOV     #SECCU+62.,R3     ;BOTTOM OF THE TABLE
111 014126 012704 000037          MOV     #31.,R4          ;TOTAL 32 ELEMENT,SORT 31 TIMES
112 014132 022712 177777          CMP     #-1,(R2)         ;IS THE TABLE EMPTY ?
113 014136 001451                    BEQ     6$                ;BR IF IT IS, QUICK EXIT
114 014140 021262 000002          1$:     CMP     (R2),2(R2)     ;N TH ELEMENT : N+1 TH ELEMENT

```

```

115 014144 103430          BLO 4$          ;BR IF SMALLER
116 014146 001406          BEQ 2$          ;DELET N+1 TH ELEMENT, IF SAME
117                                ;OTHERWISE, N > N+1
118 014150 011246          MOV (R2),-(SP)  ;STORE N ELEMENT
119 014152 016712 000002  MOV 2(R2),(R2)  ;SWITCH N+1 ELEMENT TO N ELEMENT
120 014156 012662 000002  MOV (SP)+,2(R2) ;SWITCH N ELEMENT TO N+1 ELEMENT
121 014162 000421          BR 4$          ;ADJUST TOP POINTER
122 014164 022712 177777  2$:  CMP #-1,(R2)  ;TERMINATOR ?
123 014170 001416          BEQ 4$          ;BR IF IT IS
124 014172 010246          MOV R2,-(SP)  ;SAVE TOP POINTER
125 01 174 062702 000002  ADD #2,R2     ;DELET 2(R2)
126 014200 016222 000002  3$:  MOV 2(R2),(R2)+ ;
127 014204 022702 005532  CMP #SECCU+62.,R2 ;BOTTOM OF THE TABLE ?
128 014210 101373          BHI 3$          ;BR IF NOT
129 014212 012737 177777  005532  MOV #-1,SECCU+62. ;LOAD -1 TO TABLE BOTTOM
130 014220 012602          MOV (SP)+,R2  ;RESTORE THE TOP POINTER
131 014222 005304          DEC R4        ;DECREMENT ONE ELEMENT COUNT
132 014224 001416          BEQ 6$          ;BR IF ALL DONE
133 014226 062702 000002  4$:  ADD #2,R2     ;INCREMENT THE TOP POINTER
134 014232 020302          CMP R3,R2    ;REACH THE BOTTOM ?
135 014234 001341          BNE 1$          ;BR IF NOT
136 014236 005304          5$:  DEC R4        ;DECR. MENT ONE SORT COUNT
137 014240 001410          BEQ 6$          ;BR IF ALL DONE
138 014242 012702 005434  MOV #SECCU,R2 ;RESET TOP PCINTER
139 014246 162703 000002  SUB #2,R3    ;UPDATE BOTTOM POINTER
140 014252 022703 005434  CMP #SECCU,R3 ;EXIT IF TOP = BOTTOM
141 014256 001401          BEQ 6$          ;
142 014260 000727          BR 1$          ;
143 014262 000240          6$:  NOP        ;SORT THE REST ELEMENTS
144 014264 005037 005536  CLR NEXT     ;DONE
145                                ;FLAG,START OF A NEW TRACK
146 014270 005037 001316  WRTRK1: CLR SOFSW ;CLEAR ERROR COUNTER
147 014274 005037 001344  CLR RETRY   ;ZERO THE RETRY COUNTER
148 014300 105037 005550  CLR#B FMTDPB+10 ;RESTORE SECTOR
149 014304 013737 001354  005544  MOV #WC,FMTDPB+4 ;RESTORE WC
150 014312 012737 045774  005546  MOV #BUFP,FMTDPB+6 ;RESTORE BA
151
152                                ;THIS CODE TO SET UP WORD COUNT AND BUFFER ADDRESS
153
154 014320 022737 177777  005536  WRTRKA: CMP #-1,NEXT ;END OF OPERATION ?
155 014326 001003          BNE 1$          ;BR IF NOT
156 014330 104401 040373          TYPE ,MHALT
157 014334 000000          HALT        ;OTHERWISE SOMETHING WRONG *****
158
159 014336 113737 005536  005550  1$:  MOV#B NEXT,FMTDPB+10 ;STARTING SECTOR
160 014344 012701 005434  MOV #SECCU,R1  ;R1 POINTS TO THE BAD SECTOR TABLE
161 014350 022711 177777  2$:  CMP #-1,(R1)  ;END OF TABLE ?
162 014354 001513          BEQ WRTRKD    ;BR IF IT IS
163 014356 032711 100000  BIT #BIT15,(R1) ;THE BAD SECTOR HAS ALREADY ACCESSED ?
164 014362 001403          BEQ 3$          ;BR IF NOT
165 014364 062701 000002  ADD #2,R1     ;ADJUST THE POINTER
166 014370 000767          BR 2$          ;LOOPING BACK
167 014372 123711 005550  3$:  CMP#B FMTDPB+10,(R1) ;BAD SPOT = STARTING SECTOR ?
168 014376 001417          BEQ WRTRKB    ;BR IF IT IS
169 014400 103403          BLO 4$          ;BR IF START SEC < BAD SPOT
170 014402 104401 040373          TYPE ,MHALT
171 014406 000000          HALT        ;OTHERWISE . SOMETHING WRONG

```

```

172
173 014410 111102
174 014412 023702 001362
175 014416 001463
176 014420 103471
177
178 014422 005202
179 014424 010237 005536
180 014430 052721 100000
181 014434 000432
182
183 014436 105237 005550
184 014442 052721 100000
185 014446 123711 005550
186 014452 001771
187 014454 123737 005550 001362
188 014462 101402
189
190
191
192 014464 000137 015450
193
194 014470 022711 177777
195 014474 001443
196 014476 111102
197 014500 023702 001362
198
199 014504 001430
200 014506 103436
201 014510 005202
202 014512 010237 005536
203 014516 052711 100000
204
205 014522 013702 005536
206 014526 162702 000002
207 014532 006302
208 014534 016203 006232
209 014540 113702 005550
210 014544 006302
211 014546 166203 006232
212 014552 062703 000402
213 014556 005403
214 014560 010337 005544
215 014564 000432
216
217 014566 012737 177777 005536
218 014574 052711 100000
219 014600 005302
220 014602 000405
221
222 014604 012737 177777 005536
223 014612 013702 001362
224
225 014616 006302
226 014620 016203 006232
227 014624 113702 005550
228 014630 006302

```

4S: MOV (R1),R2 :UPDATE NEXT STARTING ADDRESS
 CMP MAXSEC,R2 :NEXT BAD SPOT IS THE LAST SECTOR?
 BEQ WRTRKF :BR IF LAST SECTOR
 BLO WRTRKD :BR IF LESS
 BLOS WRTRKD :BR IF IT IS
 INC R2 :NEXT = BAD + 1
 MOV R2,NEXT :NEXT STARTING ADDRESS
 BIS #BIT15,(R1)+ :MARK THE ACCESSED SECTOR
 BR WRTRKC :TO SET UP WORD COUNT

WRTRKB: INCB FMTDPB+10 :INCREMENT THE STARTING SECTOR
 BIS #BIT15,(R1)+ :MARK THE ACCESSED SECTOR
 CMPB FMTDPB+10,(R1) :STILL ON ANOTHER BAD SPOT ?
 BEQ WRTRKB :LOOP AGAIN
 CMPB FMTDPB+10,MAXSEC :IF START SECTOR > MAXSEC
 BLOS 1\$:IMPROPER MFG INFORMATION
 JSR PC,RESTR :START AGAIN AND REBUILD FMG FILE
 MOV #-1,PACK :RESET THE MFG FILE AVAILABLE FLAG
 JMP TABINT :RESTART POINT
 JMP HSET :EXIT

1\$: CMP #-1,(R1) :TO SEE IF NEXT BAD SPOT IN TABLE
 BEQ WRTRKD :BR IF NONE
 MOV (R1),R2 :UPDATE THE NEXT
 CMP MAXSEC,R2 :BAD SECTOR IS THE LAST SECTOR ?
 BLOS WRTRKD :BR IF IT IS
 BEQ WRTRKF :BR IF LESS
 BLO WRTRKD :BR IF LESS
 INC R2 :BAD SEC +1
 MOV R2,NEXT :NEXT STARTING ADDRESS
 BIS #BIT15,(R1) :MARK THE ACCESSED SECTOR

WRTRKC: MOV NEXT,R2 :CALCULATE THE WORD COUNT
 SUB #2,R2 :ENDSECTOR = NEXT - 2
 ASL R2 :WORD INDEX
 MOV WCTBL(R2),R3 :WORD CTR FOR SECTOR 0 TO ENDING SECTOR
 MOVB FMTDPB+10,R2 :STARTING SECTOR
 ASL R2 :WORD INDEX
 SUB WCTBL(R2),R3 :WORD COUNT FOR SECTOR 0 TO STARTING SECTOR
 ADD #258.,R3 :ADJUST ONE SECTOR
 NEG R3 :GET THE NEAGTIVE WORD COUNT
 MOV R3,FMTDPB+4 :LOAD THE WORD CTR INTO DPB
 BR WRTRKE :LOAD THE STARTING ADDRESS

WRTRKF: MOV #-1,NEXT :ALL DONE
 BIS #BIT15,(R1) :MASK THE SECTOR
 DEC R2 :PATCH FOR THE LAST SECTOR
 BR WRTRKH :PATCH

WRTRKD: MOV #-1,NEXT :NO MORE BAD SPOT
 MOV MAXSEC,R2 :LAST SECTOR

WRTRKH: ASL R2 :WORD INDEX
 MOV WCTBL(R2),R3 :WORD CTR FOR THE ENDING SECTOR
 MOVB FMTDPB+10,R2 :STARTING ADDRESS
 ASL R2 :WORD INDEX

```

229 014632 166203 006232          SUB    WCTBL(R2),R3      ;WORD CTR FOR THE STARTING SECTOR
230 014636 062703 000402          ADD    #258.,R3        ;ADJUST ONE SECTOR
231 014642 005403                    NEG    R3                ;NEGATIVE WORD COUNT
232 014644 010337 005544          MOV    R3,FMTDPB+4      ;LOAD THE WORD COUNT INTO DPB
233 014650 000240                    NOP                      ;DONE
234
235 014652 113702 005550          WRTRKE: MOVB   FMTDPB+10,R2 ;STARTING ADDRESS
236 014656 006302                    ASL    R2                ;LOCATE THE BUFFER ADDRESS
237 014660 016237 006132 005546    MOV    ADRTBL(R2),FMTDPB+6 ;BUFFER ADDRESS
238
239 014666 112737 000163 005542    WRTRK2: MOVB   #WRTHD,FMTDPB+2 ;SET WRITE HEADER & DATA COMMAND IN TBL
240 014674 012737 014674 001124    MOV    #.,$LPERR        ;SET UP LOOP ON ERROR ADDRESS
241 014702 012706 001100          MOV    #STACK,SP        ;LOAD STACK POINT
242 014706 004037 030622          1$:   JSR    RO,RM05      ;GO FORMAT A TRACK
243 014712 005540                    FMTDPB ;ADRS OF PARAMETERS - TBL
244 014714 000774                    BR     1$                ;WAIT FOR QUEUE IF FULL
245 014716 005737 005556          2$:   TST    FMTDPB+16    ;WAIT FOR COMMAND TO COMPLETE
246 014722 001775                    BEQ    2$                ;BR IF NOT DONE
247 014724 100024                    BPL    4$                ;BR IF NO ERROR
248 014726 012737 014754 001176    MOV    #3$, $ESCAPE     ;:ESCAPE TO 3$ ON ERROR
249 014734 004737 017204          JSR    PC,ERINDX        ;SEE WHICH ERROR
250 014740 104010                    EMT    10
251 014742 104011                    EMT    11
252 014744 104012                    EMT    12
253 014746 104013                    EMT    13
254 014750 104014                    EMT    14
255 014752 104015                    EMT    15
256 014754 004737 017304          3$:   JSR    PC,LOP.CK    ;LOOP ON THE ERROR ?
257 014760 022737 000003 001344    CMP    #3,RETRY         ;ERROR RETRY LIMIT ?
258 014766 001403                    BEQ    4$                ;BR IF REACHED
259 014770 005237 001344          INC    RETRY            ;COUNT THE ERROR
260 014774 000744                    BR     1$                ;TRY AGAIN
261 014776 005037 001176          4$:   CLR    $ESCAPE     ;CLEAR ERROR ESCAPE ADDRESS
262 015002 005037 001344          CLR    RETRY           ;CLEAR THE RETRY COUNTER
263
264          ;CHECK THE TRACK JUST WRITTEN
265
266 015006 112737 000153 005542    CKTRK: MOVB   #WCKHD,FMTDPB+2 ;SET WRITE CHECK HEADER & DATA COMMAND IN TBL
267 015014 012737 015014 001124    MOV    #.,$LPERR        ;SETUP LOOP ON ERROR ADDRESS
268 015022 012706 001100          MOV    #STACK,SP        ;LOAD STACK POINTER
269 015026 004037 030622          1$:   JSR    RO,RM05      ;GO CHECK THE TRACK JUST FORMATTED
270 015032 005540                    FMTDPB ;ADRS OF PARAMETERS - TBL
271 015034 000774                    BR     1$                ;WAIT FOR QUEUE IF FULL
272 015036 005737 005556          2$:   TST    FMTDPB+16    ;WAIT FOR COMMAND TO COMPLETE
273 015042 001775                    BEQ    2$                ;BR IF NOT DONE
274 015044 100131                    BPL    9$                ;BR IF NO ERROR
275 015046 113737 005566 001346    MOVB   RM.REG+RMDA,SAVSEC ;GET THE SECTOR ADDRESS
276 015054 001005                    BNE    3$                ;BR IF NOT SECTOR 0
277 015056 013737 001362 001346    MOV    MAXSEC,SAVSEC    ;RESTORE TO LAST SECTOR +1
278 015064 005237 001346          INC    SAVSEC           ;INCREMENT TO LAST SECTOR +1
279 015070 005337 001346          3$:   DEC    SAVSEC        ;ADJUST SECTOR TO THE ONE THAT FAILED
280 015074 012737 015376 001176    MOV    #11$, $ESCAPE   ;:ESCAPE TO 11$ ON ERROR
281 015102 004737 017204          JSR    PC,ERINDX        ;SEE WHICH ERROR
282 015106 104010                    EMT    10
283 015110 104011                    EMT    11
284 015112 104012                    EMT    12
285 015114 104013                    EMT    13

```

285	015116	104014			EMT	14	
286	015120	032737	040000	005570	BIT	#WCE, RM.REG+RMCS2	;WRITE CHECK ERROR ?
287	015126	001005			BNE	4\$;BR IF SET
288	015130	033727	005574		BIT	RM.REG+RMER1, (PC)+	;CHECK FOR DATA ERRORS
289	015134	130620			.WORD	DCK!OPI!DTE!MCRC!HCE!FER	;DATA ERROR BITS
290	015136	001001			BNE	4\$;BR IF SET
291	015140	104016			EMT	16	
292	015142	005737	001316		4\$: TST	SOF SW	;RETRYING THE SECTOR ?
293	015146	001413			BEQ	5\$;BR IF NOT
294	015150	012737	015422	001176	MOV	#12\$, \$ESCAPE	;ESCAPE TO 12\$ ON ERROR
295	015156	004737	020570		JSR	PC, RECORD	;RECORD THE BAD SPOT
296	015162	012737	100000	001410	MOV	#BIT15, HEADX	;RESET THE USER BIT
297	015170	004737	021416		JSR	PC, RESET	;RESET THE HEADER BITS
298	015174	104017			EMT	17	
299	015176	005037	001176		5\$: CLR	\$ESCAPE	;RESET ESCAPE ADDRESS
300	015202	032737	040000	005572	BIT	#ERR, RM.REG+RMDS	;DATA ERROR ?
301	015210	001002			BNE	6\$;BR IF IT IS
302	015212	104024			EMT	24	
303	015214	000401			BR	7\$;CONTINUE
304	015216				6\$: EMT	20	
305	015220	013701	001346		7\$: MOV	SAVSEC, R1	;FAILING SECTOR ADDRESS
306	015224	113737	001346	005550	MOVB	SAVSEC, FMTDPB+10	;SECTOR ADDRESS
307	015232	006301			ASL	R1	;SETUP INDEX TO ADDRESS WORDS
308	015234	016137	006132	005546	MOV	ADRTBL(R1), FMTDPB+6	;BUFFER ADDRESS FOR FAILING SECTOR
309	015242	013701	001362		MOV	MAXSEC, R1	;R1 ENDING SECTOR #
310	015246	005737	005536		TST	NEXT	;WHOLE TRACK DONE ?
311	015252	100404			BMI	8\$;BR IF IT IS
312	015254	013701	005536		MOV	NEXT, R1	;LOAD CURRENT ENDING SECTOR
313	015260	162701	000002		SUB	#2, R1	;ADJUST TWO SECTORS
314	015264	006301			8\$: ASL	R1	;WORD INDEX
315	015266	016102	006232		MOV	WCTBL(R1), R2	;WORDCTR FOR ENDING SECTOR
316	015272	113701	005550		MOVB	FMTDPB+10, R1	;STARTING ADDRESS
317	015276	006301			ASL	R1	;WORD COUNT
318	015300	166102	006232		SUB	WCTBL(R1), R2	;WORD CTR FOR STARTING SECTOR
319	015304	005402			NEG	R2	
320	015306	010237	001350		MOV	R2, SAVWC	;SAVE THE WORD COUNT
321	015312	012737	177376	005544	MOV	#-258., FMTDPB+4	;WORD COUNT FOR 1 SECTOR
322	015320	005237	001316		INC	SOF SW	;INDICATE THAT A RETRY IS IN PROGRESS
323	015324	000137	014666		JMP	WRTRK2	;REFORMAT ERROR SECTOR
324	015330	005737	001316		9\$: TST	SOF SW	;RETRY IN PROGRESS ?
325	015334	001432			BEQ	12\$;BR IF NOT
326	015336	022737	000002	001316	CMP	#2, SOF SW	;SEE IF LAST RETRY
327	015344	001403			BEQ	10\$;BR IF IT IS
328	015346	005237	001316		INC	SOF SW	;INCREMENT RETRY COUNT
329	015352	000625			BR	1\$;READ AGAIN
330	015354	123737	001362	005550	10\$: CMPB	MAXSEC, FMTDPB+10	;SEE IF LAST SECTOR ON THE TRACK
331	015362	001417			BEQ	12\$;BR IF IT IS
332	015364	004737	017530		JSR	PC, SCAWC	;SETUP TO CHECK REMAINING SECTORS ON THE TRACK
333	015370	005037	001316		CLR	SOF SW	;CLEAR RETRY COUNTER
334	015374	000614			BR	1\$;FINISH CHECKING THE TRACK
335	015376	004737	017304		11\$: JSR	PC, LOP.CK	;CHECK FOR LOOP ON ERROR
336	015402	022737	000003	001344	CMP	#3, RETRY	;ERROR RETRY REACHED ?
337	015410	001404			BEQ	12\$;BR IF IT IS
338	015412	005237	001344		INC	RETRY	;COUNT THE RETRY
339	015416	000137	015026		JMP	1\$;DO THE WRITE CHECK AGAIN
340	015422	005037	001176		12\$: CLR	\$ESCAPE	;CLEAR THE ERROR RETURN ESCAPE ADDRESS

```

341 015426 005037 001344      CLR      RETRY      ;CLEAR THE RETRY COUNTER
342 015432 005037 001316      CLR      SOFSW     ;CLEAR THE SOFTWARE RETRY COUNT
343 015436 005737 005536      TST      NEXT      ;WHOLE TRACK DONE ?
344 015442 100402          BMI      .+6       ;BR IF IT IS
345 015444 000137 014320      JMP      WRTRKA
346
347 015450 032777 000002 163476 HSET:  BIT      #BIT1,@SWR   ;LOOP ON CURRENT TRACK ?
348 015456 001402          BEQ      1$        ;BR IF NOT
349 015460 000137 013676      JMP      WRTRKX    ;START AGAIN ON THE SAME TRACK
350
351
352 015464 012737 040000 001410 1$:  MOV      #BIT14,HEADX ;RESET THE MFG BIT OF THE 1ST HEADER WORD
353 015472 012704 005434          MOV      #SECCU,R4  ;TABLE FOR BAD SECTOR ON CURRENT TRACK
354 015476 022714 177777      2$:  CMP      #-1,(R4)  ;END OF TABLE ?
355 015502 001414          BEQ      4$        ;BR IF SO
356 015504 032714 040000      BIT      #BIT14,(R4) ;HAS THE SECTOR BE ASSCESSED ?
357 015510 001006          BNE      3$        ;BR IF SO
358 015512 052714 040000      BIS      #BIT14,(R4) ;MASK THE SECTOR TABLE
359 015516 111437 001346      MOVB    (R4),SAVSEC ;SECTOR ADDRESS
360 015522 004737 021416      JSR     PC,RESET   ;RESET THE HEADER
361 015526 062704 000002      3$:  ADD      #2,R4    ;INCREMENT THE TABLE POINTER
362 015532 000761          BR       2$        ;LOOPING BACK
363 015534 000240      4$:  NOP
364
365 015536 004737 017426      WRTRKZ: JSR     PC,TRKTST ;GET UPDATED TRACK AND CYLINDER ADDRESSES
366 015542 000402          BR       HDREAD    ;RETURN HERE IF DONE
367
368 015544 000137 013676      JMP      WRTRKX    ;AND DO QUICK CHECK OF DISK
369
370
371
372
373
374 015550 005737 001320      HDREAD: TST     MODE   ;CHECK MODE ?
375 015554 001022          BNE      1$        ;NO
376 015556 022737 066666 001340  CMP      #066666,PATA ;PATTERN HAS BEEN ROTATED TO WORST CASE ?
377 015564 001002          BNE      .+6       ;
378 015566 000137 016342      JMP      $EOP      ;EXIT
379
380 015572 000241          CLC
381 015574 006037 001340      ROR     PATA       ;ROTATE PATTERN
382 015600 103003          BCC     .+10      ;SET SIGN BIT IF CARRY=1
383 015602 052737 100000 001340  BIS      #BIT15,PATA ;FROM BIT 0
384 015610 013737 001340 001342  MOV     PATA,PATB  ;SET BOTH PATTERN
385 015616 000137 013462      JMP     ST1        ;CONTINUE THE VERIFY OPERATION
386
387 015622 005737 001320      1$:  TST     MODE     ;'FORMAT', 'CHECK' OR VERIFY MODE ?
388 015626 003003          BGT     2$        ;BR IF 'VERIFY ONLY' MODE
389 015630 104401 037537      TYPE   ,MSQVER    ;TYPE 'STARTING QUICK VERIFICATION'
390 015634 000402          BR      3$
391 015636 104401 037600      2$:  TYPE   ,MSVER   ;TYPE 'STARTING VERIFICATION'
392 015642          3$:
015642 013746 001222      MOV     DRIVE,-(SP) ;:SAVE DRIVE FOR TYPEOUT
;:TYPE DRIVE NUMBER
015646 104403          TYPOS
;:GO TYPE--OCTAL ASCII
015650 001          .BYTE 1
;:TYPE 1 DIGIT(S)
015651 000          .BYTE 0
;:SUPPRESS LEADING ZEROS

```

393	015652	104401	001205		TYPE	,\$CRLF	:CR-LF	
394	015656	005037	001356		CLR	HEDERR	:CLEAR HEADER CHECK ERROR INDICATOR	
395	015662	004737	017352		JSR	PC,SETTBL	:GO SET UP DRIVER TABLE	
396	015666	004737	017556		JSR	PC,SETHDR	:GO SET UP HEADERS IN CORE	
397	015672	012700	045774	4\$:	MOV	#BUFV,R0	:GET STARTING ADDRESS OF EXPECTED HEADERS	
398	015676	012701	106174		MOV	#BUFV,R1	:GET STARTING ADDRESS OF RECIEVED HEADERS	
399	015702	010137	005546		MOV	R1,FMTDPB+6	:SET UP BUFFER ADDRESS	
400	015706	112737	000173	005542	MOVB	#RDHD,FMTDPB+2	:SET UP READ HEADER & DATA COMMAND	
401	015714	013737	001354	005544	MOV	MWC,FMTDPB+4	:SETUP WORD FOR VERIFY ONLY MODE	
402	015722	005037	001322		CLR	SNGSEC	:SETUP MULTI-SECTOR READ FLAG	
403	015726	005737	001320		TST	MODE	:VERIFY MODE ONLY ?	
404	015732	003005			BGT	6\$:BR IF YES	
405	015734	012737	177776	005544	5\$:	MOV	#-2,FMTDPB+4	:SETUP WORD COUNT FOR QUICK VERIFY
406	015742	005237	001322		INC	SNGSEC	:SETUP SINGLE SECTOR READ	
407								
408	015746	012737	015746	001124	6\$:	MOV	#,,\$LPERR	:SETUP LOOP ON ERROR ADDRESS
409	015754	004037	030622		7\$:	JSR	R0,RM05	:GO READ HEADER
410	015760	005540				FMTDPB	:ADRS OF PARAMETER TBL	
411	015762	000774				BR	7\$:WAIT IF QUEUE IS FULL
412	015764	005737	005556	8\$:	TST	FMTDPB+16	:WAIT FOR COMMAND TO COMPLETE	
413	015770	001775				BEQ	8\$:BR IF NOT DONE
414	015772	100044				BPL	11\$:BR IF NOT ERROR
415								
416	015774	012737	016170	001176		MOV	#13\$, \$ESCAPE	::ESCAPE TO 13\$ ON ERROR
417	016002	004737	017204			JSR	PC,ERINDX	:SEE WHICH ERROR
418	016006	104010				EMT	10	:DRIVE OFFLINE
419	016010	104011				EMT	11	:PERSISTENT DRIVE UNSAFE ERROR
420	016012	104012				EMT	12	:UNCORRECTABLE MASSBUS PARITY ERROR
421	016014	104013				EMT	13	:SOFTWARE TIMEOUT
422	016016	104014				EMT	14	:DRIVE UNSAFE ERROR
423								
424	016020	032737	040000	005572		BIT	#ERR,RM.REG+12	:ANY DRIVE ERRORS ?
425	016026	001426				BEQ	11\$:BR IF NO
426	016030	022737	000200	005574		CMP	#HCE,RM.REG+14	:IS IT A HEADER COMPARE ERROR ?
427	016036	001411				BEQ	9\$:BR IF YES
428	016040	032737	000400	005574		BIT	#HCRC,RM.REG+14	:IS IT A HEADER CRC ERROR ?
429	016046	001005				BNE	9\$:BR IF YES
430	016050	022737	100000	005622		CMP	#BSE,RM.REG+42	:BAD SECTOR ERROR ?
431	016056	001406				BEQ	10\$:BR IF YES
432	016060	104021				EMT	21	
433	016062	005037	001176	9\$:	CLR	\$ESCAPE	:CLEAR THE ERROR ESCAPE ADDRESS	
434	016066	104022				EMT	22	
435	016070	004737	017304			JSR	PC,LOP.CK	:CHECK FOR LOOP ON ERROR
436								
437	016074	000240		10\$:	NOP			
438	016076	005737	001322		TST	SNGSEC	:ALREADY READING SINGLE SECTORS ?	
439	016102	001714				BEQ	5\$:BR IF NO
440	016104	052711	140000	11\$:	BIS	#MF'UF,(R1)	:SET MFG AND USER SECTOR GOOD BITS	
441	016110	011037	045770		MOV	(R0),RBUF+4	:GET EXPECTED CYLINDER HEADER	
442	016114	011137	045764		MOV	(R1),RBUF	:GET RECIEVED CYLINDER HEADER	
443	016120	023737	045764	045770	CMP	RBUF,RBUF+4	:SEE IF CYL READ EQUALS CYL EXPECTED	
444	016126	001402				BEQ	12\$:BR IF CYL CORRECT
445	016130	104023				EMT	23	
446	016132	000416				BR	13\$	
447								
448	016134	005737	001320	12\$:	TST	MODE	:VERIFY ONLY MODE ?	
449	016140	003451			BLE	17\$:BR IF NO	


```
450 016142 016037 000002 045772      MOV      2(R0),RBUF+6      ;GET EXPECTED TRACK/SECTOR HEADER
451 016150 016137 000002 045766      MOV      2(R1),RBUF+2      ;GET RECIEVED TRACK/SECTOR HEADER
452 016156 023737 045766 045772      CMP      RBUF+2,RBUF+6     ;IS TRACK AND SECTOR CORRECT ?
453 016164 001405                      BEQ      15$                ;BR IF YES
454 016166 104007                      EMT      7
455
456 016170 004737 017304          13$:    JSR      PC,LOP.CK        ;CHECK FOR LOOP ON ERROR
457 016174 000137 016342          14$:    JMP      $EOP              ;END OF VERIFY
458
459 016200 000240          15$:    NOP
460 016202 123737 005550 ,001362      CMPB    FMTDPB+10,MAXSEC   ;DONE WITH ALL SECTORS ?
461 016210 001412                      BEQ      16$                ;BR IF YES
462 016212 105237 005550          INCB    FMTDPB+10         ;UPDATE SECTOR ADDRESS
463 016216 062700 001004          ADD     #516.,R0          ;ADJUST POINTER TO NEXT EXPECTED HEADER
464 016222 005737 001322          TST     SNGSEC           ;READING ONE SECTOR AT A TIME ?
465 016226 001252                      BNE     7$                  ;BR IF YES
466 016230 062701 001004          ADD     #516.,R1          ;ADJUST POINTER TO NEXT RECIEVED HEADER
467 016234 000723                      BR      11$                ;GO COMPARE NEXT SECTOR
468 016236 105037 005550          16$:    CLRB    FMTDPB+10         ;START AT SECTOR 0
469 016242 105237 005551          INCB    FMTDPB+11         ;INCREMENT TRACK NUMBER
470 016246 123737 005551 001332      CMPB    FMTDPB+11,ENDTRK   ;WAS IT LAST TRACK ?
471 016254 101003                      BHI     17$                ;BR IF YES
472 016256 004737 017756          JSR     PC,UPDTRK         ;UPDATE TRACK ADDRESS IN BUFFER
473 016262 000603                      BR      4$                  ;GO READ NEXT TRACK
474
475 016264 113737 001334 005551 17$:    MOVVB   BEGTRK,FMTDPB+11    ;GET BEGINNING TRACK ADDRESS
476 016272 005737 001400          TST     WRAP              ;SEEKING FROM HI TO LO CYLINDERS ?
477 016276 001407                      BEQ     18$                ;BR IF NO
478 016300 005337 005552          DEC     FMTDPB+12         ;DECREMENT CYLINDER ADDRESS
479 016304 023737 005552 001326      CMP     FMTDPB+12,ENDCYL   ;DONE WITH VERIFY ?
480 016312 002730                      BLT     14$                ;BR IF YES
481 016314 000406                      BR      19$
482 016316 005237 005552          18$:    INC     FMTDPB+12         ;INCREMENT CYLINDER ADDRESS BEING CHECKED
483 016322 023737 005552 001326      CMP     FMTDPB+12,ENDCYL   ;DONE WITH VERIFY ?
484 016330 101321                      BHI     14$                ;BR IF YES
485 016332 004737 017644          19$:    JSR     PC,UPDCYL        ;UPDATE CYLINDER ADDRESS IN BUFFER
486 016336 000137 015672          JMP     4$                  ;GO READ NEXT CYLINDER
487
567
568
```

.SBTTL END OF PASS ROUTINE

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

```
016342          $EOP:
016342 005737 001406          1$:    TST     EDIT              ;UPDATE LAST TRACK ?
016346 001537                      BEQ     8$                  ;BR IF NO
016350 005037 001344          CLR     RETRY              ;CLEAR RETRY COUNT
016354 004537 020710          JSR     R5,SORT2           ;SORT THE USTAB IN ASCENDING FORM
016360 003434          USTAB+8.                  ;TABLE ENTRY POINT
016362 004737 021050          JSR     PC,TEXT           ;SET UP THE BUFFER
016366 112737 000020 005541      MOVVB   #20,FMTDPB+1       ;16 BIT MODE
016374 112737 000143 005542      MOVVB   #SETFMT,FMTDPB+2   ;SET FORMAT COMMAND
016402 004037 030622          2$:    JSR     R0,RM05           ;CALL THE DRIVER
016406 005540          FMTDPB
```

```

016410 000774
016412 005737 005556      3$:  BR      2$      ;LOOP IF QUEQU FAILS
016416 001775              TST      FMTDPB+16 ;ALL DONE ?
                                BEQ      3$      ;BR IF NOT

016420 012737 001466 005552      MOV      #822.,FMTDPB+12 ;CYLINDER ADDRESS
016426 113737 001324 005551      MOVB     LSTRK,FMTDPB+11 ;TRACK NUMBER
016434 105037 005550              CLRB     FMTDPB+10     ;SECTOR 0
016440 012737 157700 005544      MOV      #-<258.*32.>,FMTDPB+4 ;WORD COUNT
016446 012737 045774 005546      MOV      #BUFP,FMTDPB+6     ;BUFFER ADDRESS
016454 112737 000163 005542      MOVB     #WRTHD,FMTDPB+2 ;WRITE HEAD AND DATA COMMAND
016462 000240              NOP
016464 004037 030622      4$:  JSR      R0,RM05      ;CALL THE DRIVER
016470 005540              FMTDPB
016472 000773              BR      4$      ;LOOP IF QUEUE FAILS
016474 005737 005556      5$:  TST      FMTDPB+16 ;ALL DONE ?
016500 001775              BEQ      5$      ;BR IF NOT
016502 100061              BPL      8$      ;EXIT IF NO ERROR

016504 012737 016532 001176      MOV      #6$, $ESCAPE   ;ERROR ESCAPE ADDRESS
016512 004737 017204              JSR      PC,ERINDX     ;ERROR INDICATOR RT.
016516 104010              EMT      10
016520 104011              EMT      11
016522 104012              EMT      12
016524 104013              EMT      13
016526 104014              EMT      14
016530 104015              EMT      15
016532 000240              6$:  NOP
016534 005237 001344              INC
016540 022737 000003 001344      CMP      #3,RETRY      ;OVER 3 TIMES ?
016546 101345              BHI      4$      ;BR IF IT IS
016550 000137 017162              JMP      REJCT3        ;WRITE THE LAST TRACK FAILS THREE TIMES
016554 122737 000037 005550      CMPB     #31.,FMTDPB+10 ;LAST SECTOR ?
016562 003431              BLE      8$      ;BR IF ALL SET
016564 113737 005566 001346      MOVB     RM.REG+RMDA,SAVSEC ;FOUND THE FALLING SPOT
016572 001003              BNE      7$      ;BR IF NOT 0
016574 012737 000037 001346      MOV      #31.,SAVSEC    ;THE LAST SECTOR IS A BAD SECTOR
016602 113737 001346 005550      7$:  MOVB     SAVSEC,FMTDPB+10 ;LOAD THE NEW STARTING SECTOR
016610 113701 001346              MOVB     SAVSEC,R1      ;LOAD THE WORD CTR AND BUFFER
016614 006301              ASL      R1            ;WORD INDEX
016616 016137 006132 005546      MOV      ADRTBL(R1),FMTDPB+6 ;BUFFER ADDRESS
016624 012737 157700 005544      MOV      #-<258.*32.>,FMTDPB+4 ;WORD COUNT
016632 066137 006232 005544      ADD      WCTBL(R1),FMTDPB+4 ;WORD COUNT
016640 005037 001344              CLR      RETRY        ;RESET THE RETRY FLAG
016644 000706              BR      4$      ;BR BACK

016646 005737 001320      8$:  TST      MODE        ;FORMAT, CHECK OR VERIFY MODE
016652 003007              BGT      10$       ;BR IF VERIFY ONLY MODE
016654 001403              BEQ      9$      ;BR IF CHECK
016656 104401 037633              TYPE     ,MFCMPT     ;FORMAT COMPLETE
016662 000405              BR      11$
016664 104401 037656      9$:  TYPE     ,MCCMPT     ;CHECK COMPLETE
016670 000402              BR      11$
016672 104401 037700      10$: TYPE     ,MVCMPT     ;VERIFY COMPLETE
016676 104401 037723      11$: TYPE     ,NUMERR     ;TOTAL ERRORS
016702 013746 001126      MOV      $ERTTL,-(SP)  ;SAVE $ERTTL FOR TYPEOUT
016706 104405              TYPDS
016710 005037 001126      CLR      $ERTTL      ;CLEAR THE ERROR ACCUMULATOR

```

016714	104401	037753			TYPE	,STARS	
016720	005737	001402			TST	DEVMP	; ALL ASSIGNED DEVICE DONE ?
016724	001030				BNE	DONE	; TO FORMAT NEXT DRIVE
016726	005337	001216			DEC	\$PASS	; RESTORE THE PASS COUNT
016732	000240			12\$:	NOP		
016734	005237	001216			INC	\$PASS	:: INCREMENT THE PASS NUMBER
016740	042737	100000	001216		BIC	#100000,\$PASS	:: DON'T ALLOW A NEG. NUMBER
016746	005327				DEC	(PC)+	:: LOOP?
016750	000001			\$EOPCT:	.WORD	1	
016752	003013				BGT	\$DOAGN	:: YES
016754	012737				MOV	(PC)+,@(PC)+	:: RESTORE COUNTER
016756	000001			\$ENDCT:	.WORD	1	
016760	016750				\$EOPCT		
016762	013700	000042		\$GET42:	MOV	@#42,R0	:: GET MONITOR ADDRESS
016766	001405				BEQ	\$DOAGN	:: BRANCH IF NO MONITOR
016770	000005				RESET		:: CLEAR THE WORLD
016772	004710			\$ENDAD:	JSR	PC,(R0)	:: GO TO MONITOR
016774	000240				NOP		:: SAVE ROOM
016776	000240				NOP		:: FOR
017000	000240				NOP		:: ACT11
017002				\$DOAGN:			
017002	000137				JMP	@(PC)+	:: RETURN
017004	017006			\$RTNAD:	.WORD	DONE	
569							
570	017006	105737	001150	DONE:	TSTB	\$AUTOB	; RUNNING IN AUTO MODE ?
571	017012	001405			BEQ	13\$; NO
572	017014	005737	001402		TST	DEVMP	; ARE ALL DRIVES DONE ?
573	017020	001427			BEQ	16\$; BR IF YES
574	017022	000137	010466		JMP	M5FX	; NEXT DRIVE FROM APT DEVICE MAP
575							
576	017026	032777	000001	162120	13\$:	BIT	#SW0,@SWR
577	017034	001002			BNE	14\$; BR IF SET
578	017036	000137	011342		JMP	M0	; ASK FOR NEXT DRIVE
579							
580	017042	012706	001100		14\$:	MOV	#STACK,SP
581	017046	005737	001320		TST	MODE	; CHECK MODE ?
582	017052	001402			BEQ	15\$; YES
583	017054	000137	013462		JMP	ST1	; DO THE FORMAT OR CHECK AGAIN
584							
585	017060	012737	133331	001340	15\$:	MOV	#133331,PATA
586	017066	012737	133331	001342		MOV	#133331,PATB
587	017074	000137	013462		JMP	ST1	; LOOP ON CHECK
588							
589	017100	005237	001216		16\$:	INC	\$PASS
590	017104	005237	001214		17\$:	INC	\$TESTN
591	017110	000775			BR	17\$; GARBAGE CODE FOR APT
592							
593	017112	104401	001205		REJECT1:	TYPE	,\$CRLF
594	017116	104401	040354			TYPE	,MMSG4
595	017122	104401	040373			TYPE	,MHALT
596	017126	000000			HALT		
597	017130	000137	007020		JMP	BEGIN2	
598							
599	017134	104401	001205		REJECT2:	TYPE	,\$CRLF
600	017140	113737	005551	001366		MOVB	FMTDPB+11,DS.TRK ; TRACK NUMBER FOR ERROR LOG
601	017146	104025			EMT	25	
602	017150	104401	040373		TYPE	,MHALT	

603 017154 000000
604 017156 000137 007020
605
606 017162 104401 001205
607 017166 104401 042126
608 017172 104401 040373
609 017176 000000
610 017200 000137 007020

HALT
JMP BEGIN2
REJCT3: TYPE ,SCLF
TYPE ,MESG17
TYPE ,MHALT
HALT
JMP BEGIN2

611
612
613
614
615
616
617
618
619

.SBTTL SUPPORT SUBROUTINES

;THIS ROUTINE DETERMINES THE ERROR TYPE

620 017204 010146
621 017206 005001
622 017210 033727 005556
623 017214 060006
624 017216 001025
625 017220 033727 005556
626 017224 010000
627 017226 001020
628 017230 033727 005556
629 017234 006000
630 017236 001013
631 017240 033727 005556
632 017244 001400
633 017246 001006
634 017250 033727 005556
635 017254 000020
636 017256 001001
637 017260 005201
638 017262 005201
639 017264 005201
640 017266 005201
641 017270 005201
642 017272 006301
643 017274 060166 000002
644 017300 012601
645 017302 000207

ERINDX: MOV R1,-(SP) ;STORE R1
CLR R1 ;CLEAR R1
BIT FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
.WORD BIT14!BIT13!BIT2!BIT1 ;DRIVE OFFLINE
BNE 5\$;BR IF OFFLINE
BIT FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
.WORD BIT12 ;DRIVE PERSISTENTLY UNSAFE
BNE 4\$;BR IF UNSAFE
BIT FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
.WORD BIT11!BIT10 ;UNCORRECTABLE MASSBUS PARITY ERROR ?
BNE 3\$;BR IF PARITY ERROR
BIT FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
.WORD BIT09!BIT08 ;SOFTWARE TIMEOUT ?
BNE 2\$;BR IF YES
BIT FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
.WORD BIT04 ;DRIVE UNSAFE ERROR ?
BNE 1\$;BR IF YES
INC R1 ;INCREMENT THE RETURN INDEX
1\$: INC R1 ;INCREMENT THE RETURN INDEX
2\$: INC R1 ;INCREMENT THE RETURN INDEX
3\$: INC R1 ;INCREMENT THE RETURN INDEX
4\$: INC R1 ;INCREMENT THE RETURN INDEX
5\$: ASL R1 ;DOUBLE THE INCREMENT
ADD R1,2(SP) ;DEVELOP THE RETURN ADDRESS
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

646
647
648

;ROUTINE TO CHECK FOR LOOP ON ERROR

649 017304 032777 001000 161642
650 017312 001402
651 017314 000177 161604
652
653 017320 005037 001176
654 017324 033727 005556
655 017330 072002
656 017332 001004
657 017334 032737 004000 005574
658 017342 001402
659 017344 000137 016342

LOP.CK: BIT #SW09,@SWR ;LOOP ON ERROR ?
BEQ 1\$;BR IF NOT
JMP @SLPERR ;GO TO THE LOOP ON ERROR ADDRESS
1\$: CLR \$ESCAPE ;CLEAR THE ERROR ESCAPE ADDRESS
BIT FMTDPB+16,(PC)+ ;CHECK FOR 'FATAL' ERROR
.WORD BIT14!BIT13!BIT12!BIT10!BIT01 ;'FATAL' ERROR BITS
BNE 2\$;BR IF NOT
BIT #WLE,RM.REG+RMER1 ;WRITE LOCK ERROR ?
BEQ 3\$;BR IF NOT
2\$: JMP \$EOP ;TERMINATE THE FORMAT

```
660
661 017350 000207      3$:   RTS      PC      ;RETURN
662
663                   ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
664
665 017352 113737 001222 005540 SETTBL: MOV     DRIVE,FMTDPB ;SET UP DRIVE #
666 017360 105037 005543          CLR     FMTDPB+3      ;CLEAR HIGH ORDER ADRS BITS
667 017364 013737 001354 005544          MOV     MWC,FMTDPB+4    ;LOAD UP WORD COUNT
668 017372 012737 045774 005546          MOV     #BUFP,FMTDPB+6  ;LOAD UP CURRENT ADRS
669 017400 105037 005550          CLR     FMTDPB+10     ;SET SECTOR TO ZERO
670 017404 113737 001334 005551          MOV     BEGTRK,FMTDPB+11 ;SET UP STARTING TRK ADRS
671 017412 013737 001330 005552          MOV     BEGCYL,FMTDPB+12 ;SET UP STARTING CYL
672 017420 005037 005556          CLR     FMTDPB+16     ;CLEAR RM STATUS
673 017424 000207          RTS      PC      ;RETURN FROM SETUP
674
675                   ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
676                   ;IT IS ENTERED AFTER EVERY TRK OPERATION
677
678 017426                   TRKTST:
679 017426 105237 005551          INCB    FMTDPB+11      ;INCREMENT TRACK NUMBER
680 017432 123737 005551 001332          CMPB    FMTDPB+11,ENDTRK ;LAST TRACK TO BE FORMATTED ON THIS CYLINDER ?
681 017440 101003          BHI     1$           ;BR IF YES
682 017442 004737 017756          JSR     PC,UPDTRK    ;UPDATE TRACK ADDRESS IN BUFFER
683 017446 000425          BR      4$           ;EXIT
684
685 017450 113737 001334 005551 1$:   MOV     BEGTRK,FMTDPB+11 ;GET BEGINNING TRACK ADDRESS
686 017456 005737 001400          TST     WRAP         ;SEEKING FROM HI TO LO CYLINDER ?
687 017462 001407          BEQ     2$           ;BR IF NO
688 017464 005337 005552          DEC     FMTDPB+12    ;DECREMENT CYLINDER ADDRESS
689 017470 023737 005552 001326          CMP     FMTDPB+12,ENDCYL ;DONE WITH FORMATTING ?
690 017476 002413          BLT     5$           ;BR IF YES
691 017500 000406          BR      3$
692 017502 005237 005552          INC     FMTDPB+12    ;INCREMENT CYLINDER ADDRESS
693 017506 023737 005552 001326          CMP     FMTDPB+12,ENDCYL ;DONE WITH FORMATTING ?
694 017514 101004          BHI     5$           ;BR IF YES
695 017516 004737 017644          3$:   JSR     PC,UPDCYL  ;UPDATE CYLINDER ADDRESS IN BUFFER
696
697 017522 062716 000002          4$:   ADD     #2,(SP)    ;ADD TWO TO RETURN ADRS (NOT DONE)
698 017526 000207          5$:   RTS      PC      ;RETURN, DONE FORMAT !!
699
700                   ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
701                   ;AFTER A WRITE CHECK ERROR
702
703 017530 013737 001350 005544 SCAWC: MOV     SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
704 017536 062737 001004 005546          ADD     #516.,FMTDPB+6 ;SET UP BA FOR REMAINING SECTORS
705 017544 005237 005550          INC     FMTDPB+10    ;ADVANCE TO NEXT SECTOR IN TBL
706 017550 005037 001316          CLR     SQFSW        ;RESET RETRY COUNTER
707 017554 000207          RTS      PC      ;RETURN TO COMPLETE TRK FORMAT
708
709
710                   ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
711                   ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
712
713 017556 013701 001362          SE^HDR: MOV     MAXSEC,R1 ;SET UP SECTOR COUNT
714 017562 012737 000001 001214          MOV     #1,$TESTN    ;GARBAGE CODE FOR APT
715 017570 012700 045774          MOV     #BUFP,R0     ;SET UP HEADER POINTER IN R0
716 017574 013702 001330          MOV     BEGCYL,R2    ;PUT STARTING CYL # IN R2
```

```

717 017600 052702 140000      BIS      #MFG!UF,R2      ;SET MFG AND USR GOOD BITS (DEC STANDARD 144)
718 017604 013703 001334      MOV      BEGTRK,R3     ;PUT STARTING TRK # IN R3
719 017610 000303              SWAB     R3            ;JUSTIFY TRACK ADRS
720 017612 005737 001360      TST     SEC30         ;SEE IF 30 OR 32 SECTOR MODE
721 017616 001402              BEQ     1$            ;BR IF 30 SECTOR MODE
722 017620 052702 010000      BIS      #FMT,R2      ;SET THE 32 SECTOR FORMAT BIT
723 017624 010220      1$:  MOV     R2,(R0)+     ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
724 017626 010320      MOV     R3,(R0)+     ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
725 017630 062700 001000      ADD     #512.,R0     ;SET UP FOR NEXT HEADER
726 017634 005203              INC     R3            ;UPDATE SECTOR ADRS FOR NEXT HEADER
727 017636 005301              DEC     R1            ;MAINTAIN COUNT OF SECTORS
728 017640 002371              BGE    1$            ;BR IF NOT LAST SECTOR
729 017642 000207              RTS     PC            ;EXIT - HEADERS ARE LOADED INTO CORE

```

;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS
;AND FILLS DATA BUFFER FIELD IN CORE

```

730
731
732
733
734 017644      UPDCYL:
      017644 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      017646 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
      017650 010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
      017652 010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
735 017654 013701 001362      MOV     MAXSEC,R1    ;SET UP SECTOR COUNT
736 017660 012700 045774      MOV     #BUFP,R0     ;SET UP HEADER POINTER IN R0
737 017664 013703 001334      MOV     BEGTRK,R3    ;GET BEGINNING TRACK
738 017670 000303              SWAB     R3            ;STARTING SECTOR = 0
739 017672 005737 001400      1$:  TST     WRAP         ;SEEKING FROM HI TO LO CYLINDERS ?
740 017676 001402              BEQ     2$            ;BR IF NO
741 017700 162710 000002      SUB     #2,(R0)      ;DECREMENT CYLINDER NUMBER
742 017704 005220      2$:  INC     (R0)+        ;INCREMENT FOR NEXT CYLINDER
743 017706 010320      MOV     R3,(R0)+     ;WRITE TRACK AND SECTOR HEADER
744 017710 012702 001000      MOV     #512.,R2     ;DATA FIELD LENGTH
745 017714 013720 001340      3$:  MOV     PATA,(R0)+   ;FILL BUFFER
746 017720 013720 001342      MOV     PATB,(R0)+   ;FILL BUFFER
747 017724 162702 000004      SUB     #4,R2        ;END OF DATA FIELD ?
748 017730 001371              BNE    3$            ;NO
749 017732 005203              INC     R3            ;INCREMENT TO NEXT SECTOR
750 017734 005301              DEC     R1            ;COUNT SECTORS
751 017736 002355              BGE    1$            ;BR IF NOT LAST SECTOR
752 017740 012603      MOV     (SP)+,R3     ;;POP STACK INTO R3
      017742 012602      MOV     (SP)+,R2     ;;POP STACK INTO R2
      017744 012601      MOV     (SP)+,R1     ;;POP STACK INTO R1
      017746 012600      MOV     (SP)+,R0     ;;POP STACK INTO R0
753 017750 005237 001214      INC     $TESTN       ;INCREMENT TST NUMBER FOR EACH CYLINDER CHANGE
754
755 017754 000207              RTS     PC            ;GARBAGE CODE FOR APT
756
757
758
759 017756      UPDTRK:
      017756 010046      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
      017760 010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
760 017762 013701 001362      MOV     MAXSEC,R1    ;SET UP SECTOR COUNT
761 017766 012700 045774      MOV     #BUFP,R0     ;SET UP HEADER POINTER IN R0
762 017772 005720      TST     (R0)+        ;POINT HEADER POINTER TO TRK - SEC ADRS
763 017774 062710 000400      1$:  ADD     #400,(R0)    ;INDEX TRK ADRS
764 020000 062700 001004      ADD     #516.,R0     ;SET UP FOR NEXT HEADER

```

;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE

```

765 020004 005301          DEC      R1          ;COUNT SECTORS
766 020006 002372          BGE      1$          ;BR IF NOT LAST SECTOR
767 020010 012601          MOV      (SP)+,R1    ;POP STACK INTO R1
      020012 01260G        MOV      (SP)+,R0    ;POP STACK INTO R0
768 020014 000207          RTS      PC          ;EXIT
769
770          ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
771
772 020016 012737 020074 000004 ST.CLK: MOV      #STCLK1,@#ERRVEC ;SET UP VECTOR FOR P CLK
773 020024 005037 000006          CLR      @#ERRVEC+2 ;NEW PSW
774 020030 005777 161244          TST      @SLKCSR     ;CHECK FOR KW11-P
775 020034 013746 001304          MOV      $LPVEC,-(SP) ;VECTOR ADDRESS
776 020040 012776 020250 000000 MOV      #CLOCK,@(SP) ;SET UP KW11-P VECTOR
777 020046 062716 000002          ADD      #2,(SP)     ;POINT TO PSW
778 020052 012736 000300          MOV      #PR6,@(SP)+ ;PSW - PRI 6
779 020056 012777 177777 161216 MOV      #-1,@SLKCSB ;LOAD COUNTER BUFFER
780 020064 012777 000135 161206 MOV      #135,@SLKCSR ;SET CLK - CNT UP
781 020072 000426          BR      STCLK3
782 020074 062706 000004          STCLK1: ADD     #4,SP ;RESTORE THE STACK POINTER
783 020100 012737 020144 000004 MOV      #STCLK2,@#ERRVEC ;CHANGE ERROR VECTOR
784 020106 005777 161176          TST      @SLKS      ;LOOK FOR KW11-L
785 020112 013746 001312          MOV      $LLVEC,-(SP) ;KW11-L VECTOR ADDRESS
786 020116 012776 020250 000000 MOV      #CLOCK,@(SP) ;SET UP KW11-L VECTOR
787 020124 062716 000002          ADD      #2,(SP)     ;INCREMENT VECTOR ADDRESS
788 020130 012736 000300          MOV      #PR6,@(SP)+ ;PSW - PRI 6
789 020134 012777 000100 161146 MOV      #100,@SLKS  ;SET KW11-L INTERRUPT ENABLE
790 020142 000402          BR      STCLK3
791 020144 062706 000004          STCLK2: ADD     #4,SP ;RESTORE THE STACK POINTER
792 020150 012737 000006 000004 STCLK3: MOV      #6,@#ERRVEC ;RESTORE THE ERROR VECTOR
793 020156 000207          RTS      PC
794
795          ;THIS CODE PRINT THE ABORT MESSAGE AND LET O.P. RESTART
796
797 020160 105737 001150          RESTRT: TSTB     $AUTOB ;RUN UNDER APT ?
798 020164 001024          BNE      1$          ;BR IF SO
799 020166 004737 025576          JSR      PC,$TKINT ;ENABLE TO READ
800 020172 104401 040212          TYPE     ,MSG2      ;IMPROPER MFG INFORMATION
801 020176 104410          RDCHR
802 020200 012637 001174          MOV      (SP)+,$TMPO ;GET CHARACTER
803 020204 023727 001174 000015 CMP      $TMPO,#CR   ;CARRIAGE RETURN ?
804 020212 001411          BEQ      1$          ;BR IF YES
805 020214 123727 001174 000131 CMPB     $TMPO,#'Y   ;ENTER Y ?
806 020222 001405          BEQ      1$          ;BR IF YES
807 020224 104401 040373          TYPE     ,MHALT
808 020230 000000          HALT
809 020232 000137 007020          JMP      BEGIN2     ;JUMP TO BEGIN2
810
811 020236 104401 036417          1$:      TYPE     ,Y ;TYPE 'Y'
812 020242 104401 001205          TYPE     ,$CRLF    ;CR-LF
813 020246 000207          RTS      PC          ;EXIT
814
815          ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
816
817 020250 012746 000020          CLOCK: MOV      #16,-(SP) ;PUT MILLISECONDS ON THE STACK
818 020254 004737 034272          JSR      PC,$RMTMR ;GO REPORT TIME
819 020260 000002          RTI
820

```

```

821          ;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME
822
823 020262 012706 001100 C.ENTR: MOV #STACK,SP ;INITIALIZE THE STACK
824 020266 005737 027746 1$: TST TRNSWT ;ALL ACTIVITY STOPPED ?
825 020272 001375 BNE 1$ ;BR IF NOT
826 020274 000005 RESET ;CLEAR THE BUS
827 020276 000137 011306 JMP M1B ;START AGAIN WITH DRIVE SELECTION
828
829          ;ROUTINE TO INTERCEPT 'CONTROL O' TYPED WHILE PROGRAM IS FORMATTING
830          ;THE DISK PACK. THE CURRENT DISK ADDRESS IS TYPED TO THE TERMINAL
831          ;DURING THIS ROUTINE IN THE FOLLOWING FORMAT.
832          ;
833          ;PRESENT ADDRESS IS: CXXX TXXX ;WHERE 'C' IS THE CYLINDER ADDRESS
834          ;AND 'T' IS THE TRACK ADDRESS.(IN DECIMAL)
835
836 020302 117746 160654 O.ENTR: MOVB @STKB,-(SP) ;READ FROM TTY BUFFER
837 020306 042716 177600 BIC #^C177,(SP) ;STRIP PARITY
838 020312 021627 000003 CMP (SP),#3 ;CONTROL-C ?
839 020316 001005 BNE 1$ ;BR IF NO
840 020320 104401 026767 TYPE ,%CNTLC ;TYPE A CONTROL-C (^C)
841 020324 005726 TST (SP)+ ;RESTORE STACK
842 020326 000177 161042 JMP @CNTLC ;JUMP TO CONTROL-C RESTART
843 020332
844 020332 022627 000017 1$: CMP (SP)+,#17 ;CONTROL-O ?
845 020336 001406 BEQ 2$ ;YES
846 020340 005777 160616 TST @STKB ;CLEAR DONE BIT
847 020344 012777 000100 160606 MOV #100,@STKS ;ENABLE TTY INTERRUPT
848 020352 000002 RTI
849 020354
850
851 020354 005037 177776 TYPADR: CLR PS ;SET PROCESSOR TO PRIORITY 0
852 020360 012737 020262 001374 MOV #C.ENTR,CNTLC ;CHANGE 'CONTROL C' RETURN ADDRESS
853 020366 104401 037775 TYPE ,ADDRIS ;'PRESENT ADDRESS IS: '
854 020372 104401 036411 TYPE ,C ;'C'
855 020376 013746 005552 MOV FMTDPB+12,-(SP) ;PUT THE CYLINDER ADDRESS ON THE STACK
856 020402 004737 027256 JSR PC,$SB2D ;CONVERT IT TO DECIMAL
857 020406 004737 027216 JSR PC,$SUPRS ;TYPE IT
858 020412 104401 036425 TYPE ,BLNKS2 ;TYPE 2 SPACES
859 020416 104401 036415 TYPE ,T ;'T'
860 020422 005046 CLR -(SP) ;CLEAR THE STACK
861 020424 113716 005551 MOVB FMTDPB+11,(SP) ;PUT THE TRACK ADDRESS ON THE STACK
862 020430 004737 027256 JSR PC,$SB2D ;CONVERT IT TO DECIMAL
863 020434 004737 027216 JSR PC,$SUPRS ;TYPE IT
864 020440 104401 001205 TYPE ,%CRLF ;CR-LF
865 020444 012737 020302 000060 MOV #O.ENTR,@TKVEC ;RESTORE ENTRANCE FOR CONTROL-O
866 020452 000002 RTI ;RETURN
867
868          ;PARAMETER ENTRY ROUTINE
869          ;CALL
870          ; JSR R4,PARENT ;THE CALLING SEQ
871          ; TABLE ;PARAMETERS TABLE
872          ;
873
874 020454 010346 PARENT: MOV R3,-(SP) ;SAVE R3
875          ; MOV #TABLE,R3 ;PARAMETER TABLE ADDRESS
876 020456 011403 MOV (R4),R3 ;PARAMETER TABLE ADDRESS
877 020460 012337 020470 1$: MOV (R3)+,2$ ;ADDRESS OF PARAMETER NAME

```



```

878 020464 001435          BEQ      7$          ;BR IF AT END OF TABLE
879 020466 104401          TYPE     ;TYPE THE PARAMETER NAME
880 020470 000000          2$: .WORD 0          ;ADDRESS OF PARAMETER NAME TEXT
881 020472 012302          MOV      (R3)+,R2      ;MAXIMUM PARAMETER VALUE
882 020474 012305          MOV      (R3)+,R5      ;ADDRESS OF PARAMETER
883 020476 011546          MOV      (R5),-(SP)    ;CURRENT VALUE OF PARAMETER
884 020500 104405          TYPDS   ;TYPE THE CURRENT VALUE OF THE PARAMETER
885 020502 104401 036405   TYPE     .SLASH      ;' / '
886 020506 104411          RDLIN   ;READ THE KEYBOARD
887 020510 012601          MOV      (SP)+,R1      ;INPUT ASCII STRING ADDRESS
888 020512 004537 023462   JSR     R5,CK.DIG     ;CHECK THE DIGIT(S)
      020516 020460          1$      ;CARRIAGE RETURN ONLY ENTERED
      020520 020560          7$      ;$ ONLY ENTERED
      020522 020536          4$      ;ILLEGAL INPUT
      020524 020532          3$      ;TERMINATED WITH A CARRIAGE RETURN
      020526 020536          4$      ;TERMINATED WITH A ''
      020530 020550          5$      ;TERMINATED WITH A ''
889 020532 010215          3$: MOV      R2,(R5)    ;MOVE NEW VALUE TO PARAMETER LOCATION
890 020534 000751          BR       1$          ;GET MORE PARAMETERS
891 020536 104401 037211   4$: TYPE     .BADENT   ;'BAD ENTRY'
892 020542 162703 000006   SUB      #6,R3       ;DECREMENT THE TABLE POINTER
893 020546 000744          BR       1$          ;TRY AGAIN
894 020550 010215          5$: MOV      R2,(R5)    ;NEW VALUE
895 020552 000402          BR       7$          ;EXIT
896          :6$: MOV      #TABLE,R3 ;RELOAD THE PARAMETER TABLE ADDRESS
897 020554 011403          6$: MOV      (R4),R3  ;RELOAD THE PARAMETER TABLE ADDRESS
898 020556 000740          BR       1$          ;TRY AGAIN
899 020560 012603          7$: MOV      (SP)+,R3  ;RESTORE R3
900 020562 062704 000002   ADD      #2,R4       ;ADJUST THE RETURN ADDRESS
901 020566 000204          RTS      R4         ;RETURN FROM THE R4
902          :          PC          ;RETURN
903
904          ;THIS ROUTINE LOAD THE BAD SECTOR INTO USTAB TABLE
905          ;CALL SEQ JSR PC,RECORD
906          ;ALL REGISTER USED ARE DESTORIED
907
908 020570 C12701 003434   RECORD: MOV      #USTAB+10,R1 ;TABLE ENTRY IN R1
909 020574 022711 177777   1$:  CMP      #-1,(R1) ;AN OPENING IN THE TABLE
910 020600 001421          BEQ      3$          ;BR IF IT IS
911 020602 023711 005552   CMP      FMTDPB+12,(R1) ;CYLINDER NUMBER MATCHES ?
912 020606 001010          BNE      2$          ;BR IF NOT
913 020610 123761 005551 000003   CMPB    FMTDPB+11,3(R1) ;TRK NUMBER MATCHES ?
914 020616 001004          BNE      2$          ;BR IF NOT
915 020620 123761 005550 000002   CMPB    FMTDPB+10,2(R1) ;SECTOR NUMBER MATCHES ?
916 020626 001416          BEQ      4$          ;BR TO EXIT,IF THE SPOT HAS RECORDED
917 020630 062701 000004   2$:  ADD      #4,R1     ;INCREMENT 4 BYTES
918 020634 022701 004424   CMP      #USTAB+512.,R1 ;END OF TABLE ?
919 020640 101355          BHI      1$          ;BR IF NOT
920 020642 000411          BR       5$          ;THROW AWAY THE PACK
921 020644 013711 005552   3$:  MOV      FMTDPB+12,(R1) ;LOAD THE CYLINDER #
922 020650 113761 005551 000003   MOVB    FMTDPB+11,3(R1) ;LOAD THE TRK NUMBER
923 020656 113761 001346 000002   MOVB    SAVSEC,2(R1)   ;LOAD THE SECTOR NUMBER
924 020664 000207          4$:  RTS      PC          ;EXIT
925 020666 104401 001205   5$:  TYPE     ,SCLRF
926 020672 104401 040112   TYPE     ,MESG3
927 020676 104401 040373   TYPE     ,MHALT
928 020702 000000          HALT

```

```

929 020704 000137 007020          JMP      BEGIN2          ;RESTART IF DESIRED ?
930
931          ;THIS ROUTINE SORT THE USTAB IN ASCENDING ORDER
932          ;CALL SEQ      JSR      'PC, SORT2
933
934 020710          SORT2:
935          :      MOV      #USTAB+8.,R1      ;TOP POINTER
936 020710 011501          :      MOV      (R5),R1      ;TOP POINTER
937 020712 012702 000175          :      MOV      #125.,R2      ;TOTAL 126 ELEMENTS
938 020716 010103          :      MOV      R1,R3      ;LOCATE THE BOTTOM POINTER
939 020720 062703 000764          :      ADD      #500.,R3      ;254-4) X 2
940          :      MOV      #USTAB+508.,R3      ;THE LAST ELEMENT
941 020724 022711 177777          :      CMP      #-1,(R1)      ;THE TABLE IS EMPTY ?
942 020730 001444          :      BEQ      4$      ;QUICK EXIT
943 020732 021161 000004          1$:      CMP      (R1),4(R1)      ;COMPARE THE CYLINDER NUMBER
944 020736 101013          :      BHI      2$      ;SWITCH THE TWO ELEMENT
945 020740 103426          :      BLO      3$      ;INCREMENT POINTER
946 020742 126161 000003 000007          :      CMPB     3(R1),7(R1)      ;COMPARE THE TRK NUMBER
947 020750 101006          :      BHI      2$      ;SWITCH ELEMENT
948 020752 103421          :      BLO      3$      ;INCREMENT POINTER
949 020754 126161 000002 000006          :      CMPB     2(R1),6(R1)      ;COMPARE THE SECTOR NUMBER
950 020762 101001          :      BHI      2$      ;SWITCH ELEMENT
951 020764 000414          :      BR       3$      ;INCREMENT POINTER
952 020766 011146          2$:      MOV      (R1),-(SP)      ;SAVE THE N BLOCK
953 020770 016146 000002          :      MOV      2(R1),-(SP)      ;INTO STACK
954 020774 016111 000004          :      MOV      4(R1),(R1)      ;SWITCH THE N+1 BLOCK TO N BLOCK
955 021000 016101 000006 000002          :      MOV      6(R1),2(R1)      ;
956 021006 012661 000006          :      MOV      (SP)+,6(R1)      ;LOAD THE N+1 BLOCK
957 021012 012661 000004          :      MOV      (SP)+,4(R1)      ;
958 021016 062701 000004          3$:      ADD      #4,R1      ;UPDATE THE TOP POINTER
959 021022 020103          :      CMP      R1,R3      ;REACH THE BOTTOM ?
960 021024 001342          :      BNE      1$      ;BR IF NOT
961 021026 005302          :      DEC      R2      ;DECREMENT ONE SORT COUNT
962 021030 001404          :      BEQ      4$      ;BR IF ALL DONE
963 021032 162703 000004          :      SUB      #4,R3      ;ASJUST THE BOTTOM POINTER
964          :      MOV      #USTAB+8.,R1      ;RESET TOP POINTER
965 021036 011501          :      MOV      (R5),R1      ;RESET THE TOP POINTER
966 021040 000734          :      BR       1$      ;BR IF NOT ALL DONE
967 021042          4$:
968 021042 062705 000002          :      ADD      #2,R5      ;ADJUST RETURN ADDRESS
969 021046 000205          :      RTS      R5      ;EXIT
970          :      RTS      PC      ;EXIT
971
972          ;THIS ROUTINE TEXT THE LAST TRACK CYL 822, LAST TRACK
973          ;SECTORS 0,2,4,6,8 16 BIT MFG
974          ;SECTORS 1,3,5,7,9 18 BIT MFG
975          ;SECTORS 10,12,14,16,.....30, 16 BIT USER
976          ;SECTORS 11,13,15,17,.....31 18 BIT USER
977
978 021050 012700 045774          TEXT:  MOV      #BUFF,R0      ;BUFFER ADDRESS
979 021054 012701 020100          :      MOV      #<258.*32.>,R1      ;TOTAL WORD COUNT
980 021060 012702 177777          :      MOV      #-1,R2      ;SET ALL LOCATIONS TO -1
981 021064 010220          1$:      MOV      R2,(R0)+      ;FULL THE BUFFER
982 021066 005301          :      DEC      R1      ;ALL DONE ?
983 021070 001375          :      BNE      1$      ;LOOP, IF NOT
984 021072 013746 001214          :      MOV      $TESTN,-(SP)      ;SAVE TESTN,BEGCYL,BEGTRK
985 021076 013746 001330          :      MOV      BEGCYL,-(SP)      ;

```

```

986 021102 013746 001334      MOV      BEGTRK,-(SP)      ;
987 021106 013746 001362      MOV      MAXSEC,-(SP)    ;SAVE FLAG MAXSEC AND SEC30
988 021112 013746 001360      MOV      SEC30,-(SP)    ;
989 021116 012737 000037 001362  MOV      #37,MAXSEC      ;SET MAX 31 SECTORS
990 021124 012737 177777 001360  MOV      #-1,SEC30      ;ALWAYS IN 16 BIT MODE
991 021132 012737 001466 001330  MOV      #822.,BEGCYL   ;LOAD LAST CYLINDER
992 021140 013737 001324 001334  MOV      LSTRK,BEGTRK   ;LAST TRACK
993 021146 004737 017556      JSR      PC,SETHDR      ;SET UP THE HEAD FIELD IN THE BUFFER
994 021152 012637 001360      MOV      (SP)+,SFC30    ;RESTORE SEC30 AND MAXSEC
995 021156 012637 001362      MOV      (SP)+,MAXSEC   ;
996 021162 012637 001334      MOV      (SP)+,BEGTRK  ;
997 021166 012637 001330      MOV      (SP)+,BEGCYL  ;
998 021172 012637 001214      MOV      (SP)+,$TESTN  ;
999 021176 012701 001414      MOV      #BAD16,R1     ;LOAD SERIAL NUMBER TO EVERY SECTOR
1000 021202 012702 046000      MOV      #BUF+4,R2     ;BUFFER LOCATION + 4
1001 021206 012703 000040      MOV      #32.,R3      ;TOTAL 32 SECTORS
1002 021212 011112          MOV      (R1),(R2)     ;1 ST SN #
1003 021214 016162 000002 000002 2$:  MOV      2(R1),2(R2)   ;2 ND SN #
1004 021222 005062 000004      CLR      4(R2)        ;THIRD WORD = 0
1005 021226 005062 000006      CLR      6(R2)        ;ID = DATA PACK
1006 021232 062702 001004      ADD      #516.,R2     ;ADVANCE TO NEXT SECTOR
1007 021236 005303          DEC      R3           ;DECREMENT ONE SECTOR COUNT
1008 021240 001364          BNE      2$          ;BR IF NOT DONE
1009 021242 005046          CLR      -(SP)        ;LOAD TABLE INTO SECTORS
1010 021244 005046          CLR      -(SP)        ;DUMMY PAIRS
1011 021246 005046          CLR      -(SP)
1012 021250 012746 001414      MOV      #BAD16,-(SP)  ;TABLE ADDRESS
1013 021254 012746 046000      MOV      #BUF+4,-(SP) ;BUFFER ADDRESS
1014 021260 012746 000005      MOV      #5,-(SP)    ;SECTOR COUNT
1015 021264 012746 002420      MOV      #BAD18,-(SP) ;TABLE ADDRESS
1016 021270 012746 047004      MOV      #BUF+4+516.,-(SP) ;BUFFER ADDRESS
1017 021274 012746 000005      MOV      #5,-(SP)    ;SECTOR COUNT
1018 021300 012746 003424      MOV      #USTAB,-(SP) ;USER DEFINED TABLE
1019 021304 012746 060050      MOV      #BUF+4+<516.*10.>,-(SP) ;BUFFER ADDRESS
1020 021310 005737 001360      TST      SEC30
1021 021314 100402          BMI      3$          ;BR IF IN 16 BIT MODE
1022 021316 012716 061054      MOV      #BUF+4+<516.*11.>,(SP) ;RESET THE BUFFER ADDRESS
1023 021322 012746 000013 3$:  MOV      #11.,-(SP)   ;SECTOR COUNT
1024 021326 012701 004430      MOV      #USSAV,R1    ;TABLE ADDRESS IN R1
1025 021332 012702 061054      MOV      #BUF+4+<516.*11.>,R2 ;BUFFER ADDRESS IN R2
1026 021336 005737 001360      TST      SEC30      ;BR IF IN 16 BIT MODE
1027 021342 100402          BMI      4$
1028 021344 012702 060050      MOV      #BUF+4+<516.*10.>,R2 ;BUFFER ADDRESS FOR 18 BIT MODE
1029 021350 012703 000013 4$:  MOV      #11.,R3     ;SECTOR COUNT IN R3
1030 021354 010105          MOV      R1,R5      ;TEMPOR STORAGE
1031 021356 012704 000400 5$:  MOV      #256.,R4    ;WORD COUNT = DATA FIELD OF ONE SECTOR
1032 021362 012122 6$:  MOV      (R1)+,(R2)+ ;LOAD TABLE INTO SECTOR DATA FIELD
1033 021364 005304          DEC      R4         ;ALL DONE ?
1034 021366 001375          BNE      6$         ;BR IF NOT
1035 021370 010501          MOV      R5,R1     ;RELOAD THE TABLE ADDRESS
1036 021372 062702 001010      ADD      #520.,R2   ;SKIP ONE SECTOR
1037 021376 005303          DEC      R3        ;DECREMENT ONE SECTOR COUNT
1038 021400 001366          BNE      5$        ;BR IF NOT DONE
1039 021402 012603          MOV      (SP)+,R3  ;RETRIEVE NEXT SET OF SECTOR #
1040 021404 012602          MOV      (SP)+,R2  ;BUFFER ADDRESS
1041 021406 012601          MOV      (SP)+,R1  ;TABLE ADDRESS
1042 021410 010105          MOV      R1,R5     ;TEMPOR STORAGE

```

```

1043 021412 001361          BNE      5$          ;BR IF NOT DUMMY PAIR
1044 021414 000207          RTS       PC          ;EXIT
1045
1046          ;THIS ROUTINE RESET:
1047          ;RESET THE BIT 14/15 OF THE 1 ST HEADER WORD OF THE BAD SPOT
1048          ;CALLING SEQ:          JSR       PC,RESET
1049
1050 021416 013737 005552 045764 RESET:  MOV     FMTDPB+12,RBUF ;CYLINDER ADDRESS
1051 021424 001006          BNE      1$          ;BR IF NOT CN TRACK 0
1052 021426 122737 000001 005551      CMPB    #1,FMTDPB+11 ;BR IF NOT CYLO,TRK 0 OR 1
1053 021434 103402          BLO     1$          ;BR IF NOT
1054 021436 000137 017134          JMP     REJCT2      ;OTHERWISE, ILLEGAL BAD SECTORS
1055 021442 000240          1$:      NOP
1056 021444 113737 005551 045767      MOVB   FMTDPB+11,RBUF+3 ;TRACK ADDRESS
1057 021452 113737 001346 045766      MOVB   SAVSEC,RBUF+2  ;SECTOR ADDRESS
1058 021460 053737 001410 045764      BIS    HEADX,RBUF    ;SET MFG BIT,RESET USER BIT
1059 021466 113737 001222 005630      MOVB   DRIVE,FMTX    ;SETUP THE DPB FOR OPERATION
1060 021474 012737 177776 005634      MOV    #-2,FMTX+4    ;WORD COUNT =2
1061 021502 112737 000163 005632      MOVB   #WRTHD,FMTX+2 ;WRITE HEAD AND DATA COMMAND
1062 021510 012737 045764 005636      MOV    #RBUF,FMTX+6  ;BUFFER ADDRESS
1063 021516 113737 001346 005640      MOVB   SAVSEC,FMTX+10;SECTOR ADDRESS
1064 021524 113737 005551 005641      MOVB   FMTDPB+11,FMTX+11;TRACK ADDRESS
1065 021532 013737 005552 005642      MOV    FMTDPB+12,FMTX+12;CYLINDER ADDRESS
1066 021540 052737 010000 045764      BIS    #FMT,RBUF    ;ASSUM 16 BIT MODE
1067 021546 005737 001360          TST    SEC30        ;IN 16 BIT MODE ?
1068 021552 100403          BMI     2$          ;BR IF IT IS
1069 021554 042737 010000 045764      BIC    #FMT,RBUF    ;RESET THE FMT BIT
1070 021562 004037 030622          2$:      JSR    RC,RM05     ;CALL THE DRIVER
1071 021566 005630          FMTX
1072 021570 000774          BR     2$          ;BR IF QUEUE FAILS
1073 021572 005737 005646          3$:      TST    FMTX+16  ;DONE ?
1074 021576 001775          BEQ    3$          ;BR IF NOT
1075 021600 000207          RTS     PC          ;EXIT
1076
1077          ;THE ROUTINE INSERT:
1078          ;INSERT A BAD SECTOR ADDRESS INTO ONE OF THE USTAB,USSAV,BAD16,BAD18 TABLES
1079          ;CALLING SEQ
1080          ;          JSR    R5,INSERT
1081          ;          TABLE + 8.
1082
1083 021602 010446          INSERT: MOV    R4,-(SP) ;SAVE THE R4
1084 021604 012746 000176      MOV    #126,-(SP) ;ENTRIES OF THE TABLE
1085 021610 011504          MOV    (R5),R4    ;TABLE ADDRESS + 8 BYTES
1086 021612 022714 177777          1$:      CMP    #-1,(R4)  ;AN OPEN ENTRY ?
1087 021616 001420          BEQ    3$          ;BR IF SO
1088 021620 023714 001364          CMP    DS.CYL,(R4) ;CHECK IF THE BAD SECTOR HAS RECORDED
1089 021624 001010          BNE    2$          ;BR IF NOT
1090 021626 123764 001366 000003      CMPB   DS.TRK,3(R4) ;CYL AND TRK MATCH ?
1091 021634 001004          BNE    2$          ;BR IF NOT
1092 021636 123764 001346 000002      CMPB   SAVSEC,2(R4) ;CYL ,TRK AND SEC MATCH ?
1093 021644 001415          BEQ    4$          ;BR IF SO
1094 021646 005316          2$:      DEC    (SP)      ;DECREMENT THE ENTRY COUNT
1095 021650 001413          BEQ    4$          ;BR IF EXHAUST
1096 021652 062704 000004          ADD    #4,R4      ;LOCATE THE NEXT ENTRY
1097 021656 000755          BR     1$          ;BR BACK
1098 021660 013714 001364          3$:      MOV    DS.CYL,(R4) ;LOAD INTO TABLE
1099 021664 113764 001366 000003      MOVB   DS.TRK,3(R4) ;LOAD THE TRACK NUMBER

```

```

1100 021672 113764 001346 000002      MOV#B  SAVSEC,2(R4)      ;LOAD THE SECTOR NUMBER
1101 021700 062706 000002      4$:  ADD  #2,SP          ;CLEAR UP THE STACK
1102 021704 012604              MOV  (SP)+,R4          ;RESTORE R4
1103 021706 062705 000002      ADD  #2,R5            ;ASJUST THE RETURN ADDRESS
1104 021712 000205              RTS   R5              ;EXIT
1105
1106 021714 104401 041236      FUNCT1: TYPE  ,MESG5
1107 021720 104401 001205      TYPE  ,%CRLF          ;CR-LF
1108 021724 012737 177777 001364 1$:  MOV  #-1,DS.CYL      ;RESET THE INPUT VALUES
1109 021732 012737 177777 001366      MOV  #-1,DS.TRK      ;
1110 021740 012737 177777 001346      MOV  #-1,SAVSEC      ;
1111 021746 004437 020454      JSR  R4,PARENT       ;ENTER THE VALUE FROM THE KEYBOARD
1112 021752 005702              TABLE2
1113
1114 021754 005737 001364              TST  DS.CYL          ;UPDATE VALUES ?
1115 021760 100412              BMI  2$             ;BR IF NOT
1116 021762 005737 001366              TST  DS.TRK          ;
1117 021766 100407              BMI  2$             ;
1118 021770 005737 001346              TST  SAVSEC          ;
1119 021774 100404              BMI  2$             ;
1120 021776 004537 021602      JSR  R5,INSERT       ;INSERT INTO THE BAD16 TABLE
1121 022002 001424              BAD16+8.
1122 022004 000747              BR   1$             ;NEXT SET
1123 022006 004537 020710      2$:  JSR  R5,SORT2      ;SORT THE TABLE
1124 022012 001424              BAD16+8.
1125
1126 022014 104401 041321      TYPE  ,MESG6
1127 022020 104401 001205      TYPE  ,%CRLF          ;CR-LF
1128 022024 012737 177777 001364 3$:  MOV  #-1,DS.CYL      ;RESET THE INPUT VALUES
1129 022032 012737 177777 001366      MOV  #-1,DS.TRK      ;
1130 022040 012737 177777 001346      MOV  #-1,SAVSEC      ;
1131 022046 004437 020454      JSR  R4,PARENT       ;INPUT VALUES FROM THE KEYBOARD
1132 022052 005726              TABLE3             ;18 BIT TABLE
1133
1134 022054 005737 001364              TST  DS.CYL          ;UPDATED VALUES ?
1135 022060 100412              BMI  4$             ;BR IF NOT
1136 022062 005737 001366              TST  DS.TRK          ;
1137 022066 100407              BMI  4$             ;
1138 022070 005737 001346              TST  SAVSEC          ;
1139 022074 100404              BMI  4$             ;
1140 022076 004537 021602      JSR  R5,INSERT       ;INSERT THE BAD SECTOR INTO TABLE BAD18
1141 022102 002430              BAD18+8.
1142 022104 000747              BR   3$             ;NEXT SET
1143 022106 004537 020710      4$:  JSR  R5,SORT2      ;SORT THE BAD18 TABLE
1144 022112 002430              BAD18+8.
1145 022114 000207              RTS   PC            ;EXIT FROM THE CALLING OF TABCAL
1146
1147 022116 012737 003434 022272  FUNCT2: MOV  #USTAB+8.,3$ ;SETUP TABLES FOR 16 BITS MODE
1148 022124 012737 003434 022302      MOV  #USTAB+8.,5$   ;
1149 022132 012737 004440 022372      MOV  #USSAV+8.,7$   ;
1150 022140 012737 004440 022402      MOV  #USSAV+8.,9$   ;
1151 022146 005737 001360              TST  SEC30          ;IN 16 BIT MODE ?
1152 022152 100414              BMI  1$             ;BR IS SO,OTHERWISE RELOAD THE TABLES
1153 022154 012737 003434 022372      MOV  #USTAB+8.,7$   ;SET UP TABLES FOR 18 BITS MODE
1154 022162 012737 003434 022402      MOV  #USTAB+8.,9$   ;
1155 022170 012737 004440 022272      MOV  #USSAV+8.,3$   ;
1156 022176 012737 004440 022302      MOV  #USSAV+8.,5$   ;

```

```

1157
1158 022204 104401 041404      1$:  TYPE      ,MESG7      ;MESSAGE FOR ENTER 16 BITS USR TABLE
1159 022210 104401 001205      TYPE      , $CRLF      ;CR-LF
1160 022214 012737 177777      001364 2$:  MOV        #-1,DS.CYL    ;RESET THE INPUT VALUES
1161 022222 012737 177777      001366      MOV        #-1,DS.TRK    ;
1162 022230 012737 177777      001346      MOV        #-1,SAVSEC    ;
1163 022236 004437 020454      JSR        R4,PARENT     ;ENTER BAD SECTOR ADDRESS FROM THE KEYBOARD
1164 022242 005702      TABLE2
1165 022244 005737 001364      TST        DS.CYL        ;UPDATED INPUT VALUE ?
1166 022250 100412      BMI        4$            ;BR IF NOT
1167 022252 005737 001366      TST        DS.TRK        ;
1168 022256 100407      BMI        4$            ;
1169 022260 005737 001346      TST        SAVSEC        ;
1170 022264 100404      BMI        4$            ;
1171 022266 004537 021602      JSR        R5,INSERT     ;INSERT THE BAD SECTOR INTO TABLE
1172 022272 000000      3$:  .WORD      0            ;TABLE ADDRESS
1173 022274 000747      BR         2$            ;ENTER NEXT BAD SECTOR ADDRESS
1174 022276 004537 020710      4$:  JSR        R5,SORT2    ;SORT THE TABLE
1175 022302 000000      5$:  .WORD      0            ;TABLE ADDRESS
1176
1177 022304 104401 041467      TYPE      ,MESG8      ;MESSAGE FOR 18 BIT USR TABLE
1178 022310 104401 001205      TYPE      , $CRLF      ;CR-LF
1179 022314 012737 177777      001364 6$:  MOV        #-1,DS.CYL    ;RESET THE INPUT VALUES
1180 022322 012737 177777      001366      MOV        #-1,DS.TRK    ;
1181 022330 012737 177777      001346      MOV        #-1,SAVSEC    ;
1182 022336 004437 020454      JSR        R4,PARENT     ;ENTER BAD SECTOR FORM KEYBOARD
1183 022342 005726      TABLE3
1184
1185 022344 005737 001364      TST        DS.CYL        ;UPDATED INPUT VALUE ?
1186 022350 100412      BMI        8$            ;BR IF NOT
1187 022352 005737 001366      TST        DS.TRK        ;BR IF NOT UPDATE
1188 022356 100407      BMI        8$            ;
1189 022360 005737 001346      TST        SAVSEC        ;
1190 022364 100404      BMI        8$            ;BR IF NOT UPDATED
1191 022366 004537 021602      JSR        R5,INSERT     ;INSERT THE BAD SECTOR INTO TABLE
1192 022372 000000      7$:  .WORD      0            ;
1193 022374 000747      BR         6$            ;BR TO ENTER NEXT SECTOR
1194 022376 004537 020710      8$:  JSR        R5,SORT2    ;SORT THE TABLE
1195 022402 000000      9$:  .WORD      0            ;TABLE ADDRESS
1196 022404 012737 177777      0C1406  MOV        #-1,EDIT      ;UPDATE BAD SECTOR FILE AT END OF PASS
1197 022412 000207      RTS        PC            ;EXIT FORM THE TABCAL CALLING
1198
1199 022414 010146      FUNCT3: MOV        R1,-(SP)    ;SAVE R1
1200 022416 012701 001414      MOV        #BAD16,R1     ;ADDRESS OF THE TABLE
1201 022422 104401 041552      TYPE      ,MESG9      ;MSG TO TYPE THE 16 BIT MFG
1202 022426 104401 001205      1$:  TYPE      , $CRLF      ;CR-LF FIRST
1203 022432 012146      MOV        (R1)+,-(SP)    ;CALLING SEQ FOR TYPOC RT.
1204 022434 100402      BMI        2$            ;BR IF END OF TABLE
1205 022436 104402      TYPOC      ;TYPE THE VALUE
1206 022440 000772      BR         1$            ;LOOPING BACK
1207 022442 062706 000002      2$:  ADD        #2,SP        ;ADJUST THE RETURN ADDRESS
1208 022446 104401 041622      TYPE      ,MESG10     ;MSG TO TYPE 18 BIT MFG
1209 022452 012701 002420      MOV        #BAD18,R1     ;TABLE ADDRESS
1210 022456 104401 001205      3$:  TYPE      , $CRLF      ;
1211 022462 012146      MOV        (R1)+,-(SP)    ;CALLING SEQ FOR TYPOC RT.
1212 022464 100402      BMI        4$            ;BR IF END OF TABLE
1213 022466 104402      TYPOC      ;TYPE THE VALUE IN OCTAL

```

```

1214 022470 000772          BR      3$          ;LOOPING BACK
1215 022472 062706 000002  4$:   ADD      #2,SP      ;ADJUST THE RETURN ADDRESS
1216 022476 012601          5$:   MOV      (SP)+,R1    ;RESTORE THE R1
1217 022500 00C207          RTS      PC          ;EXIT FROM THE CALLING OF TABCAL
1218
1219 022502 010146          FUNCT4: MOV     R1,-(SP)    ;SAVE R1
1220 022504 010246          MOV     R2,-(SP)    ;SAVE R2
1221 022506 012701 003424          MOV     #USTAB,R1   ;SET 16 BIT MODE TABLE
1222 022512 012702 004430          MOV     #USSAV,R2
1223 022516 005737 001360          TST     SEC30       ;16 BIT MODE ?
1224 022522 100404          BMI     1$          ;BR IF SO
1225 022524 012702 003424          MOV     #USTAB,R2   ;OTHERWISE,RELOAD THE TABLE ADDRESS
1226 022530 012701 004430          MOV     #USSAV,R1
1227 022534 104401 041672          1$:   TYPE     ,MSG11    ;MSG FOR 16 BIT
1228 022540 104401 001205          2$:   TYPE     ,$CRLF
1229 022544 012146          MOV     (R1)+,-(SP) ;
1230 022546 100402          BMI     3$          ;BR IF END OF TABLE
1231 022550 104402          TYPOC
1232 022552 000772          BR      2$          ;
1233 022554 062706 000002          3$:   ADD      #2,SP      ;ADJUST STACK POINTER
1234 022560 104401 041742          TYPE     ,MSG12    ;MESSAGE FOR 18 BIT MODE
1235 022564 104401 001205          4$:   TYPE     ,$CRLF    ;CR-LF
1236 022570 012246          MOV     (R2)+,-(SP) ;
1237 022572 100402          BMI     5$          ;BR IF END OF TABLE
1238 022574 104402          TYPOC
1239 022576 000772          BR      4$          ;NEXT SECTOR ADDRESS
1240 022600 062706 000002          5$:   ADD      #2,SP      ;ADJUST THE STACK POINT
1241 022604 012602          MOV     (SP)+,R2    ;RESTORE R2,R1
1242 022606 012601          MOV     (SP)+,R1
1243 022610 000207          RTS      PC          ;EXIT FROM THE CALLING OF TABCAL
1244
1245 022612 010146          FUNCT5: MOV     R1,-(SP) ;SAVE R1,R2
1246 022614 010246          MOV     R2,-(SP)
1247 022616 005037 001420          CLR     BAD16+4.
1248 022622 005037 001422          CLR     BAD16+6.
1249 022626 012701 001424          MOV     #BAD16+8.,R1 ;START AT THE 5TH WORD
1250 022632 012702 000374          MOV     #252.,R2    ;WORD CTR
1251 022636 012721 177777          1$:   MOV     #-1,(R1)+ ;RESET TO -1
1252 022642 005302          DEC     R2          ;DECREMENT WORD CTR
1253 022644 001374          BNE     1$          ;BR IF NOT DONE
1254 022646 005037 002424          CLR     BAD18+4.
1255 022652 005037 002426          CLR     BAD18+6.
1256 022656 012701 002430          MOV     #BAD18+8.,R1 ;START AT THE 5TH WORD
1257 022662 012702 000374          MOV     #252.,R2    ;WORD CTR
1258 022666 012721 177777          2$:   MOV     #-1,(R1)+ ;RESET TO -1
1259 022672 005302          DEC     R2          ;DECREMENT WORD CTR
1260 022674 001374          BNE     2$          ;BR IF NOT DONE
1261 022676 012602          MOV     (SP)+,R2    ;RESTORE R2,R1
1262 022700 012601          MOV     (SP)+,R1
1263 022702 104401 042041          TYPE     ,MSG14    ;DONE MESSAGE
1264 022706 000207          RTS      PC          ;EXIT FROM THE TABCAL CALLING
1265
1266 022710 010146          FUNCT6: MOV     R1,-(SP) ;STORE R1,R2
1267 022712 010246          MOV     R2,-(SP)
1268 022714 005037 003430          CLR     USTAB+4
1269 022720 005037 003432          CLR     USTAB+6
1270 022724 012701 003434          MOV     #USTAB+8.,R1

```

```

1271 022730 012702 000374      MOV      #252.,R2      ;WORD CTR
1272 022734 012721 177777      1$:     MOV      #-1,(R1)+  ;RESET TO -1
1273 022740 005302              DEC      R2           ;DECREMENT WORD CTR
1274 022742 001374              BNE     1$           ;BR IF NOT DONE
1275 022744 005037 004434      CLR     USSAV+4
1276 022750 005037 004436      CLR     USSAV+6
1277 022754 012701 004440      MOV     #USSAV+8.,R1 ;TABLE ADDRESS
1278 022760 012702 000374      MOV     #252.,R2
1279 022764 012721 177777      2$:     MOV     #-1,(R1)+  ;RESET TO -1
1280 022770 005302              DEC     R2           ;DECREMENT WORD CTR
1281 022772 001374              BNE     2$           ;BR IF NOT DONE
1282 022774 012602              MOV     (SP)+,R2     ;RESTORE R1,R2
1283 022776 012601              MOV     (SP)+,R1
1284 023000 104401 042041      TYPE    ,MSG14       ;DONE MESSAGE
1285 023004 012737 177777 001406  MOV     #-1,EDIT     ;UPDATE BAD SECTOR FILE AT END OF PASS
1286 023012 000207              RTS     PC           ;EXIT FORM THE TABCAL CALLING
1287
1288 023014 104401 042012      FUNCT7: TYPE    ,MSG13 ;MESSAGE ENTER SECTOR #
1289 023020 104411      1$:     RDLIN              ;READ THE SECTOR NUMBER
1290 023022 012601              MOV     (SP)+,R1     ;ADDRESS OF READ-IN BUFFER
1291 023024 012702 000040      MOV     #32.,R2      ;MAX VALUE
1292 023030 004537 023462      JSR     R5,CK.DIG    ;CHECK THE READ-IN VALUE
1293 023034 023050              2$
1294 023036 023050              2$
1295 023040 023020              1$
1296 023042 023050              2$
1297 023044 023050              2$
1298 023046 023050              2$
1299 023050 006302      2$:     ASL     R2           ;FOUND THE WORD INDEX
1300 023052 016201 006132      MOV     ADRTBL(R2),R1 ;ADDRESS OF THE BUFFER
1301 023056 012702 000402      MOV     #258.,R2     ;WC FOR WHOLE SECTOR INCLUDING HEADER
1302 023062 104401 001205      3$:     TYPE    ,%CRLF   ;CR-LF
1303 023066 012705 000010      MOV     #10,R5       ;8 WORDS ON A LINE
1304 023072 012146      4$:     MOV     (R1)+,-(SP) ;CONTENTS OF THE BUFFER
1305 023074 104402              TYPOC
1306 023076 104401 036425      TYPE    ,BLNKS2     ;TYPE 2 SPACES
1307 023102 005302              DEC     R2           ;DECREMENT WORD CTR
1308 023104 001403              BEQ     5$           ;BR IF FINISH
1309 023106 005305              DEC     R5           ;START ANOTHER LINE ?
1310 023110 001370              BNE     4$           ;BR IF NOT
1311 023112 000763              BR      3$           ;BR IF SO
1312 023114 104401 001205      5$:     TYPE    ,%CRLF   ;CR-LF
1313 023120 000207              RTS     PC           ;EXIT FROM TABCAL CALLING
1314
1315 023122 104401 042051      FUNCT8: TYPE    ,MSG15 ;RESET ALL INPUT VALUES
1316 023126 012737 177777 001364 1$:     MOV     #-1,DS.CYL ;:TRACK NUMBER
1317 023134 012737 177777 001366      MOV     #-1,DS.TRK  ;:BYTE NUMBER
1318 023142 012737 047301 001346      MOV     #20161.,SAVSEC ;ENTER THE VALUE FROM KEYBOARD
1319 023150 004437 020454      JSR     R4,PARENT
1320 023154 005752      TABLE4
1321
1322 023156 005737 001364      TST     DS.CYL       ;UPDATED ?
1323 023162 100455              BMI     6$           ;EXIT IF NOT
1324 023164 005737 001366      TST     DS.TRK       ;BR IF NOT UPDATED
1325 023170 100452              BMI     6$           ;
1326 023172 005737 001346      TST     SAVSEC       ;BR TO EXIT IF NEG
1327 023176 100447              BMI     6$           ;EXIT

```



```

1328 023200 022737 047301 001346      CMP      #20161.,SAVSEC ;TOO LARGE ?
1329 023206 003443                      BLE      6$           ;BR IF SO,EXIT
1330 023210 010446                      MOV      R4,-(SP)    ;SAVE R4
1331 023212 013746 001346              MOV      SAVSEC,-(SP) ;SAVE SAVSEC
1332 023216 005004                      CLR      R4         ;INDEX STARTS FROM 0
1333 023220 023764 001346 006332 2$:  CMP      SAVSEC,BYTE16(R4) ;LOCATE THE SECTOR ADDRESS
1334 023226 101403                      BLOS    3$         ;BR IF FOUND
1335 023230 062704 000002              ADD      #2,R4     ;TRY NEXT SECTOR
1336 023234 000771                      BR       2$         ;LOOPING
1337 023236 006204                      3$:      ASR      R4         ;SECTOR ADDRESS
1338 023240 010437 001346              MOV      R4,SAVSEC ;16 BIT MODE BAD SECTOR ADD
1339 023244 004537 021602              JSR      R5,INSERT ;INSERT INTO TABLE
1340 023250 001424                      BAD16+8.
1341 023252 012637 001346              MOV      (SP)+,SAVSEC ;LOCATE SECTOR FOR 18 BIT
1342 023256 005004                      CLR      R4         ;RESET INDEX VALUE
1343 023260 023764 001346 006432 4$:  CMP      SAVSEC,BYTE18(R4) ;USE TABLE 18 BIT
1344 023266 101403                      BLOS    5$         ;BR IF FOUND
1345 023270 062704 000002              ADD      #2,R4     ;TRY NEXT SECTOR IF NOT FOUND
1346 023274 000771                      BR       4$         ;LOOPIN BACK
1347 023276 006204                      5$:      ASR      R4         ;SECTOR ADDRESS
1348 023300 010437 001346              MOV      R4,SAVSEC ;
1349 023304 004537 021602              JSR      R5,INSERT ;INSET INTO TABLE
1350 023310 002430                      BAD18+8.
1351 023312 012604                      MOV      (SP)+,R4   ;RESTORE R4
1352 023314 000704                      BR       1$         ;LOOPING BACK
1353 023316 004537 020710              6$:      JSR      R5,SORT2 ;SORT BOTH TABLE ANYWAY
1354 023322 001424                      BAD16+8.
1355 023324 004537 020710              JSR      R5,SORT2 ;
1356 023330 002430                      BAD18+8.
1357 023332 000207                      RTS      PC         ;RETURN TO SERVICE LOOP

```

```

1358
1359 ;THIS ROUTINE IS USED TO CHECK IF AN
1360 ;ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.
1361 ;CALL
1362 ;
1363 ;      MOV      #ADR,R1 ;ADDRESS OF ASCII CHARACTER
1364 ;      JSR      R5,CK.OCT ;CHECK THE CHARACTER
1365 ;      RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
1366 ;      RETURN2 ;CHARACTER IS IN R2 AS A
1367 ;      ;OCTAL DIGIT

```

```

1368 023334 121127 000060      CK.OCT: CMPB    (R1),#'0 ;LESS THAN ZERO?
1369 023340 103407                      BLO     1$         ;YES
1370 023342 121127 000067      CMPB    (R1),#'7 ;GREATER THAN SEVEN?
1371 023346 101004                      BHI     1$         ;YES
1372 023350 111102                      MOVB   (R1),R2    ;GET THE CHARACTER
1373 023352 042702 177770      BIC    #'C7,R2   ;STRIP AWAY THE ASCII
1374 023356 005725                      TST    (R5)+     ;ADJUST FOR RETURN
1375 023360 000205      1$:      RTS      R5         ;RETURN

```

```

1376
1377 ;THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
1378 ;AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
1379 ;CALL
1380 ;
1381 ;      MOV      #ADR,R1 ;ADDRESS OF ASCII CHARACTER
1382 ;      JSR      R5,CK.DEC ;CHECK THE CHARACTER
1383 ;      RETURN1 ;NOT BETWEEN 0 AND 9
1384 ;      RETURN2 ;BETWEEN 0 AND 9
;      ;R2 = DIGIT

```

```

1385
1386 023362 121127 000060      CK.DEC: CMPB    (R1),#'0      ;LESS THAN ZERO?
1387 023366 103407              BLO     1$                  ;YES
1388 023370 121127 000071      CMPB    (R1),#'9      ;GREATER THAN NINE?
1389 023374 101004              BHI     1$                  ;YES
1390 023376 111102              MOVB   (R1),R2          ;GET THE CHARACTER
1391 023400 042702 000060      BIC     #'0,R2        ;STRIP AWAY THE ASCII
1392 023404 005725              TST    (R5)+           ;ADJUST FOR RETURN
1393 023406 000205      1$:    RTS     R5        ;RETURN
1394
1395      ;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
1396      ;DETERMINE WHAT IT IS.
1397      ;CALL
1398      ;
1399      ;
1400      ;
1401      ;
1402      ;
1403      ;
1404      ;
1405      ;
1406      ;
1407      ;
1408 023410 105711      CK.CHR: TSTB   (R1)        ;'CARRIAGE RETURN'?
1409 023412 001417              BEQ    3$                ;YES
1410 023414 121127 000054      CMPB   (R1),#',        ;'COMMA'?
1411 023420 001413              BEQ    2$                ;YES
1412 023422 121127 000056      CMPB   (R1),#'.        ;'PERIOD'?
1413 023426 001407              BEQ    1$                ;YES
1414 023430 004537 023362      JSR    R5,CK.DEC       ;'DIGIT'?
1415 023434 000410              BR     4$                ;NO
1416 023436 004537 023334      JSR    R5,CK.OCT       ;OCTAL ?
1417 023442 005725              TST   (R5)+            ;DIGIT BETWEEN 8-9
1418 023444 005725              TST   (R5)+            ;DIGIT BETWEEN 0-7
1419 023446 005725      1$:    TST   (R5)+            ;PERIOD
1420 023450 005725      2$:    TST   (R5)+            ;COMMA
1421 023452 005725      3$:    TST   (R5)+            ;CARRIAGE RETURN
1422 023454 005201              INC    R1               ;MOVE POINTER TO NEXT CHARACTER
1423 023456 011505      4$:    MOV    (R5),R5      ;UNKNOWN CHARACTER
1424 023460 000205              RTS     R5              ;RETURN
1425
1426      ;THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
1427      ;CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
1428      ;CALL
1429      ;
1430      ;
1431      ;
1432      ;
1433      ;
1434      ;
1435      ;
1436      ;
1437      ;
1438      ;
1439 023462 010446      CK.DIG: MOV    R4,-(SP)    ;SAVE R4
1440 023464 010346              MOV    R3,-(SP)        ;SAVE R3
1441 023466 010246              MOV    R2,-(SP)        ;SAVE THE MAX. SIZE ON THE STACK

```

```

1442 023470 005002          CLR      R2          ;START WITH 0
1443 023472 005003          CLR      R3
1444 023474 005004          CLR      R4
1445 023476 004537 023410   JSR      R5,CK.CHR   ;CHECK ONE CHARACTER
                          6$          ;ILLEGAL CHARACTER
                          9$          ;CARRIAGE RETURN
                          6$          ;'.'
                          7$          ;'..'
                          1$          ;DIGIT 0-7
                          1$          ;DIGIT 8-9
1446 023516 062705 000004   1$:     ADD      #4,R5   ;STEP RETURN POINTER PAST 'CR' & 'PERIOD' RETURNS
1447 023522 006303         2$:     ASL      R3          ;INPUT NUMBER *2
1448 023524 010346         MOV      R3,-(SP)   ;SAVE *2
1449 023526 006303         ASL      R3          ;*4
1450 023530 006303         ASL      R3          ;*8
1451 023532 062603         ADD      (SP)+,R3   ;(*2)+(*8) = *10
1452 023534 060203         ADD      R2,R3     ;UPDATE THE INPUT NUMBER
1453 023536 004537 023410   JSR      R5,CK.CHR   ;CHECK ONE CHARACTER
                          8$          ;ILLEGAL CHARACTER
                          5$          ;CARRIAGE RETURN
                          4$          ;'.'
                          3$          ;'..'
                          2$          ;DIGIT 0-7
                          2$          ;DIGIT 8-9
1454 023556 105711         3$:     TSTB     (R1)   ;DOES A 'CR' FOLLOW THE 'PERIOD'
1455 023560 001010         BNE      8$          ;BR IF NOT
1456 023562 005724         TST      (R4)+     ;INCREMENT THE RETURN
1457 023564 005724         4$:     TST      (R4)+     ;INCREMENT THE RETURN
1458 023566 005724         5$:     TST      (R4)+     ;INCREMENT THE RETURN
1459 023570 020316         CMP      R3,(SP)   ;CHECK THE MAGNITUDE OF THE NUMBER
1460 023572 002004         BGE      9$          ;BR IF ENTERED NUMBER TOO LARGE
1461 023574 000402         BR       8$          ;BYPASS INCREMENT
1462 023576 005725         6$:     TST      (R5)+     ;INCREMENT RETURN PAST INVALID RETURN
1463 023600 005725         7$:     TST      (R5)+     ;INCREMENT RETURN
1464 023602 060405         8$:     ADD      R4,R5   ;SETUP RETURN POINTER
1465 023604 010302         9$:     MOV      R3,R2   ;ENTERED VALUE
1466 023606 005726         TST      (SP)+     ;CLEAN MAX. SIZE OFF OF STACK
1467 023610 012603         MOV      (SP)+,R3   ;RESTORE R3
1468 023612 012604         MOV      (SP)+,R4   ;RESTORE R4
1469 023614 011505         MOV      (R5),R5   ;GET RETURN ADDRESS
1470 023616 000205         RTS      R5       ;RETURN

```

1471
1472 .SBTTL MACRO ROUTINES

1473
1485 .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
$ERROR:
023620      104407      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
023620      010137      001370      MOV      R1,DDRIVE      ;;DRIVE ADDRESS IF DRIVE ERROR CALL
023622      010337      001372      MOV      R3,ATTN      ;;ATTENTION REGISTER CONTENTS
023626      010337      001372      MOV      FMTDPB+12,DS.CYL      ;;CURRENT CYLINDER ADDRESS
023632      013737      005552      001364      MOV      FMTDPB+11,DS.TRK      ;;REQUESTED TRACK ADDRESS
023640      113737      005551      001366      MOV      RM.REG+RMBA      ;;NON-ZERO BUFFER ADDRESS ?
023646      005737      005564      TST      7$      ;;BR IF NO BUFFER ADDRESS
023652      001406      BEQ      7$
023654      013746      005564      MOV      RM.REG+RMBA,-(SP)      ;;BUFFER ADDRESS
023660      162716      000002      SUB      #2,(SP)      ;;DECREMENT THE ADDRESS
023664      013637      001140      MOV      @$(SP)+,$GDDAT      ;;GET THE BUFFER WORD WHICH DIDN'T COMPARE
023670      105237      001117      7$:      INCB      $ERFLG      ;;SET THE ERROR FLAG
023674      001775      BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
023676      013777      001116      155252      MOV      $STNM,@DISPLAY      ;;DISPLAY TEST NUMBER AND ERROR FLAG
023704      032777      002000      155242      BIT      #BIT10,@SWR      ;;BELL ON ERROR?
023712      001402      BEQ      1$      ;;NO - SKIP
023714      104401      001200      TYPE      ,SBELL      ;;RING BELL
023720      005237      001126      1$:      INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
023724      011637      001132      MOV      (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
023730      162737      000002      001132      SUB      #2,$ERRPC
023736      117737      155170      001130      MOV      @$ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
023744      032777      020000      155202      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
023752      001004      BNE      20$      ;;SKIP TYPEOUTS
023754      004737      024054      JSR      PC,TYPERR      ;;GO TO USER ERROR ROUTINE
023760      104401      001205      TYPE      ,$CRLF
023764      20$:      CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
023764      122737      000001      001230      BNE      2$      ;;NO,SKIP APT ERROR REPORT
023772      001007      MOV      $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
023774      113737      001130      024006      JSR      PC,$ATY4      ;;REPORT FATAL ERROR TO APT
024002      004737      024656      21$:      .BYTE      0
024006      000      .BYTE      0
024007      000      BR      22$      ;;APT ERROR LOOP
024010      000777      22$:      TST      @SWR      ;;HALT ON ERROR
024012      005777      155136      2$:      BPL      3$      ;;SKIP IF CONTINUE
024016      100002      HALT      ;;HALT ON ERROR!
024020      000000      CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
024022      104407      3$:      BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
024024      032777      001000      155122      BEQ      4$      ;;BR IF NO
024032      001402      MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
024034      013716      001124      4$:      TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
024040      005737      001176      BEQ      5$      ;;BR IF NONE
024044      001402      MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
024046      013716      001176      5$:
024052      000002      RTI      ;;RETURN
024052      000002

;
;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.
TYPERR: SAVREG      ;SAVE R0-R5
          CLR      R0      ;CLEAR R0 FOR ERROR NUMBER
          MOV      $ITEMB,R0      ;ERROR NUMBER
  
```

1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494 024054 104412
 1495 024056 005000
 1496 024060 113700 001130

1497	024064	005300			DEC	R0		:FORM INDEX FOR ERROR TABLE
1498	024066	006300			ASL	R0		
1499	024070	006300			ASL	R0		
1500	024072	006300			ASL	R0		
1501	024074	062700	006526	18:	ADD	#SERRTB,R0		:FORM ADDRESS
1502	024100	012037	024114		MOV	(R0)+,28		:GET ERROR MESSAGE (EM) POINTER
1503	024104	001404			BEQ	38		:BR IF THERE ISN'T ONE
1504	024106	104401	001205		TYPE	,8CRLF		:CARRIAGE RETURN - LINE FEED
1505	024112	104401			TYPE			
1506	024114	000000		28:	.WORD	0		:EM POINTER GOES HERE
1507	024116	012037	024132	38:	MOV	(R0)+,48		:PICK UP DATA HEADER (DH) POINTER
1508	024122	001404			BEQ	58		:BR IF NONE
1509	024124	104401	001205		TYPE	,8CRLF		:CARRIAGE RETURN-LINE FEED
1510	024130	104401			TYPE			
1511	024132	000000		48:	.WORD	0		:DH POINTER GOES HERE
1512	024134	012001		58:	MOV	(R0)+,R1		:PICKUP DATA TABLE (DT) POINTER
1513	024136	001460			BEQ	158		:BR IF NONE
1514	024140	005005			CLR	R5		:SET INDENT SWITCH
1515	024142	012000			MOV	(R0)+,R0		:DATA FORMAT (DF) POINTER
1516	024144	012002			MOV	(R0)+,R2		:NUMBER OF DH'S TO TYPE
1517	024146	001451			BEQ	148		:BR IF DH NUMBER IS 0
1518	024150	005105			COM	R5		:NO INDENT
1519	024152	104401	001205		TYPE	,8CRLF		:CARRIAGE RETURN-LINE FEED
1520	024156	112003		68:	MOVB	(R0)+,R3		:NUMBER OF DATA WORDS TO TYPE
1521	024160	112004			MOVB	(R0)+,R4		:AND HOW TO TYPE THEM
1522	024162	006004		78:	ROR	R4		:OCTAL OR DECIMAL?
1523	024164	103403			BCS	88		:DECIMAL
1524	024166	013146			MOV	@(R1)+,-(SP)		:SAVE @(R1)+ FOR TYPEOUT
	024170	104402			TYPE	OC		:GO TYPE--OCTAL ASCII(ALL DIGITS)
1525	024172	000402			BR	98		
1526	024174			88:				
	024174	013146			MOV	@(R1)+,-(SP)		:SAVE @(R1)+ FOR TYPEOUT
	024176	104405			TYPDS			:GO TYPE--DECIMAL ASCII WITH SIGN
1527	024200	005303		98:	DEC	R3		:MORE NUMBERS TO TYPE?
1528	024202	001403			BEQ	108		:NO
1529	024204	104401	036425		TYPE	,BLNKS2		:TYPE 2 SPACES
1530	024210	000764			BR	78		:LOOP
1531	024212	005302		108:	DEC	R2		:MORE DH'S?
1532	024214	003431			BLE	158		:NO
1533	024216	104401	001205		TYPE	,8CRLF		:YES--START A NEW LINE
1534	024222	005760	000002		TST	2(R0)		:ONLY A 'DH' IN THIS REQUEST ?
1535	024226	001404			BEQ	118		:BR IF YES - BYPASS THE INDENT
1536	024230	005105			COM	R5		:INDENT?
1537	024232	001002			BNE	118		:NO
1538	024234	104401	036425		TYPE	,BLNKS2		:TYPE 2 SPACES
1539	024240	012037	024246	118:	MOV	(R0)+,128		:GET NEXT DH
1540	024244	104401			TYPE			:AND TYPE IT
1541	024246	000000		128:	.WORD	0		:DH POINTER GOES HERE
1542	024250	005710			TST	(R0)		:TYPE A 'DT' ?
1543	024252	001003			BNE	138		:BR IF A 'DT'
1544	024254	062700	000004		ADD	#4,R0		:INCREMENT THE 'DF' POINTER
1545	024260	000754			BR	108		:SEE IF END OF 'DF' BLOCK
1546	024262	104401	001205	138:	TYPE	,8CRLF		:CARRIAGE RETURN-LINE FEED
1547	024266	005705			TST	R5		:INDENT?
1548	024270	001332			BNE	68		:NO
1549	024272	104401	036425	148:	TYPE	,BLNKS2		:TYPE 2 SPACES
1550	024276	000727			BR	68		:LOOP

1551 024300 104413
1552 024302 000207
1553
1554

15\$: RESREG ;RESTORE R0-R5
RTS PC ;RETURN

.SBTTL TYPE ROUTINE

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

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

024304	105737	001173	15\$:	TSTB	\$TPFLG	:: IS THERE A TERMINAL?
024310	100002			BPL	1\$:: BR IF YES
024312	000000			HALT		:: HALT HERE IF NO TERMINAL
024314	000430			BR	3\$:: LEAVE
024316	010046		15\$:	MOV	R0,-(SP)	:: SAVE R0
024320	017600	000002		MOV	@2(SP),R0	:: GET ADDRESS OF ASCIZ STRING
024324	122737	000001 001230		CMPB	#APTENV,\$ENV	:: RUNNING IN APT MODE
024332	001011			BNE	62\$:: NO,GO CHECK FOR APT CONSOLE
024334	132737	000100 001231		BITB	#APTSPOOL,\$ENVM	:: SPOOL MESSAGE TO APT
024342	001405			BEQ	62\$:: NO,GO CHECK FOR CONSOLE
024344	010037	024354		MOV	R0,61\$:: SETUP MESSAGE ADDRESS FOR APT
024350	004737	024646		JSR	PC,\$ATY3	:: SPOOL MESSAGE TO APT
024354	000000		61\$:	.WORD	0	:: MESSAGE ADDRESS
024356	132737	000040 001231	62\$:	BITB	#APTCSUP,\$ENVM	:: APT CONSOLE SUPPRESSED
024364	001003			BNE	60\$:: YES,SKIP TYPE OUT
024366	112046		2\$:	MOVB	(R0)+,-(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK
024370	001005			BNE	4\$:: BR IF IT ISN'T THE TERMINATOR
024372	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK
024374	012600		60\$:	MOV	(SP)+,R0	:: RESTORE R0
024376	062716	000002	3\$:	ADD	#2,(SP)	:: ADJUST RETURN PC
024402	000002			RTI		:: RETURN
024404	122716	000011	4\$:	CMPB	#HT,(SP)	:: BRANCH IF <HT>
024410	001430			BEQ	8\$	
024412	122716	000200		CMPB	#CRLF,(SP)	:: BRANCH IF NOT <CRLF>
024416	001006			BNE	5\$	
024420	005726			TST	(SP)+	:: POP <CR><LF> EQUIV
024422	104401			TYPE		:: TYPE A CR AND LF
024424	001205			\$CRLF		
024426	105037	024634		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT
024432	000755			BR	2\$:: GET NEXT CHARACTER
024434	004737	024516	5\$:	JSR	PC,\$TYPEC	:: GO TYPE THIS CHARACTER
024440	123726	001172	6\$:	CMPB	\$FILLC,(SP)+	:: IS IT TIME FOR FILLER CHARS.?
024444	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.
024446	013746	001170		MOV	\$NULL,-(SP)	:: GET # OF FILLER CHARS. NEEDED
						:: AND THE NULL CHAR.
024452	105366	000001	7\$:	DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?
024456	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK


```

024706 132737 000100 001231 BITB #APTSPOOL,$ENVM ;; SHOULD SPOOL MESSAGES?
024714 001425 BEQ 3$ ;; IF NOT: BR
024716 017600 000004 MOV @4(SP),RO ;; GET MESSAGE ADDR.
024722 062766 000002 C00004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
024730 005737 001210 1$: TST $MSGTYPE ;; SEE IF DONE W/ LAST XMISSION?
024734 001375 BNE 1$ ;; IF NOT: WAIT
024736 010037 001224 MOV RO,$MSGAD ;; PUT ADDR IN MAILBOX
024742 105720 2$: TSTB (RO)+ ;; FIND END OF MESSAGE
024744 001376 BNE 2$
024746 163700 001224 SUB $MSGAD,RO ;; SUB START OF MESSAGE
024752 006200 ASP RO ;; GET MESSAGE LGTH IN WORDS
024754 010037 001226 MOV RO,$MSGGLT ;; PUT LENGTH IN MAILBOX
024760 012737 000004 00'210 MOV #4,$MSGTYPE ;; TELL APT TO TAKE MSG.
024766 000413 BR 5$
024770 017637 000004 025014 3$: MOV @4(SP),4$ ;; PUT MSG ADDR IN JSR LINKAGE
024776 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDRESS
025004 013746 177776 MOV 177776,-(SP) ;; PUSH 177776 ON STACK
025010 004737 024304 JSR PC,$TYPE ;; CALL TYPE MACRO
025014 000000 4$: .WORD 0
025016 5$:
025016 105737 025104 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
025022 001416 BEQ 12$ ;; IF NOT: BR
025024 005737 001230 TST $ENV ;; RUNNING UNDER APT?
025030 001413 BEQ 12$ ;; IF NOT: BR
025032 005737 001210 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
025036 001375 BNE 11$ ;; IF NOT: WAIT
025040 017637 000004 001212 MOV @4(SP),$FATAL ;; GET ERROR #
025046 062766 000002 000004 ADD #2,4(SP) ;; BUMP RETURN ADDR.
025054 005237 001210 INC $MSGTYPE ;; TELL APT TO TAKE ERROR
025060 105037 025104 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
025064 105037 025103 CLRB $LFLG ;; CLEAR LOG FLAG
025070 105037 025102 CLRB $MFLG ;; CLEAR MESSAGE FLAG
025074 012601 MOV (SP)+,R1 ;; POP STACK INTO R1
025076 012600 MOV (SP)+,RO ;; POP STACK INTO RO
025100 000207 RTS PC ;; RETURN
025102 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
025103 000 $LFLG: .BYTE 0 ;; LOG FLAG
025104 000 $FFLG: .BYTE 0 ;; FATAL FLAG

```

```

000200
000001
000100
000040
APTSIZE = 200
APTENV = 001
APTSPOOL = 100
APTCSUP = 040

```

1557
1558

.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

```


1559
 1560

025322 012616
 025324 000002
 025326 000
 025327 000
 025330 000
 025331 000
 025332 000000

```

MOV      (SP)+,(SP)
RTI
8$:      .BYTE 0           ;;RETURN
        .BYTE 0           ;;STORAGE FOR ASCII DIGIT
        .BYTE 0           ;;TERMINATOR FOR TYPE ROUTINE
SOCNT:   .BYTE 0           ;;OCTAL DIGIT COUNTER
$OFILL:  .BYTE 0           ;;ZERO FILL SWITCH
$OMODE:  .WORD 0           ;;NUMBER OF DIGITS TO TYPE
  
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

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

025334
 025334 010046
 025336 010146
 025340 010246
 025342 010346
 025344 010546
 025346 012746 020200
 025352 016605 000020
 025356 100004
 025360 005405
 025362 112766 000055 000001
 025370 005000
 025372 012703 025550
 025376 112723 000040
 025402 005002
 025404 016001 025540
 025410 160105
 025412 002402
 025414 005202
 025416 000774
 025420 060105
 025422 005702
 025424 001002
 025426 105716
 025430 100407
 025432 106316
 025434 103003
 025436 116663 000001 177777
 025444 052702 000060
 025450 052702 000040
 025454 110223
 025456 005720
 025460 020027 000010
 025464 002746
 025466 003002
 025470 010502
 025472 000764

```

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5     ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
1$:      CLR      R0      ;;ZERO THE CONSTANTS INDEX
MOV      #$DBLK,R3     ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2      ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
3$:      SUB      R1,R5    ;;FORM THIS BCD DIGIT
BLT      4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
4$:      ADD      R1,R5    ;;ADD BACK THE CONSTANT
TST      R2            ;;CHECK IF BCD DIGIT=0
BNE      5$            ;;FALL THROUGH IF 0
TSTB     (SP)         ;;STILL DOING LEADING 0'S?
BMI      7$            ;;BR IF YES
5$:      ASLB     (SP)    ;;MSD?
BCC      6$            ;;BR IF NO
MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
6$:      BIS      #'0,R2  ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2  ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+        ;;JUST INCREMENTING
CMP      R0,#10       ;;CHECK THE TABLE INDEX
BLT      2$            ;;GO DO THE NEXT DIGIT
BGT      8$            ;;GO TO EXIT
MOV      R5,R2        ;;GET THE LSD
BR       6$           ;;GO CHANGE TO ASCII
  
```

```

025474 105726      8$:  TSTB   (SP)+      ;; WAS THE LSD THE FIRST NON-ZERO?
025476 100003      BPL    9$              ;; BR IF NO
025500 116663 177777 177776  MOVB   -1(SP),-2(R3)  ;; YES--SET THE SIGN FOR TYPING
025506 105013      9$:  CLRB   (R3)         ;; SET THE TERMINATOR
025510 012605      MOV    (SP)+,R5       ;; POP STACK INTO R5
025512 012603      MOV    (SP)+,R3       ;; POP STACK INTO R3
025514 012602      MOV    (SP)+,R2       ;; POP STACK INTO R2
025516 012601      MOV    (SP)+,R1       ;; POP STACK INTO R1
025520 012600      MOV    (SP)+,R0       ;; POP STACK INTO R0
025522 104401 025550  TYPE   $DBLK      ;; NOW TYPE THE NUMBER
025526 016666 000002 000004  MOV    2(SP),4(SP)   ;; ADJUST THE STACK
025534 012616      MOV    (SP)+,(SP)
025536 000002      RTI                    ;; RETURN TO USER
025540 023420      $DTBL: 10000.
025542 001750      1000.
025544 000144      100.
025546 000012      10.
025550      $DBLK: .BLKW 4
    
```

1561
1562

.SBTTL TTY INPUT ROUTINE

```

;*****
.ENABL  LSB
025560 000000  $TKCNT: .WORD 0      ;; NUMBER OF ITEMS IN QUEUE
025562 000000  $TKQIN: .WORD 0      ;; INPUT POINTER
025564 000000  $TKQOUT: .WORD 0     ;; OUTPUT POINTER
025566      $TKQSRV: .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
;
025576 005037 025560  $TKINT: CLR    $TKCNT      ;; CLEAR COUNT OF ITEMS IN QUEUE
025602 012737 025566 025562  MOV    # $TKQSRV,$TKQIN  ;; MOVE THE STARTING ADDRESS OF THE
025610 013737 025562 025564  MOV    $TKQIN,$TKQOUT   ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
025616 012737 025646 000060  MOV    # $TKSRV,@ $TKVEC ;; INITIALIZE THE KEYBOARD VECTOR
025624 012737 000200 000062  MOV    #200,@ $TKVEC+2  ;; 'BR' LEVEL 4
025632 005777 153324  TST    @ $TKB           ;; CLEAR DONE FLAG
025636 012777 000100 153314  MOV    #100,@ $TKS      ;; ENABLE TTY KEYBOARD INTERRUPT
025644 000207      RTS    PC           ;; RETURN TO CALLER
    
```

```

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

```

025646 117746 153310  $TKSRV: MOVB   @ $TKB, -(SP)  ;; PICKUP THE CHARACTER
025652 042716 177600  BIC    # ^C(177), (SP)     ;; STRIP THE JUNK
025656 021627 000003  CMP    (SP), #3           ;; IS IT A CONTROL C?
    
```

```
025662 001007          BNE      1$          ;;BRANCH IF NO
025664 104401 026767   TYPE      ,SCNTLC  ;;TYPE A CONTROL-C (^C)
025670 004737 025576   JSR      PC,$TKINT  ;;INIT THE KEYBOARD
025674 005726          TST      (SP)+      ;;CLEAN UP STACK
025676 000177 153472   JMP      @CNTLC     ;;CONTROL C RESTART
025702 021627 000007   1$:    CMP      (SP),#7    ;;IS IT A CONTROL G?
025706 001004          BNE      2$          ;;BRANCH IF NO
025710 022737 000176 001154   CMP      #SWREG,SWR  ;;IS SOFT-SWR SELECTED?
025716 001500          BEQ      6$          ;;GO TO SWR CHANGE

025720          2$:    CMP      #7,$TKCNT  ;;IS THE QUEUE FULL?
025720 022737 000007 025560   BNE      3$          ;;BRANCH IF NO
025726 001004          TYPE      ,SBELL    ;;RING THE TTY BELL
025730 104401 001200   TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
025734 005726          BR       5$          ;;EXIT
025736 000451          3$:    CMP      (SP),#23    ;;IS IT A CONTROL-S?
025740 021627 000023   BNE      32$         ;;BRANCH IF NO
025744 001021          CLR      @STKS      ;;DISABLE TTY KEYBOARD INTERRUPTS
025746 005077 153206   TST      (SP)+      ;;CLEAN CHAR OFF STACK
025752 005726          31$:   TSTB     @STKS      ;;WAIT FOR A CHAR
025754 105777 153200   BPL      31$        ;;LOOP UNTIL ITS THERE
025760 100375          MOVB    @STKB,-(SP)  ;;GET THE CHARACTER
025762 117746 153174   BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
025766 042716 177600   CMP      (SP)+,#21  ;;IS IT A CONTROL-Q?
025772 022627 000021   BNE      31$        ;;BRANCH IF NO
025776 001366          MOV     #100,@STKS  ;;REENABLE TTY KEYBOARD INTERRUPTS
026000 012777 000100 153152   RTI          ;;RETURN
026006 000002          32$:   INC     $TKCNT  ;;COUNT THIS CHARACTER
026010 005237 025560   CMP      (SP),#140  ;;IS IT UPPER CASE?
026014 021627 000140   BLT     4$          ;;BRANCH IF YES
026020 002405          CMP     (SP),#175  ;;IS IT A SPECIAL CHAR?
026022 021627 000175   BGT     4$          ;;BRANCH IF YES
026026 003002          BIC     #40,(SP)   ;;MAKE IT UPPER CASE
026030 042716 000040   MOVB   (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
026034 112677 177522   4$:    INC     $TKQIN  ;;UPDATE THE POINTER
026040 005237 025562   CMP     $TKQIN,#$TKQEND ;;GO OFF THE END?
026044 023727 025562 025575   BNE     5$          ;;BRANCH IF NO
026052 001003          MOV     #,$TKQRT,$TKQIN ;;RESET THE POINTER
026054 012737 025566 025562   5$:    RTI          ;;RETURN
026062 000002
```

```
*****
;*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.
026064 022737 000176 001154 $CKSWR CMP     #SWREG,SWR  ;;IS THE SOFT-SWR SELECTED
026072 001124          BNE     15$         ;;EXIT IF NOT
026074 105777 153060   TSTB   @STKS      ;;IS A CHAR WAITING?
026100 100121          BPL     15$        ;;IF NOT, EXIT
026102 117746 153054   MOVB   @STKB,-(SP) ;;YES
026106 042716 177600   BIC     #^C177,(SP) ;;MAKE IT 7-BIT ASCII
026112 021627 000007   CMP     (SP),#7    ;;IS IT A CONTROL-G?
026116 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.
026120 123727 001150 000001 6$:  CMPB  $AUTOB,#1  ;;ARE WE RUNNING IN AUTO-MODE?
026126 001674  BEQ  2$  ;;BRANCH IF YES
026130 005726  TST  (SP)+  ;;CLEAR CONTROL-G OFF STACK
026132 004737 025576  JSR  PC,$TKINT  ;;FLUSH THE TTY INPUT QUEUE
026136 005077 153016  CLR  @STKS  ;;DISABLE TTY KEYBOARD INTERRUPTS
026142 112737 000001 001151  MOVB  #1,$INTAG  ;;SET INTERRUPT MODE INDICATOR

026150 104401 027001  SGT$SWR: TYPE  ,$CNTLG  ;;ECHO THE CONTROL-G (^G)
026154 104401 027006  TYPE  ,$MSWR  ;;TYPE CURRENT CONTENTS
026160 013746 000176  MOV  SWREG,-(SP)  ;;SAVE SWREG FOR TYPEOUT
026164 104402  TYPOC  ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
026166 104401 027017  TYPE  ,$MNEW  ;;PROMPT FOR NEW SWR
026172 005046 19$:  CLR  -(SP)  ;;CLEAR COUNTER
026174 005046  CLR  -(SP)  ;;THE NEW SWR
026176 105777 152756  7$:  TSTB @STKS  ;;CHAR THERE?
026202 100375  BPL  7$  ;;IF NOT TRY AGAIN

026204 117746 152752  MOVB  @STKB,-(SP)  ;;PICK UP CHAR
026210 042716 177600  BIC  #^C177,(SP)  ;;MAKE IT 7-BIT ASCII

026214 021627 000003  CMP  (SP),#3  ;;IS IT A CONTROL-C?
026220 001015  BNE  9$  ;;BRANCH IF NOT
026222 104401 026767  TYPE  ,$CNTLC  ;;YES, ECHO CONTROL-C (^C)
026226 062706 000006  ADD  #6,SP  ;;CLEAN UP STACK
026232 123727 001151 000001  CMPB  $INTAG,#1  ;;REENABLE TTY KEYBOARD INTERRUPTS?
026240 001003  BNE  8$  ;;BRANCH IF NO
026242 012777 000100 152710  MOV  #100,@STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
026250 000177 153120  8$:  JMP  @CNTLC  ;;CONTROL-C RESTART

026254 021627 000025  9$:  CMP  (SP),#25  ;;IS IT A CONTROL-U?
026260 001005  BNE  10$  ;;BRANCH IF NOT
026262 104401 026774  TYPE  ,$CNTLU  ;;YES, ECHO CONTROL-U (^U)
026266 062706 000006  20$:  ADD  #6,SP  ;;IGNORE PREVIOUS INPUT
026272 000737  BR  19$  ;;LET'S TRY IT AGAIN

026274 021627 000015  10$:  CMP  (SP),#15  ;;IS IT A <CR>?
026300 001022  BNE  16$  ;;BRANCH IF NO
026302 005766 000004  TST  4(SP)  ;;YES, IS IT THE FIRST CHAR?
026306 001403  BEQ  11$  ;;BRANCH IF YES
026310 016677 000002 152636  MOV  2(SP),@SWR  ;;SAVE NEW SWR
026316 062706 000006  11$:  ADD  #6,SP  ;;CLEAR UP STACK
026322 104401 001205  14$:  TYPE  ,$CRLF  ;;ECHO <CR> AND <LF>
026326 123727 001151 000001  CMPB  $INTAG,#1  ;;RE-ENABLE TTY KBD INTERRUPTS?
026334 001003  BNE  15$  ;;BRANCH IF NOT
026336 012777 000100 152614  MOV  #100,@STKS  ;;RE-ENABLE TTY KBD INTERRUPTS
026344 000002  15$:  RTI  ;;RETURN
026346 004737 024516  16$:  JSR  PC,$TYPEC  ;;ECHO CHAR
026352 021627 000060  CMP  (SP),#60  ;;CHAR < 0?
026356 002420  BLT  18$  ;;BRANCH IF YES
026360 021627 000067  CMP  (SP),#67  ;;CHAR > 7?
026364 003015  BGT  18$  ;;BRANCH IF YES
026366 042726 000060  BIC  #60,(SP)+  ;;STRIP-OFF ASCII

```

```

026372 005766 000002          TST      2(SP)          ;; IS THIS THE FIRST CHAR
026376 001403                BEQ      17$           ;; BRANCH IF YES
026400 006316                ASL      (SP)          ;; NO, SHIFT PRESENT
026402 006316                ASL      (SP)          ;; CHAR OVER TO MAKE
026404 006316                ASL      (SP)          ;; ROOM FOR NEW ONE.
026406 005266 000002          17$: INC      2(SP)          ;; KEEP COUNT OF CHAR
026412 056616 177776          BIS      -2(SP),(SP)   ;; SET IN NEW CHAR
026416 000667                BR       7$           ;; GET THE NEXT ONE
026420 104401 001204          18$: TYPE  $QUES       ;; TYPE ?<CR><LF>
026424 000720                BR       20$          ;; 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

```

```

026426 011646 000004 000002 $RDCHR: MOV      (SP),-(SP)  ;; PUSH DOWN THE PC AND
026430 016666 000004          MOV      4(SP),2(SP)    ;; THE PS
026436 005066 000004          CLR      4(SP)          ;; GET READY FOR A CHARACTER
026442 005046 026452          CLR      -(SP)         ;; PUT NEW PS ON STACK
026444 012746 026452          MOV      #64$,-(SP)   ;; PUT NEW PC ON STACK
026450 000002          RTI              ;; POP NEW PC AND PS
026452
026452 005737 025560          64$: TST      $TKCNT     ;; WAIT ON A CHARACTER
026456 001775                BEQ      1$           ;;
026460 005337 025560          1$: DEC      $TKCNT     ;; DECREMENT THE COUNTER
026464 117766 177074 000004  MOVB     @ $TKQOUT,4(SP) ;; GET ONE CHARACTER
026472 005237 025564          INC      $TKQOUT      ;; UPDATE THE POINTER
026476 023727 025564 025575  CMP      $TKQOUT,$TKQEND ;; DID IT GO OFF OF THE END?
026504 001003                BNE      2$           ;; BRANCH IF NO
026506 012737 025566 025564  MOV      #$TKQSRRT,$TKQOUT ;; RESET THE POINTER
026514 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

```

```

026516 010346 026752          $RDLIN: MOV      R3,-(SP)  ;; SAVE R3
026520 005046                CLR      -(SP)         ;; CLEAR THE RUBOUT KEY
026522 012703 026752          1$: MOV      #$TTYIN,R3  ;; GET ADDRESS
026526 022703 026767          2$: CMP      #$TTYIN+15,R3 ;; BUFFER FULL?
026532 101456                BLOS     4$           ;; BR IF YES
026534 104410                RDCHR    ;; GO READ ONE CHARACTER FROM THE TTY
026536 112613                MOVB     (SP)+,(R3)    ;; GET CHARACTER
026540 122713 000177          10$: CMPB     #177,(R3)   ;; IS IT A RUBOUT
026544 001022                BNE      5$           ;; BR IF NO
026546 005716                TST      (SP)         ;; IS THIS THE FIRST RUBOUT?
026550 001007                BNE      6$           ;; BR IF NO
026552 112737 000134 026750  MOVB     #' \,9$      ;; TYPE A BACK SLASH
026560 104401 026750          TYPE     ,9$

```

```

026564 012716 177777      MOV      #-1,(SP)      ;;SET THE RUBOUT KEY
026570 005303      6$: DEC      R3        ;;BACKUP BY ONE
026572 020327 026752      CMP      R3,#$TTYIN   ;;STACK EMPTY?
026576 103434      BLO     4$           ;;BR IF YES
026600 111337 026750      MOV     (R3),9$      ;;SETUP TO TYPEOUT THE DELETED CHAR.
026604 104401 026750      TYPE   9$           ;;GO TYPE
026610 000746      BR      2$           ;;GO READ ANOTHER CHAR.
026612 005716      5$: TST     (SP)      ;;RUBOUT KEY SET?
026614 001406      BEQ     7$           ;;BR IF NO
026616 112737 000134 026750      MOV     #'\",9$      ;;TYPE A BACK SLASH
026624 104401 026750      TYPE   9$           ;;
026630 005016      CLR     (SP)        ;;CLEAR THE RUBOUT KEY
026632 122713 000025      7$: CMP     #25,(R3)   ;;IS CHARACTER A CTRL U?
026636 001003      BNE     8$           ;;BR IF NO
026640 104401 026774      TYPE   ,SCNTLU      ;;TYPE A CONTROL 'U'
026644 000726      BR      1$           ;;GO START OVER
026646 122713 000022      8$: CMP     #22,(R3)   ;;IS CHARACTER A '^R'?
026652 001011      BNE     3$           ;;BRANCH IF NO
026654 105013      CLRB   (R3)        ;;CLEAR THE CHARACTER
026656 104401 001205      TYPE   ,SCRLF      ;;TYPE A 'CR' & 'LF'
026662 104401 026752      TYPE   ,TTYIN      ;;TYPE THE INPUT STRING
026666 000717      BR      2$           ;;GO PICKUP ANOTHER CHACTER
026670 104401 001204      4$: TYPE   ,QUES     ;;TYPE A '?'
026674 000712      BR      1$           ;;CLEAR THE BUFFER AND LOOP
026676 111337 026750      3$: MOV     (R3),9$   ;;ECHO THE CHARACTER
026702 104401 026750      TYPE   9$           ;;
026706 122723 000015      CMP     #15,(R3)+   ;;CHECK FOR RETURN
026712 001305      BNE     2$           ;;LOOP IF NOT RETURN
026714 105063 177777      CLRB   -1(R3)      ;;CLEAR RETURN (THE 15)
026720 104401 001206      TYPE   ,LF         ;;TYPE A LINE FEED
026724 005726      TST   (SP)+        ;;CLEAN RUBOUT KEY FROM THE STACK
026726 012603      MOV   (SP)+,R3     ;;RESTORE R3
026730 011646      MOV   (SP),-(SP)   ;;ADJUST THE STACK AND PUT ADDRESS OF THE
026732 016666 000004 000002      MOV   4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
026740 012766 026752 000004      MOV   #$TTYIN,4(SP)
026746 000002      RTI                ;;RETURN
026750 000      19$: .BYTE 0      ;;STORAGE FOR ASCII CHAR. TO TYPE
026751 000      .BYTE 0      ;;TERMINATOR
026752      $TTYIN: .BLKB 15  ;;RESERVE 15 BYTES FOR TTY INPUT
026767 136 103 015 $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
026774 136 125 015 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
027001 136 107 015 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
027006 015 012 123 $MSWR: .ASCIZ <15><12>/SWR = /
027017 040 040 116 $MNEW: .ASCIZ / NEW = /

```

1563
1564

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
$PWRDN: MOV   #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
MOV   #340,@#PWRVEC+2 ;;PRIO:7
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

```

```

027060 017746 152070      MOV    @SWR,-(SP)      ;;PUSH @SWR ON STACK
027064 010637 027200      MOV    SP,$SAVR6     ;;SAVE SP
027070 012737 027102 000024  MOV    #SPWRUP,@#PWRVEC ;;SET UP VECTOR
027076 000000      HALT
027100 000776      BR     .-2           ;;HANG UP

```

POWER UP ROUTINE

```

027102 012737 027174 000024 $PWRUP: MOV    #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
027110 013706 027200      MOV    $SAVR6,SP     ;;GET SP
027114 005037 027200      CLR    $SAVR6        ;;WAIT LOOP FOR THE TTY
027120 005237 027200      1$:  INC    $SAVR6     ;;WAIT FOR THE INC
027124 001375      BNE    1$           ;;OF WORD
027126 012677 152022      MOV    (SP)+,@SWR    ;;POP STACK INTO @SWR
027132 012605      MOV    (SP)+,R5     ;;POP STACK INTO R5
027134 012604      MOV    (SP)+,R4     ;;POP STACK INTO R4
027136 012603      MOV    (SP)+,R3     ;;POP STACK INTO R3
027140 012602      MOV    (SP)+,R2     ;;POP STACK INTO R2
027142 012601      MOV    (SP)+,R1     ;;POP STACK INTO R1
027144 012600      MOV    (SP)+,R0     ;;POP STACK INTO R0
027146 012737 027030 000024  MOV    #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
027154 012737 000340 000026  MOV    #340,@#PWRVEC+2 ;;PRIO:7
027162 104401      TYPE    $PWRMG      ;;REPORT THE POWER FAILURE
027164 027202      $PWRMG: .WORD    $POWER ;;POWER FAIL MESSAGE POINTER
027166 012716      MOV    (PC)+,(SP)   ;;RESTART AT BEGIN2
027170 007020      $PWRAD: .WORD    BEGIN2 ;;RESTART ADDRESS
027172 000002      RTI
027174 000000      $SILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
027176 000776      BR     .-2         ;; BEFORE THE POWER DOWN WAS COMPLETE
027200 000000      $SAVR6: 0          ;;PUT THE SP HERE
1565 027202      200      042      120 $POWER: .ASCIZ <CRLF>/'POWER UP'/
1566
1567

```

.SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS

*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
*LEADING NUMBERS.

*CALL

```

*      MOV    #NUMADR,-(SP) ;;FIRST ADDRESS OF ASCIZ STRING
*      JSR    PC,@#$SUPRS

```

```

027216 010046      $SUPRS: MOV    R0,-(SP)      ;;SAVE R0
027220 016600 000004      MOV    4(SP),R0     ;;PICKUP THE POINTER
027224 105710      1$:  TSTB   (R0)      ;;TERMINATEOR?
027226 001403      BEQ    2$          ;;BR IF YES
027230 122720 000060      CMPB   #'0',(R0)+  ;;IS THIS AN ASCII '0' ?
027234 001773      BEQ    1$          ;;BR IF YES
027236 005300      2$:  DEC    R0       ;;BACKUP BY '1'
027240 010037 027246      MOV    R0,3$       ;;SAVE FOR TYPING
027244 104401      TYPE    $POWER     ;;GO TYPE
027246 000000      3$:  .WORD    0      ;;ASCIZ POINTER GOES HERE
027250 012600      MOV    (SP)+,R0     ;;RESTORE R0
027252 012616      MOV    (SP)+,(SP)  ;;RESTORE THE STACK
027254 000207      RTS    PC          ;;RETURN

```

.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
    
```

```

027256 016637 000002 027306 $SB2D: MOV      2(SP),1$      ;;SAVE BINARY NUMBER
027264 012746 027306          MOV      #1$,-(SP)      ;;SET POINTER
027270 004737 027312          JSR      PC,@#$DB2D      ;;CALL DOUBLE LENGTH CONVERT
027274 062716 000005          ADD      #5,(SP)        ;;ONLY ALLOW FIVE CHARACTERS
027300 012666 000002          MOV      (SP)+,2(SP)    ;;PICKUP POINTER
027304 000207                      RTS      PC              ;;RETURN
027306 000000 000000          1$:      .WORD      0,0
    
```

1570
1571

.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

```

*****
*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
*POSITIVE.
*CALL
*      MOV      #PNTR,-(SP)      ;;POINTER TO LOW WORD OF BINARY NUMBER
*      JSR      PC,@#$DB2D      ;;THE FIRST ADDRESS OF ASCIZ
*      RETURN                      ;;IS ON THE STACK
    
```

```

027312 104412          $DB2D: SAVREG      ;;SAVE REGISTERS
027314 016602 000002      MOV      2(SP),R2      ;;PICKUP THE DATA POINTER
027320 012700 027472      MOV      #$DECVL,R0    ;;GET ADDRESS OF "$DECVL" STRING
027324 010066 000002      MOV      R0,2(SP)      ;;PUT ADDRESS OF ASCIZ STRING ON STACK
027330 012201          MOV      (R2)+,R1      ;;PICKUP THE BINARY NUMBER
027332 012202          MOV      (R2)+,R2
027334 012737 000012 027410      MOV      #10,4$        ;;SET UP TO DO 10 CONVERSIONS
027342 012704 027422      MOV      #$TNPWR,R4    ;;ADDRESS OF TEN POWER
027346 012705 027424      MOV      #$TNPWR+2,R5
027352 005003          1$:      CLR      R3              ;;CLEAR PARTIAL
027354 161401          2$:      SUB      (R4),R1      ;;SUBTRACT TEN POWER
027356 005602          SBC      R2
027360 161502          SUB      (R5),R2
027362 002402          BLT      3$           ;;BR IF TEN POWER TO LARGE
027364 005203          INC      R3          ;;ADD 1 TO PARTIAL
027366 000772          BR      2$           ;;LOOP
027370 062401          3$:      ADD      (R4)+,R1      ;;RESTORE SUBTRACTED VALUE
027372 005502          ADC      R2
027374 062402          ADD      (R4)+,R2
027376 022525          CMP      (R5)+,(R5)+  ;;MOVE TO NEXT TEN POWER
027400 052703 000060      BIS      #'0,R3        ;;CHANGE PARTIAL TO ASCII
027404 110320          MOVB     R3,(R0)+      ;;SAVE IT
027406 005327          DEC      (PC)+       ;;DONE?
027410 000000          4$:      .WORD      0
027412 001357          BNE     1$            ;;BR IF NO
027414 105020          CLRB     (R0)+       ;;TERMINATOR
    
```

```

027416 104413 RESREG          ;;RESTORE REGISTERS
027420 000207 RTS          PC      ;;RETURN
027422 145000 $TNPWR: 145000  ;;:1.0E09
027424 035632 35632          ;;:1.0E08
027426 160400 160400          ;;:1.0E07
027430 002765 2765           ;;:1.0E06
027432 113200 113200          ;;:1.0E05
027434 000230 230            ;;:1.0E04
027436 041100 041100          ;;:1.0E03
027440 000017 17             ;;:1.0E02
027442 103240 103240          ;;:1.0E01
027444 000001 1              ;;:1.0E00
027446 023420 23420          ;;:1.0E00
027450 000000 0              ;;:1.0E00
027452 001750 1750           ;;:1.0E00
027454 000000 0              ;;:1.0E00
027456 000144 144            ;;:1.0E00
027460 000000 0              ;;:1.0E00
027462 000012 12             ;;:1.0E00
027464 000000 0              ;;:1.0E00
027466 000001 1              ;;:1.0E00
027470 000000 0              ;;:1.0E00
027472 000000 0              ;;:1.0E00

```

1572
1573

```

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

```

```

027506 010046 $SAVREG: MOV R0,-(SP)  ;;PUSH R0 ON STACK
027510 010146 MOV R1,-(SP)  ;;PUSH R1 ON STACK
027512 010246 MOV R2,-(SP)  ;;PUSH R2 ON STACK
027514 010346 MOV R3,-(SP)  ;;PUSH R3 ON STACK
027516 010446 MOV R4,-(SP)  ;;PUSH R4 ON STACK
027520 010546 MOV R5,-(SP)  ;;PUSH R5 ON STACK
027522 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
027524 016646 000022 MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
027532 016646 000022 MOV 22(SP),-(SP) ;;SAVE PS OF CALL
027536 016646 000022 MOV 22(SP),-(SP) ;;SAVE PC OF CALL
027542 000002 RTI

```

```

*RESTORE R0-R5
*CALL:
* RESREG

```

```

027544          $RESREG:
027544 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF CALL
027550 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF CALL
027554 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PC OF MAIN FLOW
027560 012666 000022      MOV      (SP)+,22(SP)      ;;RESTORE PS OF MAIN FLOW
027564 012605              MOV      (SP)+,R5          ;;POP STACK INTO R5
027566 012604              MOV      (SP)+,R4          ;;POP STACK INTO R4
027570 012603              MOV      (SP)+,R3          ;;POP STACK INTO R3
027572 012602              MOV      (SP)+,R2          ;;POP STACK INTO R2
027574 012601              MOV      (SP)+,R1          ;;POP STACK INTO R1
027576 012600              MOV      (SP)+,R0          ;;POP STACK INTO R0
027600 000002              RTI
  
```

1574
 1575

.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.
  
```

```

027602 010046          $TRAP: MOV      R0,-(SP)          ;;SAVE R0
027604 016600 000002      MOV      2(SP),R0          ;;GET TRAP ADDRESS
027610 005740              TST      -(R0)            ;;BACKUP BY 2
027612 111000              MOVVB   (R0),R0          ;;GET RIGHT BYTE OF TRAP
027614 006300              ASL     R0              ;;POSITION FOR INDEXING
027616 016000 027636      MOV      $TRPAD(R0),R0      ;;INDEX TO TABLE
027622 000200              RTS     R0              ;;GO TO ROUTINE
  
```

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

```

027624 011646          $TRAP2: MOV      (SP),-(SP)        ;;MOVE THE PC DOWN
027626 016666 000004 000002      MOV      4(SP),2(SP)      ;;MOVE THE PSW DOWN
027634 000002              RTI                    ;;RESTORE THE PSW
  
```

.SBTTL TRAP TABLE

```

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

```

; ROUTINE
;-----
027636 027624          $TRPAD: .WORD   $TRAP2
027640 024304          $TYPE   ;;CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
027642 025132          $TYPOC  ;;CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
027644 025106          $TYPOS  ;;CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
027646 025146          $TYPON  ;;CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
027650 025334          $TYPDS  ;;CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)

027652 026154          $GTSWR  ;;CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING

027654 026064          $CKSWR  ;;CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
027656 026426          $RDCHR  ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
027660 026516          $RDLIN  ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
027662 027506          $SAVREG ;;CALL=SAVREG     TRAP+12(104412) SAVE R0-R5 ROUTINE
027664 027544          $RESREG ;;CALL=RESREG     TRAP+13(104413) RESTORE R0-R5 ROUTINE
  
```

CZRMLAO RM05/3/2 FORMATTER
TRAP TABLE

MACRO V03.01 11-APR-80 13:01:23 PAGE 10-43

^D
8

SEQ 0094

1576
1583

```
1 .SBTTL SINGLE/DUAL PORT RH/RM DRIVER (REV 6.2) FEBRUARY 1980
2
3 ;NEW DRIVE TYPE ID FOR RM02 *****
4 ;10-AUG-77 *****
5 ;10-MAR-78 THE SC, SC5 CHANGES
6 ;NEW DRIVE TYPE ID FOR RM05 *****
7 ;FEBRUARY 1980 *****
8
9 ;COPYRIGHT (C) 1977
10 ;DIGITAL EQUIPMENT CORP.
11 ;MAYNARD, MA 01754
12 ;AUTHOR(S): JIM LACEY/CHUCK HESS/
13
14 ;*****
15
16 ;STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
17 ;RMERRS = RMD5
18 ;RMERRS+2 = RMER1
19 ;RMERRS+4 = RMER2
20 ;RMERRS+6 = RMMR2
21
22 027666 000000 000000 000000 RMERRS: .WORD 0,0,0,0
23
24 ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
25 ;DRVACT=0 IF DRIVE IS IDLE
26 ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
27 ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
28
29 027676 000 DRVACT: .BYTE 0 ;DRIVE 0
32 027677 000 .BYTE 0 ;DRIVE 1
027700 000 .BYTE 0 ;DRIVE 2
027701 000 .BYTE 0 ;DRIVE 3
027702 000 .BYTE 0 ;DRIVE 4
027703 000 .BYTE 0 ;DRIVE 5
027704 000 .BYTE 0 ;DRIVE 6
027705 000 .BYTE 0 ;DRIVE 7
33
34 ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
35 ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
36 ;DRVSTA>0 IF DRIVE IS ONLINE
37 ;DRVSTA<0 IF DRIVE IS UNSAFE
38
39 027706 000 DRVSTA: .BYTE 0 ;DRIVE 0
42 027707 000 .BYTE 0 ;DRIVE 1
027710 000 .BYTE 0 ;DRIVE 2
027711 000 .BYTE 0 ;DRIVE 3
027712 000 .BYTE 0 ;DRIVE 4
027713 000 .BYTE 0 ;DRIVE 5
027714 000 .BYTE 0 ;DRIVE 6
027715 000 .BYTE 0 ;DRIVE 7
43
44 ;TABLE OF DRIVE TYPES (DRVTYP=8 BYTES)
45 ;DRVTYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
46 ;DRVTYP=7 IF DRIVE IS RM05 *****
47 ;DRVTYP=5 IF DRIVE IS RM02 *****
48 ;DRVTYP=4 IF DRIVE IS RM03
49 ;DRVTYP=-1 IF NOT RM05/3/2
```

```

50
51 027716      000      DRVTYP: .BYTE 0          ;DRIVE 0
54 027717      000          .BYTE 0          ;DRIVE 1
    027720      000          .BYTE 0          ;DRIVE 2
    027721      000          .BYTE 0          ;DRIVE 3
    027722      000          .BYTE 0          ;DRIVE 4
    027723      000          .BYTE 0          ;DRIVE 5
    027724      000          .BYTE 0          ;DRIVE 6
    027725      000          .BYTE 0          ;DRIVE 7
55
56              ;TABLE OF DUAL PORT INITIALIZATION INDICATORS
57              ;DPINT=0 IF INITIALIZATION IS NOT ACTIVE ON THE DRIVE
58              ;DPINT<0 IF INITIALIZATION IS IN PROGRESS
59
60 027726      000      DPINT: .BYTE 0          ;DRIVE 0
63 027727      000          .BYTE 0          ;DRIVE 1
    027730      000          .BYTE 0          ;DRIVE 2
    027731      000          .BYTE 0          ;DRIVE 3
    027732      000          .BYTE 0          ;DRIVE 4
    027733      000          .BYTE 0          ;DRIVE 5
    027734      000          .BYTE 0          ;DRIVE 6
    027735      000          .BYTE 0          ;DRIVE 7
64
65              ;TABLE OF PENDING DUAL PORT REQUESTS
66              ;DPRQS=0 IF THAT A DUAL PORT REQUEST IS NOT PENDING FOR THAT DRIVE
67              ;DPRQS<0 IF THAT A DUAL PORT REQUEST IS PENDING FOR THAT DRIVE
68
69 027736      000      DPRQS: .BYTE 0          ;DRIVE 0
72 027737      000          .BYTE 0          ;DRIVE 1
    027740      000          .BYTE 0          ;DRIVE 2
    027741      000          .BYTE 0          ;DRIVE 3
    027742      000          .BYTE 0          ;DRIVE 4
    027743      000          .BYTE 0          ;DRIVE 5
    027744      000          .BYTE 0          ;DRIVE 6
    027745      000          .BYTE 0          ;DRIVE 7
73
74              ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
75              ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
76              ;"DPB" OF THE I/O OPERATION.
77
78 027746      000000    TRNSWT: .WORD 0
79
80              ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
81              ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
82              ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
83              ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
84              ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
85
86 027750      000000    SRCHWT: .WORD 0
87
88              ;RM DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
89              ;ACTDRV=0 IF DRIVER IS INACTIVE
90              ;ACTDRV>0 IF DRIVER IS ACTIVE
91
92 027752      000      ACTDRV: .BYTE 0
93
94              ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
    
```

```

95                                     ;ACTSTR=J IF SOFTWARE TIMER ROUTINE IS INACTIVE
96                                     ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
97
98 027753      000      ACTSTR: .BYTE  0
99
100                                     ;UNLOAD FLAG (ULDFLG=8 BYTES)
101                                     ;ULDFLG=0 IF NO UNLOAD COMMAND
102                                     ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
103                                     ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
104
105 027754      000      ULDFLG: .BYTE  0          ;DRIVE 0
108 027755      000      .BYTE  0          ;DRIVE 1
      027756      000      .BYTE  0          ;DRIVE 2
      027757      000      .BYTE  0          ;DRIVE 3
      027760      000      .BYTE  0          ;DRIVE 4
      027761      000      .BYTE  0          ;DRIVE 5
      027762      000      .BYTE  0          ;DRIVE 6
      027763      000      .BYTE  0          ;DRIVE 7
109
110                                     ;LOOK AHEAD COUNT (LACNT=8 BYTES)
111                                     ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
112
113 027764      000      LACNT:  .BYTE  0          ;DRIVE 0
116 027765      000      .BYTE  0          ;DRIVE 1
      027766      000      .BYTE  0          ;DRIVE 2
      027767      000      .BYTE  0          ;DRIVE 3
      027770      000      .BYTE  0          ;DRIVE 4
      027771      000      .BYTE  0          ;DRIVE 5
      027772      000      .BYTE  0          ;DRIVE 6
      027773      000      .BYTE  0          ;DRIVE 7
117
118                                     ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
119                                     ;SAVEFG <0 IF SAVE THE RH/RM REGISTERS WHEN THE
120                                     ;OPERATION IS COMPLETED AS PER (DPB+14).
121                                     ;SAVEFG=0 IF SAVE THE RH/RM REGISTERS, AS PER
122                                     ;(DPB+14), AFTER AN ERROR.
123
124 027774      000000  SAVEFG: .WORD  0
125
126                                     ;SEEK FLAG (SEEKFG=1 WORD)
127                                     ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
128                                     ;FOR A DATA TRANSFER START A SEARCH COMMAND
129                                     ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
130                                     ;DISREGARD THE WINDOW
131
132 027776      177777  SEEKFG: .WORD  -1
133
134                                     ;TIMEOUT TABLE (TIMER=8 WORDS)
135                                     ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
136
137 030000      177777  TIMER:   .WORD  -1          ;DRIVE 0
140 030002      177777  .WORD  -1          ;DRIVE 1
      030004      177777  .WORD  -1          ;DRIVE 2
      030006      177777  .WORD  -1          ;DRIVE 3
      030010      177777  .WORD  -1          ;DRIVE 4
      030012      177777  .WORD  -1          ;DRIVE 5
      030014      177777  .WORD  -1          ;DRIVE 6
    
```

```

030016 177777          .WORD  -1          ;DRIVE 7
141
142          ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
143          ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
144          ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
145
146 030020 177777      DTUW:  .WORD  -1
147
148          ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
149          ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
150          ;ATTENTION BIT
151
152 030022      001      ATABIT:  .BYTE  1          ;DRIVE 0
153 030023      002          .BYTE  2          ;DRIVE 1
154 030024      004          .BYTE  4          ;DRIVE 2
155 030025      010          .BYTE 10          ;DRIVE 3
156 030026      020          .BYTE 20          ;DRIVE 4
157 030027      040          .BYTE 40          ;DRIVE 5
158 030030      100          .BYTE 100         ;DRIVE 6
159 030031      200          .BYTE 200         ;DRIVE 7
160
161          ;FSRM TO RH11/RH70 'MASSBUS CONTROL BUS PARITY ERRORS' (MCPE) ALLOWED BEFORE
162          ;CALLING IT FATAL (MCPEMX=1 WORD)
163
164 030032 000003      MCPEMX: .WORD  3
165
166          ;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH/RM),
167          ;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).
168
169 030034 176700      RMADR:  .WORD 176700
170 030036 000254 000240  RMVEC:  .WORD 254,5*32.
171
172          ;MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
173 030042 000004      MXLACT: .WORD  4
174
175          ;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
176 030044 001000      MXDLTA: .WORD  8.*64.
177
178          ;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
179 030046 000200      MNDLTA: .WORD  2*64.
180
181          ;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
182 030050 000005      MXWNDW: .WORD  5
183
184          ;DEFINITIONS OF THE RH/RM ADDRESS INDEXES
185
186          000000      RMCS1=0          ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
187          000002      RMWC=2          ;WORD COUNT REGISTER (NOT A DRIVE REG)
188          000004      RMBA=4          ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
189          000006      RMDA=6          ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
190          000010      RMCS2=10         ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
191          000012      RMDS=12         ;DRIVE STATUS REGISTER (DRIVE REG 01)
192          000014      RMER1=14        ;ERROR REGISTER #1 (DRIVE REG. 02)
193          000016      RMAS=16         ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
194          000020      RMLA=20         ;LOOK AHEAD REGISTER (DRIVE REG. 07)
195          000022      RMDB=22        ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
196          000024      RMMR1=24       ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
    
```



```

197      000026      RMDT=26      ;DRIVE TYPE REGISTER (DRIVE REG. 06)
198      000030      RMSN=30      ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
199      000032      RMOF=32      ;OFFSET REGISTER (DRIVE REG. 11)
200      000034      RMDC=34      ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
201      0C0036      RMHR=36      ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
202      000040      RMMR2=40     ;MAINTENANCE REGISTER #2
203      000042      RMR2=42      ;ERROR REGISTER #2 (DRIVE REG. 15)
204      0C0044      RMEC1=44     ;ECC POSITION REGISTER (DRIVE REG. 16)
205      000046      RMEC2=46     ;ECC PATTERN REGISTER (DRIVE REG. 17)
    
```

.SBTTL RH/RM DRIVER INITIALIZATION CODE

```

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

CALL

```

        JSR    PC,RMINIT
        RETURN
    
```

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

```

221 030052 104412      RMINIT: SAVREG      ;SAVE R0 - R5
222 030054 013746 177776      MOV    PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
223 030060 012737 000240 177776      MOV    #<5*32.>,PS   ;CHANGE THE PRIORITY TO 5
224 030066 004737 035740      JSR    PC,CLRQUE     ;CLEAR ALL REQUEST QUEUES
225 030072 012701 027666      MOV    #RMERRS,R1    ;FIRST ADDRESS TO BE CLEARED
226 030076 012702 027776      MOV    #SEEKFG,R2    ;LAST ADDRESS TO BE CLEARED
227 030102 005021      1$: CLR    (R1)+        ;CLEAR
228 030104 020102      CMP    R1,R2         ;ARE WE DONE?
229 030106 103775      BLO   1$             ;BR IF NO
230 030110 012702 030020      MOV    #DTUW,R2      ;LAST ADDRESS
231 030114 012721 177777      2$: MOV    #-1,(R1)+   ;INITIALIZE
232 030120 020102      CMP    R1,R2         ;DONE?
233 030122 101774      BLOS  2$             ;LOOP IF NO
234 030124 005037 027706      CLR    DRVSTA        ;SET ALL DRIVES TO OFFLINE
235 030130 005037 027710      CLR    DRVSTA+2
236 030134 005037 027712      CLR    DRVSTA+4
237 030140 005037 027714      CLR    DRVSTA+6
238 030144 013703 030036      MOV    RMVEC,R3      ;SETUP THE RH/RM VECTOR
239 030150 012723 032706      MOV    #ISR,(R3)+
240 030154 013713 030040      MOV    RMVEC+2,(R3)
241 030160 013704 030034      MOV    RMADR,R4      ;FIRST ADDRESS OF RH/RM
242 030164 012764 000040 000010      MOV    #CLR,RMCS2(R4);MASSBUS INIT
243 030172 005001      CLR    R1             ;START WITH DRIVE 0
244 030174 004037 030264      3$: JSR    R0,DRVINT   ;INIT THE DRIVE
245 030200 000401      BR    4$              ;'DVA' NOT SET OR PARITY ERROR
246 030202 000402      BR    5$              ;NORMAL RETURN
247 030204 105061 027706      4$: CLRB  DRVSTA(R1)   ;SET DRIVE STATUS TO OFFLINE
248 030210 005201      5$: INC    R1           ;GO TO NEXT DRIVE
249 030212 042701 177770      BIC    #^C7,R1       ;MASK OUT UNUSED BITS
250 030216 001366      BNE   3$              ;BR IF MORE DRIVES TO GO
251 030220 012701 000007      MOV    #7,R1         ;START WITH DRIVE 7
252 030224 005037 177776      CLR    PS            ;CLEAR THE PROCESSOR STATUS
253 030230 105761 027726      6$: TSTB  DPINT(R1)   ;WAITING FOR DRIVE TO SWITCH PORTS ?
    
```

```

254 030234 001405          BEQ      8$          ;BR NOT WAITING
255 030236 004737 035374   JSR      PC,SET.IE    ;SET INTERRUPT
256 030242 105761 027726   7$:    TSTB     DPINT(R1) ;DRIVE SWITCHED PORTS ?
257 030246 001375          BNE      7$          ;BR IF NOT
258 030250 005301          8$:    DEC      R1      ;GO TO THE NEXT DRIVE
259 030252 100366          BPL      6$          ;CHECK NEXT DRIVE
260 030254 012637 177776   MOV      (SP)+,PS     ;RESTORE THE PROCESSOR STATUS
261 030260 104413          RESREG   ;RESTORE R0 - R5
262 030262 000207          RTS      PC          ;BYE-BYE
263
264                      ;DRIVE INITIALIZATION ROUTINE
265                      ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
266                      ;AN RM05/3/2. IF IT IS, A 'READ-IN PRESET' IS ISSUED AND FMT16
267                      ;IS SET TO A '1'. THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
268                      ;INSURE THEY ARE ALL ON A '1'. AND DEPENDING ON THEIR STATE,
269                      ;DRVSTA IS SET TO THE PROPER CONDITION.
270                      ;CALL
271                      ;
272                      ;      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
273                      ;      MOV      RMADR,R4       ;UNIBUS ADDRESS OF RH/PM (RMCS1)
274                      ;      JSR      R0,DRVINT      ;CALLED BY A JSR
275                      ;      RETURN1     ;ERROR OCCURRED (PARITY)
276                      ;      RETURN2     ;NORMAL RETURN
277
278 030264 010546          DRVINT: MOV      R5,-(SP)   ;SAVE R5
279 030266 105061 027706   DULP:  CLRB     DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
280 030272 105061 027716   CLRB     DRVSTYP(R1)     ;CLEAR THE DRIVE TYPE INDICATOR
281 030276 105061 027754   CLRB     ULDFLG(R1)     ;CLEAR THE UNLOAD FLAG
282 030302 010164 000010   MOV      R1,RMCS2(R4)   ;SELECT A DRIVE
283 030306 112764 000111 000000 MOVVB    #111,RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
284 030314 032764 010000 000010 BIT      #BIT12,RMCS2(R4) ;NONEXISTENT DRIVE?
285 030322 001403          BEQ      1$          ;NO
286 030324 004737 035374   JSR      PC,SET.IE    ;GO SET 'IE' WITHOUT A 'TRE'
287 030330 000520          BR      6$          ;LEAVE THIS ROUTINE
288
289 030332 105061 027706   1$:    CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
290 030336 032764 004000 000000 BIT      #BIT11,RMCS1(R4) ;SEE IF DRIVE AVAILABLE
291 030344 001750          BEQ      DULP        ;BR IF DRIVE NOT AVAILABLE
292 030346 004037 034704   JSR      R0,RD.RM     ;READ THE DRIVE TYPE REG.
293 030352 000026          RMDT
294 030354 030616          8$
295 030356 012605          MOV      (SP)+,R5     ;ERROR RETURN ADDRESS
296 030360 112761 000004 027716 MOVVB    #4,DRVSTYP(R1) ;PUT DRIVE TYPE IN R5
297 030366 022705 020024   CMP      #20024,R5    ;SET RM03 INDICATOR
298 030372 001431          BEQ      2$          ;SINGLE PORT RM03 ?
299 030374 022705 024024   CMP      #24024,R5    ;BR IF YES
300 030400 001426          BEQ      2$          ;DUAL PORT RM03 ?
301 030402 112761 000005 027716 MOVVB    #5,DRVSTYP(R1) ;BR IF YES
302 030410 022705 020025   CMP      #20025,R5    ;SET RM02 INDICATOR
303 030414 001420          BEQ      2$          ;SINGLE PORT RM02 ?
304 030416 022705 024025   CMP      #24025,R5    ;BR IF SO
305 030422 001415          BEQ      2$          ;DUAL PORT RM02 ?
306 030424 112761 000007 027716 MOVVB    #7,DRVSTYP(R1) ;BR IF SO
307 030432 022705 020027   CMP      #20027,R5    ;SET RM05 INDICATOR
308 030436 001407          BEQ      2$          ;SINGLE PORT RM05 ?
309 030440 022705 024027   CMP      #24027,R5    ;BR IF YES
310 030444 001404          BEQ      2$          ;DUAL PORT RM05 ?
310 030444 001404          BEQ      2$          ;BR IF YES

```

```

311 030446 112761 177777 027716      MOVB  #-1,DRVSTYP(R1)  ;SET INDICATOR TO 'OTHER'
312 030454 000446                    BR    6$              ;EXIT
313
314 030456 012746 000121 2$:      MOV    #121,-(SP)      ;DO A 'READ-IN PRESET'
315 030462 004037 035064          JSR    RO,WRT.RM
316 030466 000000                    RMCS1
317 030470 030616                    8$
318 030472 012746 010000          MOV    #BIT12,-(SP)  ;SET FMT16=1
319 030476 004037 035064          JSR    RO,WRT.RM
320 030502 000032                    RMOF
321 030504 030616                    8$
322 030506 004037 034704          JSR    RO,RD.RM      ;READ RMDS
323 030512 000012                    RMDS
324 030514 030616                    8$
325 030516 012605                    MOV    (SP)+,R5      ;AND SAVE IT IN R5
326 030520 100015                    BPL    4$            ;BR IF ATA=0
327 030522 116164 030022 000016  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
328 030530 004037 034704          JSR    RO,RD.RM      ;FIND OUT WHY ATA=1
329 030534 000014                    RMER1
330 030536 030616                    8$
331 030540 006126                    ROL    (SP)+         ;IS IT UNSAFE?
332 030542 100004                    BPL    4$            ;BR IF NOT
333 030544 112761 177777 027706  MOVB  #-1,DRVSTA(R1)  ;SET UNSAFE INDICATOR
334 030552 000407                    BR    6$              ;EXIT
335
336 030554 005105 4$:      COM    R5              ;CHECK MOL, DPR, DRY, AND VV
337 030556 042705 167077          BIC    #^C<BIT12!BIT08!BIT07!BIT06>,R5
338 030562 001003                    BNE    6$            ;BR IF MOL, DPR, DRY, OR VV IS CLEAR
339 030564 112761 000001 027706  MOVB  #1,DRVSTA(R1)  ;SET DRIVE STATUS TO ONLINE
340 030572 005720 6$:      TST    (R0)+         ;STEP OVER THE ERROR RETURN
341 030574 000410                    BR    8$              ;EXIT
342 030576 006301 7$:      ASL    R1              ;CHANGE INDEX TO ADDRESS WORDS
343 030600 012761 060000 030000  MOV    #60000,TIMER(R1) ;START 2 SEC TIMER
344 030606 006201                    ASR    R1              ;RESTORE R1
345 030610 112761 177777 027726  MOVB  #-1,DPINT(R1)  ;SET PORT INITIALIZE INIDICATOR
346 030616 012605 8$:      MOV    (SP)+,R5      ;RESTORE R5
347 030620 000200                    RTS    RO              ;EXIT
348
349                                ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
350
351                                ;CALL
352
353                                ;
354                                ;
355                                ;
356                                ;
357                                ;
358                                ;
359 030622 013746 177776 RM05:  MOV    PS,-(SP)        ;SAVE THE CALLING STATUS
360 030626 013737 030040 177776  MOV    RMVEC+2,PS    ;DON'T ALLOW ANY RM INTERRUPTS
361 030634 112737 000001 027752  MOVB  #1,ACTDRV      ;SET 'ACTIVE DRIVER' FLAG
362 030642 104412                    SAVREG                ;SAVE R0 - R5
363 030644 011002                    MOV    (R0),R2        ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
364 030646 005062 000016          CLR    16(R2)         ;CLEAR THE STATUS/ERROR INDICATOR
365 030652 111201                    MOVB  (R2),R1         ;PICKUP THE DRIVE NUMBER
366 030654 013704 030034          MOV    RMADR,R4       ;UNIBUS ADDRESS OF RMCS1
367 030660 105761 027706          TSTB  DRVSTA(R1)     ;CHECK DRIVES STATUS

```

```

368 030664 003014          BGT      1$          ;BR IF ONLINE
369 030666 105761 027754   TSTB    ULDFLG(R1)  ;UNLOAD COMMAND IN QUELE?
370 030672 001036          BNE     3$          ;BR IF YES
371 030674 105761 027726   TSTB    DPINT(R1)  ;TRYING TO INIT THE DRIVE
372 030700 001042          BNE     5$          ;BR IF YES
373 030702 004037 030264   JSR     RO,DRVINT  ;GO INIT. THE DRIVE
374 030706 000434          BR      4$          ;ERROR RETURN
375 030710 105761 027706   TSTB    DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
376 030714 003445          BLE     6$          ;BR IF NOT
377 030716 105761 027736   1$:    TSTB    DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
378 030722 001031          BNE     5$          ;BR IF YES
379 030724 010164 000010   MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
380 030730 004037 036036   JSR     RO,DRVQUE  ;PUT THIS REQUEST IN CUEUE
381 030734 000460          BR      9$          ;QUEUE IS FULL
382 030736 122762 000103 000002  CMPB    #103,2(R2)  ;IS THIS REQ. FOR AN UNLOAD?
383 030744 001003          BNE     2$          ;BR IF NO
384 030746 112761 177777 027754   MOVB    #-1,ULDFLG(R1) ;SET THE 'UNLOAD IN QUEUE' FLAG
385 030754 105761 027676   2$:    TSTB    DRVACT(R1) ;IS THIS DRIVE ACTIVE?
386 030760 001043          BNE     8$          ;BR IF YES
387 030762 004737 031114   JSR     PC,OPT     ;CALL THE OPTIMIZER
388 030766 000440          BR      8$
389 030770 012762 120000 000016  3$:    MOV     #BIT15!BIT13,16(R2) ;SET THE 'UNLOAD IN QUEUE' ERROR FLAG
390 030776 000434          BR      8$          ;EXIT
391 031000 004737 032174   4$:    JSR     PC,C17   ;GO HANDLE THE PARITY ERROR
392 031004 000431          BR      8$
393 031006 004037 036036   5$:    JSR     RO,DRVQUE  ;PUT REQUEST IN QUEUE
394 031012 000431          BR      9$          ;QUEUE IS FULL
395 031014 032714 000100   BIT     #BIT06,(R4) ;IE BIT SET ?
396 031020 001023          BNE     8$          ;YES
397 031022 004737 035374   JSR     PC,SET.IE  ;SET THE INTERRUPT
398 031026 000420          BR      8$          ;RETURN
399 031030 105761 027706   6$:    TSTB    DRVSTA(R1) ;SEE IF DRIVE OFFLINE OR UNSAFE
400 031034 002412          BLT     7$          ;BR IF UNSAFE
401 031036 012762 140000 000016  MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
402 031044 105761 027716   TSTB    DRVTP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
403 031050 001007          BNE     8$          ;BR IF OFFLINE
404 031052 012762 100002 000016  MOV     #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
405 031060 000403          BR      8$          ;GO TO EXIT
406 031062 012762 110000 000016  7$:    MOV     #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
407 031070 104413          8$:    RESREG   ;RESTORE R0 - R5
408 031072 005720          TST     (R0)+      ;SETUP FOR NORMAL RETURN
409 031074 000401          BR      10$         ;FINISH UP, THEN EXIT
410 031076 104413          9$:    RESREG   ;RESTORE R0 - R5
411 031100 005720          10$:   TST     (R0)+      ;CORRECT THE RETURN ADDRESS
412 031102 105037 027752   CLRB   ACTDRV     ;CLEAR 'ACTIVE DRIVER' FLAG
413 031106 012637 177776   MOV     (SP)+,PS   ;RETURN 'PS' TO USER LEVEL
414 031112 000200          RTS     RO         ;RETURN TO CALLER
415
416 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
417
418 ;CALL
419
420 ;
421 ;
422 031114 104412          OPT:   SAVREG   ;SAVE R0 - R5
423 031116 013746 177776   MOV     PS,-(SP)   ;SAVE PROC. STATUS
424 031122 146137 030022 027750  BICB   ATABIT(R1),SRCHWT ;CLEAR LA SEACH FLAG

```

```

425 031130 105061 027736          CLRB   DPRQS(R1)      ;RESET THE PORT REQ FLAG ****
426 031134 004737 036112          JSR    PC,GETREQ    ;GET "DPB" POINTER OF REQUEST
427 031140 005702                   TST    R2           ;IS THERE A REQUEST IN QUEUE?
428 031142 001472                   BEQ    7$           ;NO--BR TO EXIT
429 031144 010164 000010          MOV    R1,RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****
430 031150 012764 000111 000000  MOV    #11,RMCS1(R4) ;CLEAR THE DRIVE
431 031156 032764 004000 000000  BIT    #BIT11,RMCS1(R4) ;DVA SET ?
432 031164 001446                   BEQ    5$           ;TO PROT REQUEST ,IF NOT
433 031166 105761 027706          10$:  TSTB   DRVSTA(R1)   ;IS DRIVE ONLINE?
434 031172 003014                   BGT    1$           ;YES
435 031174 004737 036134          JSR    PC,POPQUE    ;NO--REMOVE REQUEST FROM QUEUE
436 031200 012762 140000 000016  MOV    #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
437 031206 105761 027706          TSTB   DRVSTA(R1)   ;IS DRIVE UNSAFE ?
438 031212 100053                   BPL    8$           ;BR TO EXIT IF NOT
439 031214 012762 110000 000016  MOV    #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
440 031222 000447                   BR     8$           ;BR TO EXIT
441 031224                   1$:
442                   :
443                   :
444                   :
445                   :
446                   :
447                   :
448 031224 122762 000150 000002  MOV    #111,-(SP)    ;LOAD COMMAND ONTO THE STACK
449 031232 002403                   JSR    RO,WRT.RM    ;LOAD THE REGISTER
450 031234 004737 031560          RMCS1  ;REGISTER INCREMENT
451 031240 000440                   6$
452 031242 005737 030020          2$:  BIT    #BIT11,(R4)  ;DRIVE AVAILABLE ?
453 031246 002012                   BEQ    9$           ;BR IF NOT
454 031250 005737 027776          CMPB   #150,2(R2)   ;IS THE REQUEST FOR I/O?
455 031254 100404                   BLT    2$           ;YES
456 031256 004037 032532          JSR    PC,C14       ;CALL THE COMMAND INITIATOR
457 031262 000427                   BR     8$           ;BR TO EXIT
458 031264 000403                   TST    DTUW         ;DATA TRANSFER UNDERWAY?
459 031266 004737 031352          3$:  BGE    4$           ;YES--GO START A SEARCH
460 031272 000423                   TST    SEEKFG       ;DO IMPLIED SEEKS?
461 031274 004737 031460          4$:  BMI    3$           ;YES
462 031300 000420                   JSR    RO,LA        ;NO--DO LOOK AHEAD
463 031302 112761 177777 027736  5$:  BR     8$           ;RETURN HERE ON A PARITY ERROR
464 031310 010103                   BR     4$           ;GO START A SEARCH
465 031312 006303                   JSR    PC,C11       ;START A DATA TRANSFER
466 031314 012763 060000 030000  6$:  BR     8$           ;START A SEARCH
467 031322 000402                   JSR    PC,C13       ;GO TO THE EXIT
468 031324 004737 032174          7$:  BR     8$           ;SET PORT REQUEST INDICATOR
469 031330 032714 000100          MOV    #-1,DPRQS(R1) ;SET UP TO ADDRESS WORDS
470 031334 001002                   MOV    R1,R3        ;SET UP TO ADDRESS WORDS
471 031336 004737 035374          ASL    R3           ;CONVERT TO WORD INDEX
472 031342 012637 177776          8$:  MOV    #60000,TIMER(R3) ;START 10 SEC TIMER
473 031346 104413                   BR     7$           ;EXIT
474 031350 000207                   JSR    PC,C17       ;PROCESS THE PARITY ERROR
475                   :
476                   :
477                   :
478                   :
479                   :
480                   :
481                   :

```

:COMMAND INITIATOR

:CALL

```

MOV    #DRVNUM,R1    ;DRIVE NUMBER
MOV    #DPB,R2       ;ADDRESS OF DPB
JSR    PC,C1?        ;C1?= C11,C13, OR C14

```


539	031612	000006			RMDA			
540	031614	032174			C17			
541	031616	000403			BR	2\$:GO LOAD CYLINDER
542	031620	122703	000105	1\$:	CMPB	#105,R3		:IS IT A SEEK COMMAND
543	031624	001007			BNE	3\$:BR IF NO
544	031626	016246	000012	2\$:	MOV	12(R2),-(SP)		:LOAD DESIRED CYLINDER
545	031632	004037	035064		JSR	RO,WRT.RM		
546	031636	000034			RMDC			
547	031640	032174			C17			
548	031642	000546			BR	C16		
549	031644	122703	000115	3\$:	CMPB	#115,R3		:IS IT AN 'OFFSET' COMMAND?
550	031650	001013			BNE	4\$:BR IF NO
551	031652	004037	034704		JSR	RO,RD.RM		:MERGE THE OFFSET VALUE INTO RMOF
552	031656	000032			RMOF			:BUT DON'T CHANGE THE UPPER
553	031660	032174			C17			
554	031662	116216	000001		MOVB	1(R2),(SP)		:BYTE WHEN LOADING THE
555	031666	004037	035064		JSR	RO,WRT.RM		:REGISTER (RMOF)
556	031672	000032			RMOF			
557	031674	032174			C17			
558	031676	000530			BR	C16		:GO START THE COMMAND
559	031700	122703	000107	4\$:	CMPB	#107,R3		:IS IT A 'RECALIBRATE' COMMAND?
560	031704	001525			BEQ	C16		:BR IF YES
561	031706	122703	000117		CMPB	#117,R3		:IS IT A RETURN TO CENTER?
562	031712	001522			BEQ	C16		:BR IF YES
563	031714	122703	000103		CMPB	#103,R3		:IS IT AN 'UNLOAD' COMMAND?
564	031720	001016			BNE	5\$:BR IF NO
565	031722	112761	000001	027676	MOVB	#1,DRVACT(R1)		:SET THE DRIVE ACTIVE INDICATOR
566	031730	105061	027706		CLRB	DRVSTA(R1)		:PUT DRIVE STATUS TO OFFLINE
567	031734	112761	000001	027754	MOVB	#1,ULDFLG(R1)		:SET 'UNLOAD IN PROGRESS' FLAG
568	031742	010346			MOV	R3,-(SP)		:START THE 'UNLOAD' COMMAND
569	031744	004037	035064		JSR	RO,WRT.RM		
570	031750	000000			RMCS1			
571	031752	032174			C17			
572	031754	000207			RTS	PC		:RETURN TO USER
573	031756	122703	000143	5\$:	CMPB	#143,R3		:IS IT A 'SET FORMAT' COMMAND?
574	031762	001014			BNE	6\$:BR IF NO
575	031764	004037	034704		JSR	RO,RD.RM		:READ THE OFFSET REGISTER
576	031770	000032			RMOF			
577	031772	032174			C17			
578	031774	116266	000001	000001	MOVB	1(R2),1(SP)		:COMBINE 'FMT16','ECI', AND 'HCI'
579	032002	004037	035064		JSR	RO,WRT.RM		:LOAD 'FMT16', 'ECI', AND/OR 'HCI'.
580	032006	000032			RMOF			
581	032010	032174			C17			
582	032012	000436			BR	12\$		
583	032014	122703	000141	6\$:	CMPB	#141,R3		:IS IT A 'GET REGISTER' COMMAND?
584	032020	001023			BNE	10\$:BR IF NO
585	032022	016203	000006	7\$:	MOV	6(R2),R3		:POINTS TO 1ST ADDRESS OF WHERE
586								:TO PUT THE REGISTER(S)
587	032026	116237	000010	032044	MOVB	10(R2),9\$:INIT. THE INDEX FOR THE FIRST REG.
588	032034	116205	000011		MOVB	11(R2),R5		:INDEX OF LAST REG. TO MOVE
589	032040	004037	034704	8\$:	JSR	RO,RD.RM		:READ RH/RM REGISTER
590	032044	000000		9\$:	RMCS1			:INDEX OF REG. TO READ
591	032046	032174			C17			
592	032050	012623			MOV	(SP)+,(R3)+		:GET THE CONTENTS OF RH/RM REG.
593	032052	023705	032044		CMP	9\$,R5		:LAST REG. BEEN READ?
594	032056	001414			BEQ	12\$:GET OUT IF YES
595	032060	062737	000002	032044	ADD	#2,9\$:INCREASE THE INDEX BY 2

```

596 032066 000764          BR      8$          ;LOOP--MORE TO READ
597 032070 122703 000145 10$:  CMPB   #145,R3      ;IS IT A 'SELECT DRIVE' COMMAND?
598 032074 001405          BEQ    12$          ;BR IF YES
599 032076 010346          11$:  MOV    R3,-(SP)    ;LOAD THE COMMAND
600 032100 004037 035064    JSR    R0,WRT.RM
601 032104 000000          RMCS1
602 032106 032174          CI7
603 032110 004737 036134          12$:  JSR    PC,POPQUE   ;REMOVE REQ. FROM QUEUE
604 032114 052762 000200 000016  BIS    #BIT07,16(R2) ;SET THE 'DONE' BIT
605 032122 005737 027774          TST    SAVEFG      ;SAVE THE RH/RM REGISTERS?
606 032126 100002          BPL    13$          ;BR IF NO
607 032130 004737 035256          JSR    PC,SVRH70   ;YES--GO SAVE THE REGISTERS
608 032134 000207          13$:  RTS    PC          ;RETURN TO USER
609 032136 006301          CI5:  ASL    R1
610 032140 012761 060000 030000  MOV    #60000,TIMER(R1) ;SET A ONE SECOND TIMER
611 032146 006201          ASR    R1
612 032150 112761 000001 027676  MOVB   #1,DRVACT(R1) ;SET THE DRIVE ACTIVE
613 032156 000207          RTS    PC          ;RETURN TO THE USER
614 032160 010346          CI6:  MOV    R3,-(SP)    ;LOAD THE COMMAND
615 032162 004037 035064    JSR    R0,WRT.RM
616 032166 000000          RMCS1
617 032170 032174          CI7
618 032172 000761          BR
619 032174 032764 010000 000010  CI17: BIT    #BIT12,RMCS2(R4) ;DRIVE NON-EXISTENT ?
620          :          CI8
621 032202 005702          1$:  TST    R2          ;BR IF YES
622          :          ;ANYTHING IN QUEUE ?
623 032204 001001          :          BEQ    L17B     ;BR IF NOT
624 032206 000207          BNE    2$          ;BR IF QUEUE IS THERE
625 032210 012762 104000 000016  2$:  RTS    PC          ;OTHERWISE EXIT
626          :          MOV    #BIT15!BIT11,16(R2) ;SET 'PARITY' ERROR INDICATOR
627 032216 012746 000111          CI17B: JSR    PC,SVRH70   ;GO SAVE THE RH/RM REGISTERS
628 032222 004037 035064          MOV    #111,-(SP)  ;DO A 'DRIVE CLEAR'
629 032226 000000          JSR    R0,WRT.RM
630 032230 032274          RMCS1
631 032232 004737 036016          CI8
632 032236 105061 027736          2$:  JSR    PC,EMPTYQ  ;EMPTY THE QUEUE
633 032242 105061 027754          CLRB   DPRQS(R1)   ;CLEAR THE PORT REQUEST FLAG
634 032246 105061 027676          CLRB   ULDFLG(R1) ;CLEAR THE UNLOAD IN QUEUE FLAG
635 032252 020237 027746          CLRB   DRVACT(R1) ;DRIVE IS IDLE
636          :          CMP    R2,TRNSWT ;IF THIS DRIVE HAD AN I/O REQUEST
637 032256 001005          :          CMP    R1,DTUW  ;IF THIS DRIVE HAD AN I/O REQUEST
638 032260 005037 027746          BNE    1$          ;IN PROGRESS CLEAR ALL OF THE FLAGS
639 032264 012737 177777 030020  CLR    TRNSWT
640 032272 000207          1$:  RTS    PC
641 032274 104412          CI18: SAVREG
642 032276 032764 010000 000010  BIT    #BIT12,RMCS2(R4) ;SAVE R0 - R5
643          :          ;IS 'NED' SET ?
644 032304 005001          BNE    1$          ;BR IF YES
645 032306 005003          CLR    R1
646 032310 105761 027676          1$:  CLR    R3
647          :          TSTB   DRVACT(R1) ;DRIVE ACTIVE?
648 032314 001003          :          BEQ    5$          ;BR IF NO
649 032316 105761 027736          BNE    22$         ;BR IF IN ACTIVE
650 032322 001443          TSTB   DPRQS(R1)  ;PORT REQUEST
651 032324 013702 027746          BEQ    5$          ;BR IF NOT
652 032330 020137 030020  22$:  MOV    TRNSWT,R2  ;GET THE 'TRANSFER WAIT' QUEUE
          :          CMP    R1,DTUW  ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?

```



```

653 032334 001402          BEQ      2$          ;BR IF YES
654 032336 004737 036112   JSR      PC,GETREQ  ;GET THE DPB POINTER
655 032342 005702          2$:    TST      R2      ;QUEUE ENTRY FOR DRIVE ?
656 032344 001413          BEQ      4$          ;BR IF NOT
657 032346 032764 010000 000010  BIT     #BIT12,RMCS2(R4) ;'NED' SET ?
658 032354 001404          BEQ      3$          ;BR IF NOT
659 032356 012762 100002 000016  MOV     #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
660 032364 000403          BR       4$          ;CONTINUE
661 032366 012762 102000 000016  3$:    MOV     #BIT15!BIT10,16(R2) ;SET 'NON-CLEARABLE PARITY' ERROR INDICATOR
662                                ;
663 032374 012763 177777 030000  4$:    JSR      PC,SVRH70  ;SAVE RH/RM REGISTERS
664 032402 105061 027676          MOV     #-1,TIMER(R3) ;STOP THE TIMER
665 032406 105061 027736          CLRB   DRVACT(R1)   ;SET 'DRIVE ACTIVE' TO IDLE
666 032412 020137 030020          CLRB   DPRQS(R1)   ;CLEAR PORT REQUEST FLAG
667 032416 001005          CMP     R1,DTUW    ;IS THIS DRIVE SETUP FOR A TRANSFER
668 032420 012737 177777 030020  BNE     5$          ;BR IF NOT
669 032426 005037 027746          MOV     #-1,DTUW   ;RESET THE INDICATOR
670 032432 105061 027754          CLR    TRNSWT      ;CLEAR THE TRANSFER QUEUE
671 032436 032764 010000 000010  5$:    CLRB   ULDFLG(R1) ;CLEAR UNLOAD FLAG
672                                BIT     #BIT12,RMCS2(R4) ;'NED' SET ?
673 032444 005201          BNE     6$          ;BR IF YES
674 032446 062703 000002          INC    R1          ;MOVE TO THE NEXT DRIVE
675 032452 042701 177770          ADD    #2,R3
676 032456 001314          BIC    #^C7,R1
677 032460 012737 177777 030020  BNE     1$          ;BR IF MORE DRIVES
678 032466 005037 027746          MOV     #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
679 032472 004737 035740          CLR    TRNSWT      ;CLEAR THE 'TRANSFER WAIT' QUEUE
680 032476 012764 000040 000010  JSR     PC,CLRQUE   ;CLEAR ALL OF THE REQUEST QUEUES
681 032504 000406          MOV     #CLR,RMCS2(R4) ;DO A MASSBUS INIT.
682 032506 004737 036016          BR     7$          ;CONTINUE
683 032512 105061 027706          6$:    JSR     PC,EMPTYQ ;CLEAR THE DRIVE'S QUEUE
684 032516 105061 027716          CLRB   DRVSTA(R1)  ;SET DRIVE TO OFFLINE
685 032522 004737 035374          CLRB   DRVTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
686 032526 104413          7$:    JSR     PC,SET.IE ;SET 'IE' WITHOUT 'TRE'
687 032530 000207          RESREG ;RESTORE R0 - R5
688                                RTS     PC          ;RETURN
689                                ;LOOK AHEAD ROUTINE
690                                ;
691                                ;CALL
692                                ;
693                                MOV     #DRVNUM,R1   ;DRIVE NUMBER
694                                MOV     #DPB,R2      ;POINT TO DPB
695                                JSR     RO,LA        ;GO CHECK THE WINDOW
696                                RETURN1           ;ERROR RETURN
697                                RETURN2           ;START A SEARCH
698                                RETURN3           ;START A DATA TRANSFER
699 032532 013704 030034          LA:    MOV     RMADR,R4 ;GET RMCS1'S ADDRESS
700 032536 010164 000010          MOV     R1,RMCS2(R4) ;SELECT DRIVE
701 032542 004037 034704          JSR     RO,RD.RM    ;READ DRIVE STATUS
702 032546 000012          RMDS   4$          ;
703 032550 032700          4$:    ;ERROR RETURN ADDRESS
704 032552 042716 157577          BIC    #^C020200,(SP) ;ON CYLINDER ?
705 032556 022726 000200          CMP    #200,(SP)+  ;PIP=0,DRY=1?
706 032562 001044          BNE    3$          ;NO
707 032564 105261 027764          INCB   LACNT(R1)   ;INCREMENT THE LOOK AHEAD COUNT
708 032570 126137 027764 030042  CMPB   LACNT(R1),MXLACT ;EXCEED MAX?
709 032576 003033          BGT    2$          ;BR IF YES

```

```

710 032600 116203 000010          MOVB 10(R2),R3      ;GET DESIRED SECTOR ADDRESS AND
711 032604 000303          SWAB  R3           ;MULT. BY 64--ALIGN WITH
712 032606 006203          ASR   R3           ;LOOK AHEAD REGISTER
713 032610 006203          ASR   R3
714 032612 012737 000340 177776 6$:  MOV  #340,PS      ;PRIORITY LEVEL '7'
715 032620 004037 034704      JSR  RO,RD.RM     ;READ LOOK AHEAD REGISTER
716 032624 000020          RMLA 4$
717 032626 032700          4$
718 032630 021664 000020          CMP  (SP),RMLA(R4) ;CORRECT LA NUMBER ?
719 032634 001402          BEQ  7$           ;YES
720 032636 005726          TST  (SP)+        ;NO,CLEAR STACK
721 032640 000415          BR   3$
722 032642 162603          7$:  SUB  (SP)+,R3     ;CALCULATE THE DELTA
723 032644 002002          BGE  1$
724 032646 062703 004000          ADD  #<32.*64.>,R3 ;MAKE THE DELTA POSITIVE
725 032652 023703 030044          1$:  CMP  MXDLTA,R3  ;CHECK THE DELTA TO SEE
726 032656 002406          BLT  3$           ;IF IT IS WITHIN THE
727 032660 023703 030046          CMP  MNDLTA,R3   ;WINDOW---IF YES, ZERO
728 032664 002003          BGE  3$           ;THE LOOK AHEAD COUNT
729 032666 105061 027764          2$:  CLRB LACNT(R1) ;AND TAKE THE I/O EXIT
730 032672 005720          TST  (R0)+
731 032674 005720          3$:  TST  (R0)+   ;ADJUST THE RETURN ADDRESS
732 032676 000402          BR   5$           ;EXIT
733 032700 004737 032174          4$:  JSR  PC,C17    ;PROCESS THE ERROR
734 032704 000200          5$:  RTS  RO       ;RETURN
735
736          ;INTERRUPT SERVICE ROUTINE
737
738 032706 112737 000001 027752 1SR: MOVB #1,ACTDRV   ;SET 'ACTIVE DRIVER' FLAG
739 032714 104412          SAVREG          ;SAVE R0 - R5
740 032716 013704 030034          MOV  RMADR,R4    ;ADDRESS OF RHSCS1
741 032722 013701 030020          MOV  DTUW,R1     ;GET 'DATA TRANSFER UNDERWAY' INDICATOR
742 032726 002403          BLT  1$           ;BR IF NO DATA TRANSFER UNDERWAY
743 032730 004737 032752          JSR  FC,TD       ;CALL TRANSFER DONE
744 032734 000402          BR   2$
745 032736 004737 033122          1$:  JSR  PC,SC     ;CALL SPECIAL CONDITIONS
746 032742 104413          2$:  RESREG          ;RESTORE R0 - R5
747 032744 105037 027752          CLRB ACTDRV     ;CLEAR 'ACTIVE DRIVER' FLAG
748 032750 000002          RTI           ;RETURN
749
750          ;TRANSFER DONE ROUTINE
751
752 032752 105061 027676          TD:  CLRB DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
753 032756 012737 177777 030020      MOV  #-1,DTUW     ;NO DATA TRANSFERS UNDERWAY
754 032764 006301          ASL  R1
755 032766 012761 177777 030000      MOV  #-1,TIMER(R1) ;CANCEL TIMEOUT
756 032774 006201          ASR  R1
757 032776 013702 027746          MOV  TRNSWT,R2   ;GET 'DPB' ADDRESS FROM THE
758 033002 005037 027746          CLR  TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
759 033006 052762 000200 000016      BIS  #BIT07,16(R2) ;SET DONE
760 033014 010164 000010          MOV  R1,RMCS2(R4) ;SELECT THE DRIVE
761 033020 004037 034704          JSR  RO,RD.RM     ;TRANSFER ERROR(TRE=1)?
762 033024 000000          RMCS1
763 033026 032174          C17
764 033030 006126          ROL  (SP)+
765 033032 100417          BMI  3$
766 033034 005737 027774          TST  SAVEFG      ;BR IF YES
                          ;SAVE THE RH/RM REGISTERS?

```

```

767 033040 100002          BPL      1$          ;BR IF NO
768 033042 004737 035256    JSR      PC,SVRH70    ;YES--SAVE THE REGISTERS
772 033046          1$:          ;
774 033046 004737 036112    JSR      PC,GETREQ    ;GET DPB POINTER
775 033052 005702          TST      R2          ;ENTRY FOR DRIVE ?
776 033054 001403          BEQ      2$          ;BR IF NOT
777 033056 004737 031114    JSR      PC,OPT      ;CALL OPTIMIZER
778 033062 000417          BR       SC          ;CHECK OTHER DRIVES
779          ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
780 033064 012714 000113    2$:      MOV      #113,(R4) ;RELEASE THE DRIVE
781 033070 000414          GR       SC          ;CHECK FOR OTHER DRIVES
782 033072 052762 100100 000016 3$:      BIS      #BIT15!BIT06,16(R2) ;SET DATA ERROR FLAG
783 033100 004737 036016    JSR      PC,EMPTYQ   ;EMPTY THE 'DRIVE'S WAIT' QUEUE
784 033104 004737 035256    JSR      PC,SVRH70   ;SAVE THE RH/RM REGISTERS
785 033110 012714 040111    MOV      #40111,(R4) ;ISSUE A 'DRIVE CLEAR'
786 033114 012714 000113    MOV      #113,(R4)  ;ISSUE A RELEASE TO THE DRIVE
787 033120 000400          BR       SC          ;CHECK FOR OTHER DRIVES
788
789
806
807          ;SPECIAL CONDITION ROUTINE
808
809 033122 116403 000016    SC:      MOVVB   RMAS(R4),R3 ;READ 'RMAS'
810 033126 001014          BNE      2$          ;BR IF ANY 'ATA' BITS SET
811 033130 004037 034704    JSR      RO,RD.RM    ;READ CONTROL AND STATUS REGISTER
812 033134 000000          RMCS1
813 033136 032274          C18
814          ;
815 033140 106126          1$:      ROLB    (SP)+      ;EXIT IF FAIL TO READ
816 033142 100405          BMI     1$          ;IS 'IE'=1?
817 033144 004037 036202    JSR      RO,ES.SAV   ;YES, NO DRIVES TO CHECK
818 033152 004737 035374    EMT     1            ;SAVE THE ADDRESS IN '$ESCAPE'
819 033156 000207          JSR      PC,SET.IE  ;REPORT AN ILLEGAL INTERRUPT
820 033160 005046          1$:      RTS     PC          ;SET INTERRUPT ENABLE
821 033162 110316          2$:      CLR     -(SP)    ;RETURN
822 033164 012703 000001    MOVVB   R3,(SP)     ;PROCESS ALL DRIVES THAT HAVE
823 033170 005001          MOV     #1,R3       ;AN 'ATA'=1
824 033172 030316          CLR     R1
825 033174 001005          SC3:    BIT     R3,(SP) ;ATA=1?
826 033176 005201          BNE     SC5         ;YES
827 033200 106303          SC4:    INC     R1     ;MOVE TO THE NEXT DRIVE
828 033202 001373          ASLB   R3
829 033204 005726          BNE     SC3         ;BR IF MORE TO CHECK?
830 033206 000207          TST    (SP)+       ;CLEAN OFF THE STACK
831 033210 105761 027726    RTS     PC          ;RETURN TO USER
832 033214 001402          SC5:    TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
833 033216 000137 034134    BEQ     1$          ;BR IF NOT
834 033222 105761 027736    JMP     SC13        ;PROCESS THE DRIVE
835 033226 001402          1$:      BEQ     2$          ;PORT REQUEST OUTSTANDING ?
836 033230 000137 034134    JMP     SC13        ;BR IF NOT
837 033234 105761 027706    2$:      TSTB   DRVSTA(R1) ;START THE OUTSTANDING COMMAND
838 033240 003023          BGT     5$          ;CHECK THE DRIVE STATUS
839 033242 105761 027754    TSTB   ULDFLG(R1)  ;BR IF ONLINE
840 033246 003420          BLE     5$          ;UNLOAD IN PROGRESS?
841 033250 004737 036112    JSR     PC,GETREQ   ;BR IF NOT
842 033254 004737 035256    JSR     PC,SVRH70   ;GET DPB POINTER
                        JSR     PC,SVRH70   ;SAVE THE RH/RM REGISTERS

```

```

843 033260 004737 034064      JSR      PC,SC12      ;SAVE RMD5, RMER1, RMER2, AND RMMR2
844                                ;ALSO DO A DRIVE INIT (DRVINT)
845 033264 105761 027706      TSTB    DRVSTA(R1)   ;DID DRIVE COME ONLINE?
846 033270 003414                BLE     6$           ;NO
847 033272 032737 040000 027666  BIT     #BIT14,RMERRS ;WAS THERE AN ERROR?
848 033300 001000                BNE     3$           ;BR IF ERROR
849                                JMP     SC11         ;NO ERROR
850 033302 013705 027670      3$:    MOV     RMERRS+2,R5 ;YES -- PICKUP RMER1 AND
851 033306 000504                BR      SC6A        ;GO PROCESS THE ERROR
852 033310 105761 027676      5$:    TSTB    DRVACT(R1) ;DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
853 033314 001033                BNE     SC6         ;BR IF EITHER
854 033316 004737 034064      JSR      PC,SC12      ;SAVE RMD5, RMER1, RMER2, AND RMMR2
855                                ;ALSO DO A DRVINT
856 033322 105761 027726      6$:    TSTB    DPINT(R1) ;TRYING TO INIT THE DRIVE ?
857 033326 001323                BNE     SC4         ;BR IF YES, CHECK ON MORE DRIVES
858 033330 105761 027706      TSTB    DRVSTA(R1)   ;CHECK ON DRIVE'S STATUS
859 033334 100412                BMI     7$           ;BR IF UNSAFE
860 033336 032737 020000 027672  BIT     #BIT13,RMERRS+4 ;ADDRESS PLUG CHANGED ?
861 033344 001013                BNE     8$           ;BR IF YES
862                                MOV     #113,-(SP)   ;RELEASE COMMAND
863 033346 012746 000111      MOV     #111,-(SP)   ;DRIVE CLEAR
864 033352 004037 035064      JSR      RO,WRT.RM   ;WRITE THE COMMAND INTO RMCS1
865 033356 000000                RMCS1
866 033360 033724                SC8              ;REGISTER INDEX
867 033362 011605                MOV     (SP),R5     ;PARITY EXIT ADDRESS
868 033364 004037 036202      7$:    JSR      RO,ES.SAV ;PICKUP (RMAS) BEFORE THE ERROR CALL
869 033370 104002                EMT     2           ;SAVE THE ADDRESS IN '$ESCAPE'
870 033372 000701                BR      SC4         ;REPORT THE UNEXPECTED ATTENTION
871 033374 004037 036202      8$:    JSR      RO,ES.SAV ;GO CHECK FOR MORE ATA'S
872 033400 104005                EMT     5           ;SAVE THE ADDRESS IN '$ESCAPE'
873 033402 000675                BR      SC4         ;REPORT THE ADDRESS PLUG CHANGE
874 033404 006301                ASL     R1          ;CHECK FOR MORE DRIVES
875 033406 012761 177777 030000  SC6:   MOV     #-1,TIMER(R1) ;SETUP TO ADDRESS WORDS
876 033414 006201                ASR     R1          ;STOP THE TIMER
877 033416 004737 036112      JSR      PC,GETREQ   ;RESTORE THE DRIVE ADDRESS
878 033422 010164 000010      MOV     R1,RMCS2(R4) ;GET THE DPB POINTER FROM THE QUEUE
879 033426 000137 033754      JMP     SC11         ;SELECT DRIVE
880 033432 004037 034704      JSR      RO,RD.RM   ;PROCESS THE SEARCH
881 033436 000012                RMD5
882 033440 033724                SC8              ;READ THE RM'S STATUS REG.
883 033442 011605                MOV     (SP),R5     ;AND PUT IT IN R5
884 033444 006126                ROL     (SP)+       ;WAS THERE AN ERROR?
885 033446 100407                BMI     1$         ;BR IF ERROR
886 033450 105761 027676      TSTB    DRVACT(R1)   ;CHECK DRIVE'S STATE
887 033454 003137                BGT     SC11        ;BR IF DRIVE ACTIVE WITH ORDER
888 033456 052762 100210 000016  BIS     #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
889 033464 000470                BR      SC7         ;
890 033466 004037 034704      1$:    JSR      RO,RD.RM   ;READ ERROR REGISTER #1
891 033472 000014                RMER1
892 033474 033724                SC8
893 033476 012605                MOV     (SP)+,R5    ;AND SAVE IT IN R5
894 033500 004737 035256      JSR      PC,SVRH70   ;SAVE RH/RM REGISTERS
895 033504 012746 000111      MOV     #111,-(SP)  ;ISSUE A DRIVE CLEAR
896 033510 004037 035064      JSR      RO,WRT.RM
897 033514 000000                RMCS1
898 033516 033724                SC8

```

```

897 033520 006105          SC6A:  ROL    R5          ;WAS 'UNSAFE' CONDITION =1?
898 033522 100406          BMI    1$          ;BR IF YES
899 033524 005702          TST   R2          ;ANYTHING IN QUEUE ?
900 033526 001447          BEQ   SC7          ;BR IF NOT
901 033530 052762 100240 000016  BIS   #BIT15:BIT07.BIT05,16(R2) ;INFORM USER OF ERROR
902 033536 000443          BR    SC7
903 033540 004037 034704 1$:  JSR   RO,RD.RM    ;READ DRIVE STATUS REG. #1
904 033544 000012          RMDS
905 033546 033724          SC8
906 033550 011605          MOV   (SP),R5    ;SAVE RMDS IN R5
907 033552 006126          ROL   (SP)+      ;'ERR'=1?
908 033554 100011          BPL   2$          ;BR IF NO--UNSAFE CLEARED
909 033556 112761 177777 027706  MOVB  #-1,DRVSTA(R1) ;DRIVE IS UNSAFE
910 033564 004737 035256          JSR   PC,SVRH70  ;SAVE RH/RM REGISTERS
911 033570 052762 110000 000016  BIS   #BIT15:BIT12,16(R2) ;INFORM USER OF UNSAFE ERROR
912 033576 000423          BR    SC7
913 033600 032705 010000 2$:  BIT   #BIT12,R5   ;'MOL' = 1 ?
914 033604 001015          BNE   3$          ;BR IF YES
915 033606 112761 177777 027676  MOVB  #-1,DRVALT(R1) ;ACTIVE ERROR RECOVER
916 033614 112761 000001 027706  MOVB  #1,DRVSTA(R1) ;ONLINE
917 033622 006301          ASL   R1
918 033624 012761 072460 030000  MOV   #30000.,TIMER(R1) ;START 30 SECOND TIMER
919 033632 006201          ASR   R1
920 033634 000137 033176          JMP   SC4
921 033640 052762 100220 000016 3$:  BIS   #BIT15.BIT07.BIT04,16(R2) ;INFORM USER OF ERROR
922 033646 105061 027676          SC7:  CLRB  DRVACT(R1) ;DRIVE IS IDLE
923          ; JSR   PC,EMPTYQ ;DUMP THE QUEUE
924 033652 004737 036134          ; JSR   PC,POPQUE ;REMOVE THE QUEUE
925 033656 105761 027754          TSTB  ULDFLG(R1) ;UNLOAD IN RMOGRESS OR QUEUE?
926 033662 003002          BGT   1$          ;BR IF NOT
927 033664 105061 027754          CLRB  ULDFLG(R1) ;CLEAR UNLOAD FLAG
928 033670 116164 030022 000016 1$:  MOVB  ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
929 033676 105761 027706          TSTB  DRVSTA(R1) ;IS THE DRIVE UNSAFE ?
930 033702 100406          BMI   2$          ;BR IF IT IS
931          ; MOV   #113,-(SP) ;RELEASE COMMAND
932 033704 012746 000111          MOV   #111,-(SP) ;DRIVE CLEAR COMMAND
933 033710 004037 035064          JSR   RO,WRT.RM  ;WRITE THE COMMAND INTO RPCS1
934 033714 000000          RMCS1
935 033716 033724          SC8
936 033720 000137 033176          2$:  JMP   SC4          ;CHECK FOR MORE DRIVES
937 033724 105761 027676          SC8:  TSTB  DRVACT(R1) ;IS DRIVE IDLE?
938 033730 001405          BEQ   1$          ;YES
939 033732 004737 036112          JSR   PC,GETREQ ;GET DPB POINTER
940 033736 004737 032174          JSR   PC,C17    ;PROCESS THE PARITY ERROR
941 033742 000402          BR    2$          ;CONTINUE
942 033744          1$:
943          ; JSR   PC,C17 ;PROCESS THE PARITY ERROR
944 033744 004737 032216          ; JSR   PC,C17B ;PROCESS THE UNCORRECTABLE PARITY ERROR
945 033750 000137 033176          2$:  JMP   SC4          ;CHECK MORE DRIVES
946 033754 105761 027754          SC11: TSTB  ULDFLG(R1) ;'UNLOAD IN PROGRESS'?
947 033760 003402          BLE   1$          ;BR IF NO
948 033762 105061 027754          CLRB  ULDFLG(R1) ;CLEAR UNLOAD FLAG
949 033766 105061 027676          1$:  CLRB  DRVACT(R1) ;SET DRIVE IDLE
950 033772 136137 030022 027750  BITB  ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
951          ; AN I/O COMMAND?
952 034000 001012          BNE   2$          ;BR IF YES
953 034002 004737 036134          JSR   PC,POPQUE ;REMOVE REQUEST FROM QUEUE

```

```

954 034006 052762 000200 000016      BIS      #BIT07,16(R2)      ;SET 'DONE' BIT
955 034014 005737 027774              TST      SAVEFG        ;SAVE THE REGISTERS?
956 034020 100002              BPL      2$           ;BR IF NO
957 034022 004737 035256      JSR      PC,SVRH70    ;YES--SAVE ALL OF THE RH/RM REG'S
958 034026 116164 030022 000016 2$:  MOVB    ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
959 034034 146137 030022 027750      BICB    ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
960 034042 006301              ASL      R1           ;WORD INDEX
961 034044 012761 177777 030000      MOV      #-1,TIMER(R1) ;STOP CLOCK
962 034052 006201              ASR      R1           ;RESTORE R1
963 034054 004737 031114      JSR      PC,OPT      ;START A REQUEST
964 034060 000137 033176      JMP      SC4         ;CHECK FOR MORE DRIVES
965 034064 010164 000010      SC12:   MOV      R1,RMCS2(R4) ;SELECT DRIVE
966 034070 016437 000012 027666      MOV      RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
967 034076 016437 000014 027670      MOV      RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
968 034104 016437 000042 027672      MOV      RMER2(R4),RMERRS+4
969 034112 016437 000040 027674      MOV      RMMR2(R4),RMERRS+6
970 034120 004037 030264      JSR      RO,DRVINT   ;INIT. THE STATE OF THE DRIVE
971 034124 000401              BR      1$          ;TAKE ERROR EXIT
972 034126 000207              RTS      PC          ;RETURN
973 034130 005726      1$:    TST      (SP)+      ;POP PC OFF OF THE STACK
974 034132 000674              BR      SC8        ;PROCESS THE PARITY ERROR
975 034134 006301      SC13:  ASL      R1           ;SETUP TO ADDRESS WORDS
976 034136 012761 177777 030000      MOV      #-1,TIMER(R1) ;STOP THE TIMER
977 034144 006201              ASR      R1           ;
978 034146 010164 000010      MOV      R1,RMCS2(R4) ;SELECT THE DRIVE
979 034152 116164 030022 000016      MOVB    ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
980 034160 105761 027726      1$:    TSTB    DPINT(R1)   ;INITIALIZING THE DRIVE ?
981 034164 001424              BEQ      2$         ;BR IF NOT
982 034166 105061 027726      CLRB    DPINT(R1)   ;CLEAR THE INIT INDICATOR
983 034172 004037 030264      JSR      RO,DRVINT   ;GO INIT THE DRIVE
984 034176 000240              NOP              ;DUMMY PARITY ERROR RETURN
985 034200 105761 027706      TSTB    DRVSTA(R1)  ;DRIVE ONLINE ?
986 034204 003014              BGT      2$         ;BR IF YES -- START ORDER
987 034206 005702              TST      R2         ;QUEUE ENTRY FOR THE DRIVE
988 034210 001426              BEQ      3$         ;BR IF NOT
989 034212 004737 036112      JSR      PC,GETREQ   ;GET DPB ADDRESS
990 034216 052762 140000 000016      BIS      #BIT15!BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
991 034224 004737 035256      JSR      PC,SVRH70   ;SAVE THE REGISTERS
992              :          JSR      PC,EMPTYQ    ;EMPTY THE REQUEST QUEUE
993 034230 004737 036134      JSR      PC,POPQUE   ;REMOVE THE QUEUE
994 034234 000414              BR      3$         ;
995 034236 032764 004000 000000 2$:  BIT      #BIT11,RMCS1(R4) ;DVA SET ?
996 034244 001006              BNE     4$         ;SET THEN CALL OPT
997 034246 006301              ASL      R1           ;
998 034250 012761 060000 030000      MOV      #60000,TIMER(R1)
999 034256 006201              ASR      R1           ;
1000 034260 000402              BR      3$         ;
1001 034262 004737 031114      4$:    JSR      PC,OPT   ;START THE PENDING REQUEST
1002 034266 000137 033176      3$:    JMP      SC4     ;PROCESS OTHER DRIVES
1003
1004              ;RM TIMER ROUTINE
1005              ;CALL
1006              :          MOV      #TIME,-(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
1007              :          JSR      PC,RMTMR   ;CALL RM05 TIME ROUTINE
1008
1009 034272 005737 027752      RMTMR: TST      ACTDRV   ;CHECK 'ACTDRV & ACTSTR'
1010 034276 001027              BNE     4$         ;IF NON ZERO EXIT

```

```

1011 034300 112737 000001 027753      MOVB    #1,ACTSTR      ;SET 'ACTSTR'
1012 034306 104412                    SAVREG                    ;SAVE R0 - R5
1013 034310 005001                    CLR     R1                ;START WITH DRIVE 0
1014 034312 005003                    CLR     R3
1015 034314 005763 030000      1$:    TST     TIMER(R3)    ;IS THE TIMER RUNNING?
1016 034320 002406                    BLT     2$                ;BR IF NO
1017 034322 166663 000002 030000    SUB     2(SP),TIMER(R3) ;COUNT THE INTERVAL
1018 034330 003002                    BGT     2$                ;BR IF NO SOFTWARE TIMEOUT
1019 034332 004737 034362      JSR     PC,STO           ;CALL SOFTWARE TIMEOUT ROUTINE
1020 034336 005201      2$:    INC     R1                ;MOVE TO NEXT DRIVE
1021 034340 005723                    TST     (R3)+
1022 034342 022701 000010      CMP     #8.,R1          ;OUT OF DRIVES?
1023 034346 003362                    BGT     1$                ;BR IF NO
1024 034350 104413      3$:    RESREG                    ;RESTORE R0 - R5
1025 034352 105037 027753      CLRB   ACTSTR          ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
1026 034356 012616      4$:    MOV     (SP)+,(SP)    ;ADJUST THE STACK
1027 034360 000207      RTS     PC                ;RETURN
1028
1029      ;SOFTWARE TIMEOUT ROUTINE
1030      ;
1031      ;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
1032      ;OR GREATER
1033      ;
1034      ;CALL:  STO
1035      ;        MOV     #DRVNUM,R1      ;DRIVE NUMBER
1036      ;        JSR     PC,STO          ;CALL
1037      ;        RETURN
1038
1039 034362 010146      STO:   MOV     R1,-(SP)    ;SAVE R1
1040 034364 010246      MOV     R2,-(SP)    ;SAVE R2
1041 034366 010346      MOV     R3,-(SP)    ;SAVE R3
1042 034370 010446      MOV     R4,-(SP)    ;SAVE R4
1043 034372 013704 030034      MOV     RMADR,R4    ;GET ADDRESS OF 'RMCS1'
1044 034376 010164 000010      MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
1045 034402 004037 034704      JSR     R0,RD.RM    ;READ 'DRIVE STATUS REG'
1046 034406 000012      RMD5
1047      ;
1048 034410 034672      STO5
1049 034412 105726      STO9
1050 034414 100436      TSTB   (SP)+        ;IS 'DRY'=1?
1051 034416 105761 027726      BMI    STO2         ;BR IF YES
1052 034422 001033      ST01: TSTB   DPINT(R1)  ;TRYING TO INTIALIZE THE DRIVE ?
1053 034424 105761 027736      BNE    STO2         ;BR IF YES
1054 034430 001030      TSTB   DPRQS(R1)   ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1055 034432 013702 027746      BNE    STO2         ;BR IF YES
1056 034436 020137 030020      MOV     TRNSWT,R2   ;PICKUP TRANSFER WAIT QUEUE
1057 034442 001404      CMP     R1,DTUW     ;TRANSFER UNDERWAY ON THIS DRIVE?
1058 034444 000137 034672      BEQ    1$          ;BR IF YES
1059 034450 004737 036112      JMP     STO9        ;IF NOT DON'T BOTHER DRIVES
1060 034454 052762 101000 000016 1$:    JSR     PC,GETREQ   ;GET DPB ADDRESS
1061 034462 004737 035256      BIS    #BIT15!BIT09,16(R2) ;SET THE ERROR FLAGS
1062      ;
1063 034466 105061 027676      JSR     PC,SVRH70   ;SAVE RH/RM REGISTERS
1064 034472 105061 027754      MOV     #CLR,RMCS2(R4) ;'INIT' THE MASS BUS
1065 034476 005037 027746      CLRB   DRVACT(R1)  ;DRIVE IS IDLE
1066 034502 012737 177777 030020      CLRB   ULDFLG(R1)  ;CLEAR THE UNLOAD FLAG
1067 034510 000470      CLR    TRNSWT      ;CLEAR DPB ADDRESS
1067 034510 000470      MOV     #-1,DTUW    ;CLEAR THE TRANSFER DRIVE #
1067 034510 000470      BR     STO9        ;DON'T BOTHER OTHER DRIVES

```

```

1068 034512 116405 000016      ST02:  MOVB   RMAS(R4),R5      ;READ ATTENTION REG
1069 034516 136105 030022      BITB   ATABIT(R1),R5      ;IS ATTENTION FOR THIS DRIVE UP ?
1070 034522 001007              BNE    ST03              ;YES
1071 034524 105761 027726      TSTB   DPINT(R1)         ;TRYING TO INTIALIZE THE DRIVE ?
1072 034530 001021              BNE    ST06              ;BR IF YES - DRIVE NOT ONLINE
1073 034532 105761 027736      TSTB   DPRQS(R1)         ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
1074 034536 001035              BNE    ST07              ;BR IF YES - NO RESPONSE TO REQUEST
1075 034540 000454              BR     ST09              ;OTHER WISE EXIT
1076 034542 105761 027726      ST03:  TSTB   DPINT(R1)         ;INITIALIZING THE DRIVE ?
1077 034546 001003              BNE    1$                ;BR IF INIT PENDING
1078 034550 105761 027736      TSTB   DPRQS(R1)         ;PORT REQUEST PENDING ?
1079 034554 001446              BEQ    ST09              ;BR IF NOT
1080 034556 012763 177777 030000 1$:  MOV     #-1,TIMER(R3)      ;STOP THE TIMER
1081 034564 000442              BR     ST09              ;EXIT
1082 034566 004737 032274      ST05:  JSR    PC,C18        ;GO HANDLE THE PARITY ERROR
1083 034572 000437              BR     ST09
1084 034574 105061 027726      ST06:  CLRB   DPINT(R1)         ;CLEAR THE INITIALIZE INDICATOR
1085 034600 105061 027706      CLRB   DRVSTA(R1)        ;SET UNIT OFFLINE
1086 034604 012763 177777 030000  MOV     #-1,TIMER(R3)      ;STOP THE TIMER
1087 034612 004737 036112      JSR    PC,GETREQ         ;GET THE DPB ADDRESS
1088 034616 005702              TST    R2                ;REQUEST IN QUEUE ?
1089 034620 001424              BEQ    ST09              ;BR IF NOT
1090 034622 052762 140000 000016  BIS    #BIT15!BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
1091 034630 000414              BR     ST08
1092 034632 012763 177777 030000  ST07:  MOV     #-1,TIMER(R3)      ;STOP THE TIMER
1093 034640 105061 027736      CLRB   DPRQS(R1)         ;CLEAR PORT REQUEST INDICATOR
1094 034644 004737 036112      JSR    PC,GETREQ         ;GET DPB ADDRESS
1095 034650 005702              TST    R2                ;QUEUE ENTRY FOR DRIVE ?
1096 034652 001407              BEQ    ST09              ;BR IF NONE
1097 034654 012762 100004 000016  MOV     #BIT15!BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
1098 034662 004737 036016      ST08:  JSR    PC,EMPTYQ      ;CLEAR THE QUEUE FOR THE DRIVE
1099 034666 004737 035256      JSR    PC,SVRH70         ;SAVE THE REGISTERS
1100 034672 012604      ST09:  MOV     (SP)+,R4        ;RESTORE R4
1101 034674 012603      MOV     (SP)+,R3        ;RESTORE R3
1102 034676 012602      MOV     (SP)+,R2        ;RESTORE R2
1103 034700 012601      MOV     (SP)+,R1        ;RESTORE R1
1104 034702 000207      RTS     PC                ;RETURN
1105
1106      ;ROUTINE TO READ A RH/RM REGISTER
1107      ;CALL
1108      ;
1109      ;JSR    RC,RD.RM      ;GO READ A REGISTER
1110      ;INDEX   ;REG. INDEX FROM BASE
1111      ;ERRADR  ;ERROR ADDRESS--PROCESS ERROR STARTING
1112      ;        ;AT THIS ADDRESS
1113      ;        ;CONTENTS OF REG. IS ON THE STACK
1114      ;RETURN
1115 034704 013737 030032 035052  RD.RM:  MOV     MCPMX,RD.RM2    ;MAX. RETRYS ALLOWED
1116 034712 011646      MOV     (SP),-(SP)       ;SAVE R0 FOR RETURN
1117 034714 013737 030034 034730      MOV     RMADR,RD.ADR     ;FORM THE DESIRED ADDRESS
1118 034722 062037 034730      ADD     (R0)+,RD.ADR     ;USING THE BASE AND THE INDEX
1119 034726 013727      RD.RM1: MOV     @(PC)+,(PC)+    ;READ THE DESIRED REGISTER OF THE RM DRIVE
1120 034730 000000      RD.ADR: .WORD 0          ;ADDRESS IS FORMED HERE
1121 034732 000000      RD.WRD: .WORD 0          ;REG. CONTENTS PUT HERE
1122 034734 013766 034732 000002      MOV     RD.WRD,2(SP)     ;RETURN IT TO THE USER
1123 034742 013746 030034      MOV     RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
1124 034746 062716 000010      ADD     #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2

```



```

1125 034752 032736 010000      BIT      #BIT12,@(SP)+      ;CHECK THE 'NED' BIT
1126 034756 001037              BNE      RD.RM3            ;BR IF DRIVE NON-EXISTENT
1127 034760 017746 173050      MOV      @RMADR,-(SP)      ;READ RMCS1
1128 034764 032716 020000      BIT      #BIT13,(SP)      ;DID MCPE SET?
1129 034770 001002              BNE      1$               ;BR IF YES
1130 034772 022620              CMP      (SP)+,(RO)+      ;ADJUST FOR RETURN
1131 034774 000432              BR       RD.RM4           ;EXIT
1132 034776              1$:
      034776 004037 036202      JSR      RO,ES.SAV        ;SAVE THE ADDRESS IN '$ESCAPE'
      035002 104003              EMT      3                ;REPORT 'MCPE' ERROR
1133 035004 005737 030020      TST      DTUW             ;DATA TRANSFER UNDERWAY?
1134 035010 100405              BMI      2$               ;NO
1135 035012 032716 040000      BIT      #BIT14,(SP)      ;'TRE' = 1 ?
1136 035016 001402              BEQ      2$               ;NO
1137 035020 005726              TST      (SP)+            ;YES--CLEAN OFF THE STACK AND
1138 035022 000415              BR       RD.RM3           ;TAKE THE FATAL ERROR EXIT
1139 035024 052716 040000      2$:      BIS      #BIT14,(SP)      ;CLEAR 'MCPE' BY SENDING A '1' TO 'TRE'
1140 035030 000316              SWAB     (SP)              ;POSITION BEFORE WRITING
1141 035032 013737 030034 035046      MOV      RMADR,3$         ;FORM ADDRESS OF HIGH BYTE
1142 035040 005237 035046              INC      3$
1143 035044 112637              MOV      (SP)+,@(PC)+     ;WRITE THE HIGH BYTE OF RMCS1
1144 035046 000000      3$:      .WORD   0                ;ADDRESS STORAGE
1145 035050 005327              DEC      (PC)+            ;EXCEEDED MAX. RETRYS
1146 035052 000003      RD.RM2: .WORD   3
1147 035054 002324              BGE      RD.RM1           ;BR IF NO
1148 035056 011000      RD.RM3: MOV      (RO),RO    ;FATAL ERROR EXIT
1149 035060 012616              MOV      (SP)+,(SP)
1150 035062 000200      RD.RM4: RTS      RO
1151
1152      ;ROUTINE TO WRITE A REGISTER
1153      ;
1154      ;CALL
1155      ;
1156      ;      MOV      DATA,-(SP)      ;DATA TO BE LOADED ON THE STACK
1157      ;      JSR      RO,WRT.RM        ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
1158      ;      INDEX     ERRADR         ;INDEX OF THE REGISTER TO BE LOADED
1159      ;      RETURN    ;ADDRESS TO RETURN TO ON AN ERROR
1160      ;      ;ERROR FREE RETURN
1161 035064 013737 030032 035242      WRT.RM: MOV      MCPEMX,WRT.R2 ;MAX RETRYS ALLOWED
1162 035072 016637 000002 035152      MOV      2(SP),WRT.WD     ;SAVE THE WORD TO WRITE
1163 035100 012616              MOV      (SP)+,(SP)       ;ADJUST THE STACK
1164 035102 012037 035154      MOV      (RO)+,WRT.AD     ;GET INDEX OF REGISTER TO BE WRITTEN
1165 035106 001015              BNE      1$               ;BR IF NOT RMCS1
1166 035110 122737 000150 035152      CMPB    #150,WRT.WD       ;IS THE COMMAND FOR DATA TRANSFERS?
1167 035116 002411              BLT      1$               ;YES--DON'T GET THE OLD A16 & A17, & PSEL
1168 035120 004037 034704      JSR      RO,RD.RM         ;NO---COMBINE A16&A17, & PSEL WITH
1169 035124 000000              RMCS1   WRT.R3           ;THE COMMAND BEFORE SENDING IT TO
1170 035126 035246              SWAB     (SP)              ;THE RH/RM
1171 035130 000316              BIC      #^C7,(SP)
1172 035132 042716 177770      MOV      (SP)+,WRT.WD+1
1173 035136 112637 035153      1$:      ADD      RMADR,WRT.AD     ;FORM THE ADDRESS OF THE DISK REG.
1174 035142 063737 030034 035154      WRT.R1: MOV      (PC)+,@(PC)+ ;LOAD THE DESIRED REG.
1175 035150 012737      WRT.WD: .WORD   0          ;WORD TO WRITE GOES HERE
1176 035152 000000      WRT.AD: .WORD   0          ;ADDRESS IS FORMED HERE
1177 035154 000000              MOV      RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
1178 035156 013746 030034      ADD      #RMCS2,(SP)      ;FORM THE ADDRESS OF RMCS2
1179 035162 062716 000010

```

```

1180 035166 032736 010000          BIT    #BIT12,(SP)+    ;CHECK THE 'NED' BIT
1181 035172 001025          BNE    WRT.R3         ;BR IF DRIVE NON-EXISTENT
1182 035174 004037 034704          JSR    RO,RD.RM      ;CHECK FOR PARITY ERROR ON WRITE
1183 035200 000014          RMER1
1184 035202 035246          WRT.R3
1185 035204 032726 000010          BIT    #BIT03,(SP)+
1186 035210 001420          BEQ    WRT.R4         ;BR IF 'PAR=0'
1187 035212 016037 177776 035224    MOV    -2(RO),1$     ;PICKUP THE INDEX
1188 035220 004037 034704          JSR    RO,RD.RM      ;READ THE REG.
1189 035224 000000          1$: .WORD 0          ;REG. INDEX
1190 035226 035246          WRT.R3              ;RETURN TO THIS ADDRESS ON ERROR
1191 035230 004037 036202          JSR    RO,ES.SAV     ;SAVE THE ADDRESS IN '$ESCAPE'
1192 035234 104004          EMT    4            ;REPORT THE PARITY ON WRITE ERROR
1193 035236 005726          TST    (SP)+        ;CLEAR OFF THE STACK
1194 035240 005327          DEC    (PC)+        ;DECREMENT THE ERROR COUNT
1195 035242 000003          WRT.R2: .WORD 3     ;PETRY COUNTER
1196 035244 002341          BGE    WRT.R1        ;TRY AGAIN IF NOT FINISHED
1197 035246 011000          WRT.R3: MOV (RO),RO ;TAKE THE 'PARITY ON WRITE' ERROR EXIT
1198 035250 000401          BR    WRT.R5         ;EXIT
1199 035252 005720          WRT.R4: TST (RO)+   ;ADJUST FOR ERROR FREE EXIT
1200 035254 000200          WRT.R5: RTS    RO
1201          ;ROUTINE TO SAVE THE RH/RM REGISTERS AS PER DPB+14
1202          ;
1203          ;CALL
1204          ;
1205          ;      MOV    #DPBNUM,R2      ;DPB POINTER TO R2
1206          ;      JSR    PC,SVRH70     ;SAVE THE DRIVES REG'S
1207 035256 104412          SVRH70: SAVREG      ;SAVE R0 - R5
1208 035260 005702          TST    R2           ;QUEUE ENTRY FOR THE DRIVE ?
1209 035262 001442          BEQ    6$           ;BR IF NONE
1210 035264 013704 030034          MOV    RMADR,R4     ;SELECT DRIVE
1211 035270 111264 000010          MOV    (R2),RMCS2(R4)
1212 035274 016203 000014          MOV    14(R2),R3   ;GET THE ERROR TABLE POINTER
1213 035300 001433          BEQ    6$           ;EXIT IF NO ADDRESS
1214 035302 005037 035336          CLR    3$          ;COUNTER & POINTER
1215 035306 023727 035336 000022 1$: CMP    3$,#RMDB     ;REACHED THE BUFFER REGISTER ?
1216 035314 001006          BNE    2$          ;BR IF NOT
1217 035316 032764 000200 000010    BIT    #BIT07,RMCS2(R4) ;'OR' SET ?
1218 035324 001002          BNE    2$          ;BR IF SET
1219 035326 005023          CLR    (R3)+       ;STORE RMDB AS ZEROES
1220 035330 000405          BR    4$          ;CONTINUE
1221 035332 004037 034704          2$: JSR    RO,RD.RM  ;READ THE SELECTED REGISTER
1222 035336 000000          3$: .WORD 0        ;REGISTER INDEX
1223 035340 035364          5$          ;ERROR RETURN ADDRESS
1224 035342 012623          MOV    (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
1225 035344 023727 035336 000046 4$: CMP    3$,#RMEC2   ;REACHED THE END ?
1226 035352 001406          BEQ    6$          ;BR IF YES
1227 035354 062737 000002 035336    ADD    #2,3$        ;INCREMENT THE REGISTER INDEX
1228 035362 000751          BR    1$          ;CONTINUE READING THE REGISTERS
1229 035364 004737 032174          5$: JSR    PC,C17    ;PROCESS THE UNCORRECTABLE PARITY ERROR
1230 035370 104413          6$: RESREG      ;RESTORE R0 - R5
1231 035372 000207          RTS    PC          ;RETURN
1232          ;
1233          ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
1234          ;CALL
1235          ;      MOV    #DRVNUM,R1     ;DRIVE NUMBER TO R1

```

```

1236      :      JSR      PC,SET.IE      :SET 'IE'
1237      :      RETURN
1238
1239 035374 010446      SET.IE: MOV      R4,-(SP)      :SAVE R4
1240 035376 013704 030034  MOV      RMADR,R4      :PICKUP ADDRESS OF RMCS1
1241 035402 010164 000010  MOV      R1,RMCS2(R4)  :SELECT DRIVE
1242 035406 011446      MOV      (R4),-(SP)    :READ RMCS1
1243 035410 052716 040000  BIS      #BIT14,(SP)   :SET THE 'TRE' BIT OF THE WORD READ
1244 035414 000316      SWAB      (SP)        :ADJUST FOR DATO
1245 035416 112714 000100  MOVB     #BIT06,(R4)   :SET 'IE'
1246 035422 032764 010000 000010  BIT      #BIT12,RMCS2(R4) :IS 'NED'=1?
1247 035430 001002      BNE      1$          :YES--CLEAR 'TRE'
1248 035432 005726      TST      (SP)+       :CLEAN OFF THE STACK
1249 035434 000402      BR       2$
1250 035436 112664 000001  1$:  MOVB     (SP)+,1(R4)  :CLEAR 'TRE'
1251 035442 012604      2$:  MOV      (SP)+,R4    :RESTORE R4
1252 035444 000207      RTS      PC         :RETURN TO CALLER
1253
1254      ;QUEUE COUNT
1255      QCNT:  .BYTE  0      :DRIVE 0
1256      .BYTE  0      :DRIVE 1
1257      .BYTE  0      :DRIVE 2
1258      .BYTE  0      :DRIVE 3
1259      .BYTE  0      :DRIVE 4
1260      .BYTE  0      :DRIVE 5
1261      .BYTE  0      :DRIVE 6
1262      .BYTE  0      :DRIVE 7
1263
1264      ;QUEUE INPUT POINTERS
1265
1266      QINPT: .WORD  QDRV0   :DRIVE 0
1267      .WORD  QDRV1   :DRIVE 1
1268      .WORD  QDRV2   :DRIVE 2
1269      .WORD  QDRV3   :DRIVE 3
1270      .WORD  QDRV4   :DRIVE 4
1271      .WORD  QDRV5   :DRIVE 5
1272      .WORD  QDRV6   :DRIVE 6
1273      .WORD  QDRV7   :DRIVE 7
1274
1275      ;QUEUE OUTPUT POINTERS
1276
1277      QOUTPT: .WORD  QDRV0   :DRIVE 0
1278      .WORD  QDRV1   :DRIVE 1
1279      .WORD  QDRV2   :DRIVE 2
1280      .WORD  QDRV3   :DRIVE 3
1281      .WORD  QDRV4   :DRIVE 4
1282      .WORD  QDRV5   :DRIVE 5
1283      .WORD  QDRV6   :DRIVE 6
1284      .WORD  QDRV7   :DRIVE 7
1285
1286      QSTART: .WORD  QDRV0   :DRIVE 0 START ADDRESS
1287      QSTOP:  .WORD  QDRV1   :DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
1288      .WORD  QDRV2   :STOP DRIVE 1--START DRIVE 2
1289      .WORD  QDRV3   :STOP DRIVE 2--START DRIVE 3
1290      .WORD  QDRV4   :STOP DRIVE 3--START DRIVE 4
1291      .WORD  QDRV5   :STOP DRIVE 4--START DRIVE 5
1292      .WORD  QDRV6   :STOP DRIVE 5--START DRIVE 6

```

```

1293 035534 035720          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
1294 035536 035740          .WORD  QTERM          ;STOP DRIVE 7
1295
1296          ;DRIVE REQUEST QUEUES
1297
1298 035540          QDRV0:  .BLKW  10
1299 035560          QDRV1:  .BLKW  10
1300 035600          QDRV2:  .BLKW  10
1301 035620          QDRV3:  .BLKW  10
1302 035640          QDRV4:  .BLKW  10
1303 035660          QDRV5:  .BLKW  10
1304 035700          QDRV6:  .BLKW  10
1305 035720          QDRV7:  .BLKW  10
1306          035740          QTERM=.
1307
1308          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
1309
1310          ;CALL
1311          ;
1312          JSR      PC,CLRQUE
1313 035740 104412          CLRQUE:  SAVREG          ;SAVE R0 - R5
1314 035742 012702 035446          MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
1315 035746 005022          CLR      (R2)+          ;DRIVES 0 & 1
1316 035750 005022          CLR      (R2)+          ;DRIVES 2 & 3
1317 035752 005022          CLR      (R2)+          ;DRIVES 4 & 5
1318 035754 005022          CLR      (R2)+          ;DRIVES 6 & 7
1319 035756 012703 000010          MOV      #8.,R3          ;MOVE THE STARTING
1320 035762 012701 035516          MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
1321 035766 012122          1$:  MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
1322 035770 005303          DEC      R3
1323 035772 001375          BNE      1$
1324 035774 012703 000010          MOV      #8.,R3          ;MOVE THE STARTING ADDRESS
1325 036000 012701 035516          MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
1326 036004 012122          2$:  MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
1327 036006 005303          DEC      R3
1328 036010 001375          BNE      2$
1329 036012 104413          RESREG          ;RESTORE R0 - R5
1330 036014 000207          RTS      PC
1331
1332          ;EMPTY THE QUEUE SPECIFIED BY R1
1333
1334          ;CALL
1335          ;
1336          MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
1337          JSR      PC,EMPTYQ
1338 036016 105061 035446          EMPTYQ:  CLRB      QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
1339 036022 006301          ASL      R1
1340 036024 016161 035456 035476          MOV      QINPT(R1),QOUTPT(R1)          ;SET OUTPUT QUEUE POINTER=INPUT POINTER
1341 036032 006201          ASR      R1
1342 036034 000207          RTS      PC
1343
1344          ;ROUTINE TO PUT A REQUEST IN QUEUE
1345
1346          ;CALL
1347          ;
1348          MOV      #DRVNUM,R1          ;DRIVE NUMBER
1349          MOV      #DPB,R2          ;ADDRESS OF PARAMETER BLOCK
          JSR      R0,DRVQUE          ;GO PUT REQUEST IN QUEUE

```

```

1350      :      RETURN1      ;RETURN HERE IF QUEUE IS FULL
1351      :      RETURN2      ;RETURN HERE IF REQUEST IS IN QUEUE
1352
1353 036036 122761 000010 035446 DRVQUE: CMPB    #10,QCNT(R1) ;IS QUEUE FULL?
1354 036044 001421          BEQ     2$      ;BR IF YES-TAKE RETURN1
1355 036046 105261 035446          INCB   QCNT(R1) ;INCREMENT QUEUE COUNT
1356 036052 006301          ASL    R1
1357 036054 010271 035456          MOV    R2,@QINPT(R1) ;PUT THIS REQUEST IN QUEUE
1358 036060 062761 000002 035456          ADD    #2,QINPT(R1) ;UPDATE THE QUEUE POINTER
1359 036066 026161 035456 035520          CMP    QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
1360 036074 001003          BNE   1$      ;BR IF NO
1361 036076 016161 035516 035456          MOV    QST .T(R1),QINPT(R1) ;YES--RESET POINTER
1362 036104 006201          1$: ASR    R1
1363 036106 005720          TST   (R1)+ ;TAKE RETURN 2
1364 036110 000200          2$: RTS    RO ;RETURN TO USER
1365
1366      ;ROUTINE TO GET THE 'DPB' ADDRESS OF NEXT REQUEST IN QUEUE
1367      :
1368      :CALL
1369      :      MOV    #DRVNUM,R1 ;DRIVE NUMBER TO R1
1370      :      JSR    PC,GETREQ ;GO GET THE REQUEST
1371      :      RETURN ;R2='DPB' ADDRESS OF THE REQUEST
1372      :      ;R2=0 IF NO REQUEST IN QUEUE
1373
1374 036112 005002          GETREQ: CLR    R2
1375 036114 105761 035446          TSTB  QCNT(R1) ;IS THERE ANY REQUEST IN QUEUE?
1376 036120 001404          BEQ   2$      ;NO
1377 036122 006301          1$: ASL    R1
1378 036124 017102 035476          MOV    @QOUTPT(R1),R2 ;PICKUP 'DPB' POINTER FOR THIS DRIVE
1379 036130 006201          ASR   R1
1380 036132 000207          2$: RTS    PC ;RETURN TO USER
1381
1382      ;ROUTINE TO 'POP' THE REQUEST FROM QUEUE
1383      :
1384      :CALL
1385      :      MOV    #DRVNUM,R1 ;DRIVE NUMBER TO R1
1386      :      JSR    PC,POPQUE ;CALL TO REMOVE REQUEST
1387      :      RETURN ;R2-ADDRESS OF DPB REMOVED
1388
1389 036134 105361 035446          POPQUE: DECB  QCNT(R1) ;DECREMENT QUEUE COUNT
1390 036140 006301          ASL   R1
1391 036142 017102 035476          MOV    @QOUTPT(R1),R2 ;GET THE 'DPB' POINTER
1392 036146 005071 035476          CLR   @QOUTPT(R1) ;REMOVE DPB ADDRESS FROM THE QUEUE
1393 036152 062761 000002 035476          ADD    #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
1394 036160 026161 035476 035520          CMP    QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
1395 036166 001003          BNE   1$      ;NO--BR TO EXIT
1396 036170 016161 035516 035476          MOV    QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
1397 036176 006201          1$: ASR   R1
1398 036200 000207          RTS   PC ;RETURN TO USER
1399
1401      ;ROUTINE TO SAVE THE CONTENTS OF '$ESCAPE' WHEN THE DRIVER
1402      ;REPORTS AN ERROR DIRECTLY.
1403      :
1404      :CALL
1405      :      JSR    RO,ES.SAV
1406      :      ERROR  N ;:THE ERROR CALL
1407      :      RETURN ;:THE RETURN IS PAST THE ERROR CALL

```

1408						
1409	036202	012037	036216	ES.SAV:	MOV	(R0)+,1\$;GET THE ERROR CALL
1410	036206	013746	001176		MOV	\$ESCAPE,-(SP) ;SAVE THE ADDRESS IN '\$ESCAPE'
1411	036212	005037	001176		CLR	\$ESCAPE ;CLEAR THE ESCAPE RETURN
1412	036216	000000		1\$:	.WORD	0 ;THE ERROR CALL IS MOVED HERE
1413	036220	012637	001176		MOV	(SP)+,\$ESCAPE ;RESTORE THE ESCAPE ADDRESS
1414	036224	000200			RTS	R0 ;RETURN
1416						

```

1          .SBTTL TELETYPE MESSAGES
2
3 036226   040   117   106 UNTOFF: .ASCIZ / OFFLINE/
4 036237   040   117   116 UNTON:  .ASCIZ / ONLINE/
5 036247   040   116   117 NOTRM:  .ASCIZ @ NOT AN RM05/3/2@
6 036270   040   116   117 NOTPRS: .ASCIZ / NOT PRESENT/
7 036305   040   125   116 NOTSAF: .ASCIZ / UNSAFE/
8 036315   040   111   123 LODEV:  .ASCIZ / IS LOAD DEVICE/
9 036335   122   115   060 $RM02: .ASCIZ /RM02/
10 036342  122   115   060 $RM03: .ASCIZ /RM03/
11 036347  122   115   060 $RM05: .ASCIZ /RM05/
12 036354  200   125   116 UNSTAT: .ASCIZ <CRLF>/UNIT STATUS:/
13 036372  200   104   122 MUNIT:  .ASCIZ <CRLF>/DRIVE: /
14 036403  060   000   ZERO:   .ASCIZ /0/
15 036405  040   057   040 SLASH:  .ASCIZ @ / @
16 036411  103   000   C:      .ASCIZ /C/
17 036413  106   000   F:      .ASCIZ /F/
18 036415  124   000   T:      .ASCIZ /T/
19 036417  131   000   Y:      .ASCIZ /Y/
20 036421  062   000   TWO:    .ASCIZ /2/
21 036423  040   BLNKS4: .ASCII / /
22 036424  040   BLNKS3: .ASCII / /
23 036425  040   BLNKS2: .ASCII / /
24 036426  040   000   BLNKS1: .ASCIZ / /
25 036430  105   116   124 ENTADR: .ASCIZ /ENTER ADDRESS LIMITS:/<CRLF>
26 036457  040   077   104 MOFFLN: .ASCIZ / ?DRIVE OFFLINE/<CRLF>
27 036500  040   077   104 MDRNP:  .ASCIZ / ?DRIVE NOT PRESENT/<CRLF>
28 036525  040   077   104 MER11:  .ASCIZ / ?DRIVE NOT AVAILABLE/<CRLF>
29 036554  040   077   104 MUSDR:  .ASCIZ / ?DRIVE UNSAFE/<CRLF>
30 036574  040   077   104 MLODEV: .ASCIZ / ?DRIVE IS LOAD DEVICE/<CRLF>
31 036624  040   123   105 MSEL:   .ASCIZ / SELECTED/
32 036636  200   120   122 MMODE:  .ASCIZ <CRLF>/PROGRAM MODE (F, C OR V): /
33 036672  200   106   117 MFORMT: .ASCIZ <CRLF>/FORMAT & VERIFY/
34 036713  200   106   117 MCHK:   .ASCIZ <CRLF>/FORMAT & VERIFY WITH ROTATING DATA PATTERN/
35 036767  200   126   105 MVERFY: .ASCIZ <CRLF>/VERIFY ONLY/
36 037004  200   117   120 MSIZE:  .ASCIZ <CRLF>/OPERATE IN 32 SECTOR MODE (Y OR N) ? /
37 037053  200   117   120 MSEC30: .ASCIZ <CRLF>/OPERATION WILL BE IN 30 SECTOR (18 BIT) MODE/<CRLF>
38 037132  200   117   120 MSEC32: .ASCIZ <CRLF>/OPERATION WILL BE IN 32 SECTOR (16 BIT) MODE/<CRLF>
39 037211  111   116   126 BADENT: .ASCIZ /INVALID ENTRY/<CRLF>
40 037230  105   116   104 MADRER: .ASCII /ENDING DISK ADRS MUST BE EQUAL TO OR GREATER/<CRLF>
41 037305  124   110   101 .ASCIZ /THAN STARTING ADRS/<CRLF>
42 037331  200   123   105 MSELP:  .ASCII <CRLF>/SELECT DATA PATTERN/<CRLF>
43 037356  040   050   060 .ASCII / (0) ZERO'S /<CRLF>
44 037373  040   050   061 .ASCII / (1) ONE'S/<CRLF>
45 037406  040   050   062 .ASCII / (2) WORST CASE/<CRLF>
46 037426  105   116   124 .ASCIZ /ENTER (0, 1 OR 2): /
47 037452  200   123   124 MSFOR:  .ASCIZ <CRLF>/STARTING FORMAT ON DRIVE /
48 037505  200   123   124 MSCHK:  .ASCIZ <CRLF>/STARTING CHECK ON DRIVE /
49 037537  200   123   124 MSQVER: .ASCIZ <CRLF>/STARTING QUICK VERIFY ON DRIVE /
50 037600  200   123   124 MSVER:  .ASCIZ <CRLF>/STARTING VERIFY ON DRIVE /
51 037633  200   106   117 MFCMPT: .ASCIZ <CRLF>/FORMAT COMPLETE, /
52 037656  200   103   110 MCCMPT: .ASCIZ <CRLF>/CHECK COMPLETE, /
53 037700  200   126   105 MVCMPT: .ASCIZ <CRLF>/VERIFY COMPLETE, /
54 037723  124   117   124 NUMERR: .ASCIZ /TOTAL ERRORS DETECTED: /
55 037753  200   052   052 STARS:  .ASCIZ <CRLF>/*****/<CRLF>
56 037775  120   122   105 ADDRIS: .ASCIZ /PRESENT ADDRESS IS: /
57 040022  105   116   124 MMSG1:  .ASCIZ /ENTER THE SERIAL NUMBER (MAX. 10 OCTAL DIGITS)/

```

58	040101	200	123	116	SN1: .ASCIZ <CRLF>/SN # : /
59	040112	200	117	126	MSG3: .ASCII <CRLF>/OVER 126. BAD SECTORS HAVE BEEN DETECTED/<CRLF>
60	040164	120	101	103	.ASCIZ /PACK NOT ACCEPTABLE !/
61	040212	200	111	116	MSG2: .ASCII <CRLF>/INCORRECT MANUFACTURES DEFINED BAD SECTOR INFORMATION/
62	040300	200	012	111	.ASCIZ <CRLF><LF>/INITIALIZE THE BAD SECTOR FILE (Y OR N) ?/
63	040354	200	101	114	MSG4: .ASCII <CRLF>/ALIGNMENT PACK/
64	040373	200	120	122	MHALT: .ASCIZ <CRLF>/PROGRAM HALTED !/
65	040415	200	123	105	MSGBLK: .ASCII <CRLF>/SELECT ONE OF THE FOLLOWING FUNCTIONS/<CRLF>
66	040464	113	105	131	.ASCII /KEYIN U.S.R. BAD SECTORS0/<CRLF>
67	040544	120	122	111	.ASCII /PRINT M.F.G. BAD SECTORS.....1/<CRLF>
68	040624	120	122	111	.ASCII /PRINT U.S.R. BAD SECTORS.....2/<CRLF>
69	040704	111	116	111	.ASCII /INITIALIZE U.S.R. BAD SECTOR FILE.....3/<CRLF>
70	040764	120	122	111	.ASCII /PRINT A SECTOR OF THE LAST TRACK.....4/<CRLF>
71	041044	120	122	111	.ASCII /PRINT THIS MESSAGE.....5/<CRLF>
72	041124	105	130	111	.ASCIZ /EXIT<CR>/
73	041204	200	105	116	MUTLTY: .ASCIZ <CRLF>/ENTER UTILITY FUNCTION: /
74	041236	200	105	116	MSG5: .ASCIZ <CRLF>/ENTER 16 BIT M.F.G. BAD SECTOR ADDRESS IN DECIMAL/
75	041321	200	105	116	MSG6: .ASCIZ <CRLF>/ENTER 18 BIT M.F.G. BAD SECTOR ADDRESS IN DECIMAL/
76	041404	200	105	116	MSG7: .ASCIZ <CRLF>/ENTER 16 BIT U.S.R. BAD SECTOR ADDRESS IN DECIMAL/
77	041467	200	105	116	MSG8: .ASCIZ <CRLF>/ENTER 18 BIT U.S.R. BAD SECTOR ADDRESS IN DECIMAL/
78	041552	200	061	066	MSG9: .ASCIZ <CRLF>/16 BIT M.F.G. BAD SECTOR FILE IN OCTAL/
79	041622	200	061	070	MSG10: .ASCIZ <CRLF>/18 BIT M.F.G. BAD SECTOR FILE IN OCTAL/
80	041672	200	061	066	MSG11: .ASCIZ <CRLF>/16 BIT U.S.R. BAD SECTOR FILE IN OCTAL/
81	041742	200	061	070	MSG12: .ASCIZ <CRLF>/18 BIT U.S.R. BAD SECTOR FILE IN OCTAL/
82	042012	105	116	124	MSG13: .ASCIZ /ENTER SECTOR NUMBER : /
83	042041	104	117	116	MSG14: .ASCIZ /DONE !/<CRLF>
84	042051	200	105	116	MSG15: .ASCIZ <CRLF>/ENTER M.F.G. BAD SECTOR ADDRESS IN DECIMAL/<CRLF>
85					;MSG16: .ASCIZ /ILLEGAL BAD SECTOR, PACK NOT ACCEPTABLE !/<CRLF>
86	042126	127	122	111	MSG17: .ASCII /WRITE BAD SECTOR FILE NOT SUCCESSFUL,/<CRLF>
87	042174	120	101	103	.ASCIZ /PACK IS NOT ACCEPTABLE !/<CRLF>
88					
89					.EVEN
90					


```

1
2
3 042226      122      110      057  EM1:  .ASCIZ  @RH/RM INTERRUPT OCCURRED (RMAS=0)@
4 042270      125      116      105  EM2:  .ASCIZ  /UNEXPECTED ATTENTION OCCURRED/
5 042326      115      101      123  EM3:  .ASCIZ  /MASSBUS PARITY ERROR (MCPE=1)/
6 042364      115      101      123  EM4:  .ASCIZ  /MASSBUS PARITY ERROR (PAR=1)/
7 042421      101      104      104  EM5:  .ASCIZ  /ADDRESS PLUG CHANGE BIT SET/
8 042455      122      110      057  EM6:  .ASCIZ  @RH/RM DIDN'T RESPOND TO ADDRESSING@
9 042520      124      122      101  EM7:  .ASCIZ  @TRACK/SECTOR FIELD IN HEADER IS INCORRECT@
10 042572     104      122      111  EM10: .ASCIZ  /DRIVE OFFLINE/
11 042610     120      105      122  EM11: .ASCIZ  /PERSISTENT DRIVE UNSAFE ERROR/
12 042646     125      116      103  EM12: .ASCIZ  /UNCORRECTABLE MASSBUS PARITY ERROR/
13 042711     123      117      106  EM13: .ASCIZ  /SOFTWARE TIMEOUT/
14 042732     104      122      111  EM14: .ASCIZ  /DRIVE UNSAFE ERROR/
15 042755     103      117      116  EM15: .ASCIZ  @CONTROLLER/DRIVE ERROR DURING WRITE@
16 043021     103      117      116  EM16: .ASCIZ  @CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
17 043073     122      105      124  EM17: .ASCIZ  @RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@
18 043151     104      101      124  EM20: .ASCIZ  @DATA ERROR DURING WRITE CHECK@
19 043207     103      117      116  EM21: .ASCIZ  @CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
20 043260     047      110      103  EM22: .ASCIZ  @'HCE' OR 'HCRC' ERROR VERIFYING HEADERS@
21 043330     103      131      114  EM23: .ASCIZ  @CYLINDER FIELD IN HEADER IS NOT CORRECT@
22 043400     127      122      111  EM24: .ASCIZ  @WRITE CHECK ERROR@
23 043422     111      114      114  EM25: .ASCIZ  /ILLEGAL BAD SECTOR, PACK NOT ACCEPTABLE/
24

```

1	043472	122	115	101	DH1:	.ASCIZ	/RMAS	RMCS1	RMER1	RMER2	RMDS	RMDC	RMDA/
2	043540	104	122	111	DH2:	.ASCIZ	/DRIVE	RMDS	RMER1	RMER2	RMAS	RMCS1/	
3	043612	104	122	111	DH3:	.ASCIZ	/DRIVE	REG ADR	DATA	RMCS1	RMER1	RMER2/	
4	043663	104	122	111	DH4:	.ASCIZ	/DRIVE	REG ADR	GOOD	BAD	RMCS1	RMER1	RMER2/
5	043746	122	110	040	DH6:	.ASCIZ	/RH ADDRESS/						
6	043761	104	122	111	DH10:	.ASCIZ	/DRIVE	ERR PC	RMCS1	RMCS2	RMDS	RMER1	RMER2/
7	044046	122	115	105	DH10A:	.ASCIZ	/RMEC1	RMEC2	RMWC	RMBA	RMDA	RMAS	RMLA/
8	044132	122	115	104	DH10B:	.ASCIZ	/RMDB	RMMR1	RMDT	RMSN	RMOF	CYLINDER	TRACK/
9	044210	104	122	111	DH17:	.ASCIZ	/DRIVE	ERR PC	CYLINDER	TRACK	SECTOR/		
10	044257	104	122	111	DH20:	.ASCIZ	/DRIVE	ERR PC	CYLINDER	TRACK	SECTOR/		
11	044326	122	115	103	DH20A:	.ASCIZ	/RMCS1	RMCS2	RMDS	RMER1	RMER2	RMMR2	RMEC1
12	044423	122	115	127	DH20B:	.ASCIZ	/RMWC	RMBA	RMDA	RMAS	RMLA	RMDB	RMMR1
13	044521	122	115	123	DH20C:	.ASCIZ	/RMSN	RMOF	CYLINDER	TRACK/			
14	044554	104	122	111	DH23:	.ASCIZ	/DRIVE	ERR PC	EXPCTD	RECVD/			
15	044613	040	040	040	DH24:	.ASCII	/				MEMORY	DISK/<CR><LF>	
16	044701	104	122	111		.ASCIZ	/DRIVE	ERR PC	CYLNDR	TRACK	SECTOR	DATA	DATA/
17	044766	104	122	111	DH25:	.ASCIZ	/DRIVE	CYLNDR	TRACK	SECTOR	/		
18													
19													
20							.EVEN						

1	045030	001372	005560	005574	DT1:	.WORD	ATTN, RM.REG, RM.REG+14, RM.REG+42, RM.REG+12, RM.REG+34, RM.REG+6
2	045046	001370	027666	027670	DT2:	.WORD	DDRIVE, RMERRS, RMERRS+2, RMERRS+4, ATTN, RMCS1
3	045062	001370	034730	034732	DT3:	.WORD	DDRIVE, RD.ADR, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
4	045076	001370	035154	035152	DT4:	.WORD	DDRIVE, WRT.AD, WRT.WD, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
5	045114	001274			DT6:	.WORD	\$RMADR
6	045116	001222	001132	045772	DT7:	.WORD	DRIVE, \$ERRPC, RBUF+6, RBUF+2
7	045126	001222	001132	005560	DT10:	.WORD	DRIVE, \$ERRPC, RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42
8	045144	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
9	045162	005602	005604	005606		.WORD	RM.REG+22, RM.REG+24, RM.REG+26, RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
10	045200	001222	001132	001364	DT17:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC
11	045212	001222	001132	001364	DT20:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC
12	045224	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
13	045244	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
14	045264	005610	005612	001364		.WORD	RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
15	045274	001222	001132	045770	DT23:	.WORD	DRIVE, \$ERRPC, RBUF+4, RBUF
16	045304	001222	001132	001364	DT24:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC, \$GDDAT, RM.REG+22
17	045322	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
18	045342	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
19	045362	005610	005612	001364		.WORD	RM.REG+30, RM.REG+32, DS.CYL, DS.TRK
20							
21	045372	001222	005552	001366	DT25:	.WORD	DRIVE, FMTDPB+12, DS.TRK, SAVSEC
22							

1	045402	000001		DF1:	.WORD	1
2	045404	007	000		.BYTE	7,0
3	045406	000001		DF2:	.WORD	1
4	045410	006	000		.BYTE	6,0
5	045412	000001		DF3:	.WORD	1
6	045414	006	000		.BYTE	6,0
7	045416	000001		DF4:	.WORD	1
8	045420	007	000		.BYTE	7,0
9	045422	000001		DF6:	.WORD	1
10	045424	001	000		.BYTE	1,0
11	045426	000003		DF10:	.WORD	3
12	045430	007	000		.BYTE	7,0
13	045432	044046			.WORD	DH10A
14	045434	007	000		.BYTE	7,0
15	045436	044132			.WORD	DH10B
16	045440	007	140		.BYTE	7,140
17	045442	000001		DF17:	.WORD	1
18	045444	005	034		.BYTE	5,34
19	045446	000004		DF20:	.WORD	4
20	045450	005	034		.BYTE	5,34
21	045452	044326			.WORD	DH20A
22	045454	010	000		.BYTE	8,0
23	045456	044423			.WORD	DH20B
24	045460	010	000		.BYTE	8,0
25	045462	044521			.WORD	DH20C
26	045464	004	014		.BYTE	4,14
27	045466	000001		DF23:	.WORD	1
28	045470	005	000		.BYTE	5,0
29	045472	000004		DF24:	.WORD	4
30	045474	007	034		.BYTE	7,34
31	045476	044326			.WORD	DH20A
32	045500	010	000		.BYTE	8,0
33	045502	044423			.WORD	DH20B
34	045504	010	000		.BYTE	8,0
35	045506	044521			.WORD	DH20C
36	045510	004	014		.BYTE	4,14
37	045512	000001		DF25:	.WORD	1
38	045514	004	000		.BYTE	4,0
39						

```

1      .SBTTL  BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH/RM
2
3      ;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
4      ;OF THE RH/RM IS SETUP FOR THE PROPER ADDRESS.
5      ;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
6      ;REQUIRED.
7      ;NOTE: THIS ROUTINE DESTROYS R0-R4
8      ;CALL
9      :
10     :       JSR      PC,BUSADR
11     :       RETURN
12
13     045516 005737 001376  BUSADR: TST      CHGADR      ;INPUT FROM TTY REQUESTED?
14     045522 001436          BEQ      3$          ;NO
15     045524 005037 001376  CLR      CHGADR      ;DISABLE CHANGE OF RH/RM ADDRESS
16     045530 104401 001205  TYPE     ,%CRLF      ;CR-LF
17     045534 012700 001274  1$:     MOV     #RMRADR,R0 ;FIRST ADDRESS
18     045540 104401 045636  TYPE     ,MRMCS1     ;"RMC1="
19     045544 011046          MOV     (R0),-(SP)   ;PRESENT RMC1 ADDRESS
20     045546 104402          TYPOC          ;TYPE IT
21     045550 104401 036425  TYPE     ,BLNKS2     ;TYPE 2 SPACES
22     045554 104411          RDLIN          ;GET THE ENTRY
23     045556 012601          MOV     (SP)+,R1     ;ADDRESS OF ASCII TEXT
24     045560 004537 045654  JSR     R5,CK.NUM    ;ENTER AND STORE NEW ADDRESS
25     045564 000763          BR      1$          ;ERROR RET
26     045566 012700 001276  2$:     MOV     #RMRVEC,R0 ;CHECK VERC
27     045572 104401 045645  TYPE     ,MRMVEC     ;"RVEC="
28     045576 011046          MOV     (R0),-(SP)   ;PRESENT RH/RM VECTOR ADDRESS ON THE STACK
29     045600 104402          TYPOC          ;TYPE IT
30     045602 104401 036425  TYPE     ,BLNKS2     ;TYPE 2 SPACES
31     045606 104411          RDLIN          ;READ THE ENTRY
32     045610 012601          MOV     (SP)+,R1     ;ASCII TEXT ADDRESS
33     045612 004537 045654  JSR     R5,CK.NUM    ;ENTER AND STORE NEW ADDRESS
34     045616 000763          BR      2$          ;ERROR RET
35     045620 012700 001274  3$:     MOV     #RMRADR,R0 ;FIRST ADDRESS OF NEW PARAMETERS
36     045624 012701 030034  MOV     #RMRADR,R1   ;FIRST ADDRESS OF WHERE TO PUT THEM
37     045630 012021          MOV     (R0)+,(R1)+ ;BUS ADDRESS
38     045632 012021          MOV     (R0)+,(R1)+ ;VECTOR ADDRESS
39     045634 000207          RTS      PC         ;RETURN
40
41     045636      122      115      103  MRMCS1: .ASCIZ @RMC1=@
42     045645      122      115      126  MRMVEC: .ASCIZ @RVEC=@
43
44     .SBTTL  CK.NUM - CHECK NUMBER (OCTAL)
45     ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
46     ;AND FORMS AN OCTAL NUMBER IN R2
47     ;CALL:
48     :       MOV     #ADR,R1      ;ADDRESS OF ASCIZ STRING
49     :       R0=ADDRESS TO STORE THE NUMBER
50     :       JSR     R5,CK.NUM
51     :       RET1   ERROR RETURN
52     :       RET2   NORMAL RET
53
54
55
56     045654 010246  CK.NUM: MOV     R2,-(SP)      ;SAVE R2
57     045656 010346          MOV     R3,-(SP)      ;SAVE R3
    
```

58	045660	010446		MOV	R4,-(SP)	;SAVE R4	
59	045662	012703	000006	MOV	#6,R3	;MAX OCTAL DIGITS IN THE NUMBER	
60	045666	005002		CLR	R2	;FINAL OCTAL VALUE	
61	045670	112104		1\$:	MOVB	(R1)+,R4	;GET CURRENT POINTED BYTE
62	045672	001424			BEQ	3\$;BR IF TERMINATOR DETECTED
63	045674	120427	000060		CMPB	R4,#'0	;SMALLER THAN ASCII-0 ?
64	045700	103425			BLO	5\$;YES,ERROR EXIT
65	045702	120427	000067		CMPB	R4,#'7	;LARGER THAN ASCII-7 ?
66	045706	101022			BHI	5\$;YES,ERROR EXIT
67	045710	006302			ASL	R2	;SHIFT LEFT
68	045712	103420			BCS	5\$.
69	045714	006302			ASL	R2	;ONE
70	045716	103416			BCS	5\$.
71	045720	006302			ASL	R2	;OCTAL DIGIT
72	045722	103414			BCS	5\$;ERROR IF CARRY BIT SET
73	045724	042704	177770		BIC	#177770,R4	;CHOP OFF HIGHER BITS
74	045730	060402			ADD	R4,R2	;APPENDING CURRENT DIGIT
75	045732	005303			DEC	R3	;DECREMENT BYTE COUNT
76	045734	001401			BEQ	2\$;BR IF LAST DIGIT
77	045736	000754			BR	1\$;LOOPING BACK
78	045740	112104		2\$:	MOVB	(R1)+,R4	;CHECK TERMINATOR
79	045742	001004			BNE	5\$;BR IF NOT FOUND
80	045744	005702		3\$:	TST	R2	;FINAL VALUE = 0 ?
81	045746	001401			BEQ	4\$;YES
82	045750	010210			MOV	R2,(R0)	;REPLACE THE ORIGINAL VALUE
83	045752	005725		4\$:	TST	(R5)+	;ADJUST FOR NORMAL RET
84	045754	012604		5\$:	MOV	(SP)+,R4	;RESTORE 4
85	045756	012603			MOV	(SP)+,R3	;RESTORE R3
86	045760	012602			MOV	(SP)+,R2	;RESTORE R2
87	045762	000205			RTS	R5	;EXIT
88							

```
1          .SBTTL  READ/WRITE BUFFERS
2
3          ;BUFFER STARTS HERE
4
5 045764 000000 000000 000000 RBUF:  .WORD  0,0,0,0          ;BUFFER FOR HEADER CHECK
6
7 045774          BUFP:  .BLKW  <258.*32.>          ;FORMAT AND CHECK BUFFER STARTS HERE AND
8          ;WILL OVERLAY ALL REMAINING CODE IN PROGRAM
9 106174          BUFV:  .BLKW  <258.*32.>          ;BUFFER FOR VERIFY ONLY MODE
10
11          000200          .END  200
```

ABASE = 176700	ATO = 000001	BYTE16 006332	DH6 = 043746	EM14 = 042732
ACDW1 = 000000	AT1 = 000002	BYTE18 006432	DISPLA 001156	EM15 = 042755
ACDW2 = 000000	AT2 = 000004	C 036411	DISPRE 000174	EM16 = 043021
ACK = 000123	AT3 = 000010	CALIN 013322	DLT = 100000	EM17 = 043073
ACPUOP = 000000	AT4 = 000020	CHGADR 001376	DMD = 000001	EM2 = 042270
ACTDRV 027752	AT5 = 000040	C11 031352	DONE 017006	EM20 = 043151
ACTSTR 027753	AT6 = 000100	C13 031460	DPINT 027726	EM21 = 043207
ADDRIS 037775	AT7 = 000200	C14 031560	DPR = 000400	EM22 = 043260
ADDW0 = 000000	AUNIT = 000000	C15 032136	DPRQS 027736	EM23 = 043330
ADDW1 = 000000	AUSWR = 000000	C16 032160	DRIVE = 001222	EM24 = 043400
ADDW10 = 000000	AVECT1 = 000254	C17 032174	DRQ = 004000	EM25 = 043422
ADDW11 = 000000	AVECT2 = 000000	C17B 032216	DRVACT 027676	EM3 = 042326
ADDW12 = 000000	A16 = 000400	C18 032274	DRVCLR = 000111	EM4 = 042364
ADDW13 = 000000	A17 = 001000	CKSWR = 104407	DRVINT 030264	EM5 = 042421
ADDW14 = 000000	BADENT 037211	CKTRK 015006	DRVQUE 036036	EM6 = 042455
ADDW15 = 000000	BAD16 001414	CK.CHR 023410	DRVSTA 027706	EM7 = 042520
ADDW2 = 000000	BAD18 002420	CK.DEC 023362	DRVSTYP 027716	ENDCYL 001326
ADDW3 = 000000	BAI = 000010	CK.DIG 023462	DRY = 000200	ENDTRK 001332
ADDW4 = 000000	BEGCYL 001330	CK.NUM 045654	DSWR = 177570	ENTADR 036430
ADDW5 = 000000	BEGIN 006776	CK.OCT 023334	DS.CYL 001364	ERINDX 017204
ADDW6 = 000000	BEGIN2 007020	CLOCK 020250	DS.TRK 001366	ERR = 040000
ADDW7 = 000000	BEGTRK 001334	CLR = 000040	DTE = 010000	ERROR = 104000
ADDW8 = 000000	BIT0 = 000001	CLRQUE 035740	DTSY = 000200	ERRVEC = 000004
ADDW9 = 000000	BIT00 = 000001	CNTLC 001374	DTUW 030020	ES.SAV 036202
ADEVCT = 000000	BIT01 = 000002	CR = 000015	DT00 = 000001	F = 036413
ADEVM = 000000	BIT02 = 000004	CRLF = 000200	DT01 = 000002	FER = 000020
ADRTBL 006132	BIT03 = 000010	C.ENTR 020262	DT02 = 000004	FMT = 010000
AENV = 000000	BIT04 = 000020	DCK = 100000	DT03 = 000010	FMTDPB 005540
AENVM = 000000	BIT05 = 000040	DDISP = 177570	DT04 = 000020	FMTX 005630
AFATAL = 000000	BIT06 = 000100	DDRIVE 001370	DT05 = 000040	FMT16 = 010000
AMADR1 = 000000	BIT07 = 000200	DEVMP 001402	DT06 = 000100	FUNCT1 021714
AMADR2 = 000000	BIT08 = 000400	DF1 045402	DT07 = 000200	FUNCT2 022116
AMADR3 = 000000	BIT09 = 001000	DF10 045426	DT08 = 000400	FUNCT3 022414
AMADR4 = 000000	BIT1 = 000002	DF17 045442	DT1 = 045030	FUNCT4 022502
AMAMS1 = 000000	BIT10 = 002000	DF2 045406	DT10 045126	FUNCT5 022612
AMAMS2 = 000000	BIT11 = 004000	DF20 045446	DT17 045200	FUNCT6 022710
AMAMS3 = 000000	BIT12 = 010000	DF23 045466	DT2 = 045046	FUNCT7 023014
AMAMS4 = 000000	BIT13 = 020000	DF24 045472	DT20 045212	FUNCT8 023122
AMSGAD = 000000	BIT14 = 040000	DF25 045512	DT23 045274	F1 = 000002
AMSGLG = 000000	BIT15 = 100000	DF3 045412	DT24 045304	F2 = 000004
AMSGTY = 000000	BIT2 = 000004	DF4 045416	DT25 045372	F3 = 000010
AMTYP1 = 000000	BIT3 = 000010	DF6 045422	DT3 045062	F4 = 000020
AMTYP2 = 000000	BIT4 = 000020	DH1 043472	DT4 045076	F5 = 000040
AMTYP3 = 000000	BIT5 = 000040	DH10 043761	DT6 045114	GETREG = 000141
AMTYP4 = 000000	BIT6 = 000100	DH10A 044046	DT7 045116	GETREQ 036112
AOE = 001000	BIT7 = 000200	DH10B 044132	DULP 030266	GO = 000001
APASS = 000000	BIT8 = 000400	DH17 044210	DVA = 004000	GTSWR = 104406
APRIOR = 000000	BIT9 = 001000	DH2 043540	ECH = 000100	HCE = 000200
APTCSU = 000040	BLNKS1 036426	DH20 044257	ECI = 004000	HCI = 002000
APTENV = 000001	BLNKS2 036425	DH20A 044326	EDIT 001406	HCRC = 000400
APTSIZ = 000200	BLNKS3 036424	DH20B 044423	EMPTYQ 036016	HDREAD 015550
APTSPO = 000100	BLNKS4 036423	DH20C 044521	EMTVEC = 000030	HEADX 001410
ASWREG = 000000	BPTVEC = 000014	DH23 044554	EM1 = 042226	HEDERR 001356
ATA = 100000	BSE = 100000	DH24 044613	EM10 042572	HSET 015450
ATABIT 030022	BUFP 045774	DH25 044766	EM11 042610	HT = 000011
ATESTN = 000000	BUFV 106174	DH3 043612	EM12 042646	IAE = 002000
ATTN 001372	BUSADR 045516	DH4 043663	EM13 042711	IE = 000100

ILF = 000001	MSELD 036624	PGE = 002000	RETRY 001344	SEC30 001360
ILR = 000002	MSELP 037331	PGM = 001000	RMADR 030034	SEEK = 000105
INSERT 021602	MSFOR 037452	PIP = 020000	RMAS = 000016	SEEKFG 027776
IOTVEC= 000020	MSGBLK 040415	PIRQ = 177772	RMBA = 000004	SELDRV= 000145
IR = 000100	MSIZE 037004	PIRQVE= 000240	RMCS1 = 000000	SELPAT 013462
ISR 032706	MSQVER 037537	POPQUE 036134	RMCS2 = 000010	SERNL 012746
LA 032532	MSTCK = 000010	PRO = 000000	RMDA = 000006	SETFMT= 000143
LACNT 027764	MSVER 037600	PR1 = 000040	RMDB = 000022	SETHDR 017556
LF = 000012	MUNIT 036372	PR2 = 000100	RMDC = 000034	SETTBL 017352
LODEV 036315	MUSDR 036554	PR3 = 000140	RMDS = 000012	SETVEC 010126
LOP.CK 017304	MUTLTY 041204	PR4 = 000200	RMDT = 000026	SET.IE 035374
LST = 002000	MVCMP 037700	PR5 = 000240	RMEC1 = 000044	SKI = 040000
LSTRK 001324	MVERFY 036767	PR6 = 000300	RMEC2 = 000046	SLASH 036405
MADRER 037230	MWC 001354	PR7 = 000340	RMERRS 027666	SNGSEC 001322
MAXSEC 001362	MWR = 000040	PS = 177776	RMER1 = 000014	SN1 040101
MCCMPT 037656	MXDLTA 030044	PSEL = 002000	RMER2 = 000042	SOF SW 001316
MCHECK 036713	MXF = 001000	PSW = 177776	RMHR = 000036	SORT1 014116
MCLK = 000002	MXLACT 030042	PWRVEC= 000024	RMINIT 030052	SORT2 020710
MCPE = 020000	MXWNDW 030050	QCNT 035446	RMLA = 000020	SRCHWT 027750
MCPEMX 030032	M0 011342	QDRV0 035540	RMMR1 = 000024	STACK = 001100
MDRNP 036500	M1 010776	QDRV1 035560	RMMR2 = 000040	STARS 037753
MER11 036525	M1A 011132	QDRV2 035600	RMOF = 000032	START 007014
MESG1 040022	M1B 011306	QDRV3 035620	RMR = 000004	STCLK1 020074
MESG10 041622	M2 011720	QDRV4 035640	RMSN = 000030	STCLK2 020144
MESG11 041672	M4 012000	QDRV5 035660	RMTMR 034272	STCLK3 020150
MESG12 041742	M4A 012026	QDRV6 035700	RMVEC 030036	STKLMT= 177774
MESG13 042012	M5 012146	QDRV7 035720	RMWC = 000002	STO 034362
MESG14 042041	M5FX 010466	QINPT 035456	RM.REG 005560	STO1 034416
MESG15 042051	NBA = 100000	QOUTPT 035476	RM05 030622	STO2 034512
MESG17 042126	NED = 010000	QSTART 035516	RNOP = 000101	STO3 034542
MESG2 040212	NEM = 004000	QSTOP 035520	RTC = 000117	STO5 034566
MESG3 040112	NEXT 005536	QTERM = 035740	R6 = %000006	STO6 034574
MESG4 040354	NOTPRS 036270	RBUF 045764	R7 = %000007	STO7 034632
MESG5 041236	NOTRM 036247	RDCHR = 104410	SAVEFG 027774	STO8 034662
MESG6 041321	NOTSAF 036305	RDDAT = 000171	SAVREG= 104412	STO9 034672
MESG7 041404	NUMERR 037723	RDHD = 000173	SAVSEC 001346	ST.CLK 020016
MESG8 041467	OFDIR = 000001	RDLIN = 104411	SAVWC 001350	ST1 013462
MESG9 041552	OFFSET= 000115	RDY = 000200	SC 033122	SVRH70 035256
MF = 100000	OP1 = 020000	RD.ADR 034730	SCAWC 017530	SWR 001154
MFCMPT 037633	OPT 031114	RD.RM 034704	SCOPE = 000004	SWREG 000176
MFORMT 036672	OR = 000200	RD.RM1 034726	SC01 = 000100	SW0 = 000001
MHALT 040373	O.ENTR 020302	RD.RM2 035052	SC02 = 000200	SW00 = 000001
MINX = 000004	PACK 001404	RD.RM3 035056	SC04 = 000400	SW01 = 000002
MLODEV 036574	PAR = 000010	RD.RM4 035062	SC10 = 001000	SW02 = 000004
MMODE 036636	PARENT 020454	RD.WRD 034732	SC11 033754	SW03 = 000010
MNDLTA 030046	PAR1 005776	READIN= 000121	SC12 034064	SW04 = 000020
MODE 001320	PAR2 006011	RECAL = 000107	SC13 034134	SW05 = 000040
MOFFLN 036457	PAR3 006024	RECORD 020570	SC20 = 002000	SW06 = 000100
MOH = 020000	PAR4 006035	REJCT2 017134	SC3 033172	SW07 = 000200
MOL = 010000	PAR5 006046	REJCT3 017162	SC4 033176	SW08 = 000400
MPE = 000400	PAR6 006057	REJEC1 017112	SC5 033210	SW09 = 001000
MRD = 000020	PAR7 006070	RELSE = 000113	SC6 033404	SW1 = 000002
MRMCS1 045636	PAR8 006101	RESET 021416	SC6A 033520	SW10 = 002000
MRMVEC 045645	PAT = 000020	RESREG= 104413	SC7 033646	SW11 = 004000
MSCHK 037505	PATA 001340	RESTRT 020160	SC8 033724	SW12 = 010000
MSEC30 037053	PATB 001342	RESVEC= 000010	SEARCH= 000131	SW13 = 020000
MSEC32 037132	PATSEL 001336	RETBAD 012230	SECCU 005434	SW14 = 040000

SW15 = 100000	UPDTRK 017756	SBDADR 001136	SLF 001206	\$RTNAD 017004
SW2 = 000004	UPE = 020000	\$BDDAT 001142	\$LFLG 025103	\$SAVRE 027506
SW3 = 000010	USSAV 004430	\$BELL 001200	\$LKCSB 001302	\$SAVR6 027200
SW4 = 000020	USTAB 003424	\$CDW1 001270	\$LKCSR 001300	\$SB2D 027256
SW5 = 000040	US1 = 000001	\$CDW2 001272	\$LKS 001310	\$SETUP= 000116
SW6 = 000100	US2 = 000002	\$CHARC 024634	\$LLVEC 001312	\$STUP = 177777
SW7 = 000200	US4 = 000004	\$CKSWR 026064	\$LPADR 001122	\$SUPRS 027216
SW8 = 000400	VV = 000100	\$CMTAG 001114	\$LPERR 001124	\$SVPC = 000210
SW9 = 001000	WC = 001352	\$CM3 = 000000	\$LPVEC 001304	\$SWR = 123000
T 036415	WCE = 040000	\$CM4 = 000001	\$MADR1 001242	\$SWREG 001232
TABCAL 006120	WCF = 000040	\$CNTLC 026767	\$MADR2 001246	\$TESTN 001214
TABINT 012532	WCKD = 000151	\$CNTLG 027001	\$MADR3 001252	\$TKB 001162
TABLD 012614	WCKHD = 000153	\$CNTLU 026774	\$MADR4 001256	\$TKCNT 025560
TABLE 005650	WCTBL 006232	\$CPUOP 001236	\$MAIL 001210	\$TKINT 025576
TABLE2 005702	WLE = 004000	\$CRLF 001205	\$MAMS1 001240	\$TKQEN= 025575
TABLE3 005726	WRAP 001400	\$DBLK 025550	\$MAMS2 001244	\$TKQIN 025562
TABLE4 005752	WRL = 004000	\$DB2D 027312	\$MAMS3 001250	\$TKQOU 025564
TAB.XY= 001114	WRDAT= 000161	\$DECVL 027472	\$MAMS4 001254	\$TKQSR 025566
TAP = 040000	WRTHD = 000163	\$DEVCT 001220	\$MBADR 001102	\$TKS 001160
TBITVE= 000014	WRTRK 013622	\$DEVN 001266	\$MLG 025102	\$TKSRV 025646
TD 032752	WRTRKA 014320	\$DOAGN 017002	\$MNEW 027017	\$TMPD 001174
TEXT 021050	WRTRKB 014436	\$DTBL 025540	\$MSGAD 001224	\$TN = 000002
TIMER 030000	WRTRKC 014522	\$ENDAD 016772	\$MSGLG 001226	\$TNPWR 027422
TKVEC = 000060	WRTRKD 014604	\$ENDCT 016756	\$MSGTY 001210	\$TPB 001166
TPVEC = 000064	WRTRKE 014652	\$ENV 001230	\$MSWR 027006	\$TPFLG 001173
TRAPVE= 000034	WRTRKF 014566	\$ENVM 001231	\$MTYP1 001241	\$TPS 001164
TRE = 040000	WRTRKH 014616	\$EOP 016342	\$MTYP2 001245	\$TRAP 027602
TRKTST 017426	WRTRKX 013676	\$EOPCT 016750	\$MTYP3 001251	\$TRAP2 027624
TRK1 = 004000	WRTRKY 013744	\$ERFLG 001117	\$MTYP4 001255	\$TRP = 000014
TRK10 = 040000	WRTRKZ 015536	\$ERMAX 001131	\$NULL 001170	\$TRPAD 027636
TRK2 = 010000	WRTRK1 014270	\$ERROR 023620	\$NWTST= 000000	\$STM 001104
TRK20 = 100000	WRTRK2 014666	\$ERRPC 001132	\$OCNT 025330	\$STNM 001116
TRK4 = 020000	WRT.AD 035154	\$ERRTB 006526	\$OMODE 025332	\$TTYIN 026752
TRNSWT 027746	WRT.RM 035064	\$ERTTL 001126	\$PASS 001216	\$TYPDS 025334
TRTVEC= 000014	WRT.R1 035150	\$ESCAP 001176	\$PASTM 001106	\$TYPE 024304
TST1 007020	WRT.R2 035242	\$ETABL 001230	\$POWER 027202	\$TYPEC 024516
TWO 036421	WRT.R3 035246	\$ETEND 001274	\$PWRAD 027170	\$TYPEX 024636
TYPADR 020354	WRT.R4 035252	\$FATAL 001212	\$PWRDN 027030	\$TYPOC 025132
TYPDS = 104405	WRT.R5 035254	\$FFLG 025104	\$PWRMG 027164	\$TYPON 025146
TYPE = 104401	WRT.WD 035152	\$FILLC 001172	\$PWRUP 027102	\$TYPOS 025106
TYPERR 024054	XSIZE 010422	\$FILLS 001171	\$QUES 001204	\$UNIT 001222
TYPOC = 104402	XXDP 001412	\$GDADR 001134	\$RDCHR 026426	\$UNITM 001110
TYPON = 104404	Y 036417	\$GDDAT 001140	\$RDLIN 026516	\$USWR 001234
TYPOS = 104403	ZERO 036403	\$GET42 016762	\$RDSZ = 000015	\$VECT1 001260
UF = 040000	\$APTHD 001100	\$GTSWR 026154	\$RESRE 027544	\$VECT2 001262
ULDFLG 027754	\$ATYC 024664	\$HD = 000000	\$RMADR 001274	\$XOFF = 000023
UNS = 040000	\$ATY1 024640	\$HIBTS 001100	\$RMVEC 001276	\$XON = 000021
UNSTAT 036354	\$ATY3 024646	\$ICNT 001120	\$RM02 036335	\$SET4= 000000
UNTOFF 036226	\$ATY4 024656	\$ILLUP 027174	\$RM03 036342	\$OFILL 025331
UNTON 036237	\$AUTOB 001150	\$INTAG 001151	\$RM05 036347	\$.SX - 001100
UPDCYL 017644	\$BASE 001264	\$ITEMB 001130		

. ABS. 146374 000
000000 001
ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 45296 WORDS (177 PAGES)

CZRMLAO RM05/3/2 FORMATTER
SYMBOL TABLE

MACRO V03.01 11-APR-80 13:01:23 PAGE 18-4

E 11

CZRMLA.BIC,CZRMLA/C=CZRMI A.DOC,CZRMLA,SYSMAC/M

SEQ 0134

SRDSZ 10-762 10-762#

H 11

SEQ 0137

STYPOS 10-258# 10-275

J 11

SEQ 0139

SUNIT	5-0#	6-0				
SUNITM	4-286#					
SUSWR	5-0#					
\$VECT1	5-0#	8-68	8-70			
\$VECT2	5-0#					
\$XOFF	10-?54	10-?54				
\$XON	10-?54	10-?54				
.SASTA	10-?56	10-?56				
.SX	4-286	4-286#				
A16	4-86#					
A17	4-87#					
ABASE	4-274#	5-0	5-0			
ACDW1	5-0	5-0				
ACDW2	5-0	5-0				
ACK	4-262#					
ACPUOP	5-0	5-0				
ACTDRV	11-92#	11-361*	11-412*	11-738*	11-747*	11-:09
ACTSTR	11-98#	11-:11*	11-:25*			
ADDRIS	10-853	12-56#				
ADDW0	5-0					
ADDW1	5-0					
ADDW10	5-0					
ADDW11	5-0					
ADDW12	5-0					
ADDW13	5-0					
ADDW14	5-0					
ADDW15	5-0					
ADDW2	5-0					
ADDW3	5-0					
ADDW4	5-0					
ADDW5	5-0					
ADDW6	5-0					
ADDW7	5-0					
ADDW8	5-0					
ADDW9	5-0					
ADEVCT	5-0	5-0				
ADEVN	5-0	5-0				
ADRTBL	6-0#	10-237	10-308	10-568	10-==00	
AENV	5-0	5-0				
AENVN	5-0	5-0				
AFATAL	5-0	5-0				
AMADR1	5-0	5-0				
AMADR2	5-0	5-0				
AMADR3	5-0	5-0				
AMADR4	5-0	5-0				
AMAMS1	5-0	5-0				
AMAMS2	5-0	5-0				
AMAMS3	5-0	5-0				
AMAMS4	5-0	5-0				
AMSGAD	5-0	5-0				
AMSGLG	5-0	5-0				
AMSGTY	5-0	5-0				
AMTYP1	5-0	5-0				
AMTYP2	5-0	5-0				
AMTYP3	5-0	5-0				
AMTYP4	5-0	5-0				

AOE

4-157#

L 11

SEQ 0141

0113 4-780

N 11

SEQ 0143

BIT4	4-78#													
BIT5	4-78#													
BIT6	4-78#													
BIT7	4-78#	8-146												
BIT8	4-78#													
BIT9	4-78#													
BLNKS1	12-24#													
BLNKS2	8-122	10-858	10-06	10-29	10-38	10-49	12-23#	17-21	17-30					
BLNKS3	12-22#													
BLNKS4	8-93	12-21#												
BPTVEC	4-78#													
BSE	4-238#	10-430												
BUFP	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0
	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0	6-0
	6-0	6-0	6-0	6-0	9-30	9-68	9-72	9-74	9-76	9-78	9-80	9-83	9-113	9-119
	9-125	9-128	10-27	10-150	10-397	10-568	10-668	10-715	10-736	10-761	10-978	10-:00	10-:13	10-:16
	10-:19	10-:22	10-:25	10-:28	18-7#									
BUFV	10-398	18-9#												
BUSADR	8-62	17-13#												
BYTE16	6-0#	10-33												
BYTE18	6-0#	10-43												
C	10-854	12-16#												
C.ENTR	8-281	10-823#	10-852											
CALIN	9-215#													
CHGADR	6-0#	8-3*	17-13	17-15*										
CI1	11-459	11-487#												
CI3	11-461	11-510#												
CI4	11-450	11-532#												
CI5	11-509	11-531	11-609#	11-618										
CI6	11-548	11-558	11-560	11-562	11-614#									
CI7	11-391	11-468	11-499	11-503	11-507	11-515	11-525	11-529	11-540	11-547	11-553	11-557	11-571	11-577
	11-581	11-591	11-602	11-617	11-619#	11-733	11-763	11-940	11-<29					
CI7B	11-627#	11-944												
CI8	11-630	11-641#	11-813	11-:82										
CK.CHR	8-293	8-384	10->08#	10->45	10->53									
CK.DEC	10-86#	10->14												
CK.DIG	10-888	10-<92	10->39#											
CK.NUM	17-24	17-33	17-56#											
CK.OCT	10-68#	10->16												
CKSWR	10->86	10->86	10-?75#											
CKTRK	10-266#													
CLOCK	10-776	10-786	10-817#											
CLR	4-106#	11-242	11-680											
CLRQUE	11-274	11-679	11-13#											
CNTLC	6-0#	8-57*	8-281*	10-842	10-852*	10-?62	10-?62							
CR	4-78#	8-219	8-248	8-287	8-378	10-803	10-?54	10-?54	14-15					
CRLF	4-78#	8-15	8-15	8-18	8-18	8-35	8-46	8-52	8-53	10-?54	10-?54	10-?65	12-12	12-13
	12-25	12-26	12-27	12-28	12-29	12-30	12-32	12-33	12-34	12-35	12-36	12-37	12-37	12-38
	12-38	12-39	12-40	12-41	12-42	12-42	12-43	12-44	12-45	12-47	12-48	12-49	12-50	12-51
	12-52	12-53	12-55	12-55	12-58	12-59	12-59	12-61	12-62	12-63	12-64	12-65	12-65	12-66
	12-67	12-68	12-69	12-70	12-71	12-73	12-74	12-75	12-76	12-77	12-78	12-79	12-80	12-81
	12-83	12-84	12-84	12-86	12-87									
DCK	4-163#	10-289												
DDISP	4-78#	5-0	8-14											
DDRIVE	6-0#	10->86*	15-2	15-3	15-4									
DEVMP	6-0#	8-88*	8-121*	8-151*	8-152*	8-153	8-184	8-192*	10-568	10-572				

DTOS 4-196#

E 12

SEQ 0147

F2

4-1270

G 12

SEQ 0149

M1A

8-2440

1 12

SEQ 0151

MSIZE 8-245 12-36#

K 12

SEQ 0153

1
1

PR7

4-78#

8-87

M 12

SEQ 0155

RM05 9-34 9-43 10-46 10-242 10-268 10-409 10-568^{B 13} 10-568 10-:70 11-359#

SEQ 0157

SEC30 6-0# 8-166* 8-176* 8-260* 8-267* 9-126 10-42^{0 13} 10-74 10-720 10-988 10-990* 10-994* 10-:20 10-:26
SEQ 0159

SW15 4-78#

F 13

SEQ 0161

SW2	4-78#													
SW3	4-78#													
SW4	4-78#													
SW5	4-78#													
SW6	4-78#													
SW7	4-78#													
SW8	4-78#													
SW9	4-78#													
SWR	5-0#	8-14	8-14	8-14*	8-14*	8-14*	8-15	8-85	10-347	10-576	10-649	10->86	10->86	10->86
SWREG	10->86	10-?62	10-?62	10-?62*	10-?64	10-?64*								
T	4-279#	8-14	8-15	10-?62	10-?62	10-?62								
TAB.XY	10-859	12-18#												
TABCAL	4-287#	5-0												
TABINT	6-0#	9-235												
TABLD	9-90#	10-98												
TABLE	9-111#													
TABLE2	6-0#	8-342*	8-344*	8-351										
TABLE3	6-0#	8-342*	10-;12	10-;64										
TABLE4	6-0#	8-342*	10-;32	10-;83										
TAP	6-0#	8-342*	10-;20											
TBITVE	4-202#													
TD	4-78#													
TEXT	11-743	11-752#												
TIMER	10-568	10-978#												
TKVEC	11-137#	11-343*	11-466*	11-610*	11-663*	11-755*	11-873*	11-918*	11-961*	11-976*	11-998*	11-;15	11-;17*	11-;80*
TPVEC	11-;86*	11-;92*												
TRAPVE	4-78#	9-5*	10-8*	10-865*	10-?62*	10-?62*								
TRE	4-78#	8-14*	8-14*											
TRK1	4-90#													
TRK10	4-213#													
TRK2	4-216#													
TRK20	4-214#													
TRK4	4-217#													
TRKTST	4-215#													
TRNSWT	10-365	10-678#												
TRIVEC	10-824	11-78#	11-488*	11-635	11-638*	11-651	11-669*	11-678*	11-757	11-758*	11-;55	11-;65*		
TST1	4-78#													
TWO	8-9#													
TYPADR	8-385	12-20#												
TYPDS	10-851#													
TYPE	10-568	10-884	10-?26	10-?75#										
	8-15	8-18	8-35	8-4	8-46	8-52	8-53	8-90	8-91	8-93	8-101	8-104	8-107	8-110
	8-117	8-120	8-122	8-133	8-139	8-216	8-221	8-228	8-230	8-233	8-237	8-238	8-244	8-245
	8-250	8-255	8-258	8-259	8-266	8-284	8-291	8-295	8-298	8-306	8-317	8-322	8-325	8-345
	8-349	8-360	8-375	8-382	8-385	8-391	8-394	9-12	9-15	9-17	9-152	9-153	9-154	9-222
	9-223	9-230	9-231	9-238	10-156	10-170	10-389	10-391	10-393	10-568	10-568	10-568	10-568	10-568
	10-593	10-594	10-595	10-599	10-602	10-606	10-607	10-608	10-800	10-807	10-811	10-812	10-840	10-853
	10-854	10-858	10-859	10-864	10-879	10-885	10-891	10-925	10-926	10-927	10-;06	10-;07	10-;26	10-;27
	10-;58	10-;59	10-;77	10-;78	10-<01	10-<02	10-<08	10-<10	10-<27	10-<28	10-<34	10-<35	10-<63	10-<84
	10-<88	10-;02	10-;06	10-;12	10-;15	10->86	10->86	10-?04	10-?05	10-?09	10-?10	10-?19	10-?29	10-?33
	10-?38	10-?40	10-?46	10-?49	10-?54	10-?58	10-?60	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62
	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62	10-?62
	17-16	17-18	17-21	17-27	17-30									
TYPERR	10->86	10->94#												
TYPDC	10-<05	10-<13	10-<31	10-<38	10-;05	10-?24	10-?62	10-?75#	17-20	17-29				

TYPON 10-275#

H 13

SEQ 0163

