

# RM03,02

FORMATTER  
CZRMAC0

AH-9254C-MC

COPYRIGHT © 1977

FICHE 1 OF 1

JAN 1978

**digital**

MADE IN USA





94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149

1. ABSTRACT  
-----

THE RMO3 FORMATTER PROGRAM PROVIDES THE FACILITIES TO FORMAT OR TO CHECK THE HEADER AND DATA FIELDS OF EACH DATA BLOCK ON THE DISC 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; WRITES THE DATA FIELD WITH SELECTED DATA PATTERN. THE PROGRAM THEN VERIFIES THE WRITTEN DATA BLOCKS VIA EXECUTING THE "WRITE CHECK HEAD AND DATA" COMMAND.

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  
TELETYPE  
PROGRAM LOAD DEVICE  
KW11-L OR KW11-P CLOCK  
RH11 OR RH70 WITH 1 - 8 RMO3 DISK DRIVES  
DISK PACK(S)

2.2 PRELIMINARY PROGRAMS

THERE ARE NO PREQUISITE PROGRAMS PROVIDING THE RMO3 SUBSYSTEM IS KNOWN TO BE OPERATIONAL. IF THE STATE OF THE RMO3 SUBSYSTEM IS UNKNOWN, THE FOLLOWING PROGRAMS SHOULD BE RUN PRIO TO USING THE FORMATTER PROGRAM  
RMO3 DISKLESS DIAGNOSTIC  
RMO3 FUNCTIONAL TEST

3. LOADING PROCEDURES  
-----

3.1 PAPER TAPE AND XXDP

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

3.2 APT

150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
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

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

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

AUTOMATIC MODE (MONITOR)

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

DUMP MODE: INPUT DIALOGUE AFTER PROGRAM STARTS

### 3.3 APT ETABLE DEFINITIONS

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

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

206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251

- 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 SECTORS FOR A TRACK IN ALL OPERATIONS.

4. STARTING PROCEDURES  
-----

4.1 STARTING ADDRESSES

THE PROGRAM IS STARTED FROM LOCATION 200(8) 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(8) 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(8) OR 204(8).
- 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 (C OR F):'

262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
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

ENTER THE APPROPRIATE CODE: 'C' FOR 'CHECK' OPERATION OR 'F' FOR 'FORMAT & VERIFY'. IF A 'CARRIAGE RETURN' IS ENTERED IN RESPONSE TO THE REQUEST, THE PROGRAM WILL ASSUME 'FORMAT & 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 D E C I M A L 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
- (1) ONE'S
- (2) WORST CASE:'

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

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  
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

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'

IF A NEW PACK IS UNDER FORMATTED, THE PROGRAM ALSO ASKS TO ENTER TWO SERIAL NUMBERS. THESE TWO SERIAL NUMBER MUST BE NON-ZERO DECIMAL NUMBERS.

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

'PRESENT ADDRESS IS: CXXX TXX'

IF A 'CONTROL C' IS TYPED WHILE THE PROGRAM IS TYPING THE CURRENT ADDRESS, THE PROGRAM WILL ABORT THE FORMAT (OR CHECK) OPERATION AND RETURN TO THE 'DRIVE' REQUEST.

11. IF THE OPERATOR DOES NOT SELECT A DIFFERENT DRIVE WHEN THE FORMAT OR CHECK OPERATION HAS COMPLETED, THE PROGRAM WILL NOT ALTER THE ADDRESS LIMITS SPECIFIED AT THE START OF THE PREVIOUS OPERATION. IF THE FORMAT OR CHECK OPERATION IS TERMINATED BY A 'CONTROL C' OR IF A DIFFERENT DRIVE IS SELECTED, THE PROGRAM WILL RESET THE ADDRESS LIMITS TO THE VALUES APPROPRIATE FOR THE DRIVE TYPE.

12. TO CHANGE EITHER THE ADDRESSING MODE OR THE OPERATION MODE, THE PROGRAM MUST BE STARTED FROM LOCATION 200(8) OR 204(8) AGAIN.

4.3 RH11 - RH70 UNIBUS ADDRESS

THE PROGRAM ASSUMES THAT THE RH11 OR RH70 ADDRESSES START AT 176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY BE CHANGED WHEN THE PROGRAM IS STARTED FROM LOCATION 204(8).



374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429

IF THE RH11 - RH70 IS NOT AT THE DEFAULT ADDRESS, THE PROGRAM MUST BE STARTED FROM 204(8) 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.

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 (8). 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 RMO3 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  
-----

1. 'RH11 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 RH11 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. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH11 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 SUCESSIVE 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 SUCESSFUL - SECTOR NOT ACCEPTABLE' - THE INDICATED SECTOR FAILED DURING RETRY FOLLOWING A DATA ERROR.

430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485

486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600

- 16. 'CONTROLLER/DRIVE ERROR VERIFYING HEADERS' - AN ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
- 17. ''HCE' ERROR VERIFYING HEADERS' - A HEADER ERROR OCCURRED DURING THE VERIFICATION PASS AFTER FORMATTING.
- 18. 'CYLINDER FIELD IN HEADER IS NOT CORRECT' - THE CYLINDER FIELD FROM A HEADER READ DURING THE VERIFICATION PASS IS NOT CORRECT.
- 19. 'WRITE CHECK ERROR' - THE RH11 REPORTED A WRITE CHECK ERROR AND NO DRIVE ERRORS WERE SET.

7. MISCELLANEOUS  
-----

7.1 FORMAT TIME

IT TAKES APPROXIMATELY 10 MINUTES TO FORMAT AN ENTIRE RMD3 PACK. THE 'CHECK' MODE TIME FOR AN ENTIRE RMD3 PACK IS 24 MINUTES.

7.2 HALTING THE PROGRAM

THE OPERATOR SHOULD NOT HALT THE PROGRAM DURING A FORMAT OPERATION. HALTING THE PROGRAM MAY LEAVE A SECTOR INCORRECTLY FORMATTED. TO TERMINATE THE FORMAT, TYPE A 'CONTROL O' AND WHILE THE PROGRAM IS TYPING THE ADDRESS, TYPE ANOTHER 'CONTROL C'; THIS SEQUENCE RETURNS THE PROGRAM TO THE DRIVE ADDRESS ENTRY ROUTINE.

7.3 SURFACE VERIFICATION

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

8. PROGRAM DESCRIPTION  
-----

8.1 FORMAT OPERATION

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

THE FORMAT OPERATION CONSISTS OF A WRITE HEADER AND DATA COMMAND FOR THE ENTIRE TRACK FOLLOWED BY A WRITE CHECK HEADER AND DATA COMMAND. IF AN ERROR OCCURS DURING THE WRITE, THE PROGRAM WILL RETRY THE WRITE BEFORE CONTINUING. IF A DATA RELATED ERROR IS DETECTED DURING THE WRITE CHECK (A DATA ERROR IS DEFINED AS ONE OF

542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597

THE FOLLOWING ERRORS: 'DCK', 'OPI', 'DTE', 'HCRC', 'HCE' OR 'FER'), THE SECTOR IN ERROR WILL BE REWRITTEN. THE PROGRAM WILL CHECK THE SECTOR TWICE; IF A DATA ERROR IS DETECTED IN THE SECTOR DURING EITHER OF THE WRITE CHECKS, THE PROGRAM WILL DECLARE THE SECTOR AS BEING 'NONRECOVERABLE'. FOLLOWING THIS SEQUENCE, THE REMAINDER OF THE TRACK IS CHECKED. IF DATA ERRORS ARE ENCOUNTERED IN ANY OF THE REMAINING SECTORS, EACH SECTOR WILL BE HANDLED AS DESCRIBED ABOVE.

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

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

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

8.2 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

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.

9. BAD SPOT FILE

SECTORS 0 THROUGH 31, ON CYLINDER 822, TRACK 4 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

598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652

NUMBER AND CONSISTS OF MAXIMUM 10 OCTAL DIGITS.  
THE FIRST WORD OF THE SERIAL NUMBER CONTAINS THE 5  
LEST 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 SPECIES THE CYLINDER ADDRESS, THE SECOND WORD  
SPECIFIES 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.  
SECTORS 0,2,4,6 AND 8 CONTAIN THE MANUFACTURE DEFINED BAD SECTORS  
FOR 16 BIT MODE;SECTORS 1,3,5,7 AND 9 FOR 18 BIT MODE.  
SECTORS 10,12...THROUGH 30 CONTAIN THE USER DEFINED BAD SECTORS  
FOR 16 BIT MODE;SECTORS 11,13...THROUGH 31 FOR 18 BIT MODE.  
THE SECTORS IN EACH GROUP ARE IDENTICAL AND SECTORS 0,1,10AND 11 ARE  
USED AS REFERENCE SECTORS BY THE PROGRAM.

THE PROGRAM PROVIDES A FEW UTILITY ROUTINES TO HANDLE THE BAD  
SECTOR FILE.

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 FOLLOWED BY <CR>.

KEYIN U.S.R. BAD SECTORS.....0  
PRINT M.F.G. BAD SECTORS.....1  
PRINT U.S.R. BAD SECTORS.....2  
INITIALIZE U.S.R. BAD SECTOR FILE.....3  
PRINT A SECTOR OF THE LAST TRACK.....4  
EXIT.....<CR>

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 . FOLLOWED BY <CR>.

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

654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668

THE SECOND AND THIRD ROUTINES TYPE THE MANUFACTURE AND USER  
DEFINED BAD SECTORS ON THE CONSOLE IN OCTAL NUMBER.  
THE FIRST FOUR WORDS ON THE LISTING ARE THE SERIAL NUMBER FOLLOWED  
BY TWO 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 ENTER .<CR> .

669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724

10.1 RH70(11)/RMO3 DRIVER

THIS DOCUMENT IS THE USER'S GUIDE FOR THE RH70/RMO3 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 RMO3
=0	OFFLINE RMO3, DRIVE IS NOT AN RMO3, OR NONEXISTENT DRIVE
<0	UNSAFE RMO3

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	RMO3
-1	NOT AN RMO3

THE 'RMINIT' ROUTINE WILL DO A READIN PRESET AND WILL SET FMT22.

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

```
CALL: JSR RO,RMO3 ;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 FMT22, ECT, AND HCI
.BYTE	0	:(2) COMMAND
.BYTE	0	:(3) PSEL AND A17 AND A16
.WORD	0	:(4) WORD COUNT (MUST BE NEG.)

```

725 .WORD 0 ;(6) BUFFER ADDRESS OR
726 ;REGISTER TABLE POINTER
727 .BYTE 0 ;(10) SECTOR ADDRESS OR
728 ;FIRST REG. INDEX
729 .BYTE 0 ;(11) TRACK ADDRESS OR
730 ;LAST REG. INDEX
731 .WORD 0 ;(12) CYLINDER ADDRESS
732 .WORD 0 ;(14) ERROR TABLE POINTER
733 ;POINTS TO THE FIRST OF TWENTY
734 ;LOCATIONS OF WHERE THE DRIVER
735 ;IS TO STORE THE RH70/RM03
736 ;REGISTERS ON AN ERROR, IF LEFT
737 ;ZERO REGISTERS ARE NOT SAVED.
738 .WORD 0 ;(16) STATUS/ERROR INDICATOR
739 ;BIT15=1=>ERROR OCCURRED
740 ;BIT07=1=>DONE
741 ;BIT14-BIT09 AND BIT06-BIT03
742 ;INDICATE TYPE OF ERROR
743

```

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
JSR PC,RMTMR ;CLOCK TICKS
;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$: JSR RO,RM03 ;CALL THE DRIVER
WRTDPB ;DPB ADDRESS
BR 1$ ;WAIT FOR QUEUE IF FULL
2$: TST WRTDPB+16 ;WAIT FOR COMMAND TO COMPLETE
BEQ 2$
BMI ERROR1 ;ERROR OCCURRED
.
.
.
WRTDPB: .BYTE 5 ;DRIVE #5
.BYTE 0
.BYTE 161 ;WRITE COMMAND
.BYTE 0
.WORD -1000. ;WORD COUNT
.WORD WRTBUF ;BUFFER ADDRESS
.BYTE 3 ;SECTOR
.BYTE 5 ;TRACK
.WORD 400 ;CYLINDER
.WORD ERRTB5 ;ERROR TABLE

```

780



781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836

.WORD 0 ;STATUS/ERROR INDICATOR

ALTERNATE DPB SETUP

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

10.5 RH70/RMD3 REGISTERS

MNEMONIC	INDEX
RMCS1	0
RMWC	2
RMBA	4
RMDA	6
RMCS2	10
RMD5	12
RMER1	14
RMA5	16
RMLA	20
RMDB	22
RMMR1	24
RMDT	26
RMSN	30
RMOF	32
RMDC	34
RMHR	36
RMMR2	40
RMER2	42
RMEC1	44
RMEC2	46

10.6 COMMANDS PERFORMED BY THE DRIVER

COMMAND	CODE	COMMAND TYPE
NO OPERATION	101	N
UNLOAD	103	N
SEEK	105	P
RECALIRATE	107	P
DRIVE CLEAR	111	N
RELEASE	113	N
OFFSET	115	P

037  
038  
039  
040  
041  
042  
043  
044  
045  
046  
047  
048  
049  
050  
051  
052  
053  
054  
055  
056  
057  
058  
059  
060  
061  
062  
063  
064  
065  
066  
067  
068  
069  
070  
071  
072  
073  
074  
075  
076  
077  
078  
079  
080  
081  
082  
083  
084  
085  
086  
087  
088  
089  
090  
091  
092

RETURN TO CENTER	117	P
READIN PRESET	121	N
PACK ACKNOWLEDGE	123	N
SEARCH	131	P
GET REGISTER(S)	141	S
SET FORMAT	143	S
SELECT DRIVE	145	S
WRITE CHECK DATA	151	D
WRITE CHECK HEADER AND DATA	153	D
WRITE DATA	161	D
WRITE HEADER & DATA	163	D
READ DATA	171	D
READ HEADER & DATA	173	D

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

10.7 DPB STATUS/ERROR INDICATOR WORD

THIS INDICATOR WILL INFORM THE USER OF THE RESULTS OF THE REQUEST.  
THIS IS ACCOMPLISHED BY SETTING VARIOUS 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 PERFORMED ON AN OFFLINE OR UNSAFE DRIVE
13(1)	USER MADE A REQUEST FOR A FUNCTION TO BE PERFORMED ON A DRIVE THAT HAS AN UNLOAD REQUEST IN QUEUE.
12(2)	PERSISTENT UNSAFE CONDITION EXIST.
11(2)	UNCORRECTABLE PARITY ERROR OCCURRED

893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948

10(2)(4)	FATAL PARITY ERROR. A MASSBUS CLEAR WAS PERFORMED. ALL QUEUES WERE EMPTIED, AND ALL DRVACT'S SET TO THE IDLE STATE
9(3)(4)	SOFTWARE TIMEOUT OCCURRED ON THIS DRIVE
8(4)	SOFTWARE TIMEOUT OCCURRED ON ANOTHER DRIVE
7	DONE
6(2)	ERROR OCCURRED DURING AN I/O OPERATION
5(2)	ERROR OCCURRED DURING AN OPERATION OTHER THAN I/O.
4(2)	CORRECTABLE UNSAFE CONDITION OCCURRED
3(2)	DRIVE ERROR OCCURRED THAT CAUSED AN AUTOMATIC "RECALIBRATE" SEQUENCE
2	PORT REQUEST TIMEOUT. THE DRIVER REQUESTED THE DRIVE BUT THE OPPOSITE PORT DID NOT RELEASE THE DRIVE WITHIN 20 SECONDS.
1	NON-EXISTENT DRIVE REQUESTED. USER MADE A REQUEST FOR A NON-EXISTENT DRIVE.
(1) =>	REQUEST WASN'T PUT IN QUEUE. (RH70/RM03 REGISTERS WERE NOT SAVED)
(2) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A "DRIVE CLEAR" TO THE DRIVE. NOTE: ALL RH70/RM03 REGISTERS ARE SAVED AS PER DPB+14 BEFORE THE "DRIVE CLEAR".
(3) =>	REQUEST QUEUE HAS BEEN EMPTIED. THE DRIVER ISSUED A MASSBUS INIT. ALL RH70/RM03 REGISTERS FOR THE DRIVE WERE SAVED AS PER DPB+14 BEFORE THE INIT.
(4) =>	A "RECALIBRATE" SHOULD BE ISSUED BEFORE ANY OTHER COMMAND.

10.8 ERROR CALLS MADE BY THE DRIVER.

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

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

N	TYPE	DATA AVAILABLE
-	----	-----

949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980

- 1 RH70 INTERRUPT OCCURRED (RHAS=0) \*R4= RMCS1'S ADDRESS
  - 2 UNEXPECTED ATTENTION OCCURRED R1= DRIVE NUMBER  
R3= ATA BIT  
\*R4= RMCS1'S ADDRESS  
R5= (RMAS)  
RMERRS =RMDS  
RMERRS+2=RMER1  
RMERRS+4=RMER2  
RMERRS+6=RMMR2
  - 3 MASSBUS PARITY ERROR (MCPE=1) RD.ADR= ADDRESS OF REG. READ  
RD.WRD= WORD READ
  - 4 MASSBUS PARITY ERROR (PAR=1) WRT.AD= ADDRESS OF REG. 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)  
%

981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036

001100

000011  
000012  
000015  
000200  
177776  
177774  
177772  
177570  
177570

000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007  
000006  
000007

000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340

100000  
040000  
020000  
010000

```
.TITLE CZRMACD RM03/2 FORMATTER
;*COPYRIGHT (C) 1977
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY C. HESS/C. CHEN
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
;*
.SBTTL BASIC DEFINITIONS

;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV TOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

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

;*GENERAL PURPOSE REGISTER DEFINITIONS
R0= %0 ;;GENERAL REGISTER
R1= %1 ;;GENERAL REGISTER
R2= %2 ;;GENERAL REGISTER
R3= %3 ;;GENERAL REGISTER
R4= %4 ;;GENERAL REGISTER
R5= %5 ;;GENERAL REGISTER
R6= %6 ;;GENERAL REGISTER
R7= %7 ;;GENERAL REGISTER
SP= %6 ;;STACK POINTER
PC= %7 ;;PROGRAM COUNTER

;*PRIORITY LEVEL DEFINITIONS
PR0= 0 ;;PRIORITY LEVEL 0
PR1= 40 ;;PRIORITY LEVEL 1
PR2= 100 ;;PRIORITY LEVEL 2
PR3= 140 ;;PRIORITY LEVEL 3
PR4= 200 ;;PRIORITY LEVEL 4
PR5= 240 ;;PRIORITY LEVEL 5
PR6= 300 ;;PRIORITY LEVEL 6
PR7= 340 ;;PRIORITY LEVEL 7

;*SWITCH REGISTER SWITCH DEFINITIONS
SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
```

1037 004000  
1038 002000  
1039 001000  
1040 000400  
1041 000200  
1042 000100  
1043 000040  
1044 000020  
1045 000010  
1046 000004  
1047 000002  
1048 000001

SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

1060  
1061 100000  
1062 040000  
1063 020000  
1064 010000  
1065 004000  
1066 002000  
1067 001000  
1068 000400  
1069 000200  
1070 000100  
1071 000040  
1072 000020  
1073 000010  
1074 000004  
1075 000002  
1076 000001

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)  
BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0

1088  
1089 000004  
1090 000010  
1091 000014  
1092 000014

.\*BASIC "CPU" TRAP VECTOR ADDRESSES  
ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS  
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS  
TRITVEC= 14 ;: "T" BIT  
TRTVEC= 14 ;: TRACE TRAP

1093 000014  
1094 000020  
1095 000024  
1096 000030  
1097 000034  
1098 000060  
1099 000064  
1100 000240  
1101 176700  
1102 000254  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118 000000  
1119  
1120  
1121  
1122 000174  
1123 000174 000000  
1124 000176 000000  
1125  
1126  
1127  
1128  
1129  
1130 000200  
1131 000046  
1132 000046 015742  
1133 000052 000052  
1134 000052 020000  
1135 000200 000200  
1136 000200 000200  
1137 000200 000137 007014  
1138 000204 000137 006776  
1139  
1140 001100  
1141  
1142  
1143  
1144  
1145  
1146 001100  
1147 000024  
1148 000024 000200

BPTVEC= 14 ;; BREAKPOINT TRAP (BPT)  
IOTVEC= 20 ;; INPUT/OUTPUT TRAP (IOT) \*\*SCOPE\*\*  
PWRVEC= 24 ;; POWER FAIL  
EMTVEC= 30 ;; EMULATOR TRAP (EMT) \*\*ERROR\*\*  
TRAPVEC= 34 ;; "TRAP" TRAP  
TKVEC= 60 ;; TTY KEYBOARD VECTOR  
TPVEC= 64 ;; TTY PRINTER VECTOR  
PIRQVEC= 240 ;; PROGRAM INTERRUPT REQUEST VECTOR  
ABASE= 176700  
AVECT1= 254

.SBTTL OPERATIONAL SWITCH SETTINGS

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

.SBTTL TRAP CATCHER

. = 0  
;\* ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"  
;\* SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS  
;\* LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS  
. = 174  
DISPREG: .WORD 0 ;; SOFTWARE DISPLAY REGISTER  
SWREG: .WORD 0 ;; SOFTWARE SWITCH REGISTER

.SBTTL ACT11 HOOKS

```
;; *****
;HOOKS REQUIRED BY ACT11
$SVPC= . ;SAVE PC
. = 46 $ENDAD ;; 1) SET LOC. 46 TO ADDRESS OF $ENDAD IN .SEOP
. = 52 .WORD 20000 ;; 2) SET LOC. 52 TO 20000
. = $SVPC ;; RESTORE PC
. = 200 JMP BEGIN1 ;NORMAT STARTING ADDRESS
JMP BEGIN ;CHANGE THE RH11 ADDRESS
```

. = 1100  
.SBTTL APT PARAMETER BLOCK

```
;; *****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
.$X= . ;;SAVE CURRENT LOCATION
. = 24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200 ;;FOR APT START UP
```

1149 000044 000044  
1150 000044 001100  
1151 001100  
1152  
1153  
1154  
1155  
1156 001100  
1157 001100 000000  
1158 001102 001206  
1159 001104 000310  
1160 001106 000310  
1161 001110 000310  
1162 001112 000032  
1163 001114  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173 000100  
1174 000200  
1175 000400  
1176 001000  
1177 002000  
1178 020000  
1179 040000  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190 000001  
1191 000002  
1192 000004  
1193 000010  
1194 000020  
1195 000040  
1196 000100  
1197 000200  
1198 000400  
1199 001000  
1200 002000  
1201 004000  
1202 010000  
1203 020000  
1204 040000

```

      =44      ;; POINT TO APT INDIRECT ADDRESS PNTR.
      $APTHDR ;; POINT TO APT HEADER BLOCK
      =.$X    ;; RESET LOCATION COUNTER
;*****
; SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
; INTERFACE SPEC.

$APTHD:
$SHIBTS: .WORD 0      ;; TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR:  .WORD $MAIL  ;; ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT:   .WORD 200.   ;; RUN TIM OF LONGEST TEST
$PASTM:  .WORD 200.   ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$SUNITM: .WORD 200.   ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
          .WORD SETEND-$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)
TAB.XY = .           ; CMTAGSTARTING ADDRESS

;*****
.SBTTL RH11 REGISTERS

;*****
; CONTROL AND STATUS REGISTER 1 (RMCS1)
IE= 100      ; INTERRUPT ENABLE (BIT #6)
RDY= 200     ; READY (BIT #7)
A16= 400     ; HIGH ORDER BUS ADDRESS BIT (BIT #8)
A17= 1000    ; HIGH ORDER BUS ADDRESS BIT (BIT #9)
PSEL= 2000   ; PORT SELECT (BIT #10)
MCPE= 20000  ; MASSBUS PARITY ERROR (BIT #13)
TRE= 40000   ; TRANSFER ERROR (BIT #14)
;SC= 100000 ; SPECIAL CONDITION (BIT #15)

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

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

; CONTROL AND STATUS REGISTER 2 (RMCS2)
US1= 1      ; UNIT SELECT (BIT #0)
US2= 2      ; UNIT SELECT (BIT #1)
US4= 4      ; UNIT SELECT (BIT #2)
BAI= 10     ; BUS ADDRESS INCREMENT INHIBIT (BIT #3)
PAT= 20     ; MASSBUS PARITY TEST (BIT #4)
CLR= 40     ; CLEAR (BIT #5)
IR= 100    ; INPUT READY (BIT #6)
OR= 200    ; OUTPUT READY (BIT #7)
MPE= 400   ; MASS BUS PARITY ERROR (BIT #8)
MXF= 1000  ; MISSED TRANSFER ERROR (BIT #9)
PGE= 2000  ; PROGRAM ERROR (BIT #10)
NEM= 4000  ; NON EXISTENT MEMORY (BIT #11)
NED= 10000 ; NON EXISTENT DRIVE (BIT #12)
UPE= 20000 ; UNIBUS PARITY ERROR (BIT #13)
WCE= 40000 ; WRITE CHECK ERROR (BIT #14)

```



1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260

100000  
  
  
  
  
  
  
  
  
000001  
000002  
000004  
000010  
000020  
000040  
004000  
  
  
000100  
000200  
000400  
001000  
002000  
004000  
010000  
020000  
040000  
100000  
  
  
000001  
000002  
000004  
000010  
000020  
000040  
000100  
000200  
000400  
001000  
002000  
004000  
010000  
020000  
040000  
100000

DLT= 100000 ;DATA LATE (BIT #15)  
;DATA BUFFER REGISTER (RMDB)  
;(EACH BIT IS CALLED BY BIT NUMBER)  
  
;\*\*\*\*\*  
.SBTTL RMO3 REGISTERS  
;\*\*\*\*\*  
;CONTROL AND STATUS 1 REGISTER. (#00)  
GO= 1 ;GO BIT (BIT #0)  
F1= 2 ;FUNCTION CODE BIT #1  
F2= 4 ;FUNCTION CODE BIT #2  
F3= 10 ;FUNCTION CODE BIT #3  
F4= 20 ;FUNCTION CODE BIT #4  
F5= 40 ;FUNCTION CODE BIT #5  
DVA= 4000 ;DEVICE AVAILABLE (BIT #11)  
  
;DRIVE STATUS REGISTER (RMDS1) (#01)  
VV= 100 ;VOLUME VALID (BIT #6)  
DRY= 200 ;DRIVE READY (BIT #7)  
DPR= 400 ;DRIVE PRESENT (BIT #8)  
PGM= 1000 ;PROGRAMABLE (BIT #9)  
LST= 2000 ;LAST SECTOR TRANSFERRED (BIT #10)  
WRL= 4000 ;WRITE LOCK (BIT #11)  
MOL= 10000 ;MEDIUM ON-LINE (BIT #12)  
PIP= 20000 ;POSITIONING OPERATION IN PROGRESS (BIT #13)  
ERR= 40000 ;COMPOSITE ERROR (BIT #14)  
ATA= 100000 ;ATTENTION ACTIVE (BIT #15)  
  
;ERROR REGISTER #01 (RMER1) (#02)  
ILF= 1 ;ILLEGAL FUNCTION (BIT #0)  
ILR= 2 ;ILLEGAL REGISTER (BIT #1)  
RMR= 4 ;REGISTER MODIFICATION REFUSED (BIT #2)  
PAR= 10 ;PARITY ERROR (BIT #3)  
FER= 20 ;FORMAT ERROR (BIT #4)  
WCF= 40 ;WRITE CLOCK FAIL (BIT #5)  
ECH= 100 ;ECC HARD ERROR (BIT #6)  
HCE= 200 ;HEADER COMPARE ERROR (BIT #7)  
HCRC= 400 ;HEADER CRC ERROR (BIT #8)  
AOE= 1000 ;ADDRESS OVERFLOW ERROR (BIT #9)  
IAE= 2000 ;INVALID ADDRESS ERROR (BIT #10)  
WLE= 4000 ;WRITE LOCK ERROR (BIT #11)  
DTE= 10000 ;DRIVE TIMING ERROR (BIT #12)  
OPI= 20000 ;OPERATION INCOMPLETE (BIT #13)  
UNS= 40000 ;DRIVE UNSAFE (BIT #14)  
DCK= 100000 ;DATA CHECK ERROR (BIT 15)  
  
;MAINTAINABILITY REGISTER (RMMR1) (#03)

## RMO3 REGISTERS

1261	000001	DMD=	1	;	DIAGINOSTIC MODE (BIT #0)
1262	000002	MCLK=	2	;	MAINTAINABILITY CLOCK (BIT #1)
1263	000004	MINX=	4	;	MAINTAINABILITY INDEX (BIT #2)
1264	000010	MSTCK=	10	;	MAINTAINABILITY SECTOR CLOCK (BIT #3)
1265	000020	MRD=	20	;	MAINTAINABILITY READ (BIT #4)
1266	000040	MWR=	40	;	MAINTAINABILITY WRITE (BIT #5)
1267	000200	DTSY=	200	;	MAINTAINABILITY SYNC DETECTED (BIT #7)
1268					
1269					
1270					
1271	000001	AT0=	1	;	DEVICE 0 (BIT #0)
1272	000002	AT1=	2	;	DEVICE 1 (BIT #1)
1273	000004	AT2=	4	;	DEVICE 2 (BIT #2)
1274	000010	AT3=	10	;	DEVICE 3 (BIT #3)
1275	000020	AT4=	20	;	DEVICE 4 (BIT #4)
1276	000040	AT5=	40	;	DEVICE 5 (BIT #5)
1277	000100	AT6=	100	;	DEVICE 6 (BIT #6)
1278	000200	AT7=	200	;	DEVICE 7 (BIT #7)
1279					
1280					
1281					
1282					
1283					
1284					
1285	000001	DT00=	1	;	DRIVE TYPE NUMBER BIT 1
1286	000002	DT01=	2	;	DRIVE TYPE NUMBER BIT 2
1287	000004	DT02=	4	;	DRIVE TYPE NUMBER BIT 3
1288	000010	DT03=	10	;	DRIVE TYPE NUMBER BIT 4
1289	000020	DT04=	20	;	DRIVE TYPE NUMBER BIT 5
1290	000040	DT05=	40	;	DRIVE TYPE NUMBER BIT 6
1291	000100	DT06=	100	;	DRIVE TYPE NUMBER BIT 7
1292	000200	DT07=	200	;	DRIVE TYPE NUMBER BIT 8
1293	000400	DT08=	400	;	DRIVE TYPE NUMBER BIT 9
1294	004000	DRQ=	4000	;	DRIVE REQUEST REQUIRED (BIT #11)
1295	020000	MOH=	20000	;	MOVING HEAD (BIT #13)
1296	040000	TAP=	40000	;	TAPE DRIVE (BIT #14)
1297	100000	NBA=	100000	;	NOT BLOCK ADDRESSED (BIT #15)
1298					
1299					
1300					
1301	000100	SC01=	100	;	SECTOR COUNT FIELD 0 (BIT #6)
1302	000200	SC02=	200	;	SECTOR COUNT FIELD 1 (BIT #7)
1303	000400	SC04=	400	;	SECTOR COUNT FIELD 2 (BIT #8)
1304	001000	SC10=	1000	;	SECTOR COUNT FIELD 3 (BIT #9)
1305	002000	SC20=	2000	;	SECTOR COUNT FIELD 4 (BIT #10)
1306	004000	TRK1=	4000	;	TRACK FIELD 1 (BIT #11)
1307	010000	TRK2=	10000	;	TRACK FIELD 2 (BIT #12)
1308	020000	TRK4=	20000	;	TRACK FIELD 3 (BIT #13)
1309	040000	TRK10=	40000	;	TRACK FIELD 4 (BIT #14)
1310	100000	TRK20=	100000	;	TRACK FIELD 5 (BIT #15)
1311					
1312					
1313					
1314	000001	OFDIR=	1	;	OFFSET DIRECTION FOR RMO3
1315	002000	HCI=	2000	;	HEADER COMPARE INHIBIT (BIT #10)
1316	004000	ECI=	4000	;	ERROR CORRECTION CODE INHIBIT (BIT #11)

```

1317      010000      FMT16= 10000      ;FORMAT BIT (BIT #12)
1318
1319      ;DESIRED CYLINDER ADDRESS (RMDC) (#12)
1320      ;(EACH BIT IS CALLED BY BIT NUMBER)
1321
1322      ;SERIAL NUMBER REGISTER (RMSN) (#14)
1323      ;(EACH IS CALLED BY BIT NUMBER)
1324
1325      ;RMO3 MAINTENANCE REGISTER #02 (RMMR2) (#15)
1326
1327      040000      SKI= 40000      ;SEEK INCOMPLETE (BIT #14)
1328
1329      ;ECC POSITION REGISTER (RMEC1) (#16)
1330      ;(EACH BIT IS CALLED BY BIT NUMBER)
1331
1332      ;ECC PATTERN REGISTER (RMEC2) (#17)
1333      ;(EACH BIT IS CALLED BY BIT NUMBER)
1334
1335      ;;*****
1336
1337      .SBTTL RMO3 DRIVER COMMANDS
1338
1339      ;;*****
1340
1341      000101      RNOP = 101      ;NO OPERATION
1342      000105      SEEK = 105      ;SEEK
1343      000107      RECAL = 107      ;RECALIBRATE
1344      000111      DRVCLR = 111      ;DRIVE CLEAR
1345      000113      RELSE = 113      ;RELEASE
1346      000115      OFFSET = 115      ;OFFSET
1347      000117      RTC = 117      ;RETURN TO CENTER LINE
1348      000121      READIN = 121      ;READ IN PRESET
1349      000123      ACK = 123      ;PACK ACKNOWLEDGE
1350      000131      SEARCH = 131      ;SEARCH
1351      000141      GETREG = 141      ;GET REGISTERS
1352      000143      SETFMT = 143      ;SET FORMAT (& ECI OR HCI)
1353      000145      SELDRV = 145      ;SELECT DRIVE
1354      000151      WCKD = 151      ;WRITE CHECK DATA
1355      000153      WCKHD = 153      ;WRITE CHECK HEADER & DATA
1356      000161      WRTDAT = 161      ;WRITE DATA
1357      000163      WRTHD = 163      ;WRITE HEADER & DATA
1358      000171      RDDAT = 171      ;READ DATA
1359      000173      RDHD = 173      ;READ HEADER & DATA
1360
1361

```

1362  
1363  
1364  
1365  
1366  
1367  
1368 001114  
1369 001114 000000  
1370 001114 000000  
1371 001116 000  
1372 001117 000  
1373 001120 000000  
1374 001122 000000  
1375 001124 000000  
1376 001126 000000  
1377 001130 000  
1378 001131 001  
1379 001132 000000  
1380 001134 000000  
1381 001136 000000  
1382 001140 000000  
1383 001142 000000  
1384 001144 000000  
1385 001146 000000  
1386 001150 000  
1387 001151 000  
1388 001152 000000  
1389 001154 177570  
1390 001156 177570  
1391 001160 177560  
1392 001162 177562  
1393 001164 177564  
1394 001166 177566  
1395 001170 000  
1396 001171 002  
1397 001172 012  
1398 001173 000  
1399 001174 000000  
1400 001176 177607 000377  
1401 001202 077  
1402 001203 015  
1403 001204 000012  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413 001206  
1414 001206 000030  
1415 001210 000000  
1416 001212 000000  
1417 001214 000000

```

.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
      . =TAB.XY
SCMTAG:      .WORD      0      ;; START OF COMMON TAGS
      .WORD      0
$TSTNM:      .BYTE      0      ;; CONTAINS THE TEST NUMBER
$ERFLG:      .BYTE      0      ;; CONTAINS ERROR FLAG
$ICNT:       .WORD      0      ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR:      .WORD      0      ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR:      .WORD      0      ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL:      .WORD      0      ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB:      .BYTE      0      ;; CONTAINS ITEM CONTROL BYTE
$ERMAX:      .BYTE      1      ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC:      .WORD      0      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR:      .WORD      0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR:      .WORD      0      ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDADR:      .WORD      0      ;; CONTAINS 'GOOD' DATA
$BDADR:      .WORD      0      ;; CONTAINS 'BAD' DATA
      .WORD      0      ;; RESERVED--NOT TO BE USED
$AUTOB:      .BYTE      0      ;; AUTOMATIC MODE INDICATOR
$INTAG:      .BYTE      0      ;; INTERRUPT MODE INDICATOR
      .WORD      0
$SWR:        .WORD      DSWR    ;; ADDRESS OF SWITCH REGISTER
$DISP:       .WORD      DDISP   ;; ADDRESS OF DISPLAY REGISTER
$TKS:        177560            ;; TTY KBD STATUS
$TKB:        177562            ;; TTY KBD BUFFER
$TPS:        177564            ;; TTY PRINTER STATUS REG. ADDRESS
$TPB:        177566            ;; TTY PRINTER BUFFER REG. ADDRESS
$NULL:       .BYTE      0      ;; CONTAINS NULL CHARACTER FOR FILLS
$FILLS:      .BYTE      2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
$FILLC:      .BYTE      12     ;; INSERT FILL CHARS. AFTER A "LINE FEED"
$STPFLG:     .BYTE      0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$ESCAPE:     0                ;; ESCAPE ON ERROR ADDRESS
$BELL:       .ASCIZ <207><377><377> ;; CODE FOR BELL
$QUES:       .ASCII  /?/      ;; QUESTION MARK
$CRLF:       .ASCII  <15>     ;; CARRIAGE RETURN
$LF:         .ASCIZ  <12>     ;; LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.NLIST ME
;
;
.EVEN
$MAIL:       .WORD      AMSGT   ;; APT MAILBOX
$MSGTY:      .WORD      AFATAL  ;; MESSAGE TYPE CODE
$FATAL:      .WORD      ATESTN  ;; FATAL ERROR NUMBER
$TESTN:      .WORD      APASS   ;; TEST NUMBER
$PASS:       .WORD      APASS   ;; PASS COUNT

```

1418	001216	000000	\$DEVCT:	.WORD	ADEVCT	::	DEVICE COUNT
1419	001220	000000	\$UNIT:	.WORD	AUNIT	::	I/O UNIT NUMBER
1420	001222	000000	\$MSGAD:	.WORD	AMSGAD	::	MESSAGE ADDRESS
1421	001224	000000	\$MSGLG:	.WORD	AMSLG	::	MESSAGE LENGTH
1422	001226		\$ETABLE:			::	APT ENVIRONMENT TABLE
1423	001226	000	\$ENV:	.BYTE	AENV	::	ENVIRONMENT BYTE
1424	001227	000	\$ENVM:	.BYTE	AENVM	::	ENVIRONMENT MODE BITS
1425	001230	000000	\$SWREG:	.WORD	ASWREG	::	APT SWITCH REGISTER
1426	001232	000000	\$USWR:	.WORD	AUSWR	::	USER SWITCHES
1427	001234	000000	\$CPUOP:	.WORD	ACPUOP	::	CPU TYPE, OPTIONS
1428			*			::	BITS 15-11=CPU TYPE
1429			*			::	11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
1430			*			::	11/70=06, PDQ=07, Q=10
1431			*			::	BIT 10=REAL TIME CLOCK
1432			*			::	BIT 9=FLOATING POINT PROCESSOR
1433			*			::	BIT 8=MEMORY MANAGEMENT
1434	001236	000	\$MAMS1:	.BYTE	AMAMS1	::	HIGH ADDRESS, M.S. BYTE
1435	001237	000	\$MTYP1:	.BYTE	AMTYP1	::	MEM. TYPE, BLK#1
1436			*			::	MEM. TYPE BYTE -- (HIGH BYTE)
1437			*			::	900 NSEC CORE=001
1438			*			::	300 NSEC BIPOLAR=002
1439			*			::	500 NSEC MOS=003
1440	001240	000000	\$MADR1:	.WORD	AMADR1	::	HIGH ADDRESS, BLK#1
1441			*			::	MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
1442	001242	000	\$MAMS2:	.BYTE	AMAMS2	::	HIGH ADDRESS, M.S. BYTE
1443	001243	000	\$MTYP2:	.BYTE	AMTYP2	::	MEM. TYPE, BLK#2
1444	001244	000000	\$MADR2:	.WORD	AMADR2	::	MEM. LAST ADDRESS, BLK#2
1445	001246	000	\$MAMS3:	.BYTE	AMAMS3	::	HIGH ADDRESS, M.S. BYTE
1446	001247	000	\$MTYP3:	.BYTE	AMTYP3	::	MEM. TYPE, BLK#3
1447	001250	000000	\$MADR3:	.WORD	AMADR3	::	MEM. LAST ADDRESS, BLK#3
1448	001252	000	\$MAMS4:	.BYTE	AMAMS4	::	HIGH ADDRESS, M.S. BYTE
1449	001253	000	\$MTYP4:	.BYTE	AMTYP4	::	MEM. TYPE, BLK#4
1450	001254	000000	\$MADR4:	.WORD	AMADR4	::	MEM. LAST ADDRESS, BLK#4
1451	001256	000254	\$VECT1:	.WORD	AVECT1	::	INTERRUPT VECTOR#1, BUS PRIORITY#1
1452	001260	000000	\$VECT2:	.WORD	AVECT2	::	INTERRUPT VECTOR#2, BUS PRIORITY#2
1453	001262	176700	\$BASE:	.WORD	ABASE	::	BASE ADDRESS OF EQUIPMENT UNDER TEST
1454	001264	000000	\$DEVN:	.WORD	ADEVN	::	DEVICE MAP
1455	001266	000000	\$CDW1:	.WORD	ACDW1	::	CONTROLLER DESCRIPTION WORD#1
1456	001270	000000	\$CDW2:	.WORD	ACDW2	::	CONTROLLER DESCRIPTION WORD#2
1457	001272		\$ETEND:			::	
1458			.MEXIT			::	
1459		000015	CR	=	15		
1460		000012	LF	=	12		
1461	001272	176700	\$RMADR:	.WORD	176700		:RH11/RM03 UNIBUS ADDRESS
1462	001274	000254	\$RMVEC:	.WORD	254		:RH11 INTERRUPT VECTOR
1463	001276	172540	\$LKCSR:	.WORD	172540		:ADDRESS OF KW11-P CSR
1464	001300	172542	\$LKCSB:	.WORD	172542		:ADDRESS OF KW11-P COUNTER BUFFER
1465	001302	000104	\$LPVEC:	.WORD	104,106		:ADDRESS OF KW11-P VECTOR
1466	001306	177546	\$LKS:	.WORD	177546		:ADDRESS OF KW11-L CONTROL REGISTER
1467	001310	000100	\$LLVEC:	.WORD	100,102		:ADDRESS OF KW11-L VECTOR
1468		001220	DRIVE	=\$UNIT			:CONTAINS DRIVE NUMBER SELECTED
1469							:SAME PARAMETER USED IN APT
1470	001314	000000	\$OFSW:	.WORD	0		:CONTENTS ARE FOR SOFTWARE DECISIONS
1471	001316	000000	MODE:	.WORD	0		: 'FORMAT & VERIFY' OR 'CHECK' MODE INDICATOR
1472	001320	001466	ENDCYL:	.WORD	822.		:ENDING CYLINDER
1473	001322	000000	BEGCYL:	.WORD	0		:STARTING CYLINDER

1474	001324	000004	ENDTRK: .WORD	4.	:ENDING TRACK
1475	001326	000000	BEGTRK: .WORD	0	:STARTING TRACK
1476	001330	000000	TTRKS: .WORD	0	:TOTAL # OF TRACKS TO BE FORMATTED
1477	001332	000000	TTRKSC: .WORD	0	:TOTAL # OF TRACKS COUNTER
1478	001334	000000	TRKCNT: .WORD	0	:COUNTS TRKS FROM 0-5 PER CYL
1479	001336	000000	PATSEL: .WORD	0	:CONTAINS PATTERN SELECTED
1480	001340	000000	PATA: .WORD	0	:1ST WORD OF PATTERN
1481	001342	000000	PATB: .WORD	0	:2ND WORD OF PATTERN
1482	001344	000000	CYLCK: .WORD	0	:STORE CYLINDER ADDRESS FOR FORMAT VERIFICATION
1483	001346	000000	RETRY: .WORD	0	:MAINTAINS # OF WRITE RETRIES MADE
1484	001350	000000	SAVSEC: .WORD	0	:CONTAINS LAST BAD SECTOR ON FORMAT
1485	001352	000000	SAVWC: .WORD	0	:CONTAINS WC FOR REMAINING SECTORS ON ERROR
1486	001354	000000	WC: .WORD	0	:30 OR 32 SECTOR TRACK SIZE (IN WORDS)
1487	001356	000000	MWC: .WORD	0	:2'S COMPLEMENT OF 'WC'
1488	001360	000000	HEDERR: .WORD	0	:POSITIONING ERROR DURING FORMAT INDICATOR
1489	001362	000000	SEC30: .WORD	0	:30 OR 32 SECTOR MODE INDICATOR
1490					:0 = 30 SECTOR MODE
1491					:1'S = 32 SECTOR MODE
1492	001364	000000	MAXSEC: .WORD	0	:MAXIMUM SECTOR ADDRESS (FOR EITHER 30 OR 32 SECTOR
1493					:FORMAT)
1494	001366	000000	DS.CYL: .WORD	0	:ADDRESS OF CURRENT CYLINDER
1495	001370	000000	DS.TRK: .WORD	0	:ADDRESS OF CURRENT TRACK
1496	001372	000000	DDRIVE: .WORD	0	:DRIVE NUMBER OF 'DRIVER' ERROR MESSAGES
1497	001374	000000	ATTN: .WORD	0	:ATTENTION REGISTER IMAGE FOR 'DRIVER'
1498					:ERROR MESSAGES
1499	001376	000000	CNTLC: .WORD	0	:ADDRESS OF 'IC' RETURN
1500	001400	000000	CHGADR: .WORD	0	: 'CHANGE RH11 ADDRESS' FLAG
1501	001402	000000	WRAP: .WORD	0	:;WRAP AROUND FLAG,FORMAT FROM HIGH TO LOW
1502					:CYLINDER NUMBER
1503	001404	000000	DEVMP: .WORD	0	:DEVICE MAP FOR APT
1504	001406	000000	PACK: .WORD	0	:;PACK>0, BAD SPOT FILE AVAILABLE FROM PACK
1505	001410	000000	EDIT: .WORD	0	:EDIT THE BAD SPOT FILE ,IF EDIT>0
1506	001412	000000	HEADX: .WORD	0	:HEAD BIT INDICATOR
1507					:BIT15:BIT14 GOOD SECTOR
1508					:BIT15 ALONE, USER DEFINED BAD SECTOR
1509					:BIT14 ALONE, MFG DEFINED BAD SECTOR
1510	001414	000400	BAD16: .BLKW	256.	:MFG 16 BIT BAD SPOT TABLE
1511	002414	177777	.WORD	-1	
1512	002416	177777	.WORD	-1	
1513	002420	000400	BAD18: .BLKW	256.	:MFG 18 BIT BAD SPOT TABLE
1514	003420	177777	.WORD	-1	
1515	003422	177777	.WORD	-1	
1516	003424	000400	USTAB: .BLKW	256.	:USER BAD SPOT TABLE
1517	004424	177777	.WORD	-1	
1518	004426	177777	.WORD	-1	
1519	004430	000400	USSAV: .BLKW	256.	:USER BAD SPOT TABLE SAVE FROM PACK
1520	005430	177777	.WORD	-1	
1521	005432	177777	.WORD	-1	
1522	005434	000040	SECCU: .BLKW	32.	:;32 WORDS MAX TO STORE BAD SECTORS
1523	005534	177777	.WORD	-1	
1524	005536	000000	NEXT: .WORD	0	:FLAG, IF NEXT=-1,ALL GOOD SPOT ON
1525					:THE CURRENT TRACK ARE FORMATTED
1526					
1527					:;RH11/RM03 REGISTERS STORED HERE AFTER AN OPERATION
1528					
1529	005540	000000	RM.REG: .WORD	0	:;RMCS1

1530	005542	000000	.WORD	0	:RMWC
1531	005544	000000	.WORD	0	:RMBA
1532	005546	000000	.WORD	0	:RMDA
1533	005550	000000	.WORD	0	:RMCS2
1534	005552	000000	.WORD	0	:RMDS
1535	005554	000000	.WORD	0	:RMER1
1536	005556	000000	.WORD	0	:RMAS
1537	005560	000000	.WORD	0	:RMLA
1538	005562	000000	.WORD	0	:RMDB
1539	005564	000000	.WORD	0	:RMMR1
1540	005566	000000	.WORD	0	:RMDT
1541	005570	000000	.WORD	0	:RMSN
1542	005572	000000	.WORD	0	:RMOF
1543	005574	000000	.WORD	0	:RMCA
1544	005576	000000	.WORD	0	:RMDC
1545	005600	000000	.WORD	0	:RMER2
1546	005602	000000	.WORD	0	:RMMR2
1547	005604	000000	.WORD	0	:RMEC1
1548	005606	000000	.WORD	0	:RMEC2

;DATA/PARAMETER BLOCK - USED FOR ALL DRIVE OPERATION

1551					
1552					
1553	005610	000	FMTDPB: .BYTE	0	:DRIVE NUMBER
1554	005611	000	.BYTE	0	:OFFSET VALUE OR FMT22,ECI, AND HCI
1555	005612	000	.BYTE	0	:COMMAND
1556	005613	000	.BYTE	0	:PSEL AND A17 AND A16
1557	005614	000000	.WORD	0	:WORD COUNT (NEG)
1558	005616	000000	.WORD	0	:BUFFER ADDRESS
1559	005620	000	.BYTE	0	:SECTOR ADDRESS
1560	005621	000	.BYTE	0	:TRACK ADDRESS
1561	005622	000000	.WORD	0	:CYLINDER ADDRESS
1562	005624	005540	.WORD	RM.REG	:ERROR TABLE POINTER
1563	005626	000000	.WORD	0	:STATUS-ERROR INDICATOR
1564					:BIT 15 = 1: ERROR OCCURRED
1565					:BIT 07 = 1: DONE
1566					:BIT 14-10 AND BIT 06-03
1567					:INDICATE TYPE OF ERROR

1568					
1569					
1570	005630	000	FMTX: .BYTE	0	:SPECIAL DPB TO WRITE HEADER OF BAD SPOT
1571	005631	000	.BYTE	0	
1572	005632	000	.BYTE	0	
1573	005633	000	.BYTE	0	
1574	005634	177776	.WORD	-2	:ONLY TWO WORDS FOR HEADER
1575	005636	043736	.WORD	RBUF	:BUFFER ADDRESS
1576	005640	000	.BYTE	0	:SECTOR
1577	005641	000	.BYTE	0	:TRACK
1578	005642	000000	.WORD	0	:CYLINDER ADDRESS
1579	005644	000000	.WORD	0	:ERROR TABLE
1580					:DONT"SAVE ANYTHING
1581	005646	000000	.WORD	0	:STATUS WORD

;PARAMETER POINTER TABLE

1582					
1583					
1584	005650	005776	001467	001322	TABLE: PAR1,823,BEGCYL
1585	005656	006011	000005	001326	PAR2,5.,BEGTRK

```

1586 005664 006024 001467 001320      PAR3,823.,ENDCYL
1587 005672 006035 000005 001324      PAR4,5.,ENDTRK,0
1588 005700 000000
1589
1590 005702 006046 001467 001366 TABLE2: PAR5,823.,DS.CYL
1591 005710 006057 000005 001370      PAR6,5.,DS.TRK
1592 005716 006070 000040 001350      PAR7,32.,SAVSEC,0
1593 005724 000000
1594
1595 005726 006046 001467 001366 TABLE3: PAR5,823.,DS.CYL
1596 005734 006057 000005 001370      PAR6,5.,DS.TRK
1597 005742 006070 000036 001350      PAR7,30.,SAVSEC,0
1598 005750 000000
1599 005752 006046 001467 001366 TABLE4: PAR5,823.,DS.CYL
1600 005760 006057 000005 001370      PAR6,5,DS.TRK
1601 005766 006101 047301 001350      PAR8,20161.,SAVSEC,0
1602 005774 000000
1603
1604 ;ASCII MESSAGES FOR ADDRESS PARAMETERS
1605 005776 052123 051101 020124 PAR1: .ASCIZ @START CYL @
1606 006004 054503 020114 000      PAR2: .ASCIZ @START TRK @
1607 006011 123 040524 052122
1608 006016 052040 045522 000040 PAR3: .ASCIZ @END CYL @
1609 006024 047105 020104 054503
1610 006032 020114 000
1611 006035 105 042116 052040 PAR4: .ASCIZ @END TRK @
1612 006042 045522 000040
1613
1614 006046 054503 044514 042116 PAR5: .ASCIZ /CYLINDER/
1615 006054 051105 000
1616 006057 124 040522 045503 PAR6: .ASCIZ /TRACK /
1617 006064 020040 000040
1618 006070 042523 052103 051117 PAR7: .ASCIZ /SECTOR /
1619 006076 020040 000
1620 006101 102 052131 020105 PAR8: .ASCIZ /BYTE POSITION /
1621 006106 047520 044523 044524
1622 006114 047117 000040
1623
1624 006120 021044 021324 021412 .EVEN TABCAL: .WORD FUNCT2,FUNCT3,FUNCT4,FUNCT6,FUNCT7
1625 006126 021620 021716
1626
1627 ;SECTOR BUFFER ADDRESS TABLE
1628 ADRTBL: .WORD BUFP ;ADDRESS OF SECTOR 0
1629 006132 043746 .WORD BUFP+(516.*1.) ;ADDRESS OF SECTOR 1
1630 006134 044752 .WORD BUFP+(516.*2.) ;ADDRESS OF SECTOR 2
1631 006136 045756 .WORD BUFP+(516.*3.) ;ADDRESS OF SECTOR 3
1632 006140 046762 .WORD BUFP+(516.*4.) ;ADDRESS OF SECTOR 4
1633 006142 047766 .WORD BUFP+(516.*5.) ;ADDRESS OF SECTOR 5
1634 006144 050772 .WORD BUFP+(516.*6.) ;ADDRESS OF SECTOR 6
1635 006146 051776 .WORD BUFP+(516.*7.) ;ADDRESS OF SECTOR 7
1636 006150 053002 .WORD BUFP+(516.*8.) ;ADDRESS OF SECTOR 8
1637 006152 054006 .WORD BUFP+(516.*9.) ;ADDRESS OF SECTOR 9
1638 006154 055012 .WORD BUFP+(516.*10.) ;ADDRESS OF SECTOR 10
1639 006156 056016 .WORD BUFP+(516.*11.) ;ADDRESS OF SECTOR 11
1640 006160 057022 .WORD BUFP+(516.*12.) ;ADDRESS OF SECTOR 12
1641 006162 060026 .WORD BUFP+(516.*13.) ;ADDRESS OF SECTOR 13
1642 006164 061032

```



1642	006166	062036	.WORD	BUFP+(516.*14.)	ADDRESS OF SECTOR 14
1643	006170	063042	.WORD	BUFP+(516.*15.)	ADDRESS OF SECTOR 15
1644	006172	064046	.WORD	BUFP+(516.*16.)	ADDRESS OF SECTOR 16
1645	006174	065052	.WORD	BUFP+(516.*17.)	ADDRESS OF SECTOR 17
1646	006176	066056	.WORD	BUFP+(516.*18.)	ADDRESS OF SECTOR 18
1647	006200	067062	.WORD	BUFP+(516.*19.)	ADDRESS OF SECTOR 19
1648	006202	070066	.WORD	BUFP+(516.*20.)	ADDRESS OF SECTOR 20
1649	006204	071072	.WORD	BUFP+(516.*21.)	ADDRESS OF SECTOR 21
1650	006206	072076	.WORD	BUFP+(516.*22.)	ADDRESS OF SECTOR 22
1651	006210	073102	.WORD	BUFP+(516.*23.)	ADDRESS OF SECTOR 23
1652	006212	074106	.WORD	BUFP+(516.*24.)	ADDRESS OF SECTOR 24
1653	006214	075112	.WORD	BUFP+(516.*25.)	ADDRESS OF SECTOR 25
1654	006216	076116	.WORD	BUFP+(516.*26.)	ADDRESS OF SECTOR 26
1655	006220	077122	.WORD	BUFP+(516.*27.)	ADDRESS OF SECTOR 27
1656	006222	100126	.WORD	BUFP+(516.*28.)	ADDRESS OF SECTOR 28
1657	006224	101132	.WORD	BUFP+(516.*29.)	ADDRESS OF SECTOR 29
1658	006226	102136	.WORD	BUFP+(516.*30.)	ADDRESS OF SECTOR 30
1659	006230	103142	.WORD	BUFP+(516.*31.)	ADDRESS OF SECTOR 31

;REMAINING WORD COUNT TABLE

1663	006232	000402	WCTBL: .WORD	258.	REMAINING WORD COUNT AFTER SECTOR 0
1664	006234	001004	.WORD	258.+(258.*1.)	REMAINING WORD COUNT AFTER SECTOR 1
1665	006236	001406	.WORD	258.+(258.*2.)	REMAINING WORD COUNT AFTER SECTOR 2
1666	006240	002010	.WORD	258.+(258.*3.)	REMAINING WORD COUNT AFTER SECTOR 3
1667	006242	002412	.WORD	258.+(258.*4.)	REMAINING WORD COUNT AFTER SECTOR 4
1668	006244	003014	.WORD	258.+(258.*5.)	REMAINING WORD COUNT AFTER SECTOR 5
1669	006246	003416	.WORD	258.+(258.*6.)	REMAINING WORD COUNT AFTER SECTOR 6
1670	006250	004020	.WORD	258.+(258.*7.)	REMAINING WORD COUNT AFTER SECTOR 7
1671	006252	004422	.WORD	258.+(258.*8.)	REMAINING WORD COUNT AFTER SECTOR 8
1672	006254	005024	.WORD	258.+(258.*9.)	REMAINING WORD COUNT AFTER SECTOR 9
1673	006256	005426	.WORD	258.+(258.*10.)	REMAINING WORD COUNT AFTER SECTOR 10
1674	006260	006030	.WORD	258.+(258.*11.)	REMAINING WORD COUNT AFTER SECTOR 11
1675	006262	006432	.WORD	258.+(258.*12.)	REMAINING WORD COUNT AFTER SECTOR 12
1676	006264	007034	.WORD	258.+(258.*13.)	REMAINING WORD COUNT AFTER SECTOR 13
1677	006266	007436	.WORD	258.+(258.*14.)	REMAINING WORD COUNT AFTER SECTOR 14
1678	006270	010040	.WORD	258.+(258.*15.)	REMAINING WORD COUNT AFTER SECTOR 15
1679	006272	010442	.WORD	258.+(258.*16.)	REMAINING WORD COUNT AFTER SECTOR 16
1680	006274	011044	.WORD	258.+(258.*17.)	REMAINING WORD COUNT AFTER SECTOR 17
1681	006276	011446	.WORD	258.+(258.*18.)	REMAINING WORD COUNT AFTER SECTOR 18
1682	006300	012050	.WORD	258.+(258.*19.)	REMAINING WORD COUNT AFTER SECTOR 19
1683	006302	012452	.WORD	258.+(258.*20.)	REMAINING WORD COUNT AFTER SECTOR 20
1684	006304	013054	.WORD	258.+(258.*21.)	REMAINING WORD COUNT AFTER SECTOR 21
1685	006306	013456	.WORD	258.+(258.*22.)	REMAINING WORD COUNT AFTER SECTOR 22
1686	006310	014060	.WORD	258.+(258.*23.)	REMAINING WORD COUNT AFTER SECTOR 23
1687	006312	014462	.WORD	258.+(258.*24.)	REMAINING WORD COUNT AFTER SECTOR 24
1688	006314	015064	.WORD	258.+(258.*25.)	REMAINING WORD COUNT AFTER SECTOR 25
1689	006316	015466	.WORD	258.+(258.*26.)	REMAINING WORD COUNT AFTER SECTOR 26
1690	006320	016070	.WORD	258.+(258.*27.)	REMAINING WORD COUNT AFTER SECTOR 27
1691	006322	016472	.WORD	258.+(258.*28.)	REMAINING WORD COUNT AFTER SECTOR 28
1692	006324	017074	.WORD	258.+(258.*29.)	REMAINING WORD COUNT AFTER SECTOR 29
1693	006326	017476	.WORD	258.+(258.*30.)	REMAINING WORD COUNT AFTER SECTOR 30
1694	006330	020100	.WORD	258.+(258.*31.)	REMAINING WORD COUNT AFTER SECTOR 31
1695	006332	001166	BYTE16: .WORD	630.	
1696	006334	002354	.WORD	630.+(630.*1.)	
1697	006336	003542	.WORD	630.+(630.*2.)	

1698	006340	004730	.WORD	630.+<630.#3.>
1699	006342	006116	.WORD	630.+<630.#4.>
1700	006344	007304	.WORD	630.+<630.#5.>
1701	006346	010472	.WORD	630.+<630.#6.>
1702	006350	011660	.WORD	630.+<630.#7.>
1703	006352	013046	.WORD	630.+<630.#8.>
1704	006354	014234	.WORD	630.+<630.#9.>
1705	006356	015422	.WORD	630.+<630.#10.>
1706	006360	016610	.WORD	630.+<630.#11.>
1707	006362	017776	.WORD	630.+<630.#12.>
1708	006364	021164	.WORD	630.+<630.#13.>
1709	006366	022352	.WORD	630.+<630.#14.>
1710	006370	023540	.WORD	630.+<630.#15.>
1711	006372	024726	.WORD	630.+<630.#16.>
1712	006374	026114	.WORD	630.+<630.#17.>
1713	006376	027302	.WORD	630.+<630.#18.>
1714	006400	030470	.WORD	630.+<630.#19.>
1715	006402	031656	.WORD	630.+<630.#20.>
1716	006404	033044	.WORD	630.+<630.#21.>
1717	006406	034232	.WORD	630.+<630.#22.>
1718	006410	035420	.WORD	630.+<630.#23.>
1719	006412	036606	.WORD	630.+<630.#24.>
1720	006414	037774	.WORD	630.+<630.#25.>
1721	006416	041162	.WORD	630.+<630.#26.>
1722	006420	042350	.WORD	630.+<630.#27.>
1723	006422	043536	.WORD	630.+<630.#28.>
1724	006424	044724	.WORD	630.+<630.#29.>
1725	006426	046112	.WORD	630.+<630.#30.>
1726	006430	047300	.WORD	630.+<630.#31.>
1727	006432	001240	BYTE18: .WORD	672.
1728	006434	002500	.WORD	672.+<672.#1.>
1729	006436	003740	.WORD	672.+<672.#2.>
1730	006440	005200	.WORD	672.+<672.#3.>
1731	006442	006440	.WORD	672.+<672.#4.>
1732	006444	007700	.WORD	672.+<672.#5.>
1733	006446	011140	.WORD	672.+<672.#6.>
1734	006450	012400	.WORD	672.+<672.#7.>
1735	006452	013640	.WORD	672.+<672.#8.>
1736	006454	015100	.WORD	672.+<672.#9.>
1737	006456	016340	.WORD	672.+<672.#10.>
1738	006460	017600	.WORD	672.+<672.#11.>
1739	006462	021040	.WORD	672.+<672.#12.>
1740	006464	022300	.WORD	672.+<672.#13.>
1741	006466	023540	.WORD	672.+<672.#14.>
1742	006470	025000	.WORD	672.+<672.#15.>
1743	006472	026240	.WORD	672.+<672.#16.>
1744	006474	027500	.WORD	672.+<672.#17.>
1745	006476	030740	.WORD	672.+<672.#18.>
1746	006500	032200	.WORD	672.+<672.#19.>
1747	006502	033440	.WORD	672.+<672.#20.>
1748	006504	034700	.WORD	672.+<672.#21.>
1749	006506	036140	.WORD	672.+<672.#22.>
1750	006510	037400	.WORD	672.+<672.#23.>
1751	006512	040640	.WORD	672.+<672.#24.>
1752	006514	042100	.WORD	672.+<672.#25.>
1753	006516	043340	.WORD	672.+<672.#26.>

CZRMACO RMD3/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY:11 30(1046) 22-NOV-77 08:16 PAGE 35  
APT MAILBOX-ETABLE

SEG 0034

1754 006520 044600  
1755 006522 046040  
1756 006524 047300  
1757  
1758  
1759

.WORD 672.+<672.\*27.>  
.WORD 672.+<672.\*28.>  
.WORD 672.+<672.\*29.>

.EVEN

		.SBTTL ERROR POINTER TABLE	
1760		;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.	
1761		;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN	
1762		;*LOCATION \$ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.	
1763		;*NOTE1: IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).	
1764		;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:	
1765			
1766			
1767			
1768		;* EM	::POINTS TO THE ERROR MESSAGE
1769		;* DH	::POINTS TO THE DATA HEADER
1770		;* DT	::POINTS TO THE DATA
1771		;* DF	::POINTS TO THE DATA FORMAT
1772			
1773			
1774	006526	\$ERRTB:	
1775		;ERROR 1	
1776			
1777	006526	040474	EM1 ;RH11 INTERRUPT OCCURRED (RMAS=0)
1778	006530	041662	DH1
1779	006532	043256	DT1
1780	006534	043622	DF1
1781			
1782			
1783			
1784	006536	040535	EM2 ;UNEXPECTED ATTENTION OCCURRED
1785	006540	041730	DH2
1786	006542	043274	DT2
1787	006544	043626	DF2
1788			
1789			
1790			
1791	006546	040573	EM3 ;MASSBUS PARITY ERROR (MCPE=1)
1792	006550	042002	DH3
1793	006552	043310	DT3
1794	006554	043632	DF3
1795			
1796			
1797			
1798	006556	040631	EM4 ;MASSBUS PARITY ERROR (PAR=1)
1799	006560	042053	DH4
1800	006562	043324	DT4
1801	006564	043636	DF4
1802			
1803			
1804			
1805	006566	000000	0 ;UNUSED
1806	006570	000000	0
1807	006572	000000	0
1808	006574	000000	0
1809			
1810			
1811			
1812	006576	040722	EM6 ;RH11 DIDN'T RESPOND TO ADDRESSING
1813	006600	042136	DH6
1814	006602	043342	DT6
1815	006604	043642	DF6

1816				
1817			;ERROR 7	
1818				
1819	006606	000000	0	;UNUSED
1820	006610	000000	0	
1821	006612	000000	0	
1822	006614	000000	0	
1823				
1824			;ERROR 10	
1825				
1826	006616	040764	EM10	;DRIVE OFFLINE
1827	006620	042151	DH10	
1828	006622	043344	DT10	
1829	006624	043646	DF10	
1830				
1831			;ERROR 11	
1832				
1833	006626	041002	EM11	;PERSISTENT DRIVE UNSAFE ERROR
1834	006630	042151	DH10	
1835	006632	043344	DT10	
1836	006634	043646	DF10	
1837				
1838			;ERROR 12	
1839				
1840	006636	041040	EM12	;UNCORRECTABLE MASSBUS PARITY ERROR
1841	006640	042151	DH10	
1842	006642	043344	DT10	
1843	006644	043646	DF10	
1844				
1845			;ERROR 13	
1846				
1847	006646	041103	EM13	;SOFTWARE TIMEOUT
1848	006650	042151	DH10	
1849	006652	043344	DT10	
1850	006654	043646	DF10	
1851				
1852			;ERROR 14	
1853				
1854	006656	041124	EM14	;DRIVE UNSAFE ERROR
1855	006660	042151	DH10	
1856	006662	043344	DT10	
1857	006664	043646	DF10	
1858				
1859			;ERROR 15	
1860				
1861	006666	041147	EM15	;CONTROLLER/DRIVE ERROR DURING WRITE
1862	006670	042151	DH10	
1863	006672	043344	DT10	
1864	006674	043646	DF10	
1865				
1866			;ERROR 16	
1867				
1868	006676	041213	EM16	;CONTROLLER/DRIVE ERROR DURING WRITE CHECK
1869	006700	042151	DH10	
1870	006702	043344	DT10	
1871	006704	043646	DF10	

1872				
1873				
1874				
1875	006706	041265		
1876	006710	042400		
1877	006712	043416		
1878	006714	043662		
1879				
1880				
1881				
1882	006716	041343		
1883	006720	042447		
1884	006722	043430		
1885	006724	043666		
1886				
1887				
1888				
1889	006726	041401		
1890	006730	042151		
1891	006732	043344		
1892	006734	043646		
1893				
1894				
1895				
1896	006736	041452		
1897	006740	042151		
1898	006742	043344		
1899	006744	043646		
1900				
1901				
1902				
1903	006746	041521		
1904	006750	042744		
1905	006752	043512		
1906	006754	043706		
1907				
1908				
1909				
1910	006756	041571		
1911	006760	043042		
1912	006762	043524		
1913	006764	043712		
1914				
1915				
1916				
1917	006766	041613		
1918	006770	043215		
1919	006772	043612		
1920	006774	043732		
1921				
1922				
1923				
1924				
1925				
1926				
1927				

```
      ;ERROR 17
      EM17      ;RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE
      DH17
      DT17
      DF17

      ;ERROR 20
      EM20      ;DATA ERROR DURING WRITE CHECK
      DH20
      DT20
      DF20

      ;ERROR 21
      EM21      ;CONTROLLER/DRIVE ERROR VERIFYING HEADERS
      DH10
      DT10
      DF10

      ;ERROR 22
      EM22      ;'HCE' ERROR VERIFYING HEADERS
      DH10
      DT10
      DF10

      ;ERROR 23
      EM23      ;CYLINDER FIELD IN HEADER IS NOT CORRECT
      DH23
      DT23
      DF23

      ;ERROR 24
      EM24      ;WRITE CHECK ERROR
      DH24
      DT24
      DF24

      ;ERROR 25
      EM25      ;ILLEGAL BAD SECTOR
      DH25
      DT25
      DF25

      ;*****
      .SBTTL  MAIN PROGRAM
      ;*****
```

```

1928 006776 012737 177777 001400 BEGIN: MOV # -1, CHGADR ; SET CHANGE 'RH70 BUS ADDRESS' INDICATOR
1929 007004 012737 177777 001266 MOV # -1, SCDW1 ; FLAG TO USE UTILITY ROUTINES
1930 007012 000410 BR BEGIN2 ; START THE PROGRAM
1931 007014 005037 001400 BEGIN1: CLR CHGADR ; CLEAR THE 'CHANGE RH70 ADDRESS' INDICATOR
1932 007020 005037 001266 CLR SCDW1 ; REST UTILITY ROUTINE FLAG
1933 *****
1934 007024 000240 †ST1: NOP
1935 007026 012737 000001 001212 MOV #1, STESTN ; SET TEST NUMBER IN APT MAIL BOX
1936 007034 000005 BEGIN2: RESET ; CLEAR THE BUS
1937 .SBTTL INITIALIZE THE COMMON TAGS
1938 ;; CLEAR THE COMMON TAGS (SCMTAG) AREA
1939 007036 012706 001114 MOV #SCMTAG, R6 ; FIRST LOCATION TO BE CLEARED
1940 007042 005026 CLR (R6)+ ; CLEAR MEMORY LOCATION
1941 007044 022706 001154 CMP #SWR, R6 ;; DONE?
1942 007050 001374 BNE -6 ; LOOP BACK IF NO
1943 007052 012706 001100 MOV #STACK, SP ; SETUP THE STACK POINTER
1944 ;; INITIALIZE A FEW VECTORS
1945 007056 012737 022516 000030 MOV #ERROR, @EMTVEC ; EMT VECTOR FOR ERROR ROUTINE
1946 007064 012737 000340 000032 MOV #340, @EMTVEC+2 ; LEVEL 7
1947 007072 012737 026412 000034 MOV #TRAP, @TRAPVEC ; TRAP VECTOR FOR TRAP CALLS
1948 007100 012737 000340 000036 MOV #340, @TRAPVEC+2 ; LEVEL 7
1949 007106 012737 025654 000024 MOV #SPWRDN, @PWRVEC ; POWER FAILURE VECTOR
1950 007114 012737 000340 000026 MOV #340, @PWRVEC+2 ; LEVEL 7
1951 007122 005037 001174 CLR $ESCAPE ; CLEAR THE ESCAPE ON ERROR ADDRESS
1952 007126 112737 000001 001131 MOV #1, $ERMAX ; ALLOW ONE ERROR PER TEST
1953 ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1954 ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1955 007134 013746 000004 MOV @ERRVEC, -(SP) ; SAVE ERROR VECTOR
1956 007140 012737 007174 000004 MOV #64$, @ERRVEC ; SET UP ERROR VECTOR
1957 007146 012737 177570 001154 MOV #DSWR, SWR ; SETUP FOR A HARDWARE SWICH REGISTER
1958 007154 012737 177570 001156 MOV #DDISP, DISPLAY ; AND A HARDWARE DISPLAY REGISTER
1959 007162 022777 177777 171764 CMP # -1, @SWR ; TRY TO REFERENCE HARDWARE SWR
1960 007170 001012 BNE 66$ ; BRANCH IF NO TIMEOUT TRAP OCCURRED
1961 ; AND THE HARDWARE SWR IS NOT = -1
1962 007172 000403 BR 65$ ; BRANCH IF NO TIMEOUT
1963 007174 012716 007202 64$: MOV #65$, (SP) ; SET UP FOR TRAP RETURN
1964 007200 000002 RTI
1965 007202 012737 000176 001154 65$: MOV #SWREG, SWR ; POINT TO SOFTWARE SWR
1966 007210 012737 000174 001156 MOV #DISPREG, DISPLAY
1967 007216 012637 000004 66$: MOV (SP)+, @ERRVEC ; RESTORE ERROR VECTOR
1968
1969 007222 005037 001214 CLR $PASS ; CLEAR PASS COUNT
1970 007226 132737 000200 001227 BITB #APTSIZE, $ENVM ; TEST USER SIZE UNDER APT
1971 007234 001403 BEQ 67$ ; YES, USE NON-APT SWITCH
1972 007236 012737 001230 001154 MOV #SSWREG, SWR ; NO, USE APT SWITCH REGISTER
1973 007244 67$:
1974 007244 000005 RESET ; CLEAR WORLD
1975 007246 012737 000240 000036 MOV #PR5, @TRAPVEC+2 ; CHANGE TRAP PRIORITY BACK TO 5
1976 007254 012737 000240 000032 MOV #PR5, @EMTVEC+2 ; CHANGE EMT PRIORITY BACK TO 5
1977 007262 012737 007014 001376 MOV #BEGIN1, CNTLC ; 'CONTROL C' ADDRESS
1978 007270 005227 177777 INC # -1 ; FIRST START ?
1979 007274 001011 BNE 1$ ; BR IF NOT
1980 007276 104401 043746 TYPE 'TITLE' ; ADRS OF 'TITLE' MESSAGE
1981 007302 122737 000077 000041 CMPB #77, @41 ; LOAD FROM RM03/RM02 ?
1982 007310 001003 BNE 1$ ; BRANCH IF NOT
1983 007312 104401 044046 TYPE .LOADRV ; TYPE MESSAGE TO CHANGE PACK

```

1984	007316	000000				HALT			;; THEN WAIT FOR RESTART
1985	007320	004737	024422		1\$:	JSR	PC, \$TKINT		;; TURN ON THE KEYBOARD INTERRUPT
1986					.SBTTL	GET	VALUE FOR SOFTWARE SWITCH REGISTER		
1987	007324	005737	000042			TST	2#42		;; ARE WE RUNNING UNDER XXDP/ACT?
1988	007330	001012				BNE	68\$		;; BRANCH IF YES
1989	007332	123727	001226	000001		CMPB	\$ENV, #1		;; ARE WE RUNNING UNDER APT?
1990	007340	001406				BEQ	68\$		;; BRANCH IF YES
1991	007342	023727	001154	000176		CMP	\$WR, #SWREG		;; SOFTWARE SWITCH REG SELECTED?
1992	007350	001005				BNE	69\$		;; BRANCH IF NO
1993	007352	104406				GTSWR			;; GET SOFT-SWR SETTINGS
1994	007354	000403				BR	69\$		
1995	007356	112737	000001	001150	68\$:	MOVB	#1, \$AUTOB		;; SET AUTO-MODE INDICATOR
1996	007364				69\$:				
1997	007364	005227	177777			INC	#-1		;; CHECK FOR FIRST START
1998	007370	001035				BNE	SETVEC		;; BR IF NOT
1999	007372	004737	044264			JSR	PC, \$BUSADR		;; CHECK THE RH11 ADDRESS
2000	007376	013737	001272	026644		MOV	\$RMADR, \$RMADR		;; RH11 ADDRESS
2001	007404	013737	001274	026646		MOV	\$RMVEC, \$RMVEC		;; RH11 VECTOR ADDRESS
2002	007412	105737	001226			TSTB	\$ENV		;; RUN UNDER APT MODE?
2003	007416	001422				BEQ	SETVEC		;; NO, DONOT LOAD FROM E-TABLE
2004	007420	105737	001256			TSTB	\$VECT1		;; UUT INTERRUPT VECTOR
2005	007424	001403				BEQ	#+10		;; NOT CHANGE IF = 0
2006	007426	113737	001256	001274		MOVB	\$VECT1, \$RMVEC		;; NEWVALUE
2007	007434	005737	001262			TST	\$BASE		;; UUT BASE REGISTER ADDRESS
2008	007440	001403				BEQ	#+10		;; NOT CHANGE IF = 0
2009	007442	013737	001262	001272		MOV	\$BASE, \$RMADR		;; NEW BASE ADDRESS
2010	007450	013737	001272	026644		MOV	\$RMADR, \$RMADR		;; RH11/70 ADDRESS
2011	007456	013737	001274	026646		MOV	\$RMVEC, \$RMVEC		;; RH11/70 VECTOR
2012									
2013									
2014									
2015									
2016	007464	004737	017020			SETVEC:	JSR	PC, \$T.CLK	;; START THE CLOCK
2017	007470	004737	026662			JSR	PC, \$RMINIT		;; INITIALIZE THE RMO3 DRIVER
2018	007474	012737	177777	026604		MOV	#-1, \$SAVEFG		;; SET THE SAVE REGISTERS FLAG
2019	007502	005227	177777			INC	#-1		;; SEE IF FIRST START
2020	007506	001404				BEQ	11\$		;; BR IF YES
2021	007510	032777	000004	171436		BIT	\$SW02, \$SWR		;; TYPEOUT THE DRIVE STATUS TABLE ?
2022	007516	001101				BNE	10\$		;; BR IF NOT
2023	007520	012737	000340	177776	11\$:	MOV	\$PR7, \$PS		;; SET PRIORITY TO 7
2024	007526	005004				CLR	R4		;; DRIVE TABLE POINTER
2025	007530	104401	001203			TYPE	, \$CRLF		;; CR-LF
2026	007534	104401	035121			TYPE	, \$SYSTAT		;; TYPE STATUS HEADING
2027	007540				1\$:				
2028	007540	010446				MOV	R4, -(SP)		;; SAVE R4 FOR TYPEOUT
2029									;; TYPE DRIVE NUMBER
2030	007542	104403				TYPOS			;; GO TYPE--OCTAL ASCII
2031	007544	002				.BYTE	2		;; TYPE 2 DIGIT(S)
2032	007545	000				.BYTE	0		;; SUPPRESS LEADING ZEROS
2033	007546	104401	035164			TYPE	, \$LIN4SP		;; SPACES
2034	007552	105764	026516			TSTB	\$DRVSTA(R4)		;; CHECK DRIVE'S STATUS
2035	007556	100416				BMI	4\$		;; BR IF UNSAFE
2036	007560	001020				BNE	5\$		;; BR IF ONLINE
2037	007562	105764	026526			TSTB	\$DRVTP(R4)		;; SEE IF OFFLINE OR NONEXISTENT
2038	007566	001404				BEQ	2\$		;; BR IF NONEXISTENT
2039	007570	100006				BPL	3\$		;; BR IF OFFLINE

;; DISPLAY DRIVE STATUS AND SET UP THE OTHER UNITS THE PROGRAM WILL USE



Line	Address	Code	Label	Op	Opnd	Comment
2040	007572	104401	035033	TYPE	NOTRM	;DRIVE NOT AN RM03
2041	007576	000443		BR	4\$	;CHECK NEXT DRIVE
2042	007600	104401	035055	2\$: TYPE	NOTPRS	;DRIVE NOT PRESENT
2043	007604	000440		BR	4\$	;CHECK NEXT DRIVE
2044	007606	104401	035012	3\$: TYPE	UNTOFF	;DRIVE OFFLINE
2045	007612	000410		BR	6\$	;PRINT DRIVE TYPE
2046	007614	104401	035072	4\$: TYPE	NOTSAF	;DRIVE UNSAFE
2047	007620	000405		BR	6\$	;PRINT DRIVE TYPE
2048	007622	104401	035023	5\$: TYPE	UNTON	;DRIVE ONLINE
2049	007626	156437	026632	001404	A:ABIT(R4),DEVMP	;SET DEVICE MAP
2050	007634	104401	035166	6\$: TYPE	LINSP	;SPACES
2051	007640	012737	035102	007704	MOV	#RM03B,8\$ ;ADDRESS OF RPO4 MESSAGE
2052	007646	132764	000001	026526	BITB	#BIT00,DRV TYP(R4) ;RM03 ?
2053	007654	001012		BNE	7\$	;BR IF YES
2054	007656	012737	035107	007704	MOV	#RPOS,8\$ ;ADDRESS OF RPOS MESSAGE
2055	007664	132764	000002	026526	BITB	#BIT01,DRV TYP(R4) ;RPOS ?
2056	007672	001003		BNE	7\$	;BR IF YES
2057	007674	012737	035114	007704	MOV	#RM03A,8\$ ;ADDRESS OF RM03 MESSAGE
2058	007702	104401		7\$: TYPE		;TYPE THE DRIVE TYPE MESSAGE
2059	007704	000000		8\$: .WORD	0	;MESSAGE ADDRESS HERE
2060	007706	104401	001203	9\$: TYPE	\$CRLF	;CR-LF
2061	007712	005204		INC	R4	;INCREMENT DRIVE NUMBER/TABLE POINTER
2062	007714	020427	000010	CMP	R4,#8.	;FINISHED ?
2063	007720	001307		BNE	1\$	;BR IF NOT
2064	007722	104401	001203	10\$: TYPE	\$CRLF	;CR-LF
2065				APT: CODING	FOR APT---	LOAD PARAMETERS FROM E-TABLE
2066	007726	105737	001150	TSTB	\$AUTOB	;RUN UNDER APT ?
2067	007732	001544		BEQ	M1	;NO, DON'T BOTHER
2068	007734	005037	001410	CLR	EDIT	;CLEAR THE TEXT LAST TRACK FLAG
2069	007740	005737	001264	TST	\$DEVM	;DEVICE MAP AVAILABLE FROM APT MAIL BOX
2070	007744	001406		BEQ	1\$	;BRANCH IF NONE
2071	007746	013737	001264	001404	MOV	\$DEVM,DEVMP ;LOAD IT FORM MAIL BOX
2072	007754	042737	177400	001404	BIC	#177400,DEVMP ;ALLOW MAX 8 DRIVES
2073	007762	005737	001404	1\$: TST	DEVMP	;CHECK DEVICE MAP
2074	007766	001002		BNE	MSFX	;BRANCH IF DRIVES ASSIGNED
2075	007770	000137	015324	JMP	\$EOP	;OTHERWISE, EXIT
2076	007774	005037	001326	MSFX: CLR	BEGTRK	;ALWAYS STARTS FROM 0 TRK
2077	010000	005037	001322	CLR	BEGCYL	;ALWAYS STARTS FROM 0 CYLINDER
2078	010004	012737	001466	001320	MOV	#822, ENDCYL ;LAST CYLINDER
2079	010012	012737	000004	001324	MOV	#4, ENDTRK ;LAST TRACK
2080	010020	005037	001402	CLR	WRAP	;CLEAR WRAP AROUND FLAG
2081	010024	012737	177777	001316	MOV	#-1, MODE ;SET ORMAT AND VERIFY MODE
2082	010032	005737	001266	TST	\$CDW1	;APT PARAMETER = 0
2083	010036	001402		BEQ	.+6	;YES, DEFAULT TO -1
2084	010040	005037	001316	CLR	MODE	;NO SET TO CHECK MODE
2085	010044	012737	177777	001362	MOV	#-1, SEC30 ;SET 32 SECTOR
2086	010052	005737	001270	TST	\$CDW2	;APT PARAMETER = 0
2087	010056	001012		BNE	1\$	;NO, 30 SECTOR
2088	010060	012737	020100	001354	MOV	#(258.*32.),WC ;COUNT OF WORDS
2089	010066	012737	157700	001356	MOV	#-(258.*32.),MWC ;WORD COUNT
2090	010074	012737	000037	001364	MOV	#31.,MAXSEC ;MAX 32 SECTOR
2091	010102	000413		BR	2\$	
2092	010104	012737	017074	001354	1\$: MOV	#(258.*30.),WC ;COUNT OF WORDS
2093	010112	012737	160704	001356	MOV	#-(258.*30.),MWC ;WORD COUNT
2094	010120	005037	001362	CLR	SEC30	;30 SECTOR
2095	010124	012737	000035	001364	MOV	#29.,MAXSEC ;MAX 30 SECTOR

```

2096 010132 005737 001316      2$:   TST      MODE          ;CHECK MODE ?
2097 010136 001006              BNE      .+16             ;NO
2098 010140 012737 133331 001340   MOV      #133331,PATA    ;INITIATE PATTERN
2099 010146 012737 133331 001342   MOV      #133331,PATB    ;INITIATE PATTERN
2100 010154 012737 000002 001336   MOV      #2,PATSEL       ;WORSE CASE
2101 010162 012700 000001      MOV      #1,R0           ;UT STARTS AT DRIVE 0
2102 010166 005001              CLR      R1              ;DRIVE NUMBER
2103 010170 030037 001404      3$:   BIT      R0,DEVMP     ;BIT OF DEVICE MAP SET ?
2104 010174 001005              BNE      4$              ;FIND THE DRIVE # FROM DEVICE MAP
2105 010176 000241              CLC
2106 010200 006100              ROL      R0              ;NEXT DRIVE
2107 010202 003416              BLE      6$              ;BRANCH IF ALL DRIVES ARE DONE
2108 010204 005201              INC      R1              ;INCREMENT DRIVE #
2109 010206 000770              BR       3$              ;LOOP
2110 010210 010137 001220      4$:   MOV      R1,DRIVE      ;LOAD R1 INTO DRIVE
2111 010214 040037 001404      BIC      R0,DEVMP       ;CLEAR THE BIT FROM DEVICE MAP
2112 010220 105761 026516      TSTB    DRVSTA(R1)      ;DRIVE ON LINE ?
2113 010224 003003              BGT      5$              ;BRANCH IF DRIVE ON LINE
2114 010226 005237 001216      INC      $DEVCT         ;INCREMENT DEVICE COUNT TO CHANGE MAIL BOX
2115 010232 000756              BR       3$              ;LOOP BACK
2116 010234 000137 011160      5$:   JMP      M5           ;START TO FORMAT OR CHECK
2117 010240 000137 015324      6$:   JMP      $EOP        ;END OF PASS FOR ALL DRIVES ON $DEVM

```

```

;SEE WHICH MODE THE PROGRAM IS TO BE RUN IN.
;MODES ARE: 'FORMAT & VERIFY' OR 'CHECK FORMAT'

```

```

2123 010244 005037 001316      M1:   CLR      MODE          ;SET MODE TO 'CHECK FORMAT'
2124 010250 104401 035376      TYPE    ,MMODE          ;TYPE 'PROGRAM MODE'
2125 010254 104411              RDLIN                    ;READ THE KEYBOARD
2126 010256 012601              MOV      (SP)+,R1        ;GET ADDRESS OF INPUT
2127 010260 105711              TSTB    (R1)            ;'CR' ENTERED (DEFAULT)
2128 010262 001414              BEQ     2$              ;BR IF YES
2129 010264 122711 000103      CMPB    #'C',(R1)       ;'CHECK' ?
2130 010270 001406              BEQ     1$              ;BR IF YES
2131 010272 122711 000106      CMPB    #'F',(R1)       ;'FORMAT' ?
2132 010276 001411              BEQ     3$              ;BR IF YES
2133 010300 104401 001202      TYPE    $QUES           ;NO CORRECT ENTRY
2134 010304 000757              BR       M1             ;TRY AGAIN
2135 010306 104401 035451      1$:   TYPE    MHECK        ;TYPE REST OF 'CHECK'
2136 010312 000410              BR       M1A            ;GET STARTING ADDRESS
2137 010314 104401 035430      2$:   TYPE    MFORMT      ;TYPE DEFAULT MESSAGE
2138 010320 000402              BR       4$            ;SLT UP MODE
2139 010322 104401 035464      3$:   TYPE    MORMAT      ;TYPE REST OF 'FORMAT'
2140 010326 012737 177777 001316 4$:   MOV      #-1,MODE      ;SET MODE TO 'FORMAT & VERIFY'

```

```

;FIND OUT IF FORMAT IS TO BE IN 30 OR 32 SECTOR MODE

```

```

2144 010334 104401 035504      M1A:  TYPE    ,MSIZE       ;TYPE FORMAT MODE REQUEST
2145 010340 104411              RDLIN                    ;READ THE KEYBOARD
2146 010342 012601              MOV      (SP)+,R1        ;ADDRESS OF ENTRY
2147 010344 122711 000131      CMPB    #'Y',(R1)       ;IS ENTRY A 'Y' ?
2148 010350 001410              BEQ     1$              ;BR IF IT IS
2149 010352 122711 000116      CMPB    #'N',(R1)       ;IS ENTRY A 'N' ?
2150 010356 001424              BEQ     2$              ;BR IF IT IS
2151 010360 105711              TSTB    (R1)            ;DEFAULT MODE ?

```

```

2152 010362 001403          BEQ      1$          ;BR IF IT IS
2153 010364 104401 001202  TYPE     $QUES     ;TYPE '?'
2154 010370 000761          BR       M1A        ;TRY AGAIN
2155 010372 104401 035555  1$:     TYPE     MSEC22 ;TYPEOUT MODE SELECTED
2156 010376 012737 020100 001354  MOV     #<258.*32.>,WC ;TRACK SIZE IN 32 SECTOR MODE
2157 010404 012737 157700 001356  MOV     #-<258.*32.>,MWC ;2'S COMPLEMENT WORD COUNT
2158 010412 012737 177777 001362  MOV     #-1,SEC30   ;32 SECTOR INDICATOR
2159 010420 012737 000037 001364  MOV     #31.,MAXSEC ;MAX SECTOR ADDRESS IN 32 SECTOR MODE
2160 010426 000415          BR       M1B        ;CONTINUE
2161 010430 104401 035634  2$:     TYPE     MSEC20 ;TYPE OUT 30 SECTOR MODE SELECTED
2162 010434 012737 017074 001354  MOV     #<258.*30.>,WC ;TRACK SIZE IN 30 SECTOR MODE
2163 010442 012737 160704 001356  MOV     #-<258.*30.>,MWC ;2'S COMPLEMENT WORD COUNT
2164 010450 005037 001362          CLR     SEC30      ;30 SECTOR INDICATOR
2165 010454 012737 000035 001364  MOV     #29.,MAXSEC ;MAX SECTOR ADDRESS IN 30 SECTOR MODE
2166 010462 005037 001326  M1B:    CLR     BEGTRK   ;CLEAR STARTING TRACK ADDRESS
2167 010466 005037 001322          CLR     BEGCTL   ;CLEAR BEGINNING CYLINDER ADDRESS
2168 010472 005037 001402          CLR     WRAP     ;CLEAR WRAP AROUND FLAG
2169 010476 012737 000004 001324  MOV     #4.,ENDTRK ;SETUP END TRACK ADDRESS
2170 010504 012737 000002 001336  MOV     #2.,PATSEL ;SETUP FOR WORST CASE PATTERN
2171 010512 112737 177777 005610  MOVB   #-1,FMTDPB ;SETUP INVALID DRIVE NUMBER
2172 010520 000400          BR       MO        ;BRANCH TO START
2173
2174          ;GO FIND OUT WHAT DRIVE
2175
2176 010522 012737 017240 001376  MO:     MOV     #OENTER,CNTLC ;CONTROL C ABORT ENTRANCE
2177 010530 005037 001126          CLR     $ERTTL    ;CLEAR THE ERROR ACCUMULATOR
2178 010534 004737 024422          JSR     PC,$TKINT ;INITIALIZE THE TTY KEYBOARD
2179 010540 104401 035140          TYPE   ,MUNIT    ;ASK FOR DRIVE NUMBER
2180 010544 104411          RDLIN          ;READ THE KEYBOARD
2181 010546 012601          MOV     (SP)+,R1  ;ADDRESS OF ENTRY
2182 010550 004537 022306          JSR     R5,CK.CHR ;CHECK ONE CHARACTER
2183 010554 010600          3$          ;ILLEGAL CHARACTER
2184 010556 010570          2$          ;CARRIAGE RETURN
2185 010560 010600          3$          ;" "
2186 010562 010570          2$          ;" "
2187 010564 010606          4$          ;DIGIT 0-7
2188 010566 010600          3$          ;DIGIT 8-9
2189 010570 005002  2$:     CLR     R2        ;SELECT DRIVE ZERO
2190 010572 104401 035162  TYPE   MDRVD     ;GO TYPE DEFAULT DRIVE NUMBER
2191 010576 000403          BR       4$       ;GO SEE IF DRIVE ZERO IS THERE
2192 010600 104401 001202  3$:     TYPE   $QUES   ;TYPE '?'
2193 010604 000746          BR       MO        ;ASK FOR DRIVE NUMBER AGAIN
2194 010606 010237 001220  4$:     MOV     R2,DRIVE ;SAVE DRIVE NUMBER
2195 010612 005702          TST    R2        ;SEE IF DRIVE 0
2196 010614 001004          BNE    5$        ;BR IF NOT
2197 010616 122737 000011 000041  CMPB   #11,41   ;PROGRAM LOADED FROM AN RMO3 ?
2198 010624 001431          BEQ    7$        ;BR IF IT WAS, CAN'T FORMAT THE DRIVE
2199 010626 004737 017020  5$:     JSR     PC,ST.CLK ;START THE CLOCK
2200 010632 004737 026662          JSR     PC,RMINIT ;GO SEE WHAT DRIVES ARE AVAILABLE
2201 010636 012737 177777 026604  MOV     #-1,SAVEFG ;SAVE THE REGISTERS
2202 010644 012737 177777 026606  MOV     #-1,SEEKFG ;SET 'NO OPTIMIZATION' FLAG
2203 010652 005037 177776          CLR     PS       ;SET PRIORITY BACK TO ZERO
2204 010656 105762 026516          TSTB   DRVSTA(R2);LOOK AT DRIVE STATUS
2205 010662 003015          BGT    8$        ;BRANCH IF ONLINE
2206 010664 001403          BEQ    6$        ;BR IF DRIVE NOT AVAILABLE
2207 010666 104401 035344  TYPE   ,MUSDR   ;'DRIVE UNSAFE'

```

```

2208 010672 000713          BR      M0          ;GO GET DRIVE NUMBER AGAIN
2209 010674 105762 026526 6$:  TSTB   DRVTP(R2) ;A DRIVE PRESENT?
2210 010700 001003          BNE     7$          ;BR IF SO
2211 010702 104401 035243     TYPE   ,MORNP      ;TYPE 'DRIVE NOT PRESENT'
2212 010706 000705          BR      M0          ;GO GET DRIVE NUMBER AGAIN
2213 010710 104401 035270     TYPE   ,MER11      ;'DRIVE NOT AVAILABLE'
2214 010714 000702          BR      M0          ;GO GET DRIVE NUMBER AGAIN
2215 010716 123737 001220 005610 8$:  CMPB   DRIVE,FMTDPB ;SAME DRIVE AS LAST TIME ?
2216 010724 001422          BEQ    M2          ;BR IF IT IS
2217 010726 113737 001220 005610 MOVB   DRIVE,FMTDPB ;SETUP DRIVE ADDRESS
2218 010734 012737 001466 001320 MOV    #822.,ENDCYL ;SETUP FOR RMO3
2219 010742 132762 000003 026526 BITB   #BIT00!BIT01,DRV ;SEE IF DRIVE RMO3
2220 010750 001003          BNE     9$          ;BR IF EITHER
2221 010752 012737 001466 001320 MOV    #822.,ENDCYL ;SETUP ENDING CYLINDER FOR RMO3
2222 010760 005037 001326     CLR   BEGTRK      ;CLEAR STARTING TRACK ADDRESS
2223 010764 005037 001322     CLR   BEGCTL      ;CLEAR STARTING CYLINDER ADDRESS
2224 010770 000400          BR      M2          ;GET ADDRESS LIMITS FROM THE OPERATOR
2225
2226 ;GET ADDRESS LIMITS
2227
2228 010772 104401 035171     M2:   TYPE   ,ENTADR ;'ENTER ADDRESS LIMITS'
2229 ;      JSR   PC,PARENT ;GET THE ADDRESS LIMITS
2230 010776 004437 017416     ;      JSR   R4,PARENT ;ENTER THE ADDRESS LIMITS
2231 011002 005650          TABLE ;TABLE PARAMETERS
2232 011004 005037 001402     CLR   WRAP        ;CLEAR WRAP FLAG
2233 011010 023737 001320 001322 CMP    ENDCYL,BEGCYL ;SEE IF ENDING CYLINDER EQ TO GT THAN BEGINNING
2234 011016 001402          BEQ    2$          ;BR IF ON THE SAME CYLINDER
2235 011020 103410          BLO    4$          ;BR IF LESS
2236 011022 000413          BR     M4          ;TO CHECK DRIVE
2237 011024 023737 001324 001326 2$:  CMP    ENDTRK,BEGTRK ;SEE IF ENDING TRACK EQ OR GT THAN BEGINNING
2238 011032 103007          BHIS   M4          ;BR IF YES
2239 011034 104401 035733     3$:   TYPE   ,MADRER ;INVALID ADDRESS ENTERED
2240 011040 000754          BR     M2          ;TRY AGAIN
2241 011042 012737 177777 001402 4$:  MOV    #-1,WRAP    ;SET WRAP AROUND FLAG
2242 011050 000400          BR     M4
2243
2244 ;GO GET DATA PATTERN FOR FORMAT
2245
2246 011052 005737 001316     M4:   TST    MODE    ;VERIFY MODE ?
2247 011056 001007          BNE    M4A        ;NO
2248 011060 012737 133331 001340 MOV    #133331,PATA ;VERIFY MODE USE WORST CASE DATA
2249 011066 012737 133331 001342 MOV    #133331,PATB ;INITIAL THE DATA PATTERN -SHIFT 3 BITS
2250 011074 000431          BR     M5          ;START TO FORMAT AND CHECK
2251 011076 104401 036035     M4A:  TYPE   ,MSELP   ;GO TYPE 'SELECT PATTERN'
2252 011102 104411          RDLIN ;READ THE KEYBOARD
2253 011104 012601          MOV    (SP)+,R1   ;ENTRY ADDRESS
2254 011106 004537 022306     JSR   R5,CK.CHR  ;CHECK ONE CHARACTER
2255 011112 011146          3$    ;ILLEGAL CHARACTER
2256 011114 011126          1$    ;CARRIAGE RETURN
2257 011116 011146          3$    ;
2258 011120 011126          1$    ;
2259 011122 011140          2$    ;
2260 011124 011146          3$    ;
2261 011126 104401 036165     1$:   TYPE   ,MPATD  ;TYPE DEFAULT PATTERN
2262 011132 012702 000002     MOV    #2,R2     ;WORST CASE PATTERN
2263 011136 000406          BR     4$        ;GO SAVE PATTERN

```

```

2264 011140 020227 000002      2$:    CMP      R2,#2          ;IS # LARGER THAN 2
2265 011144 101403              BLOS     4$          ;BRANCH IF NOT
2266 011146 104401 001202      3$:    TYPE     $QUES      ;TYPE '?'
2267 011152 000737              BR       14         ;RETYPE LINE
2268 011154 010237 001336      4$:    MOV      R2,PATSEL   ;SAVE PATTERN SELECTED
2269
2270 ;GO TYPE 'STARTING FORMAT ON DRIVE N'
2271
2272 011160 012737 017260 000060  M5:    MOV      #NEWSVC,#TKVEC ;VECTOR FOR CONTROL-0
2273 011166 005037 177776      CLR     #PSW        ;LEVEL 0
2274 011172 005737 001316      TST     MODE        ;'FORMAT' OR 'CHECK' MODE ?
2275 011176 001403              BEQ     1$          ;BR IF 'CHECK' MODE
2276 011200 104401 036200      TYPE   MSFOU       ;TYPE 'STARTING FORMAT ON DRIVE N'
2277 011204 000402              BR      2$          ;
2278 011206 104401 036235      1$:    TYPE     ,MSCHK      ;TYPE 'STARTING CHECK ON DRIVE N'
2279 011212
2280 011212 013746 001220      2$:    MOV      DRIVE,-(SP)   ;;SAVE DRIVE FOR TYPEOUT
2281 ;TYPE DRIVE NUMBER
2282 011216 104403              TYPOS   1          ;GO TYPE--OCTAL ASCII
2283 011220 001              .BYTE  1          ;TYPE 1 DIGIT(S)
2284 011221 000              .BYTE  0          ;SUPPRESS LEADING ZEROS
2285 011222 104401 001203      TYPE   ,$CRLF      ;CR-LF
2286
2287 ;THIS CODE RETRIEVES THE BAD SPOT FILE FROM THEPACK (IF FORMATTED PACK)
2288 ;SET UP DPB BLOCK READ CYL 822. TRK 4
2289
2290 011226 005037 001346      RETBAD: CLR     RETRY      ;CLEAR THE RETRY FLAG
2291 011232 013737 001220 005610  MOV     DRIVE,FMTDPB ;LOAD DRIVE #
2292 011240 012737 157700 005614  MOV     #-<32.*258.>,FMTDPB+4 ;WORD COUNT FULL TRACK INCLUDING HEAD
2293 011246 012737 043746 005616  MOV     #BUF,FMTDPB+6 ;BUFFER ADDRESS
2294 011254 105037 005620      CLR    FMTDPB+10    ;SECTOR 0
2295 011260 112737 000004 005621  MOV    #4,FMTDPB+11 ;TRACK 4
2296 011266 012737 001466 005622  MOV    #822.,FMTDPB+12 ;CYL 822
2297 011274 112737 000173 005612  MOV    #RDHD,FMTDPB+2 ;READ HEAD AND DATA COMMAND
2298 011302 012737 177777 001406  MOV    #-1,PACK     ;RESET THE MFG AVAILABLE FLAG
2299 011310 112737 000143 005612  MOV    #SETFMT,FMTDPB+2 ;SET 16 BIT MODE
2300 011316 112737 000020 005611  MOV    #20,FMTDPB+1 ;FMT SET
2301 011324 004037 027410      1$:    JSR     RO,RMO3      ;SET THE FORMAT BIT
2302 011330 005610      FMTDPB
2303 011332 000774      BR      1$
2304 011334 005737 005626      2$:    TST     FMTDPB+16    ;THE COMMAND IS DONE ?
2305 011340 001775      BEQ     2$          ;BRANCH IF NOT
2306 011342 112737 000173 005612  MOV    #RDHD,FMTDPB+2 ;RELOAD THE READ HEAD AND DATA COMMAND
2307 011350 012737 011350 001124  MOV    #,$LPERR     ;LOOP ON ERROR ADDRESS
2308 011356 012706 001100      MOV    #STACK,SP   ;RESET STACK POINT
2309 011362 004037 027410      3$:    JSR     RO,RMO3      ;READ THE LAST TRACK
2310 011366 005610      FMTDPB
2311 011370 000774      BR      3$
2312 011372 005737 005626      4$:    TST     FMTDPB+16    ;BRANCH IF QUEUE FAIL
2313 011376 001775      BEQ     4$          ;ALL DONE
2314 011400 100024      BPL     6$          ;BRANCH IF NOT
2315 011402 012737 011430 001174  MOV    #5$, $ESCAPE ;BRANCH IF NO ERROR
2316 011410 004737 016144      JSR    PC,$ERINDX  ;ESCAPE TO 5$ ON ERROR
2317 011414 104010      ERROR  10         ;SEE WHICH ERROR
2318 011416 104011      ERROR  11
2319 011420 104012      ERROR  12

```

```

2320 011422 104013 ERROR 13
2321 011424 104014 ERROR 14
2322 011426 104021 ERROR 21
2323 011430 004737 016244 5$: JSR PC,LOP.CK ;CHECK LOOP ON ERROR
2324 011434 022737 000003 001346 CMP #3,RETRY ;DONT' TRY OVER 3 TIMES
2325 011442 103457 BLO #5 ;
2326 011444 005237 001346 INC RETRY
2327 011450 000744 BR 3$ ;TRY AGAIN
2328 011452 005037 001174 6$: CLR $ESCAPE ;
2329 011456 005037 001346 CLR RETRY
2330 011462 005737 005626 TST FMTDPB+16 ;ANY ERROR ?
2331 011466 100445 BMI #5 ;BRANCH IF ANY
2332 011470 032737 140000 043746 BIT #BIT15:BIT14,#BUF ;HEADER INFORMATION OK ?
2333 011476 001441 BEQ #5 ;BRANCH IF NOT
2334 011500 032737 010000 043746 BIT #BIT12,#BUF ;FMT BET MUST BE SET
2335 011506 001435 BEQ #5 ;BRANCH IF NOT SET
2336 011510 022737 151466 043746 CMP #151466,#BUF ;ON THE LAST CYLINDER ?
2337 011516 001031 BNE #5 ;BRANCH IF NOT
2338 011520 022737 002000 043750 CMP #2000,#BUF+2 ;ON TRACK 4 AND SECTOR 0
2339 011526 001025 BNE #5 ;BRANCH IF NOT
2340 011530 005737 043752 TST #BUF+4 ;SERIAL NUMBER SHOULD NOT 0
2341 011534 001003 BNE 7$ ;BRANCH IF NOT 0
2342 011536 005737 043754 TST #BUF+6 ;SECOND WORD OF SERIAL NUMBER
2343 011542 001417 BEQ #5 ;BRANCH IF ZERO
2344 011544 005737 043756 7$: TST #BUF+10 ;THE THIRD WORD MUST BE 0
2345 011550 001014 BNE #5 ;BRANCH IF NOT
2346 011552 022737 177777 043760 CMP #-1,#BUF+12 ;AN ALIGNMENT PACK ?
2347 011560 001002 BNE +6 ;BRANCH IF NOT
2348 011562 000137 016066 JMP REJEC1
2349 011566 005737 043760 TST #BUF+12 ;A DATA PACK ?
2350 011572 001003 BNE #5 ;BRANCH IF NOT IDENTIFIED
2351 011574 012737 000400 001406 MOV #400,PACK ;SET MFG BAD SPOT FILE AVAILABLE FLAG
2352 011602 000240 8$: NOP ;DONE
2353 ;THIS CODE INITIALIZES THE BAD16,BAD18,USTAB,USSAV TABLES
2354 ;
2355
2356 011604 012706 001100 TABINT: MOV #STACK,SP ;INITIAL STACK POINT
2357 011610 005046 CLR -(SP) ;TERMINATOR
2358 011612 012746 004430 MOV #USSAV,-(SP) ;USSAV TABLE ADDRESS
2359 011616 012746 002420 MOV #BAD18,-(SP) ;MFG 18 BIT FILE ADDRESS
2360 011622 012746 001414 MOV #BAD16,-(SP) ;MFG 16 BIT FILE ADDRESS
2361 011626 012703 003424 MOV #USTAB,R3 ;USER BAD SPOT FILE ADDRESS
2362 011632 012704 000004 1$: MOV #4,R4 ;FIRST 4 WORDS SET TO 0 TEMPERARY
2363 011636 005023 2$: CLR (R3)+
2364 011640 005304 DEC R4
2365 011642 001375 BNE 2$ ;BRANCH IF NOT DONE
2366 011644 012704 000374 MOV #252,R4 ;SET -1 TO ALL THE REST LOCATIONS
2367 011650 012705 177777 3$: MOV #-1,R5
2368 011654 010523 MOV R5,(R3)+
2369 011656 005304 DEC R4 ;DECREMENT WORD COUNT
2370 011660 001375 BNE 3$ ;BRANCH IF NOT DONE
2371 011662 012603 MOV (SP)+,R3 ;NEXT TABLE
2372 011664 001362 BNE 1$ ;LOOPING BACK IF NOT TERMINATOR
2373
2374
2375 ;THIS CODE LOADS TABLE BAD16,BAD18,USSAV, FROM PACK (IF FORMATTED PACK)

```

```

2376
2377 011666 005737 001406  TABLD: TST    PACK    ;BAD SPOT FILE AVAILABLE FROM PACK ?
2378 ; BLE    5$      ;BRANCH IF NOT
2379 011672 003451 ; BLE    6$      ;BRANCH IF NOT AVAILABLE
2380 011674 012702 043752  MOV    #BUFP+4,R2 ;LOAD BAD16 TABLE FIRST
2381 011700 012703 001414  MOV    #BAD16,R3  ;TABLE ADDRESS
2382 011704 012704 000400  MOV    #256.,R4   ;WORD COUNT
2383 011710 012223 1$: MOV    (R2)+,(R3)+ ;LOAD THE WHOLE SECTOR
2384 011712 005304  DEC    R4         ;BRANCH IF NOT DONE
2385 011714 001375  BNE    1$
2386 011716 012702 044756  MOV    #BUFP+520.,R2 ;258 X 2 + 2 X 2 + BUFP
2387 011722 012703 002420  MOV    #BAD18,R3  ;TABLE ADDRESS
2388 011726 012704 000400  MOV    #256.,R4   ;WORD COUNT
2389 011732 012223 2$: MOV    (R2)+,(R3)+ ;LOAD INTO TABLE
2390 011734 005304  DEC    R4         ;DECREMENT WORD COUNT
2391 011736 001375  BNE    2$        ;BRANCH IF NOT DONE
2392 011740 012702 057026  MOV    #BUFP+4+(11.*516.)R2 ;SECTOR 11
2393 011744 005737 001362  TST    SEC30      ;FORMAT IN 16 BIT MODE
2394 011750 100402  BMI    3$        ;BRANCH IF IT IS
2395 011752 012702 056022  MOV    #BUFP+4+(10.*516.)R2 ;ECTOR 10
2396 011756 012703 004430  MOV    #USSAV,R3  ;LOAD THE TABLE
2397 011762 012704 000400  MOV    #256.,R4   ;WORD COUNT
2398 011766 012223 4$: MOV    (R2)+,(R3)+ ;
2399 011770 005304  DEC    R4
2400 011772 001375  BNE    4$
2401 011774 062702 000004  ADD    #4,R2      ;ACCESSES THE USTAB BUFFER ADD
2402 012000 012703 003424  MOV    #USTAB,R3  ;TABLE ADDRESS
2403 012004 012704 000400  MOV    #256.,R4   ;WORD COUNT
2404 012010 012223 5$: MOV    (R2)+,(R3)+ ;LOAD THE TABLE
2405 012012 005304  DEC    R4         ;DECREMENT THE COUNT
2406 012014 001375  BNE    5$        ;BRANCH IF NOT ALL DONE
2407 012016 000240 6$: NOP
2408 ;5$: NOP
2409 ;
2410 ;THIS CODE ASK O.P TO ENTER A SERIAL NUMBER FOR UN FORMATTED PACK
2411 ;
2412 ;
2413 ;
2414 012020 005737 001406  SERNL: TST    PACK    ;BAD SPOT FILE AVAILABLE ?
2415 012024 003124 4$      ;BRANCH IF AVAILABLE
2416 012026 105737 001150  TSTB   $AUTOB    ;RUN UNDER APT ?
2417 012032 001144 5$      ;BRANCH IF IT IS
2418 012034 004737 024422  JSR    PC,$TKINT ;INITIAL THE KEYBOARD
2419 012040 104401 001203  TYPE   , $CRLF   ;TYPE CRLF
2420 012044 104401 036462  TYPE   ,MSG1     ;"ENTER TWO WORDS FOR SN"
2421 012050 104401 036550 1$: TYPE   ,SN1
2422 012054 012702 000012  MOV    #10.,R2   ;MAXMUM 10 OCTAL DIGITS ALLOWED
2423 012060 005037 001414  CLR    BAD16     ;CLEAR THE LSW OF THE SERIAL NUMBER
2424 012064 005037 001416  CLR    BAD16+2   ;CLEAR THE MSW OF THE SERIAL NUMBER
2425 012070 104411  RDLIN  ;READ IN THE STRING
2426 012072 012601  MOV    (SP)+,R1  ;ADDRESS OF THE READ IN STRING
2427 012074 105711  TSTB   (R1)      ;TERMINATOR OF THE INPUT STRING
2428 012076 001471  BEQ    3$        ;BRANCH IF IT IS
2429 012100 121127 000060  CMPB   (R1),#'0  ;LESS THAN ASCIZ 0 ?
2430 012104 103761 1$      ;ENTER AGAIN IF IT IS
2431 012106 121127 000067  CMPB   (R1),#'7  ;HIGH THAN ASCIX 7 ?

```

```

2432 012112 101356          BHI      1$          ;ENTER AGAIN IF 50
2433 012114 013746 001414   MOV      BAD16, -(SP) ;SAVE THE LSW
2434 012120 042737 100000 001414   BIC      #BIT15, BAD16 ;CLEAR THE SIGN BIT OF LSW
2435 012126 006137 001414   ROL      BAD16       ;ROTATE THE LSW FIVE TIMES
2436 012132 006137 001414   ROL      BAD16       ;TO MOVE THE MOST SIGN DIGIT
2437 012136 006137 001414   ROL      BAD16       ;TO BIT 0 TO BIT 2
2438 012142 006137 001414   ROL      BAD16
2439 012146 006137 001414   ROL      BAD16
2440 012152 042737 177770 001414   BIC      #177770, BAD16 ;LEFT ON THE MS DIGIT OF THE LSW
2441 012160 006337 001416   ASL      BAD16+2     ;SHIFT THE MSW LEFT ONE OCTAL DIGIT
2442 012164 100731          BMI      1$          ;ENTER AGAIN IF SIGN BIT SET
2443 012166 006337 001416   ASL      BAD16+2
2444 012172 100726          BMI      1$
2445 012174 006337 001416   ASL      BAD16+2
2446 012200 100723          BMI      1$
2447 012202 063737 001414 001416   ADD      BAD16, BAD16+2 ;APPENDING THE DIGITS INTO MSW
2448 012210 012637 001414   MOV      (SP)+, BAD16 ;RESTORE THE LSW
2449 012214 006337 001414   ASL      BAD16       ;SHIFT THE LSW LEFT ONE OCTAL DIGIT
2450 012220 006337 001414   ASL      BAD16
2451 012224 006337 001414   ASL      BAD16
2452 012230 042737 100000 001414   BIC      #BIT15, BAD16 ;CLEAR THE SIGN BIT
2453 012236 111103          MOVVB   (R1), R3     ;LOAD THE CURRENT READ IN DIGITS
2454 012240 042703 177770   BIC      #177770, R3 ;LEFT ONLY ONE OCTAL DIGIT
2455 012244 060337 001414   ADD      R3, BAD16   ;APPEND THE READ IN CHARACTER TO LSW
2456 012250 005201          INC      R1          ;ADJUST THE READ IN STRING ADDRESS
2457 012252 005302          DEC      R2          ;DECREMENT ONE DIGIT COUNT
2458 012254 001307          BNE      2$          ;BRANCH IF NOT DONE
2459 012256 105711          TSTB   (R1)         ;IF 10 DIGITS HAVE BEEN ENTERED
2460 012260 001273          BNE      1$          ;MUST BE FOLLOWED BY <CR>
2461 012262 005737 001414 3$:   TST      BAD16      ;SERIAL NUMBER MUST BE NOT 0
2462 012266 001003          BNE      4$          ;BRANCH IF NOT 0
2463 012270 005737 001416   TST      BAD16+2
2464 012274 001665          BEQ      1$          ;ENTER AGAIN IF ZERO
2465 012276 013737 001414 002420 4$:   MOV      BAD16, BAD18 ;LOAD ALL TABLE WITH THE SERIAL NUMBER
2466 012304 013737 001414 003424   MOV      BAD16, USTAB ;
2467 012312 013737 001414 004430   MOV      BAD16, USSAV ;
2468 012320 013737 001416 003426   MOV      BAD16+2, USTAB+2 ;
2469 012326 013737 001416 004432   MOV      BAD16+2, USSAV+2 ;
2470 012334 013737 001416 002422   MOV      BAD16+2, BAD18+2 ;
2471 012342 000410          BR       6$          ;EXIT
2472 012344 005237 004430 5$:   INC      USSAV      ;CODE FOR AUTO MODE APT,ACT,ETC.
2473 012350 005237 001414   INC      BAD16
2474 012354 005237 002420   INC      BAD18
2475 012360 005237 003424   INC      USTAB
2476 012364 000240          6$:   NOP
2477 012366 000240          :   MOV      #NEWSVC, @#TKVEC ;DONE
2478 012368 000240          :   CLR      @#PSW    ;NEW INTERRUPT VECTOR FOR KEYBOARD
2479 012370 000240          :   ;THIS CODING PROVID UTILITY ROUTINE ENTRIES
2480 012372 000240          CALIN:  NOP
2481 012366 000240          CALIN:  JSR      PC, $TKINT ;INITIALIZE THE KEYBOARD
2482 012370 004737 024422          CALIN:  TSTB   $AUTOB ;AUTO BYTE SET ?
2483 012374 105737 001150          CALIN:  BNE      CKADRS ;IF SO, BRANCH
2484 012400 001033          CALIN:  TST      $CDW1  ;ALLOW ANY UTILITY ROUTINE ?
2485 012402 005737 001266          CALIN:  BEQ      CKADRS ;BRANCH IF NOT
2486 012406 001430          CALIN:  BEQ      CKADRS ;BRANCH IF NOT
2487 012410 104401 037111 1$:   TYPE   ,MSGBLK    ;ENTER MESSAGE BLOCK

```



```

2488 012414 104411 RDLIN ;READ IN THE ROUTINE NUMBER
2489 012416 012601 MOV (SP)+,R1 ;ADDRESS OF INPUT BUFFER
2490 012420 122711 000060 CMPB #'0,(R1) ;INPUT NUMBER LESS THAN 0 ?
2491 012424 101012 BHI 2$ ;BRANCH IF SO
2492 012426 122711 000064 CMPB #'4,(R1) ;INPUT NUMBER LARGER THAN 4 ?
2493 012432 103407 BLO 2$ ;BRANCH IF SO
2494 012434 042711 177760 BIC #177760,(R1) ;LEFT OF THREE BITS
2495 012440 011101 MOV (R1),R1 ;FOUND THE INDEX VALUE
2496 012442 006301 ASL R1 ;WORD INDEX
2497 012444 004771 006120 JSR PC,ATABCAL(R1) ;CALL THE DESIRED UTILITY ROUTINE
2498 012450 000757 BR 1$ ;LOOP BACK TO THE SAME POINT
2499 012452 000240 2$: NOP ;EXIT
2500 012454 005037 001366 CLR DS.CYL ;CLEAR ALL PARAMETERS
2501 012460 005037 001370 CLR DS.TRK
2502 012464 005037 001350 CLR SAVSEC
2503 ;SETUP TOTAL TRACK COUNT FOR FORMAT
2504
2505 012470 023737 001320 001322 CKADRS: CMP ENDCYL,BEGCYL ;STARTING AND ENDING CYLINDERS THE SAME ?
2506 012476 001011 BNE 5$ ;BRANCH IF BEGCYL NOT EQUAL TO ENDCYL
2507 012500 013737 001324 001330 MOV ENDTRK,TTRKS ;END TRACK ADDRESS
2508 012506 163737 001326 001330 SUB BEGTRK,TTRKS ;SUBTRACT THE STARTING TRACK ADDRESS
2509 012514 005237 001330 INC TTRKS ;MAKE THE NUMBER OF TRACKS INCLUSIVE
2510 012520 000450 BR SETPAT ;SETUP THE DATA PATTERN
2511 012522 005737 001402 5$: TST WRAP ;WRAP AROUND FLAG SET ?
2512 012526 001407 BEQ 1$ ;NO
2513 012530 012702 001466 MOV #822.,R2 ;INITIAL COUNTER TO 822
2514 012534 163702 001322 SUB BEGCYL,R2 ;FIRST PART OF CYL #
2515 012540 063702 001320 ADD ENDCYL,R2 ;SECOND PART OF CYL #
2516 012544 000405 BR 4$
2517 012546 013702 001320 1$: MOV ENDCYL,R2 ;ENDING CYLINDER
2518 012552 163702 001322 SUB BEGCYL,R2 ;SUBTRACT THE STARTING CYLINDER
2519 012556 005302 DEC R2 ;EXCLUSIVE LAST CYLINDER
2520 012560 012737 000005 001330 4$: MOV #5,TTRKS ;INITIATE COUNTER
2521 012566 163737 001326 001330 SUB BEGTRK,TTRKS ;# OF TRACK ON STARTING CYLINDER
2522 012574 063737 001324 001330 ADD ENDTRK,TTRKS ;# OF TRACK ON ENDING CYLINDER
2523 012602 005237 001330 INC TTRKS ;INCLUSIVE ONE TRACK
2524 012606 005302 3$: DEC R2 ;DECREMENT THE CYLINDER COUNT
2525 012610 100404 BMI 6$ ;BR IF DONE
2526 012612 062737 000005 001330 ADD #5.,TTRKS ;ADD CYLINDER WORTH OF TRACKS
2527 012620 000772 BR 3$ ;CONTINUE
2528 012622 005037 001410 6$: CLR EDIT ;RESET EDIT FLAG
2529 012626 023727 001330 010011 CMP TTRKS,#(821.*5) ;ALL TRACKS ARE FORMATTED ?
2530 012634 103402 BLO SETPAT ;BRANCH IF NOT
2531 012636 005237 001410 INC EDIT ;SET EDIT FLAG
2532
2533 ;THIS CODE SETS UP THE SELECTED DATA PATTERN TO BE WRITTEN ON EACH SECTOR IMAGE
2534
2535 012642 005737 001316 SETPAT: TST MODE ;VERIFY MODE ?
2536 012646 001427 BEQ 1$ ;YES
2537 012650 005037 001340 4$: CLR PATA ;CLEAR DATA PATTERN A
2538 012654 005037 001342 CLR PATB ;CLEAR DATA PATTERN B
2539 012660 005737 001336 TST PATSEL ;SEE IF PATTERN OF ONES
2540 012664 001420 BEQ 1$ ;BR IF SO
2541 012666 012737 177777 001340 MOV #177777,PATA ;PATA = AB
2542 012674 012737 177777 001342 MOV #177777,PATB ;PATB=CD
2543 012702 022737 000001 001336 CMP #1,PATSEL ;SEE IF PATTERN OF ZEROS

```

```

2544 012710 001406          BEQ      1$          ;BRANCH IF 50
2545 012712 012737 066666 001340      MOV      #066666,PATA ;SET UP WORST CASE
2546 012720 012737 066666 001342      MOV      #066666,PATB ;SET UP WORST CASE
2547 012726 013703 001354          1$:    MOV      WC,R3        ;SET UP COUNTER
2548 012732 012700 043746          MOV      #BUF,RO      ;SET UP MEMORY POINTER
2549 012736 013701 001340          3$:    MOV      PATA,R1     ;SET UP PATTERN IN R1
2550 012742 013702 001342          MOV      PATB,R2     ;SET UP PATTERN IN R2
2551 012746 010120          2$:    MOV      R1,(R0)+    ;MOV 1ST PAT INTO MEM
2552 012750 010220          MOV      R2,(R0)+    ;MOV 2ND PAT INTO MEM
2553 012752 005303          DEC      R3          ;KEEP COUNT
2554 012754 005303          DEC      R3          ;KEEP COUNT
2555 012756 001373          BNE     2$          ;DO IT AGAIN IF R3 NOT 0
2556          ;THIS CODE SETS UP FOR THE ACTUAL FORMAT
2557
2558 012760 012737 017260 000060      MOV      #NEWSVC,#TKVEC ;NEW INTERRUPT VECTOR
2559 012766 005037 177776          CLR     #PSW        ;CLEAR THE PS
2560 012772 004737 016312          WRTRK: JSR     PC,SETTBL ;GO SET UP DRIVER TABLE
2561 012776 004737 016624          JSR     PC,SETHDR   ;GO INITIALIZE HEADERS IN THE SECTOR BUFFER IN CORE
2562 013002 112737 000020 005611      MOV     #20,FMTDPB+1 ;LOAD FMT22 BIT
2563 013010 005737 001362          TST     SEC30       ;18 BIT MODE ?
2564 013014 001002          BNE     1$          ;BR IF NOT
2565 013016 105037 005611          CLRB   FMTDPB+1    ;CLEAR THE FMT22 BIT
2566 013022 112737 000143 005612  1$:    MOV     #SETFMT,FMTDPB+2 ;'LOAD FORMAT' COMMAND
2567 013030 004037 027410          2$:    JSR     RO,RM03   ;START THE COMMAND
2568 013034 005610          FMTDPB ;DPB ADDRESS
2569 013036 000774          BR      2$          ;QUEUE FULL RETURN
2570 013040 005737 005626          3$:    TST     FMTDPB+16 ;LOOK FOR DONE
2571 013044 001775          BEQ     3$          ;LOOP UNTIL FINISHED
2572
2573          ;SET UP SECCU TABLE: BAD SPOT ON CURRENT TRACK ,RETRIEVED FROM PACK
2574
2575 013046 012703 005434          WRTRKX: MOV     #SECCU,R3 ;INITILIZE THE SECCU TABLE
2576 013052 012704 000040          MOV     #32,R4      ;TOTAL 32 WORD MAX
2577 013056 012702 177777          MOV     #-1,R2     ;LOAD ALL LOCATION TO -1
2578 013062 010223          1$:    MOV     R2,(R3)+  ;
2579 013064 005304          DEC     R4          ;
2580 013066 001375          BNE     1$          ;BRANCH IF NOT DONE
2581
2582          ;LOAD THE BAD SPOT INTO SECCU TABLE IF ANY
2583
2584 013070 005737 001406          WRTRKY: TST     PACK   ;BAD SPOT FILE AVAILABLE AT ALL
2585 013074 003461          BLE     6$          ;BRANCH IF NOT
2586 013076 005046          CLR     -(SP)      ;CLEAR UP THE DUMMY RETURN ADDRESS
2587 013100 012746 003434          MOV     #USTAB+B,-(SP) ;USER BAD SECTOR TABLE
2588 013104 012702 001414          MOV     #BAD16,R2  ;R2 POINTER OF MFG BAD SPOT TABLE
2589 013110 005737 001362          TST     SEC30      ;IF 18 BIT MODE CHANGE POINTER
2590 013114 100402          BMI     1$          ;BRANCH IF 16 BIT MODE
2591 013116 012702 002420          MOV     #BAD18,R2  ;
2592 013122 010201          1$:    MOV     R2,R1     ;R1: LAST ADDRESS OF BAD SPOT TABLE
2593 013124 062701 000776          ADD     #510,R1    ;
2594 013130 012703 005434          MOV     #SECCU,R3  ;TABLE ADDRESS
2595 013134 012704 000076          MOV     #62,R4     ;LAST ADDRESS OF THE TABLE
2596 013140 060304          ADD     R3,R4      ;
2597 013142 062702 000010          ADD     #10,R2     ;BAD SPOT STARTS FROM THE 5TH WORD OF THE FILE
2598 013146 023712 005622          2$:    CMP     FMTDPB+12,(R2) ;BAD SPOT ON THIS CYLINDER ?
2599 013152 001007          BNE     3$          ;BRANCH IF NOT

```

```

2600 013154 123762 005621 000003      CMPB   FMTDPB+11,3(R2) ;ALSO, ON THE SAME TRK ?
2601 013162 001003                BNE    3$              ;BRANCH IF NOT
2602 013164 116205 000002      MOVB   2(R2),R5        ;BAD SECTOR
2603 013170 010523                MOV    R5,(R3)+       ;WORD STORAGE
2604 013172 062702 000004      3$:   ADD    #4,R2      ;ADVANCE TO NEXT SET
2605 013176 020102                CMP    R1,R2          ;ALL SPOT ARE CHECKED ?
2606 013200 103412                BLO   5$              ;BRANCH IF IT IS
2607 013202 020403                CMP    R4,R3          ;END OF SECCU TABLE ?
2608 013204 103401                BLO   4$              ;REJECT THE PACK
2609 013206 000757                BR    2$              ;LOOPING BACK
2610 013210 004737 017162      4$:   JSR    PC,RESTR  ; IMPORPER MFG INFORMATION
2611 013214 012737 177777 001406      MOV    #-1,PACK      ; RESET MFG FILE AVAILABLE FLAG
2612 013222 000137 011604      JMP    TABINT         ; AND START AGAIN
2613 013226 011601                5$:   MOV    (SP),R1     ; NEXT TABLE
2614 013230 062701 000766      ADD    #502.,R1      ; ENDING OF THE TABLE
2615 013234 012602                MOV    (SP)+,R2      ; BEGINNING OF THE TABLE
2616 013236 001343                BNE   2$              ; BRANCH IF NOT DUMMY ADD
2617 013240 000240                6$:   NOP                    ; DONE
2618
2619
2620
2621
2622
2623 013242 012702 005434      SORT1: MOV    #SECCU,R2   ; TOP OF THE TABLE
2624 013246 012703 005532      MOV    #SECCU+62.,R3 ; BOTTOM OF THE TABLE
2625 013252 012704 000037      MOV    #31.,R4 ; TOTAL 32 ELEMENT, SORT 31 TIMES
2626 013256 022712 177777      CMP    #-1,(R2)      ; IS THE TABLE EMPTY ?
2627 013262 001451                BEQ   6$              ; BRANCH IF IT IS, QUICK EXIT
2628 013264 021262 000002      1$:   CMP    (R2),2(R2) ; N TH ELEMENT : N+1 TH ELEMENT
2629 013272 001406                BLO   4$              ; BRANCH IF SMALLER
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
                BEQ   2$              ; DELET N+1 TH ELEMENT, IF SAME
                OTHERWISE, N > N+1
                ; STORE N ELEMENT
                ; SWITCH N+1 ELEMENT TO N ELEMENT
                ; SWITCH N ELEMENT TO N+1 ELEMENT
                ; ADJUST TOP POINTER
                ; TERMINATOR ?
                ; BRANCH IF IT IS
                ; SAVE TOP POINTER
                ; DELET 2(R2)
                ; BOTTOM OF THE TABLE ?
                ; BRANCH IF NOT
                ; LOAD -1 TO TABLE BOTTOM
                ; RESTORE THE TOP POINTER
                ; DECREMENT ONE ELEMENT COUNT
                ; BRANCH IF ALL DONE
                ; INCREMENT THE TOP POINTER
                ; REACH THE BOTTOM ?
                ; BRANCH IF NOT
                ; DECREMENT ONE SORT COUNT
                ; BRANCH IF ALL DONE
                ; RESET TOP POINTER
                ; UPDATE BOTTOM POINTER
                ; EXIT IF TOP = BOTTOM
                ;
                ; SORT THE REST ELEMENTS

```

```

2656 013406 000240
2657 013410 005037 005536
2658 013414 005037 001314
2659 013420 005037 001346
2660 013424 105037 005620
2661 013430 013737 001356 005614
2662 013436 012737 043746 005616
2663
2664
2665
2666
2667
2668
2669 013444 022737 177777 005536
2670 013452 001001
2671 013454 000000
2672 013456 113737 005536 005620
2673 013464 012701 005434
2674 013470 022711 177777
2675 013474 001511
2676 013476 032711 100000
2677 013502 001403
2678 013504 062701 000002
2679 013510 000767
2680 013512 123711 005620
2681 013516 001415
2682 013520 103401
2683 013522 000000
2684 013524 111102
2685 013526 023702 001364
2686 013532 001463
2687 013534 103471
2688
2689 013536 005202
2690 013540 010237 005536
2691 013544 052721 100000
2692 013550 000432
2693 013552 105237 005620
2694 013556 052721 100000
2695 013562 123711 005620
2696 013566 001771
2697 013570 123737 005620 001364
2698 013576 101402
2699
2700
2701
2702 013600 000137 014564
2703 013604 022711 177777
2704 013610 001443
2705 013612 111102
2706 013614 023702 001364
2707
2708 013620 001430
2709 013622 103436
2710 013624 005202
2711 013626 010237 005536

```

```

6S: NOP ;DONE
CLR NEXT ;FLAG START OF A NEW TRACK
WRTRK1: CLR SOFSW ;CLEAR ERROR COUNTER
CLR RETRY ;ZERO THE RETRY COUNTER
CLRB FMTDPB+10 ;RESTORE SECTOR
MOV MWC FMTDPB+4 ;RESTORE WC
MOV #BUFF FMTDPB+6 ;RESTORE BA
;DELET THE FOLLOWING TWO LINES FOR NEW VERIFY OPERATION
TST MODE ;'FORMAT' OR 'CHECK' MODE ?
BEQ CKTRK ;BR IF 'CHECK' MODE
;THIS CODE TO SET UP WORD COUNT AND BUFFER ADDRESS
WRTRKA: CMP #-1,NEXT ;END OF OPERATION ?
BNE 1S ;BRANCH IF NOT
HALT ;OTHERWISE SOMETHING WRONG *****
1S: MOVF NEXT,FMTDPB+10 ;STARTING SECTOR
MOV #SECCU,R1 ;R1 POINTS TO THE BAD SECTOR TABLE
2S: CMP #-1,(R1) ;END OF TABLE ?
BEQ WRTRKD ;BRANCH IF IT IS
BIT #BIT15,(R1) ;THE BAD SECTOR HAS ALREADY ACCESSED ?
BEQ 3S ;BRANCH IF NOT
ADD #2,R1 ;ADJUST THE POINTER
BR 2S ;LOOPING BACK
3S: CMPB FMTDPB+10,(R1) ;BAD SPOT = STARTING SECTOR ?
BEQ WRTRKB ;BRANCH IF IT I
BLO 4S ;BRANCH IF START SEC < BAD SPOT
HALT ;OTHERWISE SOMETHING WRONG
4S: MOVF (R1),R2 ;UPDATE NEXT STARTING ADDRESS
CMP MAXSEC,R2 ;NEXT BAD SPOT IS THE LAST SECTOR?
BEQ WRTRKF ;BRANCH IF LAST SECTOR *****
BLO WRTRKD ;BRANCH IF LESS *****
BLOS WRTRKD ;BRANCH 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 1S ;IMPORPER MFG INDORMATION
; JSR PC,RESTRT ;START AGAIN AND REBUILD FMG FILE
; MOV #-1,PACK ;RESET THE MFG FILE AVAILABLE FLAG
; JMP TABINT ;RESTART POINT
; JMP HSET ;EXIT
1S: CMP #-1,(R1) ;TO SEE IF NEXT BAD SPOT IN TABLE
BEQ WRTRKD ;BRANCH IF NONE
MOVF (R1),R2 ;UPDATE THE NEXT
CMP MAXSEC,R2 ;BAD SECTOR IS THE LAST SECTOR ?
BLOS WRTRKD ;BRANCH IF IT IS
BEQ WRTRKF ;*****
BLO WRTRKD ;BRANCH IF LESS ***
INC R2 ;BAD SEC +1
MOV R2,NEXT ;NEXT STARTING ADDRESS

```

2712	013632	052711	100000		BIS	#BIT15, (R1)	; MARK THE ACCESSED SECTOR
2713	013636	013702	005536		MOV	NEXT, R2	; CALCULATE THE WORD COUNT
2714	013642	162702	000002		SUB	#2, R2	; ENDSECTOR = NEXT - 2
2715	013646	006302			ASL	R2	; WORD INDEX
2716	013650	016203	006232		MOV	WCTBL(R2), R3	; WORD CTR FOR SECTOR 0 TO ENDING SECTOR
2717	013654	113702	005620		MOVB	FMTDPB+10, R2	; STARTING SECTOR
2718	013660	006302			ASL	R2	; WORD INDEX
2719	013662	166203	006232		SUB	WCTBL(R2), R3	; WORD COUNT FOR SECTOR 0 TO STARTING SECTOR
2720	013666	062703	000402		ADD	#258., R3	; ADJUST ONE SECTOR
2721	013672	005403			NEG	R3	; GET THE NEAGTIVE WORD COUNT
2722	013674	010337	005614		MOV	R3, FMTDPB+4	; LOAD THE WORD CTR INTO DPB
2723	013700	000432			BR	WTRKE	; LOAD THE STARTING ADDRESS
2724	013702	012737	177777	005536	WTRKRF: MOV	#-1, NEXT	; ALL DONE
2725	013710	052711	100000		BIS	#BIT15, (R1)	; MASK THE SECTOR
2726	013714	005302			DEC	R2	; PATCH FOR THE LAST SECTOR
2727	013716	000405			BR	WTRKH	; PATCH
2728	013720	012737	177777	005536	WTRKRD: MOV	#-1, NEXT	; NO MORE BAD SPOT
2729	013726	013702	001364		MOV	MAXSEC, R2	; LAST SECTOR
2730	013732	006302			WTRKRD: ASL	R2	; WORD INDEX
2731	013734	016203	006232		MOV	WCTBL(R2), R3	; WORD CTR FOR THE ENDING SECTOR
2732	013740	113702	005620		MOVB	FMTDPB+10, R2	; STARTING ADDRESS
2733	013744	006302			ASL	R2	; WORD INDEX
2734	013746	166203	006232		SUB	WCTBL(R2), R3	; WORD CTR FOR THE STARTING SECTOR
2735	013752	062703	000402		ADD	#258., R3	; ADJUST ONE SECTOR
2736	013756	005403			NEG	R3	; NEGATIVE WORD COUNT
2737	013760	010337	005614		MOV	R3, FMTDPB+4	; LOAD THE WORD COUNT INTO DPB
2738	013764	000240			NOP		; DONE
2739	013766	113702	005620		WTRRKE: MOVB	FMTDPB+10, R2	; STARTING ADDRESS
2740	013772	006302			ASL	R2	; LOCATE THE BUFFER ADDRESS
2741	013774	016237	006132	005616	MOV	ADRTBL(R2), FMTDPB+6	; BUFFER ADDRESS
2742	014002	112737	000163	005612	WTRRK2: MOVB	#WTRHD, FMTDPB+2	; SET WRITE HEADER & DATA COMMAND IN TBL
2743	014010	012737	014010	001124	MOV	#, SLPERR	; SET UP LOOP ON ERROR ADDRESS
2744	014016	012706	001100		MOV	#STACK, SP	; LOAD STACK POINT
2745	014022	004037	027410		JSR	RO, RMO3	; GO FORMAT A TRACK
2746	014026	005610			FMTDPB		; ADRS OF PARAMETERS - TBL
2747	014030	000774			BR	1\$	; WAIT FOR QUEUE IF FULL
2748	014032	005737	005626		2\$: TST	FMTDPB+16	; WAIT FOR COMMAND TO COMPLETE
2749	014036	001775			BEQ	2\$	; BRANCH IF NOT DONE
2750	014040	100024			BPL	4\$	; BRANCH IF NO ERROR
2751	014042	012737	014070	001174	MOV	#3\$, \$ESCAPE	; ESCAPE TO 3\$ ON ERROR
2752	014050	004737	016144		JSR	PC, \$RINDX	; SEE WHICH ERROR
2753	014054	104010			ERROR	10	; DRIVE OFFLINE
2754	014056	104011			ERROR	11	; PERSISTENT DRIVE UNSAFE ERROR
2755	014060	104012			ERROR	12	; UNCORRECTABLE MASSBUS PARITY ERROR
2756	014062	104013			ERROR	13	; SOFTWARE TIMEOUT
2757	014064	104014			ERROR	14	; DRIVE UNSAFE ERROR
2758	014066	104015			ERROR	15	; DRIVE/CONTROLLER ERROR DURING WRITE
2759	014070	004737	016244		3\$: JSR	PC, LOP.CK	; LOOP ON THE ERROR ?
2760	014074	022737	000003	001346	CMP	#3, RETRY	; ERROR RETRY LIMIT ?
2761	014102	001403			BEQ	4\$	; BR IF REACHED
2762	014104	005237	001346		INC	RETRY	; COUNT THE ERROR
2763	014110	000744			BR	1\$	; TRY AGAIN
2764	014112	005037	001174		4\$: CLR	\$ESCAPE	; CLEAR ERROR ESCAPE ADDRESS
2765	014116	005037	001346		CLR	RETRY	; CLEAR THE RETRY COUNTER
2766							
2767							; CHECK THE TRACK JUST WRITTEN

2768											
2769	014122	112737	000153	005612	CKTRK:	MOV	#WCKHD,FMTDPB+2	:	SET WRITE CHECK HEADER & DATA COMMAND IN TBL		
2770	014130	012737	014130	001124		MOV	#SLPERR	:	SETUP LOOP ON ERROR ADDRESS		
2771	014136	012706	001100			MOV	#STACK,SP	:	LOAD STACK POINTER		
2772	014142	004037	027410		1\$:	JSR	RD,RM03	:	GO CHECK THE TRACK JUST FORMATTED		
2773	014146	005610				FMTDPB		:	ADRS OF PARAMETERS - TBL		
2774	014150	000774				BR	1\$	:	WAIT FOR QUEUE IF FULL		
2775	014152	005737	005626		2\$:	TST	FMTDPB+16	:	WAIT FOR COMMAND TO COMPLETE		
2776	014156	001775				BEQ	2\$	:	BRANCH IF NOT DONE		
2777	014160	100131				BPL	8\$	:	BRANCH IF NO ERROR		
2778	014162	113737	005546	001350		MOV	RM.REG+RMDA,SAVSEC	:	GET THE SECTOR ADDRESS		
2779	014170	001005				BNE	3\$	:	BRANCH IF NOT SECTOR 0		
2780	014172	013737	001364	001350		MOV	MAXSEC,SAVSEC	:	RESTORE TO LAST SECTOR +1		
2781	014200	005237	001350			INC	SAVSEC	:	INCREMENT TO LAST SECTOR +1		
2782	014204	005337	001350		3\$:	DEC	SAVSEC	:	ADJUST SECTOR TO THE ONE THAT FAILED		
2783	014210	012737	014512	001174		MOV	#10\$,SESCAPE	:	ESCAPE TO 10\$ ON ERROR		
2784	014216	004737	016144			JSR	PC,ERINDX	:	SEE WHICH ERROR		
2785	014222	104010				ERROR	10	:	DRIVE OFFLINE		
2786	014224	104011				ERROR	11	:	PERSISTENT DRIVE UNSAFE ERROR		
2787	014226	104012				ERROR	12	:	UNCORRECTABLE MASSBUS PARITY ERROR		
2788	014230	104013				ERROR	13	:	SOFTWARE TIMEOUT		
2789	014232	104014				ERROR	14	:	DRIVE UNSAFE ERROR		
2790	014234	032737	040000	005550		BIT	#WCE, RM.REG+RMCS2	:	WRITE CHECK ERROR ?		
2791	014242	001005				BNE	4\$	:	BR IF SET		
2792	014244	033727	005554			BIT	RM.REG+RMER1,(PC)+	:	CHECK FOR DATA ERRORS		
2793	014250	130620				.WORD	DCK!OPI!DTE!ACRC!HCE!FER	:	DATA ERROR BITS		
2794	014252	001001				BNE	4\$	:	BR IF SET		
2795	014254	104016				ERROR	16	:	CONTROLLER/DRIVE ERROR DURING WRITE CHECK		
2796	014256	005737	001314		4\$:	TST	SOFSW	:	RETRYING THE SECTOR ?		
2797	014262	001413				BEQ	5\$	:	BR IF NOT		
2798	014264	012737	014536	001174		MOV	#11\$,SESCAPE	:	ESCAPE TO 11\$ ON ERROR		
2799	014272	004737	017532			JSR	PC,RECORD	:	RECORD THE BAD SPOT		
2800	014276	012737	100000	001412		MOV	#BIT15,HEADX	:	RESET THE USER BIT		
2801	014304	004737	020354			JSR	PC,RESET	:	RESET THE HEADER BITS		
2802	014310	104017				ERROR	17	:	SECTOR NOT ACCEPTABLE		
2803	014312	005037	001174		5\$:	CLR	SESCAPE	:	RESET ESCAPE ADDRESS		
2804	014316	032737	040000	005552		BIT	#ERR, RM.REG+RMD5	:	DATA ERROR ?		
2805	014324	001002				BNE	6\$	:	BR IF IT IS		
2806	014326	104024				ERROR	24	:	WRITE CHECK ERROR		
2807	014330	000401				BR	7\$	:	CONTINUE		
2808	014332	104020			6\$:	ERROR	20	:	DATA ERROR DURING WRITE CHECK		
2809	014334	013701	001350		7\$:	MOV	SAVSEC,R1	:	FAILING SECTOR ADDRESS		
2810	014340	113737	001350	005620		MOV	SAVSEC,FMTDPB+10	:	SECTOR ADDRESS		
2811	014346	006301				ASL	R1	:	SETUP INDEX TO ADDRESS WORDS		
2812	014350	016137	006132	005616		MOV	ADRTBL(R1),FMTDPB+6	:	BUFFER ADDRESS FOR FAILING SECTOR		
2813	014356	013701	001364			MOV	MAXSEC,R1	:	R1 ENDING SECTOR #		
2814	014362	005737	005536			TST	NEXT	:	WHOLE TRACK DONE ?		
2815	014366	100404				BMI	13\$	:	BRANCH IF IT IS		
2816	014370	013701	005536			MOV	NEXT,R1	:	LOAD CURRENT ENDING SECTOR		
2817	014374	162701	000002			SUB	#2,R1	:	ADJUST TWO SECTORS		
2818	014400	006301			13\$:	ASL	R1	:	WORD INDEX		
2819	014402	016102	006232			MOV	WCTBL(R1),R2	:	WORDCTR FOR ENDING SECTOR		
2820	014406	113701	005620			MOV	FMTDPB+10,R1	:	STARTING ADDRESS		
2821	014412	006301				ASL	R1	:	WORD COUNT		
2822	014414	166102	006232			SUB	WCTBL(R1),R2	:	WORD CTR FOR STARTING SECTOR		
2823	014420	005402				NEG	R2	:			

2824	014422	010237	001352			MOV	R2, SAVWC	:SAVE THE WORD COUNT
2825	014426	012737	177376	005614		MOV	#-258., FMTDPB+4	:WORD COUNT FOR 1 SECTOR
2826	014434	005237	001314			INC	SOF5W	:INDICATE THAT A RETRY IS IN PROGRESS
2827	014440	000137	014002			JMP	WRTRK2	:REFORMAT ERROR SECTOR
2828	014444	005737	001314		8\$:	TST	SOF5W	:RETRY IN PROGRESS ?
2829	014450	001432				BEQ	11\$	:BR IF NOT
2830	014452	022737	000002	001314		CMP	#2, SOFSW	:SEE IF LAST RETRY
2831	014460	001403				BEQ	9\$	:BR IF IT IS
2832	014462	005237	001314			INC	SOF5W	:INCREMENT RETRY COUNT
2833	014466	000625				BR	1\$	:READ AGAIN
2834	014470	123737	001364	005620	9\$:	CMPB	MAXSEC, FMTDPB+10	:SEE IF LAST SECTOR ON THE TRACK
2835	014476	001417				BEQ	11\$	:BR IF IT IS
2836	014500	004737	016576			JSR	PC, SCAMC	:SETUP TO CHECK REMAINING SECTORS ON THE TRACK
2837	014504	005037	001314			CLR	SOF5W	:CLEAR RETRY COUNTER
2838	014510	000614				BR	1\$	:FINISH CHECKING THE TRACK
2839	014512	004737	016244		10\$:	JSR	PC, LOP.CK	:CHECK FOR LOOP ON ERROR
2840	014516	022737	000003	001346		CMP	#3, RETRY	:ERROR RETRY REACHED ?
2841	014524	001404				BEQ	11\$	:BR IF IT IS
2842	014526	005237	001346			INC	RETRY	:COUNT THE RETRY
2843	014532	000137	014142			JMP	1\$	:DO THE WRITE CHECK AGAIN
2844	014536	005037	001174		11\$:	CLR	SESCAPE	:CLEAR THE ERROR RETURN ESCAPE ADDRESS
2845	014542	005037	001346			CLR	RETRY	:CLEAR THE RETRY COUNTER
2846	014546	005037	001314			CLR	SOF5W	:CLEAR THE SOFTWARE RETRY COUNT
2847	014552	005737	005536			TST	NEXT	:WHOLE TRACK DONE ?
2848	014556	100402				BMI	.+6	:BRANCH IF IT IS
2849	014560	000137	013444			JMP	WRTRKA	
2850	014564	032777	000002	164362	HSET:	BIT	#BIT1, JSWR	:LOOP ON CURRENT TRACK ?
2851	014572	001402				BEQ	12\$	:BR IF NOT
2852	014574	000137	013046			JMP	WRTRKX	:START AGAIN ON THE SAME TRACK
2853								:RESET THE HEADER WORD OF ALL MFG
2854								:DEFINED BAD SECTOR
2855	014600	012737	040000	001412	12\$:	MOV	#BIT14, HEADX	:RESET THE MFG BIT OF THE 1ST HEADER WORD
2856	014606	012704	005434			MOV	#SECCU, R4	:TABLE FOR BAD SECTOR ON CURRENT TRACK
2857	014612	022714	177777		14\$:	CMP	#-1, (R4)	:END OF TABLE ?
2858	014616	001414				BEQ	16\$	:BRANCH IF SO
2859	014620	032714	040000			BIT	#BIT14, (R4)	:HAS THE SECTOR BE ASSCESSED ?
2860	014624	001006				BNE	15\$	:BRANCH IF SO
2861	014626	052714	040000			BIS	#BIT14, (R4)	:MASK THE SECTOR TABLE
2862	014632	111437	001350			MOVB	(R4), SAVSEC	:SECTOR ADDRESS
2863	014636	004737	020354			JSR	PC, RESET	:RESET THE HEADER
2864	014642	062704	000002		15\$:	ADD	#2, R4	:INCREMENT THE TABLE POINTER
2865	014646	000761				BR	14\$	:LOOPING BACK
2866	014650	000240			16\$:	NOP		:ALL DONE
2867								
2868	014652	004737	016402		WRTRKZ:	JSR	PC, TRKTST	:GET UPDATED TRACK AND CYLINDER ADDRESSES
2869	014656	000402				BR	HDREAD	:RETURN HERE IF DONE - DO QUICK CHECK OF DISK
2870	014660	000137	013046			JMP	WRTRKX	:CONTINUE WITH THE FORMAT
2871								
2872								:THIS CODE MAKES SURE EACH CYLINDER FORMATTED CONTAINS THE
2873								:PROPER CYLINDER ADDRESS. THE PROGRAM IS LOOKING FOR
2874								:POSSIBLE POSITIONER ERRORS THAT MAY HAVE OCCURRED DURING THE FORMAT.
2875								
2876	014664	005737	001316		HDREAD:	TST	MODE	:VERIFY MODE ?
2877	014670	001022				BNE	9\$	:NO
2878	014672	022737	066666	001340		CMP	#066666, PATA	:PATTERN HAS BEEN ROTATED TO WORST CASE ?
2879	014700	001002				BNE	.+6	

2880	014702	000137	015324		JMP	SEOP	:EXIT
2881	014706	000241			CLC		:CLEAR THE CARRY BIT
2882	014710	005037	001340		ROR	PATA	:ROTATE PATTERN
2883	014714	103003			BCC	.+10	:SET SIGN BIT IF CARRY=1
2884	014716	052737	100000	001340	BIS	#BIT15,PATA	:FROM BIT 0
2885	014724	013737	001340	001342	MOV	PATA,PATB	:SET BOTH PATTERN
2886	014732	000137	012470		JMP	CKADRS	:CONTINUE THE VERIFY OPERATION
2887	014736	005037	001360	9\$:	CLR	HEDERR	:CLEAR HEADER CHECK ERROR INDICATOR
2888	014742	004737	016312		JSR	PC,SETTBL	:GO SET UP DRIVER TABLE
2889	014746	004737	016624		JSR	PC,SETHDR	:GO SET UP HEADERS IN CORE
2890	014752	013737	001322	001344	MOV	BEGCYL,CYLCK	:USE 'BEGCYL' FOR FORMAT VERIFICATION
2891	014760	012737	177776	005614	MOV	#-2,FMTDPB+4	:SET UP WORD COUNT
2892	014766	012737	043736	005616	MOV	#RBUF,FMTDPB+6	:SET UP BUFFER ADRS
2893	014774	005037	005620		CLR	FMTDPB+10	:CLEAR THE SECTOR & TRACK ADDRESS FIELD
2894	015000	013737	001322	005622	MOV	BEGCYL,FMTDPB+12	:SETUP THE CYLINDER FIELD
2895	015006	112737	000173	005612	MOV	#RDHD,FMTDPB+2	:SET UP READ HEADER & DATA COMMAND
2896	015014	012737	015014	001124	MOV	#.SLPERR	:SETUP LOOP ON ERROR ADDRESS
2897	015022	012706	001100		MOV	#STACK,SP	:LOAD STACK POINTER
2898	015026	004037	027410	1\$:	JSR	RD,RM03	:GO READ HEADER
2899	015032	005610			FMTDPB		:ADRS OF PARAMETER TBL
2900	015034	000774			BR	1\$	:WAIT IF QUEUE IS FULL
2901	015036	005737	005626	2\$:	TST	FMTDPB+16	:WAIT FOR COMMAND TO COMPLETE
2902	015042	001775			BEQ	2\$	:BRANCH IF NOT DONE
2903	015044	100033			BPL	4\$	:BR IF NOT ERROR
2904	015046	012737	015154	001174	MOV	#5\$,SESCAPE	:ESCAPE TO 5\$ ON ERROR
2905	015054	004737	016144		JSR	PC,ERINDX	:SEE WHICH ERROR
2906	015060	104010			ERROR	10	:DRIVE OFFLINE
2907	015062	104011			ERROR	11	:PERSISTENT DRIVE UNSAFE ERROR
2908	015064	104012			ERROR	12	:UNCORRECTABLE MASSBUS PARITY ERROR
2909	015066	104013			ERROR	13	:SOFTWARE TIMEOUT
2910	015070	104014			ERROR	14	:DRIVE UNSAFE ERROR
2911	015072	032737	177577	005626	BIT	#1C<HCE>,FMTDPB+16	:IS ONLY ERROR A HEADER COMPARE ERROR ?
2912	015100	001410			BEQ	3\$	:BR IF YES
2913	015102	005737	005554		TST	RM.REG+14	:ANY DRIVE ERROR
2914	015106	001412			BEQ	4\$	:BRANCH IF NONE
2915	015110	032737	100000	005602	BIT	#BIT15,RM.REG+42	:BAD SPOT ERROR ?
2916	015116	001006			BNE	4\$	:BRANCH IF IT IS
2917	015120	104021			ERROR	21	:CONTROLLER/DRIVE ERROR VERIFYING HEADERS
2918	015122	005037	001174	3\$:	CLR	SESCAPE	:CLEAR THE ERROR ESCAPE ADDRESS
2919	015126	104022			ERROR	22	:HEADER COMPARE ERROR VERIFYING HEADERS
2920	015130	004737	016244		JSR	PC,LOP.CK	:CHECK FOR LOOP ON ERROR
2921	015134	052737	140000	043736	BIS	#BIT15:BIT14,RBUF	:SET BIT15 AND BIT 14
2922	015142	023737	043736	043746	CMP	RBUF,BUFF	:SEE IF CYL READ EQUALS CYL EXPECTED
2923	015150	001403			BEQ	6\$	:BR IF CYL CORRECT
2924	015152	104023			ERROR	23	:CYLINDER NOT CORRECT
2925	015154	004737	016244	5\$:	JSR	PC,LOP.CK	:CHECK FOR LOOP ON ERROR
2926	015160	023737	001320	001344	6\$:	CMP	ENDCYL,CYLCK
2927	015166	001002			BNE	7\$	:BR IF NOT FINISHED
2928	015170	000137	015324		JMP	SEOP	:END OF FORMAT
2929	015174	005237	001344	7\$:	INC	CYLCK	:INCREMENT CYLINDER ADDRESS BEING CHECKED
2930	015200	022737	001466	001344	CMP	#822.,CYLCK	:LAST CYLINDER ?
2931	015206	003041			BGT	8\$	:NO
2932	015210	122737	000004	005621	CMPB	#4,FMTDPB+11	:LAST TRACK ?
2933	015216	003035			BGT	8\$	:NO THEN BRANCH
2934	015220	005237	001344		INC	CYLCK	:INCREMENT CYLINDER ADDRESS
2935	015224	023737	001320	001344	CMP	ENDCYL,CYLCK	:LAST CYLINDER ?



```

2936 015232 001752
2937 015234 005037 001344
2938 015240 005037 005622
2939 015244 005037 005620
2940 015250 005037 005626
2941 015254 013746 001322
2942 015260 013746 001326
2943 015264 005037 001326
2944 015270 005037 001322
2945 015274 004737 016624
2946 015300 012637 001326
2947 015304 012637 001322
2948 015310 000646
2949 015312 004737 016712
2950 015316 005237 005622
2951 015322 000641
2952
2953
2954
2955
2956
2957
2958
2959
2960 015324
2961 015324 005737 001410
2962 015330 001536
2963 015332 005037 001346
2964
2965 015336 004537 017646
2966 015342 003434
2967 015344 004737 020006
2968 015350 112737 000020 005611
2969 015356 112737 000143 005612
2970 015364 004037 027410
2971 015370 005610
2972 015372 000774
2973 015374 005737 005626
2974 015400 001775
2975 015402 012737 001466 005622
2976 015410 112737 000004 005621
2977 015416 105037 005620
2978 015422 012737 157700 005614
2979 015430 012737 043746 005616
2980 015436 112737 000163 005612
2981 015444 004037 027410
2982 015450 005610
2983 015452 000774
2984 015454 005737 005626
2985 015460 001775
2986 015462 100061
2987 015464 012737 015512 001174
2988 015472 004737 016144
2989 015476 104010
2990 015500 104011
2991 015502 104012

```

```

BEQ 6$ ;EXIT IF IT IS
CLR CYLCK ;YES LAST CYLINDER
CLR FMTDPB+12 ;RESER CYLINDER # TO 0
CLR FMTDPB+10 ;RESET TRACK AND SECTOR #
CLR FMTDPB+16 ;RESET CYLINDER #
MOV BEGCYL,-(SP) ;SAVE BEGCYL
MOV BEGTRK,-(SP) ;SAVE BEGTRK
CLR BEGTRK
CLR BEGCYL
JSR PC,SETHDR ;SET UP BUFFER
MOV (SP)+,BEGTRK ;RESTORE BEGTRK
MOV (SP)+,BEGCYL ;RESTORE BEGCYL
BR 1$
8$: JSR PC,UPDACY ;SET UP FOR NEXT CYL
INC FMTDPB+12 ;ADVANCE TO NEXT CYL #
BR 1$ ;GO READ NEXT HEADER

.SBTTL END OF PASS ROUTINE

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

$EOP:
5$: TST EDIT ;TEST LAST TRACK ?
BEQ 11$ ;BRANCH IF NOT
CLR RETRY ;CLEAR RETRY COUNT
; JSR PC, SORT2 ;SORT THE USTAB
; JSR RS, SORT2 ;SORT THE USTAB IN ASCENDING FORM
USTAB+B. ;TABLE ENTRY POINT
JSR PC,TEXT ;SET UP THE BUFFER
MOVB #20,FMTDPB+1 ;16 BIT MODE
MOVB #SETFMT,FMTDPB+2 ;SET FORMAT COMMAND
6$: JSR RO,RMO3 ;CALL THE DRIVER
FMTDPB
BR 6$ ;LOOP IF QUEGU FAILS
7$: TST FMTDPB+16 ;ALL DONE ?
BEQ 7$ ;BRANCH IF NOT
MOV #822,FMTDPB+12 ;CYLINDER ADDRESS
MOVB #4,FMTDPB+11 ;TRACK NUMBER
CLRB FMTDPB+10 ;SECTOR 0
MOV #-<32.*258.>,FMTDPB+4 ;WORD COUNT
MOV #BUFP,FMTDPB+6 ;BUFFER ADDRESS
MOVB #WRTHD,FMTDPB+2 ;WRITE HEAD AND DATA COMMAND
8$: JSR RO,RMO3 ;CALL THE DRIVER
FMTDPB
BR 8$ ;LOOP IF QUEUE FAILS
9$: TST FMTDPB+16 ;ALL DONE ?
BEQ 9$ ;BRANCH IF NOT
BPL 11$ ;EXIT IF NO ERROR
MOV #22$, $ESCAPE ;ERROR ESCAPE ADDRESS
JSR PC,ERINDX ;ERROR INDICATOR RT.
ERROR 10
ERROR 11
ERROR 12

```

2992	015504	104013			ERROR	13	
2993	015506	104014			ERROR	14	
2994	015510	104015			ERROR	15	
2995	015512	000240		22\$:	NOP		
2996	015514	005237	001346		INC	RETRY	
2997	015520	022737	000003	001346	CMP	#3,RETRY	;; OVER 3 TIMES ?
2998	015526	101346			BHI	8\$	;; BRANCH IF IT IS
2999	015530	000137	016126		JMP	REJCT3	;; WRITE THE LAST TRACK FAILS THREE TIMES
3000	015534	122737	000037	005620	CMPB	#31.,FMTDPB+10	;; LAST SECTOR ?
3001	015542	003431			BLE	11\$	;; BRANCH IF ALL SET
3002	015544	113737	005546	001350	MOVB	RM.REG+RMDA,SAVSEC	;; FOUND THE FALLING SPOT
3003	015552	001003			BNE	10\$	;; BRANCH IF NOT 0
3004	015554	012737	000037	001350	MOV	#31.,SAVSEC	;; THE LAST SECTOR IS A BAD SECTOR
3005	015562	113737	001350	005620	10\$:	MOVB SAVSEC,FMTDPB+10	;; LOAD THE NEW STARTING SECTOR
3006	015570	113701	001350		MOVB	SAVSEC,R1	;; COOK THE WORD CTR AND BUFFER
3007	015574	006301			ASL	R1	;; WORD INDEX
3008	015576	016137	006132	005616	MOV	ADRTBL(R1),FMTDPB+6	;; BUFFER ADDRESS
3009	015604	012737	157700	005614	MOV	#-(258.*32.)FMTDPB+4	;; WORD COUNT
3010	015612	066137	006232	005614	ADD	WCTBL(R1),FMTDPB+4	;; WORD COUNT
3011	015620	005037	001346		CLR	RETRY	;; RESET THE RETRY FLAG
3012	015624	000707			BR	8\$	;; BRANCH BACK
3013	015626	005237	001216	11\$:	INC	\$DEVCT	;; INCREMENT THE DEVICE COUNT
3014	015632	005737	001316		TST	MODE	;; FORMAT OR CHECK MODE
3015	015636	001403			BEQ	12\$	;; BR IF CHECK
3016	015640	104401	036271		TYPE	MFCMPT	;; FORMAT COMPLETE
3017	015644	000402			BR	13\$	;; FINISH THE MESSAGE
3018	015646	104401	036316	12\$:	TYPE	MCCMPT	;; CHECK COMPLETE
3019	015652	104401	036342	13\$:	TYPE	NUMERR	;; TOTAL ERRORS
3020	015656	013746	001126		MOV	\$ERTTL,-(SP)	;; SAVE \$ERTTL FOR TYPEOUT
3021	015662	104405			TYPDS		;; GO TYPE--DECIMAL ASCII WITH SIGN
3022	015664	104401	001203		TYPE	\$CRLF	
3023	015670	005737	001404		TST	DEVMP	;; ALL ASSIGNED DEVICE DONE ?
3024	015674	001402			BEQ	14\$	;; BRANCH IF ALL DONE
3025	015676	005337	001214		DEC	\$PASS	;; RESTORE THE PASS COUNT
3026	015702	000240		14\$:	NOP		
3027							
3028							
3029	015704	005237	001214		INC	\$PASS	;; INCREMENT THE PASS NUMBER
3030	015710	042737	100000	001214	BIC	#100000,\$PASS	;; DON'T ALLOW A NEG. NUMBER
3031	015716	005327			DEC	(PC)+	;; LOOP?
3032	015720	000001		SEOPCT:	.WORD	1	
3033	015722	003013			BGT	\$DOAGN	;; YES
3034	015724	012737			MOV	(PC)+,2(PC)+	;; RESTORE COUNTER
3035	015726	000001		SENDCT:	.WORD	1	
3036	015730	015720			SEOPCT		
3037	015732	013700	000042		\$GET42:	2#42,R0	;; GET MONITOR ADDRESS
3038	015736	001405			BEQ	\$DOAGN	;; BRANCH IF NO MONITOR
3039	015740	000005			RESET		;; CLEAR THE WORLD
3040	015742	004710		SENDAD:	JSR	PC,(R0)	;; GO TO MONITOR
3041	015744	000240			NOP		;; SAVE ROOM
3042	015746	000240			NOP		;; FOR
3043	015750	000240			NOP		;; ACT11
3044	015752			\$DOAGN:			
3045	015752	000137			JMP	2(PC)+	;; RETURN
3046	015754	015756		\$RTNAD:	.WORD	DONE	
3047	015756	105737	001150	DONE:	TSTB	\$AUTOB	;; RUN UNDER APT

```

3048 015762 001405          BEQ      2$          ;NO
3049 015764 005737 001404  TST     DEVMP      ;ALL DRIVES ARE DONE ?
3050 015770 001427          BEQ      4$          ;BRANCH IF ALL DONE
3051 015772 000137 007774  JMP     MSFX        ;NEXT DRIVE FROM APT DEVICE MAP
3052 015776 032777 000001 163150 2$:  BIT     #SWD,#SWR  ;SEE IF SWR 0 SET
3053 016004 001002          BNE     1$          ;BR IF SET
3054 016006 000137 010522  JMP     MD          ;ASK FOR NEXT DRIVE
3055 016012 012706 001100      1$:  MOV     #STACK,SP  ;RESET STACK POINTER
3056 016016 005737 001316  TST     MODE       ;VERIFY MODE ?
3057 016022 001402          BEQ      3$          ;YES
3058 016024 000137 012470  JMP     CKADRS      ;DO THE FORMAT OR CHECK AGAIN
3059 016030 012737 133331 001340 3$:  MOV     #133331,PATA ;INITIATE PATTERN
3060 016036 012737 133331 001342  MOV     #133331,PATB ;INITIATE PATTERN
3061 016044 000137 012470  JMP     CKADRS      ;LOOP ON CHECK
3062 016050 005237 001214      4$:  INC     $PASS      ;GARBAGE CODE FOR APT
3063 016054 005237 001212      5$:  INC     $TESTN
3064 016060 005237 001216  INC     $DEVCT
3065 016064 000773          BR      5$
3066 016066 104401 001203  REJEC1: TYPE    , $SCRLF
3067 016072 104401 037050  TYPE    , MMSG4
3068 016076 000000          HALT
3069 016100 000137 000200  JMP     @#200
3070 016104 104401 001203  REJCT2: TYPE    , $SCRLF
3071 016110 113737 005621 001370  MOV     FMTDPB+11,DS.TRK ;TRACK NUMBER FOR ERROR LOG
3072 016116 104025          ERROR   25          ;REPORT ILLEGAL BAD SECTOR
3073 016120 000000          HALT
3074 016122 000137 000200  JMP     @#200
3075 016126 104401 001203  REJCT3: TYPE    , $SCRLF
3076 016132 104401 040375  TYPE    , MMSG17
3077 016136 000000          HALT
3078 016140 000137 000200  JMP     @#200
3079
3080 ;*****
3081
3082 .SBTTL SUPPORT SUBROUTINES
3083
3084 ;*****
3085
3086 ;THIS ROUTINE DETERMINES THE ERROR TYPE
3087
3088 ERINDX: MOV     R1,-(SP) ;STORE R1
3089          CLR     R1      ;CLEAR R1
3090          BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
3091          .WORD  BIT14!BIT13!BIT2!BIT1 ;DRIVE OFFLINE
3092          BNE     5$      ;BR IF OFFLINE
3093          BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
3094          .WORD  BIT12      ;DRIVE PERSISTENTLY UNSAFE
3095          BNE     4$      ;BR IF UNSAFE
3096          BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
3097          .WORD  BIT11!BIT10 ;UNCORRECTABLE MASSBUS PARITY ERROR ?
3098          BNE     3$      ;BR IF PARITY ERROR
3099          BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
3100          .WORD  BIT09!BIT08 ;SOFTWARE TIMEOUT ?
3101          BNE     2$      ;BR IF YES
3102          BIT     FMTDPB+16,(PC)+ ;CHECK ERROR TYPE
3103          .WORD  BIT04      ;DRIVE UNSAFE ERROR ?

```

```

3104 016216 001001      BNE      1$      ;BR IF YES
3105 016220 005201      INC      R1      ;INCREMENT THE RETURN INDEX
3106 016222 005201      1$:      INC      R1      ;INCREMENT THE RETURN INDEX
3107 016224 005201      2$:      INC      R1      ;INCREMENT THE RETURN INDEX
3108 016226 005201      3$:      INC      R1      ;INCREMENT THE RETURN INDEX
3109 016230 005201      4$:      INC      R1      ;INCREMENT THE RETURN INDEX
3110 016232 006301      5$:      ASL      R1      ;DOUBLE THE INCREMENT
3111 016234 060166 000002  ADD      R1,2(SP) ;DEVELOP THE RETURN ADDRESS
3112 016240 012601      MOV      (SP)+,R1 ;RESTORE R1
3113 016242 000207      RTS      PC      ;RETURN
3114
3115 ;ROUTINE TO CHECK FOR LOOP ON ERROR
3116
3117 016244 032777 001000 162702 LOP.CK: BIT      #SW09,2SWR ;LOOP ON ERROR ?
3118 016252 001402      BEQ      1$      ;BR IF NOT
3119 016254 000177 162644      JMP      2$LPERR ;GO TO THE LOOP ON ERROR ADDRESS
3120 016260 005037 001174      1$:      CLR      $ESCAPE ;CLEAR THE ERROR ESCAPE ADDRESS
3121 016264 033727 005626      BIT      FMTDPB+16,(PC)+ ;CHECK FOR 'FATAL' ERROR
3122 016270 072002      .WORD   BIT14!BIT13!BIT12!BIT10!BIT01 ;'FATAL' ERROR BITS
3123 016272 001004      BNE      2$      ;BR IF NOT
3124 016274 032737 004000 005554  BIT      #WLE,RM.REG+RMR1 ;WRITE LOCK ERROR ?
3125 016302 001402      BEQ      3$      ;BR IF NOT
3126 016304 000137 015324      2$:      JMP      $EOP    ;TERMINATE THE FORMAT
3127 016310 000207      3$:      RTS      PC      ;RETURN
3128
3129 ;THIS ROUTINE SETS UP THE INPUT TABLE FOR THE DRIVER ROUTINE
3130
3131 016312 113737 001220 005610 SETTBL: MOVB    DRIVE,FMTDPB ;SET UP DRIVE #
3132 016320 105037 005613      CLRB    FMTDPB+3 ;CLEAR HIGH ORDER ADRS BITS
3133 016324 013737 001356 005614      MOV     MWC,FMTDPB+4 ;LOAD UP WORD COUNT
3134 016332 012737 043746 005616      MOV     #BUF,FMTDPB+6 ;LOAD UP CURRENT ADRS
3135 016340 105037 005620      CLRB    FMTDPB+10 ;SET SECTOR TO ZERO
3136 016344 113737 001326 005621      MOVB   BEGTRK,FMTDPB+11 ;SET UP STARTING TRK ADRS
3137 016352 013737 001322 005622      MOV     BEGCYL,FMTDPB+12 ;SET UP STARTING CYL
3138 016360 005037 005626      CLR     FMTDPB+16 ;CLEAR RMO3 STATUS
3139 016364 013737 001326 001334      MOV     BEGTRK,TRKCNT ;SET UP PARTIAL CYL TRACK COUNT
3140 016372 013737 001330 001332      MOV     TTRKS,TTRKSC ;SET UP TOTAL TRACKS COUNTER
3141 016400 000207      RTS     PC      ;RETURN FROM SETUP
3142
3143 ;THIS ROUTINE CONTROLS THE DISK ADDRESSING AND TOTAL TRK COUNT
3144 ;IT IS ENTERED AFTER EVERY TRK OPERATION
3145
3146 016402 005337 001332      TRKTST: DEC     TTRKSC ;SEE IF LAST TRACK
3147 016406 001472      BEQ     3$      ;BRANCH IF LAST TRACK
3148 016410 022737 000004 001334      CMP     #4.,TRKCNT ;IS THIS THE LAST TRACK IN THE CYLINDER ?
3149 016416 001423      BEQ     1$      ;BRANCH IF SO
3150 016420 005237 001334      INC     TRKCNT ;COUNT UP TRACK WITHIN CYLINDER
3151 016424 105237 005621      INCB   FMTDPB+11 ;INCREMENT TRACK NUMBER
3152 016430 022737 001466 005622      CMP     #822.,FMTDPB+12 ;LAST CYLINDER AND LAST TRACK ?
3153 016436 003010      BGT     5$      ;NO
3154 016440 122737 000004 005621      CMPB   #4,FMTDPB+11 ;LAST TRACK ?
3155 016446 003004      BGT     5$      ;NO
3156 016450 005337 001332      DEC     TTRKSC ;DECREMENT ONE TRACK
3157 016454 001447      BEQ     3$      ;BRANCH IF LAST TRACK
3158 016456 000403      BR      1$      ;DON'T DESTROY THE LAST TRACK OF CYL822
3159 016460 004737 016770      5$:      JSR     PC,UPDATK ;UPDATE TRACK ADDRESS IN BUFFER

```

```

3160 016464 000441          BR      2$          ;EXIT
3161 016466 005037 001334 1$: CLR      TRKCNT      ;CLEAR TRACK COUNT FOR NEXT CYLINDER
3162 016472 105037 005621   CLRB    FMTDPB+11    ;RESET TRACK ADDRESS TO 0
3163 016476 005237 005622   INC     FMTDPB+12    ;UPDATE CYLINDER NUMBER
3164 016502 022737 001467 005622  CMP     #823.,FMTDPB+12 ;LAST CYLINDER REACHED ?
3165 016510 003025          BGT     4$          ;NO
3166 016512 013746 001322   MOV     BEGCTL,-(SP)  ;SAVE BEGCTL
3167 016516 013746 001326   MOV     BEGTRK,-(SP) ;SAVE BEGTRK
3168 016522 005037 001322   CLR     BEGCTL      ;RESET THE STARTING CYL
3169 016526 005037 001326   CLR     BEGTRK     ;RESET THE START TRACK
3170 016532 004737 016624   JSR    PC,SETHDR    ;INITIATE HEAD WORDS
3171 016536 005037 005620   CLR     FMTDPB+10   ;CLEAR TRACK AND SECTOR
3172 016542 005037 005622   CLR     FMTDPB+12   ;CLEAR CYLINDER #
3173 016546 005037 005626   CLR     FMTDPB+16   ;CLEAR STATUS
3174 016552 012637 001326   MOV     (SP)+,BEGTRK ;RESTORE BEGTRK
3175 016556 012637 001322   MOV     (SP)+,BEGCTL ;RESTORE BEGCTL
3176 016562 000402          BR      2$
3177 016564 004737 016712 4$: JSR    PC,UPDACY   ;UPDATE CYLINDER NUMBER IN BUFFER
3178 016570 062716 000002 2$: ADD     #2,(SP)    ;ADD TWO TO RETURN ADRS
3179 016574 000207          3$: RTS     PC        ;RETURN
3180
3181 ;THIS ROUTINE SETS UP WC & BA FOR REMAINING SECTORS
3182 ;AFTER A WRITE CHECK ERROR
3183
3184 016576 013737 001352 005614 SCAWC: MOV     SAVWC,FMTDPB+4 ;SET UP WC FOR REMAINING SECTORS
3185 016604 062737 001004 005616   ADD     #516.,FMTDPB+6 ;SET UP BA FOR REMAINING SECTORS
3186 016612 005237 005620   INC     FMTDPB+10   ;ADVANCE TO NEXT SECTOR IN TBL
3187 016616 005037 001314   CLR     SOFSW      ;RESET RETRY COUNTER
3188 016622 000207          RTS     PC        ;RETURN TO COMPLETE TRK FORMAT
3189
3190
3191 ;THIS ROUTINE SETS UP THE CYLINDER ADRS, FORMAT BIT, TRACK AND
3192 ;SECTOR ADRS IN MEMORY WITH THE STARTING CYL - TRK INFORMATION
3193
3194 016624 013701 001364 001212 SETHDR: MOV     MAXSEC,R1 ;SET UP SECTOR COUNT
3195 016630 012737 000001   MOV     #1,$TESTN   ;GARBAGE CODE FOR APT
3196 016636 012700 043746   MOV     #BUFP,R0    ;SET UP HEADER POINTER IN R0
3197 016642 013702 001322   MOV     BEGCTL,R2   ;PUT STARTING CYL # IN R2
3198 016646 052702 140000   BIS     #140000,R2  ;BIT 15,BIT 14 ARE SET FOR 144 SPC 3/14/77
3199 016652 013703 001326   MOV     BEGTRK,R3   ;PUT STARTING TRK # IN R3
3200 016656 000303          SWAB    R3          ;JUSTIFY TRACK ADRS
3201 016660 005737 001362   TST    SEC30       ;SEE IF 30 OR 32 SECTOR MODE
3202 016664 001402          BEQ    1$          ;BR IF 30 SECTOR MODE
3203 016666 052702 010000   BIS     #10000,R2   ;SET THE 32 SECTOR FORMAT BIT
3204 016672 010220 1$: MOV     R2,(R0)+    ;WRITE IN HEADER AREA OF CORE THE CYL ADRS
3205 016674 010320   MOV     R3,(R0)+    ;WRITE IN HEADER AREA OF CORE THE TRK ADRS
3206 016676 062700 001000   ADD     #512.,R0    ;SET UP FOR NEXT HEADER
3207 016702 005203          INC     R3          ;UPDATE SECTOR ADRS FOR NEXT HEADER
3208 016704 005301          DEC     R1          ;MAINTAIN COUNT OF SECTORS
3209 016706 002371          BGE    1$          ;BRANCH IF NOT LAST SECTOR
3210 016710 000207          RTS     PC        ;EXIT - HEADERS ARE LOADED INTO CORE
3211
3212 ;THIS ROUTINE UPDATES THE CYLINDER ADRS OF THE HEADER WORDS IN CORE
3213
3214 016712 013701 001364 UPDACY: MOV     MAXSEC,R1 ;SET UP SECTOR COUNT
3215 016716 012700 043746   MOV     #BUFP,R0    ;SET UP HEADER POINTER IN R0

```

```

3216 016722 010246          MOV      R2, -(SP)          ;SAVE R2
3217 016724 005220          1$: INC      (R0)+          ;INCREMENT FOR NEXT CYLINDER
3218 016726 042720 177400   BIC      #177400, (R0)+    ;RESET TRK ADRS TO 0
3219 016732 012702 001000   MOV      #512, R2         ;DATA FIELD LENGTH
3220 016736 013720 001340   2$: MOV      PATA, (R0)+    ;FILL BUFFER
3221 016742 013720 001342   MOV      PATB, (R0)+     ;FILL BUFFER
3222 016746 162702 000004   SUB      #4, R2          ;END OF DATA FIELD ?
3223 016752 001371          BNE      2$              ;NO
3224 016754 005301          DEC      R1              ;COUNT SECTORS
3225 016756 002362          BGE      1$              ;BRANCH IF NOT LAST SECTOR
3226 016760 012602          MOV      (SP)+, R2       ;RESETORE THE R2
3227 016762 005237 001212   INC      $TESTN          ;INCREMENT TEST NUMBER FOR EACH CYLINDER CHANGE
3228                                     ;GARBADE CODE FOR APT
3229 016766 000207          RTS      PC              ;EXIT
3230
3231                                     ;THIS ROUTINE UPDATES THE TRACK ADRS OF THE HEADER WORDS IN CORE
3232
3233 016770 013701 001364   UPDATK: MOV      MAXSEC, R1 ;SET UP SECTOR COUNT
3234 016774 012700 043746   MOV      #BUFF, R0       ;SET UP HEADER POINTER IN R0
3235 017000 005720          TST      (R0)+          ;POINT HEADER POINTER TO TRK - SEC ADRS
3236 017002 062710 000400   1$: ADD      #400, (R0)   ;INDEX TRK ADRS
3237 017006 062700 001004   ADD      #516., R0      ;SET UP FOR NEXT HEADER
3238 017012 005301          DEC      R1              ;COUNT SECTORS
3239 017014 002372          BGE      1$              ;BRANCH IF NOT LAST SECTOR
3240 017016 000207          RTS      PC              ;EXIT
3241
3242                                     ;ROUTINE TO CHECK FOR KW11-L OR KW11-P CLOCKS
3243
3244 017020 012737 017076 000004 ST.CLK: MOV      #STCLK1, @#ERRVEC ;SET UP VECTOR FOR P CLK
3245 017026 005037 000006   CLR      @#ERRVEC+2     ;NEW PSW
3246 017032 005777 162240   TST      @SLKCSR        ;CHECK FOR KW11-P
3247 017036 013746 001302   MOV      SLPVEC, -(SP)   ;VECTOR ADDRESS
3248 017042 012776 017226 000000   MOV      #CLOCK, @ (SP) ;SET UP KW11-P VECTOR
3249 017050 062716 000002   ADD      #2, (SP)        ;POINT TO PSW
3250 017054 012736 000300   MOV      #PR6, @ (SP)+   ;PSW - PRI 6
3251 017060 012777 162212   MOV      #-1, @SLKCSB   ;LOAD COUNTER BUFFER
3252 017066 012777 000135 162202   MOV      #135, @SLKCSR  ;SET CLK - CNT UP
3253 017074 000426          BR      STCLK3
3254 017076 062706 000004   STCLK1: ADD      #4, SP   ;RESTORE THE STACK POINTER
3255 017102 012737 017146 000004   MOV      #STCLK2, @#ERRVEC ;CHANGE ERROR VECTOR
3256 017110 005777 162172   TST      @SLKS          ;LOOK FOR KW11-L
3257 017114 013746 001310   MOV      SLLVEC, -(SP)  ;KW11-L VECTOR ADDRESS
3258 017120 012776 017226 000000   MOV      #CLOCK, @ (SP) ;SET UP KW11-L VECTOR
3259 017126 062716 000002   ADD      #2, (SP)        ;INCREMENT VECTOR ADDRESS
3260 017132 012736 000300   MOV      #PR6, @ (SP)+   ;PSW - PRI 6
3261 017136 012777 000100 162142   MOV      #100, @SLKS    ;SET KW11-L INTERRUPT ENABLE
3262 017144 000402          BR      STCLK3
3263 017146 062706 000004   STCLK2: ADD      #4, SP   ;RESTORE THE STACK POINTER
3264 017152 012737 000006 000004   STCLK3: MOV      #6, @#ERRVEC ;RESTORE THE ERROR VECTOR
3265 017160 000207          RTS      PC
3266
3267                                     ;THIS CODE PRINT THE ABORT MESSAGE AND LET O.P. RESTART
3268
3269 017162 105737 001150   RESTRT: TSTB   $AUTOB    ;RUN UNDER APT ?
3270 017166 001016          BNE      1$              ;BRANCH IF SO
3271 017170 004737 024422   JSR      PC, $TKINT     ;ENABLE TO READ

```

```

3272 017174 104401 036667      TYPE      ,MESG2      ;IMPORPER MFG INFORMATION
3273 017200 104411      RDLIN
3274 017202 012601      MOV      (SP)+,R1    ;LOCATE THE READ IN LINE
3275 017204 105711      TSTB     (R1)        ;TERMINATOR BY <CR>
3276 017206 001406      BEQ      1$          ;BRANCH IF SO
3277 017210 121127 000131      CMPB     (R1),#'Y    ;ENTER Y ?
3278 017214 001403      BEQ      1$          ;BRANCH IF SO
3279 017216 000000      HALT
3280 017220 000137 000200      JMP      @#200       ;JUMP TO 200 IF HIT START SWITCH
3281 017224 000207      1$:      RTS         PC      ;EXIT
3282
3283      ;THIS CODE SERVICES A CLOCK INTERRUPT EVERY 16MS
3284
3285 017226 012746 000020      CLOCK:  MOV      #16,-(SP) ;PUT MILLISECONDS ON THE STACK
3286 017232 004737 033056      JSR      PC,RPTMR    ;GO REPORT TIME
3287 017236 000002      RTI
3288
3289      ;ROUTINE TO INTERCEPT 'CONTROL C' TYPED DURING PARAMETER ENTRY TIME
3290
3291 017240 012706 001100      OENTER: MOV      #STACK,SP ;INITIALIZE THE STACK
3292 017244 005737 026556      1$:      TST      TRNSWT    ;ALL ACTIVITY STOPPED ?
3293 017250 001375      BNE      1$          ;BR IF NOT
3294 017252 000005      RESET
3295 017254 000137 010462      JMP      M1B         ;START AGAIN WITH DRIVE SELECTION
3296
3297      ;ROUTINE TO TYPE THE PRESENT DISK ADDRESS. ENTERED BY TYPING 'CONTROL C'
3298
3299 017260 117746 161676      NEWSVC: MOVB     @STKB,-(SP) ;READ FROM TTY BUFFER
3300 017264 042716 177600      BIC      #177,(SP)    ;STRIP PARITY
3301 017270 022627 000017      CMP      (SP)+,#15.  ;CONTROL-0 ?
3302 017274 001406      BEQ      1$          ;YES
3303 017276 005777 161660      TST      @STKB       ;CLEAR DONE BIT
3304 017302 012777 000100 161650      MOV      #100,@STKS  ;ENABLE TTY INTERRUPT
3305 017310 000002      RTI
3306 017312 004737 024422      1$:      JSR      PC,$TKINT    ;INITIAL VECTOR
3307 017316 005037 177776      TYPADR: CLR      PSW     ;SET PROCESSOR TO PRIORITY 0
3308 017322 012737 017240 001376      MOV      #OENTER,CNTLC ;CHANGE 'CONTROL C' RETURN ADDRESS
3309 017330 104401 036372      TYPE     ,ADDRI     ;'PRESENT ADDRESS IS: '
3310 017334 104401 035156      TYPE     ,C         ;'C'
3311 017340 013746 005622      MOV      FMDPB+12,-(SP) ;PUT THE CYLINDER ADDRESS ON THE STACK
3312 017344 004737 026066      JSR      PC,$SB2D    ;CONVERT IT TO DECIMAL
3313 017350 004737 026026      JSR      PC,$SUPRS   ;TYPE IT
3314 017354 104401 035166      TYPE     ,LINSR     ;SPACES
3315 017360 104401 035160      TYPE     ,T         ;'T'
3316 017364 005046      CLR      -(SP)       ;CLEAR THE STACK
3317 017366 113716 005621      MOVB     FMDPB+11,(SP) ;PUT THE TRACK ADDRESS ON THE STACK
3318 017372 004737 026066      JSR      PC,$SB2D    ;CONVERT IT TO DECIMAL
3319 017376 004737 026026      JSR      PC,$SUPRS   ;TYPE IT
3320 017402 104401 001203      TYPE     ,$CRLF     ;CR-LF
3321 017406 012737 017260 000060      MOV      @NEWSVC,@TKVEC ;RESTORE ENTRANCE TO THIS ROUTINE
3322 017414 000002      RTI
3323
3324      ;PARAMETER ENTRY ROUTINE
3325      ;CALL
3326      ;      JSR      R4,PARENT ;THE CALLING SEQ
3327      ;      TABLE ;PARAMETERS TABLE

```

```

3328 ;
3329 ;
3330 017416 010346 PARENT: MOV R3, -(SP) ; SAVE R3
3331 ; MOV #TABLE, R3 ; PARAMETER TABLE ADDRESS
3332 017420 011403 ; MOV (R4), R3 ; PARAMETER TABLE ADDRESS
3333 017422 012337 017432 1$: MOV (R3)+, 3$ ; ADDRESS OF PARAMETER NAME
3334 017426 001435 BEQ 9$ ; BR IF AT END OF TABLE
3335 017430 104401 TYPE ; TYPE THE PARAMETER NAME
3336 017432 000000 3$: .WORD 0 ; ADDRESS OF PARAMETER NAME TEXT
3337 017434 012302 MOV (R3)+, R2 ; MAXIMUM PARAMETER VALUE
3338 017436 012305 MOV (R3)+, R5 ; ADDRESS OF PARAMETER
3339 017440 011546 MOV (R5), -(SP) ; CURRENT VALUE OF PARAMETER
3340 017442 104405 TYPDS ; TYPE THE CURRENT VALUE OF THE PARAMETER
3341 017444 104401 035152 TYPE , SLASH /
3342 017450 104411 RDLIN ; READ THE KEYBOARD
3343 017452 012601 MOV (SP)+, R1 ; INPUT ASCII STRING ADDRESS
3344 017454 004537 022360 JSR R5, CK.DIG ; CHECK THE DIGIT(S)
3345 017460 017422 1$ ; CARRIAGE RETURN ONLY ENTERED
3346 017462 017522 9$ ; PERIOD ONLY ENTERED
3347 017464 017500 6$ ; ILLEGAL INPUT
3348 017466 017474 5$ ; TERMINATED WITH A CARRIAGE RETURN
3349 017470 017500 6$ ; TERMINATED WITH A " "
3350 017472 017512 7$ ; TERMINATED WITH A "' '
3351 017474 010215 5$: MOV R2, (R5) ; MOVE NEW VALUE TO PARAMETER LOCATION
3352 017476 000751 BR 1$ ; GET MORE PARAMETERS
3353 017500 104401 035713 6$: TYPE , BADENT ; 'BAD ENTRY'
3354 017504 162703 000006 SUB #6, R3 ; DECREMENT THE TABLE POINTER
3355 017510 000744 BR 1$ ; TRY AGAIN
3356 017512 010215 7$: MOV R2, (R5) ; NEW VALUE
3357 017514 000402 BR 9$ ; EXIT
3358 ; 8$: MOV #TABLE, R3 ; RELOAD THE PARAMETER TABLE ADDRESS
3359 017516 011403 8$: MOV (R4), R3 ; RELOAD THE PARAMETER TABLE ADDRESS
3360 017520 000740 BR 1$ ; TRY AGAIN
3361 017522 012603 9$: MOV (SP)+, R3 ; RESTORE R3
3362 017524 062704 000002 ADD #2, R4 ; ADJUST THE RETURN ADDRESS
3363 017530 000204 RTS R4 ; RETURN FROM THE R4
3364 ; RTS PC ; RETURN
3365 ; THIS ROUTINE LOAD THE BAD SECTOR INTO USTAB TABLE
3366 ; CALL SEQ JSR PC, RECORD
3367 ; ALL REGISTER USED ARE DESTORIED
3368 ;
3369 017532 012701 003434 RECORD: MOV #USTAB+10, R1 ; TABLE ENTRY IN R1
3370 017536 022711 177777 1$: CMP #-1, (R1) ; AN OPENING IN THE TABLE
3371 017542 001421 BEQ 3$ ; BRANCH IF IT IS
3372 017544 023711 005622 CMP FMTDPB+12, (R1) ; CYLINDER NUMBER MATCHES ?
3373 017550 001010 2$ ; BRANCH IF NOT
3374 017552 123761 005621 000003 CNPB FMTDPB+11, 3(R1) ; TRK NUMBER MATCHES ?
3375 017560 001004 2$ ; BRANCH IF NOT
3376 017562 123761 005620 000002 CNPB FMTDPB+10, 2(R1) ; SECTOR NUMBER MATCHES ?
3377 017570 001416 BEQ 4$ ; BRANCH TO EXIT, IF THE SPOT HAS RECORDED
3378 017572 062701 000004 2$: ADD #4, R1 ; INCREMENT 4 BYTES
3379 017576 022701 004424 CMP #USTAB+512., R1 ; END OF TABLE ?
3380 017602 101355 BHI 1$ ; BRANCH IF NOT
3381 017604 000411 BR 5$ ; THROW AWAY THE PACK
3382 017606 013711 005622 3$: MOV FMTDPB+12, (R1) ; LOAD THE CYLINDER #
3383 017612 113761 005621 000003 MOVB FMTDPB+11, 3(R1) ; LOAD THE TRK NUMBER

```



```

3384 017620 113761 001350 000002      MOV      SAVSEC,2(R1)      ;LOAD THE SECTOR NUMBER
3385 017626 000207                RTS      PC                ;EXIT
3386 017630 104401 001203      4$:      TYPE      ,SCLF
3387 017634 104401 036566      5$:      TYPE      ,MSG3      ;THROW AWAY THE PACK
3388 017640 000000                HALT
3389 017642 000137 000200      JMP      @#200            ;RESTART IF DESIRED ?
3390
3391                ;THIS ROUTINE SORT THE USTAB IN ASCENDING ORDER
3392                ;CALL SEQ      JSR      PC,SORT2
3393
3394 017646                SORT2:
3395                ;
3396                MOV      @USTAB+8.,R1      ;TOP POINTER
3397 017646 011501 012702 000175      MOV      (R5),R1          ;TOP POINTER
3398 017654 010103                MOV      #125.,R2        ;TOTAL 126 ELEMENTS
3399 017656 062703 000764      MOV      R1,R3           ;LOCATE THE BOTTOM POINTER
3400                ADD      #500.,R3          ;254-4) X 2
3401 017662 022711 177777      MOV      @USTAB+508.,R3  ;THE LAST ELEMENT
3402 017666 001444                CMP      #-1,(R1)        ;THE TABLE IS EMPTY ?
3403 017670 021161 000004      1$:      BEQ      4$            ;QUICK EXIT
3404 017674 101013                CMP      (R1),4(R1)      ;COMPARE THE CYLINDER NUMBER
3405 017676 103426                BHI      2$            ;SWITCH THE TWO ELEMENT
3406 017700 126161 000003 000007      BLO      3$            ;INCREMENT POINTER
3407 017706 101006                CMPB     3(R1),7(R1)     ;COPMARE THE TRK NUMBER
3408 017710 103421                BHI      2$            ;SWITCH ELEMENT
3409 017712 126161 000002 000006      BLO      3$            ;INCREMENT POINTER
3410 017720 101001                CMPB     2(R1),6(R1)     ;COMPARE THE SECTOR NUMBER
3411 017722 000414                BHI      2$            ;SWITCH ELEMENT
3412 017724 011146                BR       3$            ;INCREMENT POINTER
3413 017726 016146 000002      2$:      MOV      (R1),-(SP)    ;SAVE THE N TH BLOCK
3414 017732 016111 000004      MOV      2(R1),-(SP)    ;INTO STACK
3415 017736 016161 000006 000002      MOV      4(R1),(R1)     ;SWITCH THE N+1 BLOCK TO N BLOCK
3416 017744 012661 000006      MOV      6(R1),2(R1)
3417 017750 012661 000004      MOV      (SP)+,6(R1)    ;LOAD THE N+1 BLOCK
3418 017754 062701 000004      3$:      MOV      (SP)+,4(R1)
3419 017760 020103                ADD      #4,R1          ;UPDATE THE TOP POINTER
3420 017762 001342                CMP      R1,R3          ;REACH THE BOTTOM ?
3421 017764 005302                BNE      1$            ;BRANCH IF NOT
3422 017766 001404                DEC      R2            ;DECREMENT ONE SORT COUNT
3423 017770 162703 000004      BEQ      4$            ;BRANCH IF ALL DONE
3424                SUB      #4,R3          ;ASJUST THE BOTTOM POINTER
3425 017774 011501 011501      ;      MOV      @USTAB+8.,R1  ;RESET TOP POINTER
3426 017776 000734                MOV      (R5),R1        ;RESET THE TOP POINTER
3427 020000                BR       1$            ;BRANCH IF NOT ALL DONE
3428 020000 062705 000002      4$:      ADD      #2,R5          ;ADJUST RETURN ADDRESS
3429 020004 000205                RTS      R5            ;EXIT
3430                RTS      PC            ;EXIT
3431                ;THIS ROUTINE TEXT THE LAST TRACK   CYL 822, TRK 4
3432                ;SECTORS 0,2,4,6,8           16 BIT MFG
3433                ;SECTORS 1,3,5,7,9           18 BIT MFG
3434                ;SECTORS 10,12,14,16,.....30, 16 BIT USER
3435                ;SECTORS 11,13,15,17,.....31 18 BIT USER
3436
3437 020006 012700 043746      TEXT:  MOV      #BUFF,R0  ;BUFFER ADDRESS
3438 020012 012701 020100      MOV      #<258.*32.>,R1 ;TOTAL WORD COUNT
3439 020016 012702 177777      MOV      #-1,R2        ;SET ALL LOCATIONS TO -1

```

```

3440 020022 010220          1$:  MOV      R2,(R0)+      ;FULL THE BUFFER
3441 020024 005301          DEC      R1              ;ALL DONE ?
3442 020026 001375          BNE     1$              ;LOOP, IF NOT
3443 020030 013746 001212    MOV     $TESTN,-(SP)    ;SAVE TESTN,BEGCYL,BEGTRK
3444 020034 013746 001322    MOV     BEGCYL,-(SP)
3445 020040 013746 001326    MOV     BEGTRK,-(SP)
3446 020044 013746 001364    MOV     MAXSEC,-(SP)   ;SAVE FLAG MAXSEC AND SEC30
3447 020050 013746 001362    MOV     SEC30,-(SP)
3448 020054 012737 000037 001364    MOV     #37,MAXSEC     ;SET MAX 31 SECTORS
3449 020062 012737 177777 001362    MOV     #-1,SEC30     ;ALWAYS IN 16 BIT MODE
3450 020070 012737 001466 001322    MOV     #822,BEGCYL   ;LOAD LAST CYLINDER
3451 020076 012737 000004 001326    MOV     #4,BEGTRK     ;LAST TRACK
3452 020104 004737 016624    JSR    PC,SETHDR     ;SET UP THE HEAD FIELD IN THE BUFFER
3453 020110 012637 001362    MOV     (SP)+,SEC30   ;RESTORE SEC30 AND MAXSEC
3454 020114 012637 001364    MOV     (SP)+,MAXSEC
3455 020120 012637 001326    MOV     (SP)+,BEGTRK
3456 020124 012637 001322    MOV     (SP)+,BEGCYL
3457 020130 012637 001212    MOV     (SP)+,$TESTN
3458 020134 012701 001414    MOV     #BAD16,R1     ;LOAD SERIAL NUMBER TO EVERY SECTOR
3459 020140 012702 043752    MOV     #BUFP+4,R2   ;BUFFER LOCATION + 4
3460 020144 012703 000040    MOV     #32,R3       ;TOTAL 32 SECTORS
3461 020150 011112          MOV     (R1),(R2)    ;1 ST SN #
3462 020152 016162 000002 000002 2$:  MOV     2(R1),2(R2)  ;2 ND SN #
3463 020160 005062 000004    CLR    4(R2)        ;THIRD WORD = 0
3464 020164 005062 000006    CLR    6(R2)        ;ID = DATA PACK
3465 020170 062702 001004    ADD    #516.,R2     ;ADVANCE TO NEXT SECTOR
3466 020174 005303          DEC    R3           ;DECREMENT ONE SECTOR COUNT
3467 020176 001364          BNE    2$          ;BRANCH IF NOT DONE
3468 020200 005046          CLR    -(SP)       ;LOAD TABLE INTO SECTORS
3469 020202 005046          CLR    -(SP)       ;DUMMY PAIRS
3470 020204 005046          CLR    -(SP)
3471 020206 012746 001414    MOV     #BAD16,-(SP) ;TABLE ADDRESS
3472 020212 012746 043752    MOV     #BUFP+4,-(SP);BUFFER ADDRESS
3473 020216 012746 000005    MOV     #5,-(SP)    ;SECTOR COUNT
3474 020222 012746 002420    MOV     #BAD18,-(SP);TABLE ADDRESS
3475 020226 012746 044756    MOV     #BUFP+4+516.,-(SP);BUFFER ADDRESS
3476 020232 012746 000005    MOV     #5,-(SP)    ;SECTOR COUNT
3477 020236 012746 003424    MOV     #USTAB,-(SP);USER DEFINED TABLE
3478 020242 012746 056022    MOV     #BUFP+4+(516.*10.),-(SP);BUFFER ADDRESS
3479 020246 005737 001362    TST    SEC30
3480 020252 100402          BMI    3$          ;BRANCH IF IN 16 BIT MODE
3481 020254 012716 057026    MOV     #BUFP+4+(516.*11.) (SP);RESET THE BUFFER ADDRESS
3482 020260 012746 000013 3$:  MOV     #11,-(SP)   ;SECTOR COUNT
3483 020264 012701 004430    MOV     #USSAV,R1    ;TABLE ADDRESS IN R1
3484 020270 012702 057026    MOV     #BUFP+4+(516.*11.) R2 ;BUFFER ADDRESS IN R2
3485 020274 005737 001362    TST    SEC30       ;BRANCH IF IN 16 BIT MODE
3486 020300 100402          BMI    4$          ;BRANCH IF IN 18 BIT MODE
3487 020302 012702 056022    MOV     #BUFP+4+(516.*10.) R2 ;BUFFER ADDRESS FOR 18 BIT MODE
3488 020306 012703 000013 4$:  MOV     #11,R3      ;SECTOR COUNT IN R3
3489 020312 010105          MOV     R1,R5       ;TEMPOR STORAGE
3490 020314 012704 000400 5$:  MOV     #256.,R4    ;WORD COUNT = DATA FIELD OF ONE SECTOR
3491 020320 012122 6$:  MOV     (R1)+,(R2)+ ;LOAD TABLE INTO SECTOR DATA FIELD
3492 020322 005304          DEC    R4          ;ALL DONE ?
3493 020324 001375          BNE    6$          ;BRANCH IF NOT
3494 020326 010501          MOV     R5,R1      ;RELOAD THE TABLE ADDRESS
3495 020330 062702 001010    ADD    #520.,R2    ;SKIP ONE SECTOR

```

```

3496 020334 005303
3497 020336 001366
3498 020340 012603
3499 020342 012602
3500 020344 012601
3501 020346 010105
3502 020350 001361
3503 020352 000207
3504
3505
3506
3507
3508
3509
3510 020354 013737 005622 043736
3511 020362 001006
3512 020364 122737 000001 005621
3513 020372 103402
3514 020374 000137 016104
3515 020400 000240
3516 020402 113737 005621 043741
3517 020410 113737 001350 043740
3518 020416 053737 001412 043736
3519 020424 113737 001220 005630
3520 020432 012737 177776 005634
3521 020440 112737 000163 005632
3522 020446 012737 043736 005636
3523 020454 113737 001350 005640
3524 020462 113737 005621 005641
3525 020470 013737 005622 005642
3526 020476 052737 010000 043736
3527 020504 005737 001362
3528 020510 100403
3529 020512 042737 010000 043736
3530 020520 004037 027410
3531 020524 005630
3532 020526 000774
3533 020530 005737 005646
3534 020534 001775
3535 020536 000207
3536
3537
3538
3539
3540
3541
3542 020540 010446
3543 020542 012746 000176
3544 020546 011504
3545 020550 022714 177777
3546 020554 001420
3547 020556 023714 001366
3548 020562 001010
3549 020564 123764 001370 000003
3550 020572 001004
3551 020574 123764 001350 000002

```

```

DEC R3 ; DECREMENT ONE SECTOR COUNT
BNE 5$ ; BRANCH IF NOT DONE
MOV (SP)+,R3 ; RETRIEVE NEXT SET OF SECTOR #
MOV (SP)+,R2 ; BUFFER ADDRESS
MOV (SP)+,R1 ; TABLE ADDRESS
MOV R1,R5 ; TEMPOR STORAGE
BNE 5$ ; BRANCH IF NOT DUMMY PAIR
RTS PC ; EXIT

; THIS ROUTINE RESET:
; RESET THE BIT 14/15 OF THE 1 ST HEADER WORD OF THE BAD SPOT
; CALLING SEQ; JSR PC,RESET

RESET: MOV FMTDPB+12,RBUF ; CYLINDER ADDRESS
BNE 3$ ; BRANCH IF NOT ON TRACK 0
CMPB #1,FMTDPB+11 ; BRANCH IF NOT CYLO,TRK 0 OR 1
BLO 3$ ; BRANCH IF NOT
JMP REJCT2 ; OTHERWISE, ILLEGAL BAD SECTORS

3$: NOP
MOV B FMTDPB+11,RBUF+3 ; TRACK ADDRESS
MOV B SAVSEC,RBUF+2 ; SECTOR ADDRESS
BIS HEADX,RBUF ; SET MFG BIT,RESET USER BIT
MOV B DRIVE,FMTX ; SETUP THE DPB FOR OPERATION
MOV #2,FMTX+4 ; WORD COUNT =2
MOV B WRTHD,FMTX+2 ; WRITE HEAD AND DATA COMMAND
MOV B RBUF,FMTX+6 ; BUFFER ADDRESS
MOV B SAVSEC,FMTX+10 ; SECTOR ADDRESS
MOV B FMTDPB+11,FMTX+11 ; TRACK ADDRESS
MOV B FMTDPB+12,FMTX+12 ; CYLINDER ADDRESS
BIS #10000,RBUF ; ASSUM 16 BIT MODE
TST SEC30 ; IN 16 BIT MODE ?
BMI 1$ ; BRANCH IF IT IS
BIC #10000,RBUF ; RESET THE FMT BIT
JSR R0,RMO3 ; CALL THE DRIVER

1$: FMTX ; DPB ADDRESS
BR 1$ ; BRANCH IF QUEUE FAILS

2$: TST FMTX+16 ; DONE ?
BEQ 2$ ; BRANCH IF NOT
RTS PC ; EXIT

; THE ROUTINE INSERT:
; INSERT A BAD SECTOR ADDRESS INTO ONE OF THE USTAB,USSAV,BAD16,BAD18 TABLES
; CALLING SEQ JSR R5,INSERT
; TABLE + 8.

INSERT: MOV R4,-(SP) ; SAVE THE R4
MOV #126,-(SP) ; ENTRIES OF THE TABLE
MOV (R5),R4 ; TABLE ADDRESS + 8 BYTES
1$: CMP #-1,(R4) ; AN OPEN ENTRY ?
BEQ 3$ ; BRANCH IF SO
CMP DS.CYL,(R4) ; CHECK IF THE BAD SECTOR HAS RECORDED
BNE 2$ ; BRANCH IF NOT
CMPB DS.TRK,3(R4) ; CYL AND TRK MATCH ?
BNE 2$ ; BRANCH IF NOT
CMPB SAVSEC,2(R4) ; CYL ,TRK AND SEC MATCH ?

```

3552	020602	001415			BEQ	4\$		: BRANCH IF 50
3553	020604	005316			DEC	(SP)		: DECREMENT THE ENTRY COUNT
3554	020606	001413			BEQ	4\$		: BRANCH IF EXHAUST
3555	020610	062704	000004		ADD	#4,R4		: LOCATE THE NEXT ENTRY
3556	020614	000755			BR	1\$		: BRANCH BACK
3557	020616	013714	001366		MOV	DS.CYL,(R4)		: LOAD INTO TABLE
3558	020622	113764	001370	000003	MOVB	DS.TRK,3(R4)		: LOAD THE TRACK NUMBER
3559	020630	113764	001350	000002	MOVB	SAVSEC,2(R4)		: LOAD THE SECTOR NUMBER
3560	020636	062706	000002		ADD	#2,SP		: CLEAR UP THE STACK
3561	020642	012604			MOV	(SP)+,R4		: RESTORE R4
3562	020644	062705	000002		ADD	#2,R5		: ASJUST THE RETURN ADDRESS
3563	020650	000205			RTS	R5		: EXIT
3564	020652	104401	037631		FUNCT1: TYPE	MESG5		
3565	020656	012737	177777	001366	1\$: MOV	#-1,DS.CYL		: RESET THE INPUT VALUES
3566	020664	012737	177777	001370	MOV	#-1,DS.TRK		
3567	020672	012737	177777	001350	MOV	#-1,SAVSEC		
3568	020700	004437	017416		JSR	R4,PARENT		: ENTER THE VALUE FROM THE KEYBOARD
3569	020704	005702			TABLE2			
3570	020706	005737	001366		TST	DS.CYL		: UPDATE VALUES ?
3571	020712	100412			BMI	2\$		: BRANCH IF NOT
3572	020714	005737	001370		TST	DS.TRK		
3573	020720	100407			BMI	2\$		
3574	020722	005737	001350		TST	SAVSEC		
3575	020726	100404			BMI	2\$		
3576	020730	004537	020540		JSR	R5,INSERT		: INSERT INTO THE BAD16 TABLE
3577	020734	001424			BAD16+8.			
3578	020736	000747			1\$			: NEXT SET
3579	020740	004537	017646		2\$: JSR	R5, SORT2		: SORT THE TABLE
3580	020744	001424			BAD16+8.			
3581	020746	104401	037702		TYPE	MESG6		
3582	020752	012737	177777	001366	3\$: MOV	#-1,DS.CYL		: RESET THE INPUT VALUES
3583	020760	012737	177777	001370	MOV	#-1,DS.TRK		
3584	020766	012737	177777	001350	MOV	#-1,SAVSEC		
3585	020774	004437	017416		JSR	R4,PARENT		: INPUT VALUES FROM THE KEYBOARD
3586	021000	005726			TABLE3			: 16 BIT TABLE
3587	021002	005737	001366		TST	DS.CYL		: UPDATED VALUES ?
3588	021006	100412			BMI	4\$		: BRANCH IF NOT
3589	021010	005737	001370		TST	DS.TRK		
3590	021014	100407			BMI	4\$		
3591	021016	005737	001350		TST	SAVSEC		
3592	021022	100404			BMI	4\$		
3593	021024	004537	020540		JSR	R5,INSERT		: INSERT THE BAD SECTOR INTO TABLE BAD18
3594	021030	002430			BAD18+8.			
3595	021032	000747			BR	3\$		: NEXT SET
3596	021034	004537	017646		4\$: JSR	R5, SORT2		: SORT THE BAD18 TABLE
3597	021040	002430			BAD18+8.			
3598	021042	000207			PC			: EXIT FROM THE CALLING OF TABCAL
3599	021044	012737	003434	021214	FUNCT2: MOV	#USTAB+8., 3\$		: SETUP TABLES FOR 16 BITS MODE
3600	021052	012737	003434	021224	MOV	#USTAB+8., 5\$		
3601	021060	012737	004440	021310	MOV	#USSAV+8., 7\$		
3602	021066	012737	004440	021320	MOV	#USSAV+8., 9\$		
3603	021074	005737	001362		TST	SEC30		: IN 16 BIT MODE ?
3604	021100	100414			BMI	1\$		: BRANCH IS 50, OTHERWISE RELOAD THE TABLES
3605	021102	012737	003434	021310	MOV	#USTAB+8., 7\$		: SET UP TABLES FOR 18 BITS MODE
3606	021110	012737	003434	021320	MOV	#USTAB+8., 9\$		
3607	021116	012737	004440	021214	MOV	#USSAV+8., 3\$		

3608	021124	012737	004440	021224	MOV	#USSAV+8.,5\$	:	
3609	021132	104401	037753		1\$:	TYPE	MSG7	: MESSAGE FOR ENTER 16 BITS USR TABLE
3610	021136	012737	177777	001366	2\$:	MOV	#-1,DS.CYL	: RESET THE INPUT VALUES
3611	021144	012737	177777	001370		MOV	#-1,DS.TRK	:
3612	021152	012737	177777	001350		MOV	#-1,SAVSEC	:
3613	021160	004437	017416			JSR	R4,PARENT	: ENTER BAD SECTOR ADDRESS FROM THE KEYBOARD
3614	021164	005702				TABLE2	:	
3615	021166	005737	001366			TST	DS.CYL	: UPDATED INPUT VALUE ?
3616	021172	100412				BMI	4\$	: BRANCH IF NOT
3617	021174	005737	001370			TST	DS.TRK	:
3618	021200	100407				BMI	4\$	:
3619	021202	005737	001350			TST	SAVSEC	:
3620	021206	100404				BMI	4\$	:
3621	021210	004537	020540			JSR	R5,INSERT	: INSERT THE BAD SECTOR INTO TABLE
3622	021214	000000		3\$:		.WORD	0	: TABLE ADDRESS
3623	021216	000747				BR	2\$	: ENTER NEXT BAD SECTOR ADDRESS
3624	021220	004537	017646	4\$:		JSR	R5, SORT2	: SORT THE TABLE
3625	021224	000000		5\$:		.WORD	0	: TABLE ADDRESS
3626	021226	104401	040024			TYPE	MSG8	: MESSAGE FOR 18 BIT USR TABLE
3627	021232	012737	177777	001366	6\$:	MOV	#-1,DS.CYL	: RESET THE INPUT VALUES
3628	021240	012737	177777	001370		MOV	#-1,DS.TRK	:
3629	021246	012737	177777	001350		MOV	#-1,SAVSEC	:
3630	021254	004437	017416			JSR	R4,PARENT	: ENTER BAD SECTOR FORM KEYBOARD
3631	021260	005726				TABLE3	:	
3632	021262	005737	001366			TST	DS.CYL	: UPDATED INPUT VALUE ?
3633	021266	100412				BMI	8\$	: BRANCH IF NOT
3634	021270	005737	001370			TST	DS.TRK	: BRANCH IF NOT UPDATE
3635	021274	100407				BMI	8\$	:
3636	021276	005737	001350			TST	SAVSEC	:
3637	021302	100404				BMI	8\$	: BRANCH IF NOT UPDATED
3638	021304	004537	020540			JSR	R5,INSERT	: INSERT THE BAD SECTOR INTO TABLE
3639	021310	000000		7\$:		.WORD	0	: BRANCH TO ENTER NEXT SECTOR
3640	021312	000747				BR	6\$	: SORT THE TABLE
3641	021314	004537	017646	8\$:		JSR	R5, SORT2	: TABLE ADDRESS
3642	021320	000000		9\$:		.WORD	0	: EXIT FROM THE TABCAL CALLING
3643	021322	000207				RTS	PC	: ;SAVE R1
3644	021324	010146				MOV	R1,-(SP)	: ADDRESS OF THE TABLE
3645	021326	012701	001414			MOV	#BAD16,R1	: MSG TO TYPE THE 16 BIT MFG
3646	021332	104401	040075			TYPE	MSG9	: CR-LF FIRST
3647	021336	104401	001203	1\$:		TYPE	\$CRLF	: CALLING SEQ FOR TYPOC RT.
3648	021342	012146				MOV	(R1)+,-(SP)	: BRANCH IF END OF TABLE
3649	021344	100402				BMI	2\$	: TYPE THE VALUE
3650	021346	104402				TYPOC	:	: LOOPING BACK
3651	021350	000772				BR	1\$	: ADJUST THE RETURN ADDRESS
3652	021352	062706	000002	2\$:		ADD	#2,SP	: MSG TO TYPE 18 BIT MFG
3653	021356	104401	040135			TYPE	MSG10	: TABLE ADDRESS
3654	021362	012701	002420			MOV	#BAD18,R1	:
3655	021366	104401	001203	3\$:		TYPE	\$CRLF	: CALLING SEQ FOR TYPOC RT.
3656	021372	012146				MOV	(R1)+,-(SP)	: BRANCH IF END OF TABLE
3657	021374	100402				BMI	4\$	: TYPE THE VALUE IN OCTAL
3658	021376	104402				TYPOC	:	: LOOPING BACK
3659	021400	000772				BR	3\$	: ADJUST THE RETURN ADDRESS
3660	021402	062706	000002	4\$:		ADD	#2,SP	: RESTORE THE R1
3661	021406	012601		5\$:		MOV	(SP)+,R1	: EXIT FROM THE CALLING OF TABCAL
3662	021410	000207				RTS	PC	: ;SAVE R1
3663	021412	010146				MOV	R1,-(SP)	:

3664	021414	010246		MOV	R2, -(SP)	; SAVE R2
3665	021416	012701	003424	MOV	#USTAB, R1	; SET 16 BIT MODE TABLE
3666	021422	012702	004430	MOV	#USSAV, R2	
3667	021426	005737	001362	TST	SEC30	; 16 BIT MODE ?
3668	021432	100404		BMI	1\$	; BRANCH IF SO
3669	021434	012702	003424	MOV	#USTAB, R2	; OTHERWISE, RELOAD THE TABLE ADDRESS
3670	021440	012701	004430	MOV	#USSAV, R1	
3671	021444	104401	040175	1\$:	TYPE	, MSG11
3672	021450	104401	001203	2\$:	TYPE	, \$CRLF
3673	021454	012146		MOV	(R1)+, -(SP)	
3674	021456	100402		BMI	3\$	; BRANCH IF END OF TABLE
3675	021460	104402		TYPOC		
3676	021462	000772		BR	2\$	
3677	021464	062706	000002	3\$:	ADD	#2, SP
3678	021470	104401	040235	TYPE	, MSG12	; ADJUST STACK POINTER
3679	021474	104401	001203	4\$:	TYPE	, \$CRLF
3680	021500	012246		MOV	(R2)+, -(SP)	; MESSAGE FOR 18 BIT MODE
3681	021502	100402		BMI	5\$	; CR-LF
3682	021504	104402		TYPOC		; BRANCH IF END OF TABLE
3683	021506	000772		BR	4\$	; NEXT SECTOR ADDRESS
3684	021510	062706	000002	5\$:	ADD	#2, SP
3685	021514	012602		MOV	(SP)+, R2	; ADJUST THE STACK POINT
3686	021516	012601		MOV	(SP)+, R1	; RESTORE R2, R1
3687	021520	000207		RTS	PC	; EXIT FROM THE CALLING OF TABCAL
3688	021522	010146		FUNCT5:	MOV	R1, -(SP)
3689	021524	010246		MOV	R2, -(SP)	; SAVE R1, R2
3690	021526	005037	001420	CLR	BAD16+4.	
3691	021532	005037	001422	CLR	BAD16+6.	
3692	021536	012701	001424	MOV	#BAD16+8., R1	; START AT THE 5TH WORD
3693	021542	012702	000374	MOV	#252., R2	; WORD CTR
3694	021546	012721	177777	1\$:	MOV	#-1, (R1)+
3695	021552	005302		DEC	R2	; RESET TO -1
3696	021554	001374		BNE	1\$	; DECREMENT WORD CTR
3697	021556	005037	002424	CLR	BAD18+4.	; BRANCH IF NOT DONE
3698	021562	005037	002426	CLR	BAD18+6.	
3699	021566	012701	002430	MOV	#BAD18+8., R1	; START AT THE 5TH WORD
3700	021572	012702	000374	MOV	#252., R2	; WORD CTR
3701	021576	012721	177777	2\$:	MOV	#-1, (R1)+
3702	021602	005302		DEC	R2	; RESET TO -1
3703	021604	001374		BNE	2\$	; DECREMENT WORD CTR
3704	021606	012602		MOV	(SP)+, R2	; BRANCH IF NOT DONE
3705	021610	012601		MOV	(SP)+, R1	; RESTORE R2, R1
3706	021612	104401	040324	TYPE	, MSG14	; DONE MESSAGE
3707	021616	000207		RTS	PC	; EXIT FROM THE TABCAL CALLING
3708	021620	010146		FUNCT6:	MOV	R1, -(SP)
3709	021622	010246		MOV	R2, -(SP)	; STORE R1, R2
3710	021624	005037	003430	CLR	USTAB+4	
3711	021630	005037	003432	CLR	USTAB+6	
3712	021634	012701	003434	MOV	#USTAB+8., R1	
3713	021640	012702	000374	MOV	#252., R2	; WORD CTR
3714	021644	012721	177777	1\$:	MOV	#-1, (R1)+
3715	021650	005302		DEC	R2	; RESET TO -1
3716	021652	001374		BNE	1\$	; DECREMENT WORD CTR
3717	021654	005037	004434	CLR	USSAV+4	; BRANCH IF NOT DONE
3718	021660	005037	004436	CLR	USSAV+6	
3719	021664	012701	004440	MOV	#USSAV+8., R1	; TABLE ADDRESS

3720	021670	012702	000374			MOV	#252, R2		
3721	021674	012721	177777		2\$:	MOV	#-1, (R1)+		: RESET TO -1
3722	021700	005302				DEC	R2		: DECREMENT WORD CTR
3723	021702	001374				BNE	2\$		: BRANCH IF NOT DONE
3724	021704	012602				MOV	(SP)+, R2		: RESTORE R1, R2
3725	021706	012601				MOV	(SP)+, R1		
3726	021710	104401	040324			TYPE	MESG14		: DONE MESSAGE
3727	021714	000207				RTS	PC		: EXIT FROM THE TABCAL CALLING
3728	021716	104401	040275		FUNCT7:	TYPE	, MESG13	; MESSAGE	: ENTER SECTOR #
3729	021722	104411			1\$:	RDLIN			: READ THE SECTOR NUMBER
3730	021724	012601				MOV	(SP)+, R1		: ADDRESS OF READ-IN BUFFER
3731	021726	012702	000040			MOV	#32, R2		: MAX VALUE
3732	021732	004537	022360			JSR	R5, CK.DIG		: CHECK THE READ-IN VALUE
3733	021736	021752				2\$			
3734	021740	021752				2\$			
3735	021742	021722				1\$			
3736	021744	021752				2\$			
3737	021746	021752				2\$			
3738	021750	021752				2\$			
3739	021752	006302			2\$:	ASL	R2		: FOUND THE WORD INDEX
3740	021754	016201	006132			MOV	ADRTBL(R2), R1		: ADDRESS OF THE BUFFER
3741	021760	012702	000402			MOV	#258, R2		: WORD CTR FOR WHOLE SECTOR INCLUDING HEAD
3742	021764	104401	001203		3\$:	TYPE	, SCRLF		: CR-LF
3743	021770	012705	000010			MOV	#10, R5		: 8 WORDS ON A LINE
3744	021774	012146			4\$:	MOV	(R1)+, -(SP)		: CONTENTS OF THE BUFFER
3745	021776	104402				TYPOC			
3746	022000	104401	035166			TYPE	LINSP		: TYPE THE SPACES
3747	022004	005302				DEC	R2		: DECREMENT WORD CTR
3748	022006	001403				BEQ	5\$		: BRANCH IF FINISH
3749	022010	005305				DEC	R5		: START ANOTHER LINE ?
3750	022012	001370				BNE	4\$		: BRANCH IF NOT
3751	022014	000763				BR	3\$		: BRANCH IF SO
3752	022016	000207				PC			: EXIT FROM TABCAL CALLING
3753	022020	104401	040333		5\$:	RTS	PC		
3754	022024	012737	177777	001366	FUNCT8:	TYPE	, MESG15		
3755	022032	012737	177777	001370	1\$:	MOV	#-1, DS.CYL		: RESET ALL INPUT VALUES
3756	022040	012737	047301	001350		MOV	#-1, DS.TRK		: TRACK NUMBER
3757	022046	004437	017416			MOV	#20161, SAVSEC		: BYTE NUMBER
3758	022052	005752				JSR	R4, PARENT		: ENTER THE VALUE FROM KEYBOARD
3759	022054	005737	001366			TABLE4			
3760	022060	100455				TST	DS.CYL		: UPDATED ?
3761	022062	005737	001370			BMI	6\$		: EXIT IF NOT
3762	022066	100452				TST	DS.TRK		: BRANCH IF NOT UPDATED
3763	022070	005737	001350			BMI	6\$		
3764	022074	100447				TST	SAVSEC		: BRANCH TO EXIT IF NEG
3765	022076	022737	047301	001350		BMI	6\$		: EXIT
3766	022104	003443				CMP	#20161, SAVSEC		: TOO LARGE ?
3767	022106	010446				BLE	6\$		: BRANCH IF SO, EXIT
3768	022110	013746	001350			MOV	R4, -(SP)		: SAVE R4
3769	022114	005004				MOV	SAVSEC, -(SP)		: SAVE SAVSEC
3770	022116	023764	001350	006332	2\$:	CLR	R4		: INDEX STARTS FROM 0
3771	022124	101403				CMP	SAVSEC, BY?E16(R4)		: LOCATE THE SECTOR ADDRESS
3772	022126	062704	000002			BLOS	3\$		: BRANCH IF FOUND
3773	022132	000771				ADD	#2, R4		: TRY NEXT SECTOR
3774	022134	006204			3\$:	BR	2\$		: LOOPING
3775	022136	010437	001350			ASR	R4		: SECTOR ADDRESS
						MOV	R4, SAVSEC		: 16 BIT MODE BAD SECTOR ADD

```

3776 022142 004537 020540 JSR R5,INSERT ;INSERT INTO TABLE
3777 022146 001424 BAD16+8.
3778 022150 012637 001350 MOV (SP)+,SAVSEC ;LOCATE SECTOR FOR 18 BIT
3779 022154 005004 CLR R4 ;RESET INDEX VALUE
3780 022156 023764 001350 006432 4$: CMP SAVSEC,BYTE18(R4) ;USE TABLE 18 BIT
3781 022164 101403 BLOS 5$ ;BRANCH IF FOUND
3782 022166 062704 000002 ADD #2,R4 ;TRY NEXT SECTOR IF NOT FOUND
3783 022172 000771 BR 4$ ;LOOPIN BACK
3784 022174 006204 5$: ASR R4 ;SECTOR ADDRESS
3785 022176 010437 001350 MOV R4,SAVSEC
3786 022202 004537 020540 JSR R5,INSERT ;INSET INTO TABLE
3787 022206 002430 BAD18+8.
3788 022210 012604 MOV (SP)+,R4 ;RESTORE R4
3789 022212 000704 BR 1$ ;LOOPING BACK
3790 022214 004537 017646 6$: JSR R5,SORT2 ;SORT BOTH TABLE ANYWAY
3791 022220 001424 BAD16+8.
3792 022222 004537 017646 JSR R5,SORT2 ;
3793 022226 002430 BAD18+8.
3794 022230 000207 RTS PC ;RETURN TO SERVICE LOOP
3795
3796
3797
3798
3799
3800
3801
3802
3803
3804
3805
3806
3807

```

\*\*\*\*\*

; THIS ROUTINE IS USED TO CHECK IF AN  
ASCII CHARACTER IS A DIGIT BETWEEN 0 AND 7.

```

:CALL
:      MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
:      JSR R5,CK.OCT ;CHECK THE CHARACTER
:      RETURN1 ;CHARACTER IS NOT BETWEEN 0-7
:      RETURN2 ;CHARACTER IS IN R2 AS A
: ;OCTAL DIGIT

```

```

3808 022232 121127 000060 CK.OCT: CMPB (R1),#'0 ;LESS THAN ZERO?
3809 022236 103407 BLO 1$ ;YES -- BRANCH
3810 022240 121127 000067 CMPB (R1),#'7 ;GREATER THAN SEVEN?
3811 022244 101004 BHI 1$ ;YES -- BRANCH
3812 022246 111102 MOVB (R1),R2 ;GET THE CHARACTER
3813 022250 042702 177770 BIC #1C7,R2 ;STRIP AWAY THE ASCII
3814 022254 005725 TST (R5) ;ADJUST FOR RETURN
3815 022256 000205 1$: RTS R5 ;RETURN
3816
3817
3818
3819
3820
3821
3822
3823
3824
3825

```

; THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER  
AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.

```

:CALL
:      MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
:      JSR R5,CK.DEC ;CHECK THE CHARACTER
:      RETURN1 ;NOT BETWEEN 0 AND 9
:      RETURN2 ;BETWEEN 0 AND 9
: ;R2 = DIGIT

```

```

3826 022260 121127 000060 CK.DEC: CMPB (R1),#'0 ;LESS THAN ZERO?
3827 022264 103407 BLO 1$ ;YES -- BRANCH
3828 022266 121127 000071 CMPB (R1),#'9 ;GREATER THAN NINE?
3829 022272 101004 BHI 1$ ;YES -- BRANCH
3830 022274 111102 MOVB (R1),R2 ;GET THE CHARACTER
3831 022276 042702 BIC #'0,R2 ;STRIP AWAY THE ASCII

```



```

3832 022302 005725
3833 022304 000205
3834
3835
3836
3837
3838
3839
3840
3841
3842
3843
3844
3845
3846
3847
3848 022306 105711
3849 022310 001417
3850 022312 121127 000054
3851 022316 001413
3852 022320 121127 000056
3853 022324 001407
3854 022326 004537 022260
3855 022332 000410
3856 022334 004537 022232
3857 022340 005725
3858 022342 005725
3859 022344 005725
3860 022346 005725
3861 022350 005725
3862 022352 005201
3863 022354 011505
3864 022356 000205
3865
3866
3867
3868
3869
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879 022360 010446
3880 022362 010346
3881 022364 010246
3882 022366 005002
3883 022370 005003
3884 022372 005004
3885 022374 004537 022306
3886 022400 022474
3887 022402 022502
    
```

```

1$: TST (R5)+ ;ADJUST FOR RETURN
RTS R5 ;RETURN

: THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
: DETERMINE WHAT IT IS.
CALL
MOV #ADR,R1 ;ADDRESS OF ASCII CHARACTER
JSR R5,CK.CHR ;CHECK CHARACTER
RETURN ADR1 ;UNKNOWN CHARACTER
RETURN ADR2 ;CARRIAGE RETURN * (R1)=ADR+1
RETURN ADR3 ;COMMA * (R1)=ADR+1
RETURN ADR4 ;PERIOD * (R1)=ADR+1
RETURN ADR5 ;DIGIT BETWEEN 0 AND 7.
RETURN ADR6 ;DIGIT BETWEEN 8 AND 9.
;R2 = DIGIT * (R1)=ADR+1

CK.CHR: TSTB (R1) ;"CARRIAGE RETURN"?
BEQ 3$ YES -- BRANCH
CMPB (R1),#' , ;"COMMA"?
BEQ 2$ YES -- BRANCH
CMPB (R1),#'. ;"PERIOD"?
BEQ 1$ YES -- BRANCH
JSR R5,CK.DEC ;"DIGIT"?
BR 4$ NO -- BRANCH
JSR R5,CK.OCT ;OCTAL ?
TST (R5)+ ;DIGIT BETWEEN 8-9
TST (R5)+ ;DIGIT BETWEEN 0-7
1$: TST (R5)+ ;PERIOD
2$: TST (R5)+ ;COMMA
3$: TST (R5)+ ;CARRIAGE RETURN
INC R1 ;MOVE POINTER TO NEXT CHARACTER
4$: MOV (R5),R5 ;UNKNOWN CHARACTER
RTS R5 ;RETURN
    
```

```

: THIS ROUTINE CHECKS AN ASCII STRING FOR LEGAL
: CHARACTERS AND FORMS A DECIMAL VALUE BINARY NUMBER IN R2.
CALL
MOV #ADR,R1 ;ADDRESS OF ASCII STRING
MOV #NUM,R2 ;MAX. MAGNITUDE OF INPUT NUMBER
JSR R5,CK.DIG ;CHECK DIGITS
RETURN ADR1 ;"CR" ONLY ENTERED -- R2=0
RETURN ADR2 ;"PERIOD" ONLY ENTERED -- R2=0
RETURN ADR3 ;ILLEGAL CHARACTER OR INPUT TOO LARGE -- R2=?
RETURN ADR4 ;"CR" -- R2 = NUMBER
RETURN ADR5 ;"COMMA" -- R2 = NUMBER
RETURN ADR6 ;"PERIOD" -- R2 = NUMBER

CK.DIG: MOV R4,-(SP) ;SAVE R4
MOV R3,-(SP) ;SAVE R3
MOV R2,-(SP) ;SAVE THE MAX. SIZE ON THE STACK
CLR R2 ;START WITH 0
CLR R3
CLR R4
JSR R5,CK.CHR ;CHECK ONE CHARACTER
6$ ;ILLEGAL CHARACTER
9$ ;CARRIAGE RETURN
    
```

```

3888 022404 022474      6$      ;"
3889 022406 022476      7$      ;"
3890 022410 022414      1$      ;DIGIT 0-7
3891 022412 022414      1$      ;DIGIT 8-9
3892 022414 062705 000004 1$: ADD #4,R5 ;STEP RETURN POINTER PAST "CR" & "PERIOD" RETURNS
3893 022420 006303      2$: ASL R3 ;INPUT NUMBER *2
3894 022422 010346      MOV R3,-(SP) ;SAVE *2
3895 022424 006303      ASL R3 ;*4
3896 022426 006303      ASL R3 ;*8
3897 022430 062603      ADD (SP)+,R3 ;(*2)+(*8) = *10
3898 022432 060203      ADD R2,R3 ;UPDATE THE INPUT NUMBER
3899 022434 004537 022306 JSR R5,CK.CHR ;CHECK ONE CHARACTER
3900 022440 022500      8$      ;ILLEGAL CHARACTER
3901 022442 022464      5$      ;CARRIAGE RETURN
3902 022444 022462      4$      ;"
3903 022446 022454      3$      ;"
3904 022450 022420      2$      ;DIGIT 0-7
3905 022452 022420      2$      ;DIGIT 8-9
3906 022454 105711      3$: TSTB (R1) ;DOES A "CR" FOLLOW THE "PERIOD"
3907 022456 001010      BNE 8$ ;BR IF NOT
3908 022460 005724      TST (R4)+ ;INCREMENT THE RETURN
3909 022462 005724      4$: TST (R4)+ ;INCREMENT THE RETURN
3910 022464 005724      5$: TST (R4)+ ;INCREMENT THE RETURN
3911 022466 020316      CMP R3,(SP) ;CHECK THE MAGNITUDE OF THE NUMBER
3912 022470 002004      BGE 9$ ;BR IF ENTERED NUMBER TOO LARGE
3913 022472 000402      BR 8$ ;BYPASS INCREMENT
3914 022474 005725      6$: TST (R5)+ ;INCREMENT RETURN PAST INVALID RETURN
3915 022476 005725      7$: TST (R5)+ ;INCREMENT RETURN
3916 022500 060405      8$: ADD R4,R5 ;SETUP RETURN POINTER
3917 022502 010302      9$: MOV R3,R2 ;ENTERED VALUE
3918 022504 005726      TST (SP)+ ;CLEAN MAX. SIZE OFF OF STACK
3919 022506 012603      MOV (SP)+,R3 ;RESTORE R3
3920 022510 012604      MOV (SP)+,R4 ;RESTORE R4
3921 022512 011505      MOV (R5),R5 ;GET RETURN ADDRESS
3922 022514 000205      RTS R5 ;RETURN

```

.SBTTL MACRO ROUTINES

.SBTTL ERROR HANDLER ROUTINE

```

;*****
;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL,
;AND GO TO 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

```

```

3941 022516 104407 001372 $ERROR: CKSWR ;:TEST FOR CHANGE IN SOFT-SWR
3942 022516 010137 001372 MOV R1,DDRIVE ;:DRIVE ADDRESS IF DRIVE ERROR CALL
3943 022520 010137 001372

```

```

3944 022524 010337 001374          MOV      R3,ATTN          ;ATTENTION REGISTER CONTENTS
3945 022530 013737 005622 001366  MOV      FMTDPB+12,DS.CYL ;CURRENT CYLINDER ADDRESS
3946 022536 113737 005621 001370  MOVVB   FMTDPB+11,DS.TRK ;REQUESTED TRACK ADDRESS
3947 022544 005737 005544          TST      RM.REG+RMB A    ;NON-ZERO BUFFER ADDRESS ?
3948 022550 001406          BEQ      7$              ;BR IF NO BUFFER ADDRESS
3949 022552 013746 005544          MOV      RM.REG+RMB A,-(SP) ;BUFFER ADDRESS
3950 022556 162716 000002          SUB      #2,(SP)         ;DECREMENT THE ADDRESS
3951 022562 013637 001140          MOV      @($P)+,$GDDAT   ;GET THE BUFFER WORD WHICH DIDN'T COMPARE
3952 022566 105237 001117          7$: INCB   $ERFLG        ;SET THE ERROR FLAG
3953 022572 001775          BEQ      7$              ;DON'T LET THE FLAG GO TO ZERO
3954 022574 013777 001116 156354  MOV      $STNM,@DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
3955 022602 032777 002000 156344  BIT      #BIT10,@SWR     ;BELL ON ERROR?
3956 022610 001402          BEQ      1$              ;NO - SKIP
3957 022612 104401 001176          TYPE    $BELL           ;RING BELL
3958 022616 005237 001126          1$: INC   $ERTTL         ;COUNT THE NUMBER OF ERRORS
3959 022622 011637 001132          MOV      (SP),$ERRPC     ;GET ADDRESS OF ERROR INSTRUCTION
3960 022626 162737 000002 001132  SUB      #2,$ERRPC
3961 022634 117737 156272 001130  MOVVB   @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
3962 022642 032777 020000 156304  BIT      #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
3963 022650 001004          BNE      20$            ;;SKIP TYPEOUTS
3964 022652 004737 022752          JSR      PC,TYPERR      ;;GO TO USER ERROR ROUTINE
3965 022656 104401 001203          TYPE    ,SCRLF
3966 022662          20$:
3967 022662 122737 000001 001226  CMPB    #APTENV,$ENV     ;;RUNNING IN APT MODE
3968 022670 001007          BNE      2$              ;NO SKIP APT ERROR REPORT
3969 022672 113737 001130 022704  MOVVB   $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
3970 022700 004737 023502          JSR      PC,$ATY4       ;;REPORT FATAL ERROR TO APT
3971 022704          21$: .BYTE 0
3972 022705          .BYTE 0
3973 022706 000777          22$: BR      22$              ; APT ERROR LOOP
3974 022710 005777 156240 2$: TST     @SWR            ; HALT ON ERROR
3975 022714 100002          BPL     3$              ; SKIP IF CONTINUE
3976 022716 000000          HALT                                ; HALT ON ERROR!
3977 022720 104407          CKSWR                                ; TEST FOR CHANGE IN SOFT-SWR
3978 022722 032777 001000 156224 3$: BIT     #BIT09,@SWR    ; LOOP ON ERROR SWITCH SET?
3979 022730 001402          BEQ     4$              ; BR IF NO
3980 022732 013716 001124          MOV     $!PERR,(SP)     ; FUDGE RETURN FOR LOOPING
3981 022736 005737 001174          4$: TST     $ESCAPE      ; CHECK FOR AN ESCAPE ADDRESS
3982 022742 001402          BEQ     5$              ; BR IF NONE
3983 022744 013716 001174          MOV     $ESCAPE,(SP)   ; FUDGE RETURN ADDRESS FOR ESCAPE
3984 022750          5$:
3985 022750 000002          RTI                          ;;RETURN
3986
3987
3988          ;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
3989          ;WHICH ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR
3990          ;TABLE" ($ERRTB), AND REPORTS THE APPROPRIATE INFORMATION
3991          ;CONCERNING THE ERROR.
3992
3993 022752 104412          TYPERR: SAVREG          ;SAVE R0-R5
3994 022754 005000          CLR     R0              ;CLEAR R0 FOR ERROR NUMBER
3995 022756 113700 001130  MOVVB   $ITEMB,R0      ;ERROR NUMBER
3996 022762 005300          DEC     R0              ;FORM INDEX FOR ERROR TABLE
3997 022764 006300          ASL    R0
3998 022766 006300          ASL    R0
3999 022770 006300          ASL    R0

```

4000	022772	062700	006526	1\$:	ADD	#SERRTB,RO	;FORM ADDRESS
4001	022776	012037	023012		MOV	(RO)+,2\$	;GET ERROR MESSAGE (EM) POINTER
4002	023002	001404			BEQ	3\$	;BRANCH IF THERE ISN'T ONE
4003	023004	104401	001203		TYPE	,SCLF	; "CARRIAGE RETURN - LINE FEED
4004	023010	104401			TYPE		
4005	023012	000000		2\$:	.WORD	0	; "EM" POINTER GOES HERE
4006	023014	012037	023030	3\$:	MOV	(RO)+,4\$	; PICK UP DATA HEADER (DH) POINTER
4007	023020	001404			BEQ	5\$	; BRANCH IF NONE
4008	023022	104401	001203		TYPE	,SCLF	; CARRIAGE RETURN-LINE FEED
4009	023026	104401			TYPE		
4010	023030	000000		4\$:	.WORD	0	; "DH" POINTER GOES HERE
4011	023032	012001		5\$:	MOV	(RO)+,R1	; PICKUP DATA TABLE (DT) POINTER
4012	023034	001460			BEQ	20\$	; BRANCH IF NONE
4013	023036	005005			CLR	R5	; SET INDENT SWITCH
4014	023040	012000			MOV	(RO)+,RO	; DATA FORMAT (DF) POINTER
4015	023042	012002			MOV	(RO)+,R2	; NUMBER OF DH'S TO TYPE
4016	023044	001451			BEQ	17\$	; BRANCH IF DH NUMBER IS 0
4017	023046	005105			COM	R5	; NO INDENT
4018	023050	104401	001203		TYPE	,SCLF	; CARRIAGE RETURN-LINE FEED
4019	023054	112003		10\$:	MOV	(RO)+,R3	; NUMBER OF DATA WORDS TO TYPE
4020	023056	112004			MOV	(RO)+,R4	; AND HOW TO TYPE THEM
4021	023060	006004		11\$:	ROR	R4	; OCTAL OR DECIMAL?
4022	023062	103403			BCS	12\$	; DECIMAL--BRANCH
4023	023064	013146			MOV	2(R1)+,-(SP)	; SAVE 2(R1)+ FOR TYPEOUT
4024	023066	104402			TYPOC		; GO TYPE--OCTAL ASCII(ALL DIGITS)
4025	023070	000402			BR	13\$	
4026	023072			12\$:			
4027	023072	013146			MOV	2(R1)+,-(SP)	; SAVE 2(R1)+ FOR TYPEOUT
4028	023074	104405			TYPDS		; GO TYPE--DECIMAL ASCII WITH SIGN
4029	023076	005303		13\$:	DEC	R3	; MORE NUMBERS TO TYPE?
4030	023100	001403			BEQ	14\$	; NO--BRANCH
4031	023102	104401	035166		TYPE	,LINS	; YES--TYPE SEPERATORS
4032	023106	000764			BR	11\$	; LOOP
4033	023110	005302		14\$:	DEC	R2	; MORE DH'S?
4034	023112	003431			BLE	20\$	; NO--BRANCH
4035	023114	104401	001203		TYPE	,SCLF	; YES--START A NEW LINE
4036	023120	005760	000002		TST	2(RO)	; ONLY A 'DH' IN THIS REQUEST ?
4037	023124	001404			BEQ	15\$	; BR IF YES - BYPASS THE INDENT
4038	023126	005105			COM	R5	; INDENT?
4039	023130	001002			BNE	15\$	; NO--BRANCH
4040	023132	104401	035166		TYPE	,LINS	; YES--TYPE SPACES
4041	023136	012037	023144	15\$:	MOV	(RO)+,16\$	; GET NEXT DH
4042	023142	104401			TYPE		; AND TYPE IT
4043	023144	000000		16\$:	.WORD	0	; DH POINTER GOES HERE
4044	023146	005710			TST	(RO)	; TYPE A 'DT' ?
4045	023150	001003			BNE	21\$	; BR IF A 'DT'
4046	023152	062700	000004		ADD	#4,RO	; INCREMENT THE 'DF' POINTER
4047	023156	000754			BR	14\$	; SEE IF END OF 'DF' BLOCK
4048	023160	104401	001203	21\$:	TYPE	,SCLF	; CARRIAGE RETURN-LINE FEED
4049	023164	005705			TST	R5	; INDENT?
4050	023166	001332			BNE	10\$	; NO--BRANCH
4051	023170	104401	035166	17\$:	TYPE	,LINS	; YES--TYPE SPACES
4052	023174	000727			BR	10\$	; LOOP
4053	023176	104413		20\$:	RESREG		; RESTORE RO-R5
4054	023200	000207			RTS	PC	; RETURN
4055							

.SBTTL TYPE ROUTINE

4056  
4057  
4058  
4059  
4060  
4061  
4062  
4063  
4064  
4065  
4066  
4067  
4068  
4069  
4070  
4071  
4072  
4073  
4074  
4075  
4076  
4077  
4078  
4079  
4080  
4081  
4082  
4083  
4084  
4085  
4086  
4087  
4088  
4089  
4090  
4091  
4092  
4093  
4094  
4095  
4096  
4097  
4098  
4099  
4100  
4101  
4102  
4103  
4104  
4105  
4106  
4107  
4108  
4109  
4110  
4111

023202 105737 001173  
023206 100002  
023210 000000  
023212 000430  
023214 010046  
023216 017600 000002  
023222 122737 000001 001226  
023230 001011  
023232 132737 000100 001227  
023240 001405  
023242 010037 023252  
023246 004737 023472  
023252 000000  
023254 132737 000040 001227  
023262 001003  
023264 112046  
023266 001005  
023270 005726  
023272 012600  
023274 062716 000002  
023300 000002  
023302 122716 000011  
023306 001430  
023310 122716 000200  
023314 001006  
023316 005726  
023320 104401  
023322 001203  
023324 105037 023460  
023330 000755  
023332 004737 023414  
023336 123726 001172  
023342 001350  
023344 013746 001170  
023350 105366 000001  
023354 002770  
023356 004737 023414  
023362 105337 023460

```

*****
*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
*
STYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
        BPL       1$          BR IF YES
        HALT      3$          HALT HERE IF NO TERMINAL
        BR        3$          LEAVE
        MOV       RO, -(SP)    SAVE RO
        MOV       22(SP), RO  GET ADDRESS OF ASCIZ STRING
        CMPB     #APTENV, $ENV RUNNING IN APT MODE
        BNE      62$          NO, GO CHECK FOR APT CONSOLE
        BITB     #APTPOOL, $ENVM SPPOOL MESSAGE TO APT
        BEQ      62$          NO, GO CHECK FOR CONSOLE
        MOV      RO, 61$      SETUP MESSAGE ADDRESS FOR APT
        JSR     PC, $ATY3     SPOOL MESSAGE TO APT
        .WORD    0           MESSAGE ADDRESS
        BITB     #APTCSUP, $ENVM APT CONSOLE SUPPRESSED
        BNE      60$          YES, SKIP TYPE OUT
        MOVB     (RO)+, -(SP) PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE      4$          BR IF IT ISN'T THE TERMINATOR
        TST     (SP)+        IF TERMINATOR POP IT OFF THE STACK
        MOV      (SP)+, RO    RESTORE RO
        ADD     #2, (SP)     ADJUST RETURN PC
        RTI                      RETURN
        CMPB     #HT, (SP)    ;; BRANCH IF <HT>
        BEQ      8$
        CMPB     #CRLF, (SP) ;; BRANCH IF NOT <CRLF>
        BNE      5$
        TST     (SP)+        ;; POP <CR><LF> EQUIV
        TYPE     $CRLF        ;; TYPE A CR AND LF
        CLRB     $CHARCNT    ;; CLEAR CHARACTER COUNT
        BR       2$          GET NEXT CHARACTER
        JSR     PC, $TYPEC    GO TYPE THIS CHARACTER
        CMPB     $FILLC, (SP)+ IS IT TIME FOR FILLER CHARS.?
        BNE      2$          IF NO GO GET NEXT CHAR.
        MOV      $NULL, -(SP) GET # OF FILLER CHARS. NEEDED
                          AND THE NULL CHAR.
        DECB     1(SP)       DOES A NULL NEED TO BE TYPED?
        BLT     6$          BR IF NO--GO POP THE NULL OFF OF STACK
        JSR     PC, $TYPEC    GO TYPE A NULL
        DECB     $CHARCNT    ;; DO NOT COUNT AS A COUNT

```

```

4112 023366 000770          BR      7$          ;;LOOP
4113
4114          ;HORIZONTAL TAB PROCESSOR
4115
4116 023370 112716 000040      8$:      MOVB      #' (SP)          ;; REPLACE TAB WITH SPACE
4117 023374 004737 023414      9$:      JSR      PC,$TYPEC          ;; TYPE A SPACE
4118 023400 132737 000007 023460      BITB      #7,$SCHARCNT          ;; BRANCH IF NOT AT
4119 023406 001372          BNE      9$          ;; TAB STOP
4120 023410 005726          TST      (SP)+          ;; POP SPACE OFF STACK
4121 023412 000724          BR      2$          ;; GET NEXT CHARACTER
4122 023414 105777 155544      $TYPEC: TSTB      $STPS          ;; WAIT UNTIL PRINTER IS READY
4123 023420 100375          BPL      $TYPEC
4124 023422 116677 000002 155536      MOVB      2(SP),$STPB          ;; LOAD CHAR TO BE TYPED INTO DATA REG.
4125 023430 122766 000015 000002      CMPB      #CR,2(SP)          ;; IS CHARACTER A CARRIAGE RETURN?
4126 023436 001003          BNE      1$          ;; BRANCH IF NO
4127 023440 105037 023460      CLRB      $SCHARCNT          ;; YES--CLEAR CHARACTER COUNT
4128 023444 000406          BR      $TYPEX          ;; EXIT
4129 023446 122766 000012 000002 1$:      CMPB      #LF,2(SP)          ;; IS CHARACTER A LINE FEED?
4130 023454 001402          BEQ      $TYPEX          ;; BRANCH IF YES
4131 023456 105227          INCB      (PC)+          ;; COUNT THE CHARACTER
4132 023460 000000      $SCHARCNT: WORD D          ;; CHARACTER COUNT STORAGE
4133 023462 000207      $TYPEX: RTS      PC
4134
4135
4136          .SBTTL  APT COMMUNICATIONS ROUTINE
4137
4138          ;*****
4139
4140 023464 112737 000001 023730  $ATY1:  MOVB      #1,$FFLG          ;; TO REPORT FATAL ERROR
4141 023472 112737 000001 023726  $ATY3:  MOVB      #1,$MFLG          ;; TO TYPE A MESSAGE
4142 023500 000403          BR      $ATYC
4143 023502 112737 000001 023730  $ATY4:  MOVB      #1,$FFLG          ;; TO ONLY REPORT FATAL ERROR
4144 023510
4145 023510 010046          MOV      R0,-(SP)          ;; PUSH R0 ON STACK
4146 023512 010146          MOV      R1,-(SP)          ;; PUSH R1 ON STACK
4147 023514 105737 023726          TSTB      $MFLG          ;; SHOULD TYPE A MESSAGE?
4148 023520 001450          BEQ      5$          ;; IF NOT: BR
4149 023522 122737 000001 001226      CMPB      #APTENV,$ENV          ;; OPERATING UNDER APT?
4150 023530 001031          BNE      3$          ;; IF NOT: BR
4151 023532 132737 000100 001227      BITB      #APTSPOOL,$ENVM          ;; SHOULD SPOOL MESSAGES?
4152 023540 001425          BEQ      3$          ;; IF NOT: BR
4153 023542 017600 000004          MOV      @4(SP),R0          ;; GET MESSAGE ADDR.
4154 023546 062766 000002 000004      ADD      #2,4(SP)          ;; BUMP RETURN ADDR.
4155 023554 005737 001206      1$:      TST      $MSGTYPE          ;; SEE IF DONE W/ LAST XMISSION?
4156 023560 001375          BNE      1$          ;; IF NOT: WAIT
4157 023562 010037 001222      MOV      R0,$MSGAD          ;; PUT ADDR IN MAILBOX
4158 023566 105720      2$:      TSTB      (R0)+          ;; FIND END OF MESSAGE
4159 023570 001376          BNE      2$
4160 023572 163700 001222      SUB      $MSGAD,R0          ;; SUB START OF MESSAGE
4161 023576 006200          ASR      R0          ;; GET MESSAGE LNTH IN WORDS
4162 023600 010037 001224      MOV      R0,$MSGLEN          ;; PUT LENGTH IN MAILBOX
4163 023604 012737 000004 001206      MOV      #4,$MSGTYPE          ;; TELL APT TO TAKE MSG.
4164 023612 000413          BR      5$
4165 023614 017637 000004 023640 3$:      MOV      @4(SP),4$          ;; PUT MSG ADDR IN JSR LINKAGE
4166 023622 062766 000002 000004      ADD      #2,4(SP)          ;; BUMP RETURN ADDRESS
4167 023630 013746 177776          MOV      177776,-(SP)          ;; PUSH 177776 ON STACK

```

4168 023634 004737 023202  
 4169 023640 000000  
 4170 023642  
 4171 023642 105737 023730  
 4172 023646 001416  
 4173 023650 005737 001226  
 4174 023654 001413  
 4175 023656 005737 001206  
 4176 023662 001375  
 4177 023664 017637 000004 001210  
 4178 023672 062766 000002 000004  
 4179 023700 005237 001206  
 4180 023704 105037 023730  
 4181 023710 105037 023727  
 4182 023714 105037 023726  
 4183 023720 012601  
 4184 023722 012600  
 4185 023724 000207  
 4186 023726 000  
 4187 023727 000  
 4188 023730 000  
 4189 023732  
 4190 000200  
 4191 000001  
 4192 000100  
 4193 000040  
 4194  
 4195  
 4196  
 4197  
 4198  
 4199  
 4200  
 4201  
 4202  
 4203  
 4204  
 4205  
 4206  
 4207  
 4208  
 4209  
 4210  
 4211  
 4212  
 4213  
 4214  
 4215  
 4216  
 4217  
 4218  
 4219  
 4220 023732 017646 000000  
 4221 023736 116637 000001 024155  
 4222 023744 112637 024157  
 4223 023750 062716 000002

```

JSR PC,STYPE ;;CALL TYPE MACRO
4S: .WORD 0
5S:
10S: TSTB SFFLG ;; SHOULD REPORT FATAL ERROR?
      BEQ 12S ;; IF NOT: BR
      TST SENV ;; RUNNING UNDER APT?
      BEQ 12S ;; IF NOT: BR
11S: TST SMSGTYPE ;; FINISHED LAST MESSAGE?
      BNE 11S ;; IF NOT: WAIT
      MOV #4(SP),SFATAL ;; GET ERROR #
      ADD #24(SP) ;; BUMP RETURN ADDR.
      INC SMSGTYPE ;; TELL APT TO TAKE ERROR
12S: CLRB SFFLG ;; CLEAR FATAL FLAG
      CLRB SLFLG ;; CLEAR LOG FLAG
      CLRB SMFLG ;; CLEAR MESSAGE FLAG
      MOV (SP)+,R1 ;; POP STACK INTO R1
      MOV (SP)+,R0 ;; POP STACK INTO R0
      RTS PC ;; RETURN
SMFLG: .BYTE 0 ;; MESSG. FLAG
SLFLG: .BYTE 0 ;; LOG FLAG
SFFLG: .BYTE 0 ;; FATAL FLAG
      .EVEN
APTSIZE=200
APTENV=001
APTSPool=100
APTCSUP=040

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOS  ;; CALL FOR TYPEOUT
*   .BYTE  N              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*   .BYTE  M              ;; M=1 OR 0
*                               ;; 1=TYPE LEADING ZEROS
*                               ;; 0=SUPPRESS LEADING ZEROS
*
*STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*STYPOS OR STYPOC
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPON  ;; CALL FOR TYPEOUT
*
*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*   MOV     NUM,-(SP)      ;; NUMBER TO BE TYPED
*   TYPOC  ;; CALL FOR TYPEOUT
*
STYPOS: MOV     #2(SP),-(SP)  ;; PICKUP THE MODE
        MOVB   1(SP),%OFILL  ;; LOAD ZERO FILL SWITCH
        MOVB   (SP)+,%SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
        ADD    #2,(SP)      ;; ADJUST RETURN ADDRESS

```

4224	023754	000406			BR	\$TYPON		
4225	023756	112737	000001	024155	\$TYPOC: MOV	#1,\$OFILL	::	SET THE ZERO FILL SWITCH
4226	023764	112737	000006	024157	MOV	#6,\$OMODE+1	::	SET FOR SIX(6) DIGITS
4227	023772	112737	000005	024154	\$TYPON: MOV	#5,\$OCNT	::	SET THE ITERATION COUNT
4228	024000	010346			MOV	R3,-(SP)	::	SAVE R3
4229	024002	010446			MOV	R4,-(SP)	::	SAVE R4
4230	024004	010546			MOV	R5,-(SP)	::	SAVE R5
4231	024006	113704	024157		MOV	\$OMODE+1,R4	::	GET THE NUMBER OF DIGITS TO TYPE
4232	024012	005404			NEG	R4		
4233	024014	062704	000006		ADD	#6,R4	::	SUBTRACT IT FOR MAX. ALLOWED
4234	024020	110437	024156		MOV	R4,\$OMODE	::	SAVE IT FOR USE
4235	024024	113704	024155		MOV	\$OFILL,R4	::	GET THE ZERO FILL SWITCH
4236	024030	016605	000012		MOV	12(SP),R5	::	PICKUP THE INPUT NUMBER
4237	024034	005003			CLR	R3	::	CLEAR THE OUTPUT WORD
4238	024036	006105			1\$: ROL	R5	::	ROTATE MSB INTO "C"
4239	024040	000404			BR	3\$	::	GO DO MSB
4240	024042	006105			2\$: ROL	R5	::	FORM THIS DIGIT
4241	024044	006105			ROL	R5		
4242	024046	006105			ROL	R5		
4243	024050	010503			MOV	R5,R3		
4244	024052	006103			3\$: ROL	R3	::	GET LSB OF THIS DIGIT
4245	024054	105337	024156		DECB	\$OMODE	::	TYPE THIS DIGIT?
4246	024060	100016			BPL	7\$	::	BR IF NO
4247	024062	042703	177770		BIC	#177770,R3	::	GET RID OF JUNK
4248	024066	001002			BNE	4\$	::	TEST FOR 0
4249	024070	005704			TST	R4	::	SUPPRESS THIS 0?
4250	024072	001403			BEQ	5\$	::	BR IF YES
4251	024074	005204			4\$: INC	R4	::	DON'T SUPPRESS ANYMORE 0'S
4252	024076	052703	000060		BIS	#'0,R3	::	MAKE THIS DIGIT ASCII
4253	024102	052703	000040		5\$: BIS	#' R3	::	MAKE ASCII IF NOT ALREADY
4254	024106	110337	024152		MOV	R3,\$\$	::	SAVE FOR TYPING
4255	024112	104401	024152		TYPE	8\$	::	GO TYPE THIS DIGIT
4256	024116	105337	024154		7\$: DECB	\$OCNT	::	COUNT BY 1
4257	024122	003347			BGT	2\$	::	BR IF MORE TO DO
4258	024124	002402			BLT	6\$	::	BR IF DONE
4259	024126	005204			INC	R4	::	INSURE LAST DIGIT ISN'T A BLANK
4260	024130	000744			BR	2\$	::	GO DO THE LAST DIGIT
4261	024132	012605			6\$: MOV	(SP)+,R5	::	RESTORE R5
4262	024134	012604			MOV	(SP)+,R4	::	RESTORE R4
4263	024136	012603			MOV	(SP)+,R3	::	RESTORE R3
4264	024140	016666	000002	000004	MOV	2(SP),4(SP)	::	SET THE STACK FOR RETURNING
4265	024146	012616			MOV	(SP)+,(SP)		
4266	024150	000002			RTI		::	RETURN
4267	024152	000			8\$: .BYTE	0	::	STORAGE FOR ASCII DIGIT
4268	024153	000			.BYTE	0	::	TERMINATOR FOR TYPE ROUTINE
4269	024154	000			\$OCNT: .BYTE	0	::	OCTAL DIGIT COUNTER
4270	024155	000			\$OFILL: .BYTE	0	::	ZERO FILL SWITCH
4271	024156	000000			\$OMODE: .WORD	0	::	NUMBER OF DIGITS TO TYPE

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

4279



```
4280 ;#REPLACED WITH SPACES.
4281 ;#CALL:
4282 ;#      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
4283 ;#      TYPDS      ;;GO TO THE ROUTINE
4284
4285 $TYPDS:
4286      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
4287      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
4288      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
4289      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
4290      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
4291      MOV      #20200,-(SP)   ;;SET BLANK SWITCH AND SIGN
4292      MOV      20(SP),R5     ;;GET THE INPUT NUMBER
4293      BPL      1$           ;;BR IF INPUT IS POS.
4294      NEG      R5           ;;MAKE THE BINARY NUMBER POS.
4295      MOVVB   #'-,1(SP)     ;;MAKE THE ASCII NUMBER NEG.
4296      CLR      R0           ;;ZERO THE CONSTANTS INDEX
4297      MOV      #SDBLK,R3     ;;SETUP THE OUTPUT POINTER
4298      MOVVB   #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
4299      CLR      R2           ;;CLEAR THE BCD NUMBER
4300      MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
4301      SUB      R1,R5         ;;FORM THIS BCD DIGIT
4302      BLT     4$           ;;BR IF DONE
4303      INC      R2           ;;INCREASE THE BCD DIGIT BY 1
4304      BR      3$
4305      ADD      R1,R5         ;;ADD BACK THE CONSTANT
4306      TST     R2           ;;CHECK IF BCD DIGIT=0
4307      BNE     5$           ;;FALL THROUGH IF 0
4308      TSTB   (SP)          ;;STILL DOING LEADING 0'S?
4309      BMI     7$           ;;BR IF YES
4310      ASLB   (SP)          ;;MSD?
4311      BCC     6$           ;;BR IF NO
4312      MOVVB  1(SP),-1(R3)   ;;YES--SET THE SIGN
4313      BIS    #'0,R2        ;;MAKE THE BCD DIGIT ASCII
4314      BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4315      MOVVB  R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4316      TST    (R0)+        ;;JUST INCREMENTING
4317      CMP    R0,#10       ;;CHECK THE TABLE INDEX
4318      BLT    2$           ;;GO DO THE NEXT DIGIT
4319      BGT    8$           ;;GO TO EXIT
4320      MOV    R5,R2         ;;GET THE LSD
4321      BR     6$           ;;GO CHANGE TO ASCII
4322      TSTB  (SP)+         ;;WAS THE LSD THE FIRST NON-ZERO?
4323      BPL   9$           ;;BR IF NO
4324      MOVVB -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
4325      CLRB  (R3)          ;;SET THE TERMINATOR
4326      MOV   (SP)+,R5      ;;POP STACK INTO R5
4327      MOV   (SP)+,R3      ;;POP STACK INTO R3
4328      MOV   (SP)+,R2      ;;POP STACK INTO R2
4329      MOV   (SP)+,R1      ;;POP STACK INTO R1
4330      MOV   (SP)+,R0      ;;POP STACK INTO R0
4331      TYPE  $SDBLK        ;;NOW TYPE THE NUMBER
4332      MOV   2(SP),4(SP)   ;;ADJUST THE STACK
4333      MOV   (SP)+,(SP)
4334      RTI
4335 $DTBL: 10000.

020200
000020
000055 000001
024374
000040
024364
177777
000060
000040
000010
177777 177776
024374
000002 000004
```

4336 024366 001750  
 4337 024370 000144  
 4338 024372 000012  
 4339 024374 000004  
 4340  
 4341  
 4342  
 4343  
 4344  
 4345 024404 000000  
 4346 024406 000000  
 4347 024410 000000  
 4348 024412 000007  
 4349 024421  
 4350 024422  
 4351  
 4352  
 4353  
 4354  
 4355  
 4356  
 4357  
 4358  
 4359  
 4360 024422 005037 024404  
 4361 024426 012737 024412 024406  
 4362 024434 013737 024406 024410  
 4363 024442 012737 024472 000060  
 4364 024450 012737 000200 000062  
 4365 024456 005777 154500  
 4366 024462 012777 000100 154470  
 4367 024470 000207  
 4368  
 4369  
 4370  
 4371  
 4372  
 4373  
 4374  
 4375  
 4376 024472 117746 154464  
 4377 024476 042716 177600  
 4378 024502 021627 000003  
 4379 024506 001007  
 4380 024510 104401 025613  
 4381 024514 004737 024422  
 4382 024520 005726  
 4383 024522 000177 154650  
 4384 024526 021627 000007  
 4385 024532 001004  
 4386 024534 022737 000176 001154  
 4387 024542 001500  
 4388  
 4389 024544  
 4390 024544 022737 000007 024404  
 4391 024552 001004

```

1000.
100.
10.
SDBLK: .BLKW 4

.SBTTL TTY INPUT ROUTINE

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

; *TK INITIALIZE ROUTINE
; *THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
; *SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
; *CALL:
; * JSR PC,$TKINT
; * RETURN
$TKINT: CLR $TKCNT ;: CLEAR COUNT OF ITEMS IN QUEUE
MOV $TKQSRT,$TKQIN ;: MOVE THE STARTING ADDRESS OF THE
MOV $TKQIN,$TKQOUT ;: QUEUE INTO THE INPUT & OUTPUT POINTERS.
MOV $TKSRV,$TKVEC ;: INITIALIZE THE KEYBOARD VECTOR
MOV #200,$TKVEC+2 ;: "BR" LEVEL 4
TST $TKB ;: CLEAR DONE FLAG
MOV #100,$TKS ;: ENABLE TTY KEYBOARD INTERRUPT
RTS PC ;: RETURN TO CALLER

; *TK SERVICE ROUTINE
; *THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
; *BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
; *IT IN THE QUEUE.
; *IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
; *UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (↑CNTLC)
$TKSRV: MOVB $TKB,-(SP) ;: PICKUP THE CHARACTER
BIC #↑C177,(SP) ;: STRIP THE JUNK
CMP (SP),#3 ;: IS IT A CONTROL C?
BNE 1$ ;: BRANCH IF NO
TYPE $CNTLC ;: TYPE A CONTROL-C (↑C)
JSR PC,$TKINT ;: INIT THE KEYBOARD
TST (SP)+ ;: CLEAN UP STACK
JMP ↑CNTLC ;: CONTROL C RESTART
1$: CMP (SP),#7 ;: IS IT A CONTROL G?
BNE 2$ ;: BRANCH IF NO
CMP #SWREG,SWR ;: IS SOFT-SWR SELECTED?
BEQ 6$ ;: GO TO SWR CHANGE

2$: CMP #7,$TKCNT ;: IS THE QUEUE FULL?
BNE 3$ ;: BRANCH IF NO

```

```

4392 024554 104401 001176          TYPE      $BELL          ;;RING THE TTY BELL
4393 024560 005726          TST        (SP)+        ;;CLEAN CHARACTER OFF OF STACK
4394 024562 000451          BR         5$           ;;EXIT
4395 024564 021627 000023      3$: CMP      (SP),#23    ;;IS IT A CONTROL-S?
4396 024570 001021          BNE       32$          ;;BRANCH IF NO
4397 024573 005077 154362      CLR       @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
4398 024576 005726          TST        (SP)+        ;;CLEAN CHAR OFF STACK
4399 024600 105777 154354      31$: TSTB      @STKS        ;;WAIT FOR A CHAR
4400 024604 100375          BPL       31$         ;;LOOP UNTIL ITS THERE
4401 024606 117746 154350      MOVB     @STKB, -(SP)   ;;GET THE CHARACTER
4402 024612 042716 177600      BIC      #1C17?, (SP)  ;;MAKE IT 7-BIT ASCII
4403 024616 022627 000021      CMP      (SP)+, #21    ;;IS IT A CONTROL-Q?
4404 024622 001366          BNE       31$         ;;BRANCH IF NO
4405 024624 012777 000100 154326      MOV      #100, @STKS   ;;REENABLE TTY KEYBOARD INTERRUPTS
4406 024632 000002          RTI                    ;;RETURN
4407 024634 005237 024404      32$: INC      $TKCNT     ;;COUNT THIS CHARACTER
4408 024640 021627 000140      CMP      (SP), #140    ;;IS IT UPPER CASE?
4409 024644 002405          BLT      4$           ;;BRANCH IF YES
4410 024646 021627 000175      CMP      (SP), #175    ;;IS IT A SPECIAL CHAR?
4411 024652 003002          BGT      4$           ;;BRANCH IF YES
4412 024654 042716 000040      BIC      #40, (SP)     ;;MAKE IT UPPER CASE
4413 024660 112677 177522      4$: MOVB     (SP)+, @STKQIN ;;AND PUT IT IN QUEUE
4414 024664 005237 024406      INC      $TKQIN        ;;UPDATE THE POINTER
4415 024670 023727 024406 024421      CMP      $TKQIN, @STKQEND ;;GO OFF THE END?
4416 024676 001003          BNE      5$           ;;BRANCH IF NO
4417 024700 012737 024412 024406      MOV      @STKQSR, $TKQIN ;;RESET THE POINTER
4418 024706 000002      5$: RTI                    ;;RETURN

```

```

4419
4420
4421 ;;*****
4422 ;;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4423 ;;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4424 ;;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
4425 ;;CALL WHEN OPERATING IN TTY INTERRUPT MODE.

```

```

4425 024710 022737 000176 001154 $CKSWR: CMP      @SWREG, SWR    ;;IS THE SOFT-SWR SELECTED
4426 024716 001124          BNE      15$         ;;EXIT IF NOT
4427 024720 105777 154234          TSTB     @STKS        ;;IS A CHAR WAITING?
4428 024724 100121          BPL      15$         ;;IF NOT, EXIT
4429 024726 117746 154230      MOVB     @STKB, -(SP)  ;;YES
4430 024732 042716 177600      BIC      #1C17?, (SP)  ;;MAKE IT 7-BIT ASCII
4431 024736 021627 000007      CMP      (SP), #?     ;;IS IT A CONTROL-G?
4432 024742 001300          BNE      2$           ;;IF NOT, PUT IT IN THE TTY QUEUE
4433
4434
4435
4436
4437
4438

```

```

4439 024744 123727 001150 000001 6$: CMPB     $AUTOB, #1   ;;ARE WE RUNNING IN AUTO-MODE?
4440 024752 001674          BEQ      2$           ;;BRANCH IF YES
4441 024754 005726          TST      (SP)+        ;;CLEAR CONTROL-G OFF STACK
4442 024756 004737 024422      JSR      PC, $TKINT   ;;FLUSH THE TTY INPUT QUEUE
4443 024762 005077 154172      CLR      @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
4444 024766 112737 000001 001151      MOVB     #1, $INTAG    ;;SET INTERRUPT MODE INDICATOR
4445
4446 024774 104401 025625      $GTSWR: TYPE      , $CNTLG  ;;ECHO THE CONTROL-G (!G)
4447 025000 104401 025632      TYPE      , $MSWR     ;;TYPE CURRENT CONTENTS

```

4448	025004	013746	000176		MOV	SWREG, -(SP)	:: SAVE SWREG FOR TYPEOUT
4449	025010	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
4450	025012	104401	025643		TYPE	, \$MNEW	:: PROMPT FOR NEW SWR
4451	025016	005046		19\$:	CLR	-(SP)	:: CLEAR COUNTER
4452	025020	005046			CLR	-(SP)	:: THE NEW SWR
4453	025022	105777	154132	7\$:	TSTB	\$STKS	:: CHAR THERE?
4454	025026	100375			BPL	7\$	:: IF NOT TRY AGAIN
4455							
4456	025030	117746	154126		MOVB	\$STKB, -(SP)	:: PICK UP CHAR
4457	025034	042716	177600		BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
4458							
4459	025040	021627	000003		CMP	(SP), #3	:: IS IT A CONTROL-C?
4460	025044	001015			BNE	9\$	:: BRANCH IF NOT
4461	025046	104401	025613		TYPE	\$CNTLC	:: YES, ECHO CONTROL-C (↑C)
4462	025052	062706	000006		ADD	#6, SP	:: CLEAN UP STACK
4463	025056	123727	001151	000001	CMPB	\$INTAG, #1	:: REENABLE TTY KEYBOARD INTERRUPTS?
4464	025064	001003			BNE	8\$	:: BRANCH IF NO
4465	025066	012777	000100	154064	MOV	#100, \$STKS	:: ALLOW TTY KEYBOARD INTERRUPTS
4466	025074	000177	154276	8\$:	JMP	\$CNTLC	:: CONTROL-C RESTART
4467							
4468							
4469	025100	021627	000025	9\$:	CMP	(SP), #25	:: IS IT A CONTROL-U?
4470	025104	001005			BNE	10\$	:: BRANCH IF NOT
4471	025106	104401	025620		TYPE	\$CNTLU	:: YES, ECHO CONTROL-U (↑U)
4472	025112	062706	000006	20\$:	ADD	#6, SP	:: IGNORE PREVIOUS INPUT
4473	025116	000737			BR	19\$	:: LET'S TRY IT AGAIN
4474							
4475							
4476	025120	021627	000015	10\$:	CMP	(SP), #15	:: IS IT A <CR>?
4477	025124	001022			BNE	16\$	:: BRANCH IF NO
4478	025126	005766	000004		TST	4(SP)	:: YES, IS IT THE FIRST CHAR?
4479	025132	001403			BEQ	11\$	:: BRANCH IF YES
4480	025134	016677	000002	154012	MOV	2(SP), \$SWR	:: SAVE NEW SWR
4481	025142	062706	000006	11\$:	ADD	#6, SP	:: CLEAN UP STACK
4482	025146	104401	001203	14\$:	TYPE	\$CRLF	:: ECHO <CR> AND <LF>
4483	025152	123727	001151	000001	CMPB	\$INTAG, #1	:: RE-ENABLE TTY KBD INTERRUPTS?
4484	025160	001003			BNE	15\$	:: BRANCH IF NOT
4485	025162	012777	000100	153770	MOV	#100, \$STKS	:: RE-ENABLE TTY KBD INTERRUPTS
4486	025170	000002		15\$:	RTI		:: RETURN
4487	025172	004737	023414	16\$:	JSR	PC, \$TYPEC	:: ECHO CHAR
4488	025176	021627	000060		CMP	(SP), #60	:: CHAR < 0?
4489	025202	002420			BLT	18\$	:: BRANCH IF YES
4490	025204	021627	000067		CMP	(SP), #67	:: CHAR > 7?
4491	025210	003015			BGT	18\$	:: BRANCH IF YES
4492	025212	042726	000060		BIC	#60, (SP)+	:: STRIP-OFF ASCII
4493	025216	005766	000002		TST	2(SP)	:: IS THIS THE FIRST CHAR
4494	025222	001403			BEQ	17\$	:: BRANCH IF YES
4495	025224	006316			ASL	(SP)	:: NO, SHIFT PRESENT
4496	025226	006316			ASL	(SP)	:: CHAR OVER TO MAKE
4497	025230	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
4498	025232	005266	000002	17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
4499	025236	056616	177776		BIS	-2(SP), (SP)	:: SET IN NEW CHAR
4500	025242	000667			BR	7\$	:: GET THE NEXT ONE
4501	025244	104401	001202	18\$:	TYPE	\$QUES	:: TYPE ?<CR><LF>
4502	025250	000720			BR	20\$	:: SIMULATE CONTROL-U
4503					.DSABL	LSB	

```
4504  
4505  
4506  
4507  
4508  
4509  
4510  
4511  
4512  
4513  
4514 025252 011646  
4515 025254 016666 000004 000002  
4516 025262 005066 000004  
4517 025266 005046  
4518 025270 012746 025276  
4519 025274 000002  
4520 025276  
4521 025276 005737 024404  
4522 025302 001775  
4523 025304 005337 024404  
4524 025310 117766 177074 000004  
4525 025316 005237 024410  
4526 025322 023727 024410 024421  
4527 025330 001003  
4528 025332 012737 024412 024410  
4529 025340 000002  
4530  
4531  
4532  
4533  
4534  
4535  
4536  
4537 025342 010346  
4538 025344 005046  
4539 025346 012703 025576  
4540 025352 022703 025613  
4541 025356 101456  
4542 025360 104410  
4543 025362 112613  
4544 025364 122713 000177  
4545 025370 001022  
4546 025372 005716  
4547 025374 001007  
4548 025376 112737 000134 025574  
4549 025404 104401 025574  
4550 025410 012716 177777  
4551 025414 005303  
4552 025416 020327 025576  
4553 025422 103434  
4554 025424 111337 025574  
4555 025430 104401 025574  
4556 025434 000746  
4557 025436 005716  
4558 025440 001406  
4559 025442 112737 000134 025574
```

```
*****  
*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  
*  
SRDCHR: MOV (SP), -(SP) ;: PUSH DOWN THE PC AND  
MOV 4(SP), 2(SP) ;: THE PS  
CLR 4(SP) ;: GET READY FOR A CHARACTER  
CLR -(SP) ;: PUT NEW PS ON STACK  
MOV #64$, -(SP) ;: PUT NEW PC ON STACK  
RTI ;: POP NEW PC AND PS  
  
64$:  
1$: TST $TKCNT ;: WAIT ON A CHARACTER  
BEQ 1$  
DEC $TKCNT ;: DECREMENT THE COUNTER  
MOV $TKQOUT, 4(SP) ;: GET ONE CHARACTER  
INC $TKQOUT ;: UPDATE THE POINTER  
CMP $TKQOUT, # $TKQEND ;: DID IT GO OFF OF THE END?  
BNE 2$ ;: BRANCH IF NO  
MOV # $TKQSR, $TKQOUT ;: RESET THE POINTER  
RTI ;: RETURN  
2$:  
*****  
*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  
*  
SRDLIN: MOV R3, -(SP) ;: SAVE R3  
CLR -(SP) ;: CLEAR THE RUBOUT KEY  
1$: MOV # $TTYIN, R3 ;: GET ADDRESS  
2$: CMP # $TTYIN+15, R3 ;: BUFFER FULL?  
BLOS 4$ ;: BR IF YES  
RDCHR ;: GO READ ONE CHARACTER FROM THE TTY  
MOV (SP)+, (R3) ;: GET CHARACTER  
10$: CMPB #177, (R3) ;: IS IT A RUBOUT  
BNE 5$ ;: BR IF NO  
TST (SP) ;: IS THIS THE FIRST RUBOUT?  
BNE 6$ ;: BR IF NO  
MOV #'\, 9$ ;: TYPE A BACK SLASH  
TYPE 9$  
MOV #-1, (SP) ;: SET THE RUBOUT KEY  
6$: DEC R3 ;: BACKUP BY ONE  
CMP R3, # $TTYIN ;: STACK EMPTY?  
BLOS 4$ ;: BR IF YES  
MOV (R3), 9$ ;: SETUP TO TYPEOUT THE DELETED CHAR.  
TYPE 9$ ;: GO TYPE  
BR 2$ ;: GO READ ANOTHER CHAR.  
5$: TST (SP) ;: RUBOUT KEY SET?  
BEQ 7$ ;: BR IF NO  
MOV #'\, 9$ ;: TYPE A BACK SLASH
```

```

4560 025450 104401 025574          TYPE          9$
4561 025454 005016          CLR          (SP)          ;; CLEAR THE RUBOUT KEY
4562 025456 122713 000025      7$: CMPB      #25,(R3)      ;; IS CHARACTER A CTRL U?
4563 025462 001003          BNE          8$           ;; BR IF NO
4564 025464 104401 025620          TYPE          $CNTLU      ;; TYPE A CONTROL "U"
4565 025470 000726          BR          1$           ;; GO START OVER
4566 025472 122713 000022      8$: CMPB      #22,(R3)      ;; IS CHARACTER A "R"?
4567 025476 001011          BNE          3$           ;; BRANCH IF NO
4568 025500 105013          CLRB        (R3)         ;; CLEAR THE CHARACTER
4569 025502 104401 001203          TYPE          $CRLF      ;; TYPE A "CR" & "LF"
4570 025506 104401 025576          TYPE          $TTYIN     ;; TYPE THE INPUT STRING
4571 025512 000717          BR          2$           ;; GO PICKUP ANOTHER CHACTER
4572 025514 104401 001202      4$: TYPE          $QUES     ;; TYPE A '?'
4573 025520 000712          BR          1$           ;; CLEAR THE BUFFER AND LOOP
4574 025522 111337 025574      3$: MOV      (R3),9$      ;; ECHO THE CHARACTER
4575 025526 104401 025574          TYPE          9$
4576 025532 122723 000015          CMPB      #15,(R3)+     ;; CHECK FOR RETURN
4577 025536 001305          BNE          2$           ;; LOOP IF NOT RETURN
4578 025540 105063 177777          CLRB      -1(R3)       ;; CLEAR RETURN (THE 15)
4579 025544 104401 001204          TYPE          $LF       ;; TYPE A LINE FEED
4580 025550 005726          TST        (SP)+       ;; CLEAN RUBOUT KEY FROM THE STACK
4581 025552 012603          MOV      (SP)+,R3      ;; RESTORE R3
4582 025554 011646          MOV      (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4583 025556 016666 000004 000002  MOV      4(SP),2(SP)    ;; FIRST ASCII CHARACTER ON IT
4584 025564 012766 025576 000004  MOV      #$TTYIN,4(SP)
4585 025572 000002          RTI
4586 025574 000          9$: .BYTE      0          ;; RETURN
4587 025575 000          .BYTE      0          ;; STORAGE FOR ASCII CHAR. TO TYPE
4588 025576 000015          $TTYIN: .BLKB      15   ;; TERMINATOR
4589 025613 136 006503 000012  $CNTLC: .ASCIZ  /?C/<15><12>  ;; RESERVE 15 BYTES FOR TTY INPUT
4590 025620 052536 005015 000  $CNTLU: .ASCIZ  /?U/<15><12>  ;; CONTROL "C"
4591 025625 136 006507 000012  $CNTLG: .ASCIZ  /?G/<15><12>  ;; CONTROL "U"
4592 025632 005015 053523 020122  $MSWR: .ASCIZ  <15><12>/SWR = /  ;; CONTROL "G"
4593 025640 020075 000
4594 025643 040 047040 053505  $MNEW: .ASCIZ  / NEW = /
4595 025650 036440 000040
4596
4597 .SBTTL POWER DOWN AND UP ROUTINES
4598
4599 ;; *****
4600 :POWER DOWN ROUTINE
4601 025654 012737 026020 000024  $PWRDN: MOV      #ILLUP,@PWRVEC ;; SET FOR FAST UP
4602 025662 012737 000340 000026  MOV      #340,@PWRVEC+2 ;; PRIO:7
4603 025670 010046          MOV      R0,-(SP)      ;; PUSH R0 ON STACK
4604 025672 010146          MOV      R1,-(SP)      ;; PUSH R1 ON STACK
4605 025674 010246          MOV      R2,-(SP)      ;; PUSH R2 ON STACK
4606 025676 010346          MOV      R3,-(SP)      ;; PUSH R3 ON STACK
4607 025700 010446          MOV      R4,-(SP)      ;; PUSH R4 ON STACK
4608 025702 010546          MOV      R5,-(SP)      ;; PUSH R5 ON STACK
4609 025704 017746 153244          MOV      @SWR,-(SP)    ;; PUSH @SWR ON STACK
4610 025710 010637 026024          MOV      SP,$SAVR6    ;; SAVE SP
4611 025714 012737 025726 000024  MOV      #$PWRUP,@PWRVEC ;; SET UP VECTOR
4612 025722 000000          HALT
4613 025724 000776          BR          -2          ;; HANG UP
4614
4615 ;; *****

```

```

4616      :POWER UP ROUTINE
4617 025726 012737 026020 000024 $PWRUP: MOV    $SILLUP,@#PWRVEC  ;;SET FOR FAST DOWN
4618 025734 013706 026024          MOV    $$SAVR6,SP      ;;GET SP
4619 025740 005037 026024          CLR    $$SAVR6        ;;WAIT LOOP FOR THE TTY
4620 025744 005237 026024 1$: INC    $$SAVR6        ;;WAIT FOR THE INC
4621 025750 001375          BNE    1$              ;;OF WORD
4622 025752 012677 153176          MOV    (SP)+,@SWR      ;;POP STACK INTO @SWR
4623 025756 012605          MOV    (SP)+,R5        ;;POP STACK INTO R5
4624 025760 012604          MOV    (SP)+,R4        ;;POP STACK INTO R4
4625 025762 012603          MOV    (SP)+,R3        ;;POP STACK INTO R3
4626 025764 012602          MOV    (SP)+,R2        ;;POP STACK INTO R2
4627 025766 012601          MOV    (SP)+,R1        ;;POP STACK INTO R1
4628 025770 012600          MOV    (SP)+,R0        ;;POP STACK INTO R0
4629 025772 012737 025654 000024  MOV    #SPWRDN,@#PWRVEC  ;;SET UP THE POWER DOWN VECTOR
4630 026000 012737 000340 000026  MOV    #340,@#PWRVEC+2  ;;PRIO:7
4631 026006 104401          TYPE                                ;;REPORT THE POWER FAILURE
4632 026010 036417  SPWRMG: .WORD  PMSG          ;;POWER FAIL MESSAGE POINTER
4633 026012 012716          MOV    (PC)+,(SP)      ;;RESTART AT BEGIN2
4634 026014 007034  SPWRAD: .WORD  BEGIN2      ;;RESTART ADDRESS
4635 026016 000002          RTI
4636 026020 000000  $SILLUP: HALT          ;;THE POWER UP SEQUENCE WAS STARTED
4637 026022 000776          BR    -2              ;;BEFORE THE POWER DOWN WAS COMPLETE
4638 026024 000000  $$SAVR6: 0              ;;PUT THE SP HERE
4639
4640      .SBTTL TYPE NUMERICAL ASCIZ STRING SUPPRESS LEADING ZEROS
4641
4642      ;*****
4643      ;*THIS ROUTINE IS USED TO TYPE AN ASCIZ NUMBER SUPPRESSING THE
4644      ;*LEADING NUMBERS.
4645      ;*CALL
4646      ;*      MOV    #NUMADR,-(SP)  ;;FIRST ADDRESS OF ASCIZ STRING
4647      ;*      JSR    PC,@$$SUPRS
4648
4649
4650 026026 010046          $$SUPRS: MOV    RO,-(SP)  ;;SAVE RO
4651 026030 016600 000004          MOV    4(SP),RO      ;;PICKUP THE POINTER
4652 026034 105710 1$: TSTB   (RO)        ;;TERMINATEOR?
4653 026036 001403          BEQ    2$            ;;BR IF YES
4654 026040 122720 000060          CMPB  #'0,(RO)+      ;;IS THIS AN ASCII "0" ?
4655 026044 001773          BEQ    1$            ;;BR IF YES
4656 026046 005300 2$: DEC    RO              ;;BACKUP BY "1"
4657 026050 010037 026056          MOV    RO,3$         ;;SAVE FOR TYPING
4658 026054 104401          TYPE                                ;;GO TYPE
4659 026056 000000 3$: .WORD  0              ;;ASCIZ POINTER GOES HERE
4660 026060 012600          MOV    (SP)+,RO      ;;RESTORE RO
4661 026062 012616          MOV    (SP)+,(SP)    ;;RESTORE THE STACK
4662 026064 000207          RTS    PC            ;;RETURN
4663
4664      .SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE
4665
4666      ;*****
4667      ;*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
4668      ;*UNSIGNED DECIMAL ASCIZ NUMBER.
4669      ;*CALL
4670      ;*      MOV    NUMBER,-(SP)  ;;PUT BINARY NUMBER ON THE STACK
4671      ;*      JSR    PC,@$$SB2D   ;;CALL

```

```

4672 ;* RETURN ;:ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK
4673
4674
4675 026066 016637 000002 026116 $$B2D: MOV 2(SP),1$ ;:SAVE BINARY NUMBER
4676 026074 012746 026116 ;:SET POINTER
4677 026100 004737 026122 JSR PC,2$$B2D ;:CALL DOUBLE LENGTH CONVERT
4678 026104 062716 000005 ADD #5,(SP) ;:ONLY ALLOW FIVE CHARACTERS
4679 026110 012666 000002 MOV (SP)+,2(SP) ;:PICKUP POINTER
4680 026114 000207 ;:RETURN
4681 026116 000000 000000 1$: .WORD 0,0
4682
4683 .SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE
4684
4685 ;:*****
4686 ;:THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
4687 ;:DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
4688 ;:POSITIVE.
4689 ;:CALL
4690 ;* MOV #PNTR,-(SP) ;:POINTER TO LOW WORD OF BINARY NUMBER
4691 ;* JSR PC,2$$B2D ;:THE FIRST ADDRESS OF ASCIZ
4692 ;* RETURN ;:IS ON THE STACK
4693
4694
4695
4696 026122 104412 $$B2D: SAVREG ;:SAVE REGISTERS
4697 026124 016602 000002 MOV 2(SP),R2 ;:PICKUP THE DATA POINTER
4698 026130 012700 026302 MOV #SDECLV,R0 ;:GET ADDRESS OF "SDECLV" STRING
4699 026134 010066 000002 MOV R0,2(SP) ;:PUT ADDRESS OF ASCIZ STRING ON STACK
4700 026140 012201 MOV (R2)+,R1 ;:PICKUP THE BINARY NUMBER
4701 026142 012202 MOV (R2)+,R2
4702 026144 012737 000012 026220 MOV #10,4$ ;:SET UP TO DO 10 CONVERSIONS
4703 026152 012704 026232 MOV #STNPWR,R4 ;:ADDRESS OF TEN POWER
4704 026156 012705 026234 MOV #STNPWR+2,R5
4705 026162 005003 1$: CLR R3 ;:CLEAR PARTIAL
4706 026164 161401 2$: SUB (R4),R1 ;:SUBTRACT TEN POWER
4707 026166 005602 SBC R2
4708 026170 161502 SUB (R5),R2
4709 026172 002402 BLT 3$ ;:BR IF TEN POWER TO LARGE
4710 026174 005203 INC R3 ;:ADD 1 TO PARTIAL
4711 026176 000772 BR 2$ ;:LOOP
4712 026200 062401 3$: ADD (R4)+,R1 ;:RESTORE SUBTRACTED VALUE
4713 026202 005502 ADC R2
4714 026204 062402 ADD (R4)+,R2
4715 026206 022525 CMP (R5)+,(R5)+ ;:MOVE TO NEXT TEN POWER
4716 026210 052703 000060 BIS #'0,R3 ;:CHANGE PARTIAL TO ASCII
4717 026214 110320 MOVB R3,(R0)+ ;:SAVE IT
4718 026216 005327 DEC (PC)+ ;:DONE?
4719 026220 000000 4$: .WORD 0 ;:BR IF NO
4720 026222 001357 BNE 1$ ;:TERMINATOR
4721 026224 105020 CLRB (R0)+ ;:RESTORE REGISTERS
4722 026226 104413 RESREG ;:RESTORE REGISTERS
4723 026230 000207 RTS PC ;:RETURN
4724 026232 145000 $TNPWR: 145000 ;:1.0E09
4725 026234 035632 35632
4726 026236 160400 160400 ;:1.0E08
4727 026240 002765 2765

```



```

4728 026242 113200
4729 026244 000230
4730 026246 041100
4731 026250 000017
4732 026252 103240
4733 026254 000001
4734 026256 023420
4735 026260 000000
4736 026262 001750
4737 026264 000000
4738 026266 000144
4739 026270 000000
4740 026272 000012
4741 026274 000000
4742 026276 000001
4743 026300 000000
4744 026302 000014
4745
4746
4747
4748
4749
4750
4751
4752
4753
4754
4755
4756
4757
4758
4759
4760
4761
4762
4763 026316
4764 026316 010046
4765 026320 010146
4766 026322 010246
4767 026324 010346
4768 026326 010446
4769 026330 010546
4770 026332 016646 000022
4771 026336 016646 000022
4772 026342 016646 000022
4773 026346 016646 000022
4774 026352 000002
4775
4776
4777
4778
4779 026354
4780 026354 012666 000022
4781 026360 012666 000022
4782 026364 012666 000022
4783 026370 012666 000022

```

```

113200 ;;1.0E07
230
041100 ;;1.0E06
17
103240 ;;1.0E05
1
23420 ;;1.0E04
0
1750 ;;1.0E03
0
144 ;;1.0E02
0
12 ;;1.0E01
0
1 ;;1.0E00
0
$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
$SAVREG:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R4,-(SP) ;;PUSH R4 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV 22(SP),-(SP) ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP) ;;SAVE PS OF CALL
MOV 22(SP),-(SP) ;;SAVE PC OF CALL
RTI
;RESTORE R0-R5
;CALL:
; RESREG
$RESREG:
MOV (SP)+,22(SP) ;;RESTORE PC OF CALL
MOV (SP)+,22(SP) ;;RESTORE PS OF CALL
MOV (SP)+,22(SP) ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP) ;;RESTORE PS OF MAIN FLOW

```

```

4784 026374 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
4785 026376 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
4786 026400 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
4787 026402 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
4788 026404 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
4789 026406 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
4790 026410 000002      RTI

```

.SBTTL TRAP DECODER

```

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

```

```

4800 026412 010046      $TRAP:  MOV      RO, -(SP)      ;;SAVE RO
4801 026414 016600 000002  MOV      2(SP),RO      ;;GET TRAP ADDRESS
4802 026420 005740      TST      -(RO)        ;;BACKUP BY 2
4803 026422 111000      MOV      (RO),RO      ;;GET RIGHT BYTE OF TRAP
4804 026424 006300      ASL      RO           ;;POSITION FOR INDEXING
4805 026426 016000 026446  MOV      $TRPAD(RO),RO  ;;INDEX TO TABLE
4806 026432 000200      RTS      RO           ;;GO TO ROUTINE

```

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

```

4811 026434 011646      $TRAP2: MOV      (SP), -(SP)    ;;MOVE THE PC DOWN
4812 026436 016666 000004 000002  MOV      4(SP),2(SP)    ;;MOVE THE PSW DOWN
4813 026444 000002      RTI                    ;;RESTORE THE PSW

```

.SBTTL TRAP TABLE

```

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

```

```

;
; ROUTINE
;-----
4822 026446 026434      $TRPAD: .WORD      $TRAP2
4823 026450 023202      $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
4824 026452 023756      $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
4825 026454 023732      $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
4826 026456 023772      $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
4827 026460 024160      $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
4828
4829 026462 025000      $GTSWR  ;;CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
4830
4831 026464 024710      $CKSWR  ;;CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
4832 026466 025252      $RDCHR  ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
4833 026470 025342      $RDLIN  ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
4834 026472 026316      $SAVREG ;;CALL=SAVREG     TRAP+12(104412) SAVE RO-R5 ROUTINE
4835 026474 026354      $RESREG ;;CALL=RESREG     TRAP+13(104413) RESTORE RO-R5 ROUTINE

```

.SBTTL SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 5.1)-24-AUG-77

4836  
4837  
4838  
4839

```

4840 ;NEW DRIVE TYPE ID FOR RM02 *****
4841 ; 10-AUG-77 *****
4842
4843 ;COPYRIGHT (C) 1977
4844 ;DIGITAL EQUIPMENT CORP.
4845 ;MAYNARD, MA 01754
4846 ;AUTHOR(S): JIM LACEY/CHUCK HESS/C. CHEN
4847
4848 ;*****
4849
4850 ;STORAGE FOR RMD5, RMER1, RMER2, AND RMMR2 ON AN ERROR "2"
4851 ;RMERRS = RMD5
4852 ;RMERRS+2 = RMER1
4853 ;RMERRS+4 = RMER2
4854 ;RMERRS+6 = RMMR2
4855
4856 026476 000000 000000 000000 RMERRS: .WORD 0,0,0,0
4857 026504 000000
4858
4859 ;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
4860 ;DRVACT=0 IF DRIVE IS IDLE
4861 ;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
4862 ;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
4863
4864 026506 000 DRVACT: .BYTE 0 ;DRIVE 0
4865 026507 000 .BYTE 0 ;DRIVE 1
4866 026510 000 .BYTE 0 ;DRIVE 2
4867 026511 000 .BYTE 0 ;DRIVE 3
4868 026512 000 .BYTE 0 ;DRIVE 4
4869 026513 000 .BYTE 0 ;DRIVE 5
4870 026514 000 .BYTE 0 ;DRIVE 6
4871 026515 000 .BYTE 0 ;DRIVE 7
4872
4873 ;TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
4874 ;DRVSTA=0 IF DRIVE IS OFFLINE OR NONEXISTENT
4875 ;DRVSTA>0 IF DRIVE IS ONLINE
4876 ;DRVSTA<0 IF DRIVE IS UNSAFE
4877
4878 026516 000 DRVSTA: .BYTE 0 ;DRIVE 0
4879 026517 000 .BYTE 0 ;DRIVE 1
4880 026520 000 .BYTE 0 ;DRIVE 2
4881 026521 000 .BYTE 0 ;DRIVE 3
4882 026522 000 .BYTE 0 ;DRIVE 4
4883 026523 000 .BYTE 0 ;DRIVE 5
4884 026524 000 .BYTE 0 ;DRIVE 6
4885 026525 000 .BYTE 0 ;DRIVE 7
4886
4887 ;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)
4888 ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)
4889 ;DRV TYP=5 IF DRIVE IS RM02 *****
4890 ;DRV TYP=4 IF DRIVE IS RM03
4891 ;DRV TYP=-1 IF NOT RM03
4892
4893 026526 000 DRV TYP: .BYTE 0 ;DRIVE 0
4894 026527 000 .BYTE 0 ;DRIVE 1
4895 026530 000 .BYTE 0 ;DRIVE 2

```

4896 026531 000  
4897 026532 000  
4898 026533 000  
4899 026534 000  
4900 026535 000

```
.BYTE 0 ;DRIVE 3
.BYTE 0 ;DRIVE 4
.BYTE 0 ;DRIVE 5
.BYTE 0 ;DRIVE 6
.BYTE 0 ;DRIVE 7
```

4901  
4902  
4903  
4904  
4905  
4906 026536 000  
4907 026537 000  
4908 026540 000  
4909 026541 000  
4910 026542 000  
4911 026543 000  
4912 026544 000  
4913 026545 000

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

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

4914  
4915  
4916  
4917  
4918  
4919 026546 000  
4920 026547 000  
4921 026550 000  
4922 026551 000  
4923 026552 000  
4924 026553 000  
4925 026554 000  
4926 026555 000

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

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

4927  
4928  
4929  
4930  
4931

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

4932 026556 000000  
4933  
4934  
4935  
4936  
4937  
4938  
4939

```
TRNSWT: .WORD 0

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

4940 026560 000000  
4941  
4942  
4943  
4944  
4945  
4946 026562 000

```
SRCHWT: .WORD 0

;RM03 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
;ACTDRV=0 IF DRIVER IS INACTIVE
;ACTDRV>0 IF DRIVER IS ACTIVE
```

4947  
4948  
4949  
4950  
4951

```
ACTDRV: .BYTE 0

;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
```

4952 026563 000  
 4953  
 4954  
 4955  
 4956  
 4957  
 4958  
 4959 026564 000  
 4960 026565 000  
 4961 026566 000  
 4962 026567 000  
 4963 026570 000  
 4964 026571 000  
 4965 026572 000  
 4966 026573 000  
 4967  
 4968  
 4969  
 4970  
 4971 026574 000  
 4972 026575 000  
 4973 026576 000  
 4974 026577 000  
 4975 026600 000  
 4976 026601 000  
 4977 026602 000  
 4978 026603 000  
 4979  
 4980  
 4981  
 4982  
 4983  
 4984  
 4985  
 4986 026604 000000  
 4987  
 4988  
 4989  
 4990  
 4991  
 4992  
 4993  
 4994 026606 177777  
 4995  
 4996  
 4997  
 4998  
 4999 026610 177777  
 5000 026612 177777  
 5001 026614 177777  
 5002 026616 177777  
 5003 026620 177777  
 5004 026622 177777  
 5005 026624 177777  
 5006 026626 177777  
 5007

ACTSTR: .BYTE 0  
 ;UNLOAD FLAG (ULDFLG=8 BYTES)  
 ;ULDFLG=0 IF NO UNLOAD COMMAND  
 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS  
 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE  
 ULDFLG: .BYTE 0 ;DRIVE 0  
 .BYTE 0 ;DRIVE 1  
 .BYTE 0 ;DRIVE 2  
 .BYTE 0 ;DRIVE 3  
 .BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7  
 ;LOOK AHEAD COUNT (LACNT=8 BYTES)  
 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED  
 LACNT: .BYTE 0 ;DRIVE 0  
 .BYTE 0 ;DRIVE 1  
 .BYTE 0 ;DRIVE 2  
 .BYTE 0 ;DRIVE 3  
 .BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7  
 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)  
 ;SAVEFG <0 IF SAVE THE RH70/RMO3 REGISTERS WHEN THE  
 ;OPERATION IS COMPLETED AS PER (DPB+14).  
 ;SAVEFG=0 IF SAVE THE RH70/RMO3 REGISTERS, AS PER  
 ;(DPB+14), AFTER AN ERROR.  
 SAVEFG: .WORD 0  
 ;SEEK FLAG (SEEKFG=1 WORD)  
 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW  
 ;FOR A DATA TRANSFER START A SEARCH COMMAND  
 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,  
 ;DISREGARD THE WINDOW  
 SEEKFG: .WORD -1  
 ;TIMEOUT TABLE (TIMER=8 WORDS)  
 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION  
 TIMER: .WORD -1 ;DRIVE 0  
 .WORD -1 ;DRIVE 1  
 .WORD -1 ;DRIVE 2  
 .WORD -1 ;DRIVE 3  
 .WORD -1 ;DRIVE 4  
 .WORD -1 ;DRIVE 5  
 .WORD -1 ;DRIVE 6  
 .WORD -1 ;DRIVE 7

5009  
5009  
5010  
5011  
5012 026630 177777  
5013  
5014  
5015  
5016  
5017  
5018 026632 001  
5019 026633 002  
5020 026634 004  
5021 026635 010  
5022 026636 020  
5023 026637 040  
5024 026640 100  
5025 026641 200  
5026  
5027  
5028  
5029  
5030 026642 000003  
5031  
5032  
5033  
5034  
5035 026644 176700  
5036 026646 000254 000240  
5037  
5038  
5039  
5040 026652 000004  
5041  
5042  
5043 026654 001000  
5044  
5045  
5046 026656 000200  
5047  
5048  
5049 026660 000005  
5050  
5051  
5052  
5053 000000  
5054 000002  
5055 000004  
5056 000006  
5057 000010  
5058 000012  
5059 000014  
5060 000016  
5061 000020  
5062 000022  
5063 000024

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

DTUW: .WORD -1

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

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

;FSRM03 TO RH70 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE  
;CALLING IT FATAL (MCPEM=1 WORD)

MCPEM: .WORD 3

;STORAGE FOR RMADR (THE FIRST ADDRESS (776700) OF THE RH70/RM03),  
;RMVEC (THE VECTOR ADDRESS (254)), AND RMVEC+2 (THE BR LEVEL (5)).

RMADR: .WORD 176700  
RMVEC: .WORD 254,5\*32.

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

MXLACT: .WORD 4  
;MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)

MXDLTA: .WORD 8.\*64.  
;MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)

MNDLTA: .WORD 2.\*64.  
;MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)

MXWNDW: .WORD 5

;DEFINITIONS OF THE RH70/RM03 ADDRESS INDEXES

RMCS1=0	;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
RMWC=2	;WORD COUNT REGISTER (NOT A DRIVE REG)
RMBA=4	;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
RMDA=6	;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
RMCS2=10	;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
RMS=12	;DRIVE STATUS REGISTER (DRIVE REG 01)
RMR1=14	;ERROR REGISTER #1 (DRIVE REG. 02)
RMS=16	;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
RMLA=20	;LOOK AHEAD REGISTER (DRIVE REG. 07)
RMDB=22	;DATA BUFFER REGISTER (NOT A DRIVE REG.)
RMMR1=24	;MAINTAINABILITY REGISTER (DRIVE REG. 03)

```

5064      000026      RMDT=26      ;DRIVE TYPE REGISTER (DRIVE REG. 06)
5065      000030      RMSN=30      ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
5066      000032      RMOF=32      ;OFFSET REGISTER (DRIVE REG. 11)
5067      000034      RMDC=34      ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
5068      000036      RMHR=36      ;DUMMY ADDRESS REGISTER (DRIVE REG. 13)
5069      000040      RMMR2=40     ;MAINTENANCE REGISTER #2
5070      000042      RMER2=42     ;ERROR REGISTER #2 (DRIVE REG. 15)
5071      000044      RMEC1=44     ;ECC POSITION REGISTER (DRIVE REG. 16)
5072      000046      RMEC2=46     ;ECC PATTERN REGISTER (DRIVE REG. 17)
5073
5074      ;RH70/RMO3 DRIVER INITIALIZATION CODE
5075      ;THIS ROUTINE WILL DETERMINE WHICH RMO3 DRIVES ARE
5076      ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
5077      ;TO THE PROPER STATE FOR EACH DRIVE.
5078      ;NOTE: THIS ROUTINE CALLS DRVINT
5079
5080      ;CALL
5081
5082      ;       JSR      PC,RMINIT
5083      ;       RETURN
5084
5085      ;NOTE: THE 'P' OR 'L' CLOCK MUST BE STARTED
5086
5087      026662      104412      RMINIT: SAVREG      ;SAVE R0 - R5
5088      026664      013746      177776      MOV      @#PS,-(SP)      ;SAVE THE PRESENT PROCESSOR STATUS
5089      026670      012737      000240      177776      MOV      @<5*32.>,@#PS      ;CHANGE THE PRIORITY TO 5
5090      026676      004737      034524      JSR      PC,CLIQUE      ;CLEAR ALL REQUEST QUEUES
5091      026702      012701      026476      MOV      #RMERRS,R1      ;FIRST ADDRESS TO BE CLEARED
5092      026706      012702      026606      MOV      #SEEKFG,R2      ;LAST ADDRESS TO BE CLEARED
5093      026712      005021      1$:      CLR      (R1)+      ;CLEAR
5094      026714      020102      CMP      R1,R2      ;ARE WE DONE?
5095      026716      103775      BLO     1$      ;BRANCH IF NO
5096      026720      012702      026630      MOV      #DTUW,R2      ;LAST ADDRESS
5097      026724      012721      177777      2$:      MOV      #-1,(R1)+      ;INITIALIZE
5098      026730      020102      CMP      R1,R2      ;DONE?
5099      026732      101774      BLOS    2$      ;LOOP IF NO
5100      026734      005037      026516      CLR      DRVSTA      ;SET ALL DRIVES TO OFFLINE
5101      026740      005037      026520      CLR      DRVSTA+2
5102      026744      005037      026522      CLR      DRVSTA+4
5103      026750      005037      026524      CLR      DRVSTA+6
5104      026754      013703      026646      MOV      RMVEC,R3      ;SETUP THE RH70/RMO3 VECTOR
5105      026760      012723      031474      MOV      #ISR,(R3)+
5106      026764      013713      026650      MOV      RMVEC+2,(R3)
5107      026770      013704      026644      MOV      RMADR,R4
5108      026774      012764      000040      000010      MOV      #BIT05,RMCS2(R4) ;FIRST ADDRESS OF RH70/RMO3
5109      027002      005001      109:      CLR      R1      ;MASSBUS INIT
5110      027004      004037      027074      3$:      JSR      R0,DRVINT      ;START WITH DRIVE 0
5111      027010      000401      BR      4$      ;INIT THE DRIVE
5112      027012      000402      BR      5$      ;'DVA' NOT SET OR PARITY ERROR
5113      027014      105061      026516      4$:      CLRB    DRVSTA(R1)      ;NORMAL RETURN
5114      027020      005201      5$:      INC      R1      ;SET DRIVE STATUS TO OFFLINE
5115      027022      042701      177770      BIC      #1C7,R1      ;GO TO NEXT DRIVE
5116      027026      001366      BNE     3$      ;MASK OUT UNUSED BITS
5117      027030      012701      000007      MOV      #7,R1      ;BR IF MORE DRIVES TO GO
5118      027034      005037      177776      CLR      @#PS      ;START WITH DRIVE 7
5119      027040      105761      026536      6$:      TSTB   DPINT(R1)      ;CLEAR THE PROCESSOR STATUS
;WAITING FOR DRIVE TO SWITCH PORTS ?

```

```

S120 027044 001405          BEQ      8$          ;BR NOT WAITING
S121 027046 004737 034160   JSR      PC,SET.IE  ;SET INTERRUPT
S122 027052 105761 026536   7$:     TSTB      DP,INT(R1) ;DRIVE SWITCHED PORTS ?
S123 027056 001375          BNE      7$          ;BR IF NOT
S124 027060 005301          DEC      R1          ;GO TO THE NEXT DRIVE
S125 027062 100366          BPL      6$          ;CHECK NEXT DRIVE
S126 027064 012637 177776   MOV      (SP)+,2#PS ;RESTORE THE PROCESSOR STATUS
S127 027070 104413          RESREG   ;RESTORE RO - RS
S128 027072 000207          RTS      PC          ;BYE-BYE
S129
S130 ;DRIVE INITIALIZATION ROUTINE
S131 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
S132 ;AN RMO3. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
S133 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
S134 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
S135 ;DRVSTA IS SET TO THE PROPER CONDITION.
S136
S137 ;CALL
S138 :
S139 :
S140 :
S141 :
S142 :
S143 :
S144 :
S145 027074 010546          MOV      #DRVNUM,R1 ;DRIVE NUMBER TO R1
S146 027076 105061 026516   MOV      RMADR,R4   ;UNIBUS ADDRESS OF RH70/RMO3 (RMCS1)
S147 027102 105061 026526   JSR      RO,DRVINT  ;CALLED BY A JSR
S148 027106 105061 026554   RETURN1 ;ERROR OCCURRED (PARITY)
S149 027112 010164 000010   RETURN2 ;NORMAL RETURN
S150 027116 112764 000111 000000
S151 027124 032764 010000 000010
S152 027132 001403          DRVINT: MOV      R5,-(SP) ;SAVE R5
S153 027134 004737 034160   DULP:  CLRB     DRVSTA(R1) ;START DRIVE STATUS AS OFFLINE
S154 027142 105061 026516   CLRB     DRVSTYP(R1) ;CLEAR THE DRIVE TYPE INDICATOR
S155 027146 032764 004000 000000   CLRB     ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
S156 027154 001750          MOV      R1, RMCS2(R4) ;SELECT A DRIVE
S157 027156 004037 033470   MOV      #11, RMCS1(R4) ;DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
S158 027162 000026          BIT      #BIT12, RMCS2(R4) ;NONEXISTENT DRIVE?
S159 027164 027404          BEQ      1$          ;NO---BRANCH
S160 027166 012605          JSR      PC,SET.IE  ;GO SET "IE" WITHOUT A "TRE"
S161 027170 112761 000004 026526   BR      6$          ;LEAVE THIS ROUTINE
S162 027176 022705 020024          CLRB     DRVSTA(R1) ;SET DRIVE STATUS TO OFFLINE
S163 027202 001420          BIT      #BIT11, RMCS1(R4) ;SEE IF DRIVE AVAILABLE
S164 027204 022705 024024          BEQ      DULP       ;BR IF DRIVE NOT AVAILABLE
S165 027210 001415          JSR      RO,RO.RM   ;READ THE DRIVE TYPE REG.
S166 027212 112761 000005 026526   RMDT     8$          ;ERROR RETURN ADDRESS
S167 027220 022705 020025          MOV      (SP)+,R5   ;PUT DRIVE TYPE IN R5
S168 027224 001407          MOV      #4,DRVSTYP(R1) ;SET RMO3 INDICATOR
S169 027226 022705 024025          CMP      #20024,R5  ;SINGLE PORT RMO3 ?
S170 027232 001404          BEQ      2$          ;BR IF YES
S171 027234 112761 177777 026526   CMP      #24024,R5  ;DUAL PORT RMO3 ?
S172 027242 000446          BEQ      2$          ;BR IF YES
S173 027244 012746 000121          MOV      #5,DRVSTYP(R1) ;SET RMO2 INDICATOR
S174 027250 004037 033650          CMP      #20025,R5  ;SINGLE PPRT RMO2 ?
S175 027254 000000          BEQ      2$          ;BRANCH IF SO
                          BEQ      #24025,R5 ;DUAL PORT RMO2 ?
                          BEQ      2$          ;BRANCH IF SO
                          MOV      #-1,DRVSTYP(R1) ;SET INDICATOR TO 'OTHER'
                          BR      6$          ;EXIT
                          MOV      #121,-(SP) ;DO A "READ-IN PRESET"
                          JSR      RO,WRT.RM
                          RMCS1

```



F08

CZRMAC0 RMO3/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY11 30(1046) 22-NOV-77 08:16 PAGE 97  
SINGLE/DUAL PORT RM70/RMO3 DRIVER (REV 5.1)-24-AUG-77

SEQ 0096

```

5176 027256 027404      8$
5177 027260 012746 010000  MOV      #BIT12,-(SP)      ;SET FMT22=1
5178 027264 004037 033650  JSR      RO,WRT.RM
5179 027270 000032      RMOF
5180 027272 027404      8$
5181 027274 004037 033470  JSR      RO,RO.RM      ;READ RMD5
5182 027300 000012      RMD5
5183 027302 027404      8$
5184 027304 012605      MOV      (SP)+,R5      ;AND SAVE IT IN R5
5185 027306 100015      BPL      4$           ;BRANCH IF ATA=0
5186 027310 116164 026632 000016  MOVB    ATABIT(R1),RMA5(R4) ;CLEAR ATTENTION BIT
5187 027316 004037 033470  JSR      RO,RO.RM      ;FIND OUT WHY ATA=1
5188 027322 000014      RMER1
5189 027324 027404      8$
5190 027326 006126      ROL      (SP)+      ;IS IT UNSAFE?
5191 027330 100004      BPL      4$           ;BR IF NOT
5192 027332 112761 177777 026516  MOVB    #-1,DRVSTA(R1) ;SET UNSAFE INDICATOR
5193 027340 000407      BR      6$           ;EXIT
5194 027342 005105      4$: COM     R5           ;CHECK MOL, DPR, DRY, AND VV
5195 027344 042705 167077      BIC     #1<BIT12:BIT08:BIT07:BIT06>,R5
5196 027350 001003      BNE     6$           ;BRANCH IF MOL DPR, DRY, OR VV IS CLEAR
5197 027352 112761 000001 026516  MOVB    #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
5198 027360 005720      6$: TST     (R0)+      ;STEP OVER THE ERROR RETURN
5199 027362 000410      BR      8$           ;EXIT
5200 027364 006301      7$: ASL     R1           ;CHANGE INDEX TO ADDRESS WORDS
5201 027366 012761 060000 026610  MOV     #60000,TIMER(R1) ;START 2 SEC TIMER
5202 027374 006201      ASR     R1           ;RESTORE R1
5203 027376 112761 177777 026536  MOVB    #-1,DPINT(R1) ;SET PORT INITIALIZE INIDICATOR
5204 027404 012605      8$: MOV     (SP)+,R5      ;RESTORE R5
5205 027406 000200      RTS     RO           ;EXIT
5206
5207      ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
5208
5209      ;CALL
5210
5211      ;
5212      JSR     RO,@#RMO3 ;CALL THE RMO3 DRIVER
5213      PNTADR ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
5214      RETURN1 ;RETURN HERE IF QUEUE IS FULL
5215      RETURN2 ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
5216      ;IS AN ERROR CONDITION
5217
5218      RMO3: MOV     @#PS,-(SP) ;SAVE THE CALLING STATUS
5219      MOV     RMVEC+2,@#PS ;DON'T ALLOW ANY RMO3 INTERRUPTS
5220      MOVB    #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
5221      SAVREG ;SAVE RO - R5
5222      MOV     (R0),R2 ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
5223      CLR     16(R2) ;CLEAR THE STATUS/ERROR INDICATOR
5224      MOVB    (R2),R1 ;PICKUP THE DRIVE NUMBER
5225      MOV     RMADR,R4 ;UNIBUS ADDRESS OF RMCS1
5226      TSTB   DRVSTA(R1) ;CHECK DRIVES STATUS
5227      BGT     1$ ;BRANCH IF ONLINE
5228      TSTB   ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
5229      BNE     3$ ;BRANCH IF YES
5230      TSTB   DPINT(R1) ;TRYING TO INIT THE DRIVE
5231      BNE     5$ ;BR IF YES
5231      JSR     RO,DRVINT ;GO INIT. THE DRIVE

```

```

5232 027474 000434          BR      4$          ;ERROR RETURN
5233 027476 105761 026516  TSTB   DRVSTA(R1)  ;IS DRIVE STATUS ONLINE?
5234 027502 003445          BLE    6$          ;BR IF NOT
5235 027504 105761 026546  1$:   TSTB   DPRQS(R1)  ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
5236 027510 001031          BNE   5$          ;BR IF YES
5237 027512 010164 000010  MOV    R1, RMCS2(R4) ;SELECT THE DRIVE
5238 027516 004037 034622  JSR    RD, DRVQUE  ;PUT THIS REQUEST IN QUEUE
5239 027522 000460          BR    9$          ;QUEUE IS FULL
5240 027524 122762 000103 000002  CMPB  #103,2(R2)  ;IS THIS REQ. FOR AN UNLOAD?
5241 027532 001003          BNE   2$          ;BR IF NO
5242 027534 112761 177777 026564  MOVB  #-1, ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
5243 027542 105761 026506  2$:   TSTB   DRVACT(R1) ;IS THIS DRIVE ACTIVE?
5244 027546 001043          BNE   8$          ;BR IF YES
5245 027550 004737 027702  JSR    PC, OPT     ;CALL THE OPTIMIZER
5246 027554 000440          BR    8$          ;
5247 027556 012762 120000 000016  3$:   MOV    #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
5248 027564 000434          BR    8$          ;EXIT
5249 027566 004737 030762  4$:   JSR    PC, CI7    ;GO HANDLE THE PARITY ERROR
5250 027572 000431          BR    8$          ;
5251 027574 004037 034622  5$:   JSR    RD, DRVQUE  ;PUT REQUEST IN QUEUE
5252 027600 000431          BR    9$          ;QUEUE IS FULL
5253 027602 032714 000100  BIT    #BIT06, (R4) ;IE BIT SET ?
5254 027606 001023          BNE   8$          ;YES
5255 027610 004737 034160  JSR    PC, SET.IE  ;SET THE INTERRUPT
5256 027614 000420          BR    8$          ;RETURN
5257 027616 105761 026516  6$:   TSTB   DRVSTA(R1)  ;SEE IF DRIVE OFFLINE OR UNSAFE
5258 027622 002412          BLT   7$          ;BR IF UNSAFE
5259 027624 012762 140000 000016  MOV    #BIT15!BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
5260 027632 105761 026526  TSTB   DRVTP(R1)  ;SEE IF OFFLINE OR NONEXISTENT
5261 027636 001007          BNE   8$          ;BR IF OFFLINE
5262 027640 012762 100002 000016  MOV    #BIT15!BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
5263 027646 000403          BR    8$          ;GO TO EXIT
5264 027650 012762 110000 000016  7$:   MOV    #BIT15!BIT12,16(R2) ;DRIVE IS UNSAFE
5265 027656 104413          8$:   RESREG ;RESTORE R0 - R5
5266 027660 005720          TST   (R0)+       ;SETUP FOR NORMAL RETURN
5267 027662 000401          BR    10$         ;FINISH UP, THEN EXIT
5268 027664 104413          9$:   RESREG ;RESTORE R0 - R5
5269 027666 005720          10$:  TST   (R0)+       ;CORRECT THE RETURN ADDRESS
5270 027670 105037 026562  CLRB  ACTDRV      ;CLEAR "ACTIVE DRIVER" FLAG
5271 027674 012637 177776  MOV    (SP)+, #PS ;RETURN "PS" TO USER LEVEL
5272 027700 000200          RTS   R0          ;RETURN TO CALLER
5273
5274 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
5275
5276 ;CALL
5277
5278 MOV    #DRVNUM, R1 ;DRIVE NUMBER TO R1
5279 JSR    PC, OPT     ;SETUP A COMMAND
5280
5281 OPT:  SAVREG ;SAVE R0 - R5
5282 MOV    #PS, -(SP) ;SAVE PROC. STATUS
5283 BICB  ATABIT(R1), SRCHWT ;CLEAR LA SEACH FLAG
5284 CLRB  DPRQS(R1)  ;RESET THE PORT REQ FLAG ****
5285 JSR    PC, GETREQ ;GET "DPB" POINTER OF REQUEST
5286 TST   R2         ;IS THERE A REQUEST IN QUEUE?
5287 BEQ   7$        ;NO--BRANCH TO EXIT
5288 MOV    R1, RMCS2(R4) ;LOAD THE DRIVE ADDRESS *****

```

```

5288 027736 012764 000111 000000   MOV      #111,RMCS1(R4) ;CLEAR THE DRIVE
5289 027744 032764 004000 000000   BIT     #BIT11,RMCS1(R4) ;DVA SET ?
5290 027752 001446          BEQ     5$ ;TO PROT REQUEST,IF NOT
5291 027756 105761 026516      10$:    TSTB   DRVSTA(R1) ;IS DRIVE ONLINE?
5292 027760 003014          BGT     1$ ;YES--BRANCH
5293 027762 004737 034720      JSR     PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
5294 027766 012762 140000 000016   MOV     #BIT15!BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
5295 027774 105761 026516      TSTB   DRVSTA(R1) ;IS DRIVE UNSAFE ?
5296 030000 100053          BPL     8$ ;BR TO EXIT IF NOT
5297 030002 012762 110000 000016   MOV     #BIT15!BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
5298 030010 000447          BR     8$ ;BRANCH TO EXIT
5299 030012          1$:
5300          MOV     #111, -(SP) ;LOAD COMMAND ONTO THE STACK
5301          JSR     RO,WRT.RM ;LOAD THE REGISTER
5302          RMCS1  ;REGISTER INCREMENT
5303          6$ ;ERROR RETURN ADDRESS
5304          BIT     #BIT11,(R4) ;DRIVE AVAILABLE ?
5305          BEQ     9$ ;BR IF NOT
5306 030012 122762 000150 000002   CMPB   #150,2(R2) ;IS THE REQUEST FOR I/O?
5307 030020 002403          BLT     2$ ;YES--BRANCH
5308 030022 004737 030346      JSR     PC,C14 ;CALL THE COMMAND INITIATOR
5309 030026 000440          BR     8$ ;BRANCH TO EXIT
5310 030030 005737 026630      2$:    TST     DTUW ;DATA TRANSFER UNDERWAY?
5311 030034 002012          BGE     4$ ;YES--GO START A SEARCH
5312 030036 005737 026606      TST     SEEKFG ;DO IMPLIED SEEKS?
5313 030042 100404          BMI     3$ ;YES---BRANCH
5314 030044 004037 031320      JSR     RO,LA ;NO--DO LOOK AHEAD
5315 030050 000427          BR     8$ ;RETURN HERE ON A PARITY ERROR
5316 030052 000403          BR     4$ ;GO START A SEARCH
5317 030054 004737 030140      3$:    JSR     PC,C11 ;START A DATA TRANSFER
5318 030060 000423          BR     8$
5319 030062 004737 030246      4$:    JSR     PC,C13 ;START A SEARCH
5320 030066 000420          BR     8$ ;GO TO THE EXIT
5321 030070 112761 177777 026546   5$:    MOV     #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
5322 030076 010103          MOV     R1,R3 ;SET UP TO ADDRESS WORDS
5323 030100 006303          ASL     R3 ;CONVERT TO WORD INDEX
5324 030102 012763 060000 026610   MOV     #60000,TIMER(R3) ;START 10 SEC TIMER
5325 030110 000402          BR     7$ ;EXIT
5326 030112 004737 030762      6$:    JSR     PC,C17 ;PROCESS THE PARITY ERROR
5327 030116 032714 000100      7$:    BIT     #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
5328 030122 001002          BNE     8$ ;BR IF SET
5329 030124 004737 034160      JSR     PC,SET.IE ;SET "IE" WITHOUT A "TRE"
5330 030130 012637 177776      8$:    MOV     (SP)+,@#PS ;RESTORE PROC. STATUS
5331 030134 104413          RESREG ;RESTORE RO - R5
5332 030136 000207          RTS     PC

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

```

5344									
5345	030140	004737	034720	CI1:	JSR	PC,POPQUE			; REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
5346	030144	010237	026556		MOV	R2,TRANSW			; PUT REQ. IN TRANSFER WAIT QUEUE
5347	030150	010203			MOV	R2,R3			; DPB ADDRESS TO R3
5348	030152	013704	026644		MOV	RMADR,R4			; RMCS1 ADDRESS
5349	030156	010164	000010		MOV	R1,RMCS2(R4)			; SELECT DRIVE
5350	030162	062703	000004		ADD	#4,R3			; DESIRED WORD COUNT
5351	030166	062704	000002		ADD	#2,R4			; RMWC ADDRESS
5352	030172	012324			MOV	(R3)+,(R4)+			; LOAD WORD COUNT
5353	030174	012324			MOV	(R3)+,(R4)+			; LOAD BUFFER ADDRESS
5354	030176	012346			MOV	(R3)+,-(SP)			; LOAD SECTOR AND TRACK
5355	030200	004037	033650		JSR	RO,WRT.RM			; CALL THE LOAD(WRITE) ROUTINE
5356	030204	000006			RMDA				; INDEX OF REGISTER TO LOAD
5357	030206	030762			CI7				; ERROR RETURN ADDRESS
5358	030210	012346			MOV	(R3)+,-(SP)			; LOAD CYLINDER ADDRESS
5359	030212	004037	033650		JSR	RO,WRT.RM			
5360	030216	000034			RMDC				
5361	030220	030762			CI7				
5362	030222	016246	000002		MOV	2(R2),-(SP)			; LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
5363	030226	004037	033650		JSR	RO,WRT.RM			
5364	030232	000000			RMCS1				
5365	030234	030762			CI7				
5366	030236	010137	026630		MOV	R1,DTUW			; SET "DATA TRANSFER UNDERWAY"
5367	030242	000137	030724		JMP	CI5			
5368	030246	013704	026644	CI3:	MOV	RMADR,R4			; RMCS1 ADDRESS
5369	030252	010164	000010		MOV	R1,RMCS2(R4)			; SELECT DRIVE
5370	030256	016246	000012		MOV	12(R2),-(SP)			; DESIRED CYLINDER ADDRESS
5371	030262	004037	033650		JSR	RO,WRT.RM			
5372	030266	000034			RMDC				
5373	030270	030762			CI7				
5374	030272	116203	000010		MOVB	10(R2),R3			; PICKUP SECTOR ADDRESS
5375					SUB	MXWNDW,R3			; BACKUP BY MAX. SEARCH FOR I/O WINDOW
5376					BGE	1\$			
5377					ADD	#32,R3			
5378	030276	010346		is:	MOV	R3,-(SP)			; COMBINE THE ADJUSTED SECTOR WITH
5379	030300	042716	177740		BIC	#177740,(SP)			; CHOP OF ALL EXENTED BITS ,IF ANY
5380	030304	116266	000011	000001	MOVB	11(R2),1(SP)			; THE DESIRED TRACK
5381	030312	004037	033650		JSR	RO,WRT.RM			; LOAD DESIRED TRACK & SECTOR
5382	030316	000006			RMDA				
5383	030320	030762			CI7				
5384	030322	012746	000131		MOV	#131,-(SP)			; START A SEARCH
5385	030326	004037	033650		JSR	RO,WRT.RM			
5386	030332	000000			RMCS1				
5387	030334	030762			CI7				
5388	030336	156137	026632	026560	BISB	ATABIT(R1),SRCHWT			; SET "SEARCH WAIT" KEY
5389	030344	000567			BR	CI5			
5390	030346	013704	026644	CI4:	MOV	RMADR,R4			; RMCS1 ADDRESS
5391	030352	010164	000010		MOV	R1,RMCS2(R4)			; SELECT DRIVE
5392	030356	116203	000002		MOVB	2(R2),R3			; PICKUP THE REQUESTED COMMAND
5393	030362	122703	000131		CMPB	#131,R3			; IS IT A SEARCH COMMAND?
5394	030366	001007			BNE	1\$			; BRANCH IF NO
5395	030370	016246	000010		MOV	10(R2),-(SP)			; LOAD DESIRED TRACK & SECTOR
5396	030374	004037	033650		JSR	RO,WRT.RM			
5397	030400	000006			RMDA				
5398	030402	030762			CI7				
5399	030404	000403			BR	2\$			; GO LOAD CYLINDER

# JOB

CZRMAC0 RMO3/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY11 30(1046) 22-NOV-77 08:16 PAGE 101  
SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 5.1)-24-AUG-77

SEQ 0100

5400	030406	122703	000105	1\$:	CMPB	#105,R3	; IS IT A SEEK COMMAND
5401	030412	001007			BNE	3\$	; BRANCH IF NO
5402	030414	016246	000012	2\$:	MOV	12(R2),-(SP)	; LOAD DESIRED CYLINDER
5403	030420	004037	033650		JSR	RO,WRT.RM	
5404	030424	000034			RMDC		
5405	030426	030762			CI7		
5406	030430	000546			BR	CI6	
5407	030432	122703	000115	3\$:	CMPB	#115,R3	; IS IT AN "OFFSET" COMMAND?
5408	030436	001013			BNE	4\$	; BR IF NO
5409	030440	004037	033470		JSR	RO,RD.RM	; MERGE THE OFFSET VALUE INTO RMOF
5410	030444	000032			RMOF		; BUT DON'T CHANGE THE UPPER
5411	030446	030762			CI7		
5412	030450	116216	000001		MOVB	1(R2),(SP)	; BYTE WHEN LOADING THE
5413	030454	004037	033650		JSR	RO,WRT.RM	; REGISTER (RMOF)
5414	030460	000032			RMOF		
5415	030462	030762			CI7		
5416	030464	000530			BR	CI6	; GO START THE COMMAND
5417	030466	122703	000107	4\$:	CMPB	#107,R3	; IS IT A "RECALIBRATE" COMMAND?
5418	030472	001525			BEQ	CI6	; BRANCH IF YES
5419	030474	122703	000117		CMPB	#117,R3	; IS IT A RETURN TO CENTER?
5420	030500	001522			BEQ	CI6	; BRANCH IF YES
5421	030502	122703	000103		CMPB	#103,R3	; IS IT AN "UNLOAD" COMMAND?
5422	030506	001016			BNE	5\$	; BRANCH IF NO
5423	030510	112761	000001	026506	MOVB	#1,DRVACT(R1)	; SET THE DRIVE ACTIVE INDICATOR
5424	030516	105061	026516		CLRB	DRVSTA(R1)	; PUT DRIVE STATUS TO OFFLINE
5425	030522	112761	000001	026564	MOVB	#1,ULDFLG(R1)	; SET "UNLOAD IN PROGRESS" FLAG
5426	030530	010346			MOV	R3,-(SP)	; START THE "UNLOAD" COMMAND
5427	030532	004037	033650		JSR	RO,WRT.RM	
5428	030536	000000			RMCS1		
5429	030540	030762			CI7		
5430	030542	000207			RTS	PC	; RETURN TO USER
5431	030544	122703	000143	5\$:	CMPB	#143,R3	; IS IT A "SET FORMAT" COMMAND?
5432	030550	001014			BNE	6\$	; BRANCH IF NO
5433	030552	004037	033470		JSR	RO,RD.RM	; READ THE OFFSET REGISTER
5434	030556	000032			RMOF		
5435	030560	030762			CI7		
5436	030562	116266	000001	000001	MOVB	1(R2),1(SP)	; COMBINE "FMT22" "ECI" AND "HCI"
5437	030570	004037	033650		JSR	RO,WRT.RM	; LOAD "FMT22", "ECI", AND/OR "HCI".
5438	030574	000032			RMOF		
5439	030576	030762			CI7		
5440	030600	000436			BR	12\$	
5441	030602	122703	000141	6\$:	CMPB	#141,R3	; IS IT A "GET REGISTER" COMMAND?
5442	030606	001023			BNE	10\$	; BRANCH IF NO
5443	030610	016203	000006	7\$:	MOV	6(R2),R3	; POINTS TO 1ST ADDRESS OF WHERE
5444							; TO PUT THE REGISTER(S)
5445	030614	116237	000010	030632	MOVB	10(R2),9\$	; INIT. THE INDEX FOR THE FIRST REG.
5446	030622	116205	000011		MOVB	11(R2),R5	; INDEX OF LAST REG. TO MOVE
5447	030626	004037	033470	8\$:	JSR	RO,RD.RM	; READ RH70/RMO3 REGISTER
5448	030632	000000		9\$:	RMCS1		; INDEX OF REG. TO READ
5449	030634	030762			CI7		
5450	030636	012623			MOV	(SP)+,(R3)+	; GET THE CONTENTS OF RH70/RMO3 REG.
5451	030640	023705	030632		CMP	9\$,R5	; LAST REG. BEEN READ?
5452	030644	001414			BEQ	12\$	; GET OUT IF YES
5453	030646	062737	000002	030632	ADD	#2,9\$	; INCREASE THE INDEX BY 2
5454	030654	000764			BR	8\$	; LOOP--MORE TO READ
5455	030656	122703	000145	10\$:	CMPB	#145,R3	; IS IT A "SELECT DRIVE" COMMAND?

K08

CZRMACD RMO3/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY11 30(1046) 22-NOV-77 08:16 PAGE 102  
SINGLE/DUAL PORT RH70/RMO3 DRIVER (REV 5.1)-24-AUG-77

SEQ 0101

5456	030662	001405				BEQ	12\$		;BRANCH IF YES
5457	030664	010346			11\$:	MOV	R3,-(SP)		;LOAD THE COMMAND
5458	030666	004037	033650			JSR	RO,WRT.RM		
5459	030672	000000				RMCS1			
5460	030674	030762				CI7			
5461	030676	004737	034720		12\$:	JSR	PC,POPQUE		;REMOVE REQ. FROM QUEUE
5462	030702	052762	000200	000016		BIS	#BIT07,16(R2)		;SET THE "DONE" BIT
5463	030710	005737	026604			TST	SAVEFG		;SAVE THE RH70/RMO3 REGISTERS?
5464	030714	100002				BPL	13\$		;BRANCH IF NO
5465	030716	004737	034042			JSR	PC,SVRH70		;YES--GO SAVE THE REGISTERS
5466	030722	000207			13\$:	RTS	PC		;RETURN TO USER
5467	030724	006301			CI5:	ASL	R1		
5468	030726	012761	060000	026610		MOV	#60000,TIMER(R1)		;SET A ONE SECOND TIMER
5469	030734	006201				ASR	R1		
5470	030736	112761	000001	026506		MOV	#1,DRVACT(R1)		;SET THE DRIVE ACTIVE
5471	030744	000207				RTS	PC		;RETURN TO THE USER
5472	030746	010346			CI6:	MOV	R3,-(SP)		;LOAD THE COMMAND
5473	030750	004037	033650			JSR	RO,WRT.RM		
5474	030754	000000				RMCS1			
5475	030756	030762				CI7			
5476	030760	000761				BR	CI5		
5477	030762	032764	010000	000010	CI7:	BIT	#BIT12,RMCS2(R4)		;DRIVE NON-EXISTENT ?
5478						BNE	CI8		;BR IF YES
5479	030770	005702			1\$:	TST	R2		;ANYTHING IN QUEUE ?
5480						BEQ	CI7B		;BR IF NOT
5481	030772	001001				BNE	2\$		;BRANCH IF QUEUE IS THERE
5482	030774	000207				RTS	PC		;OTHERWISE EXIT
5483	030776	012762	104000	000016	2\$:	MOV	#BIT15!BIT11,16(R2)		;SET "PARITY" ERROR INDICATOR
5484						JSR	PC,SVRH70		;GO SAVE THE RH70/RMO3 REGISTERS
5485	031004	012746	000111		CI7B:	MOV	#111,-(SP)		;DO A "DRIVE CLEAR"
5486	031010	004037	033650			JSR	RO,WRT.RM		
5487	031014	000000				RMCS1			
5488	031016	031062				CI8			
5489	031020	004737	034602		2\$:	JSR	PC,EMPTYQ		;EMPTY THE QUEUE
5490	031024	105061	026546			CLRB	DPRQS(R1)		;CLEAR THE PORT REQUEST FLAG
5491	031030	105061	026564			CLRB	ULDFLG(R1)		;CLEAR THE UNLOAD IN QUEUE FLAG
5492	031034	105061	026506			CLRB	DRVACT(R1)		;DRIVE IS IDLE
5493	031040	020237	026556			CMP	R2,TRNSWT		;IF THIS DRIVE HAD AN I/O REQUEST
5494						CMP	R1,DTUW		;IF THIS DRIVE HAD AN I/O REQUEST
5495	031044	001005				BNE	1\$		;IN PROGRESS CLEAR ALL OF THE FLAGS
5496	031046	005037	026556			CLR	TRNSWT		
5497	031052	012737	177777	026630		MOV	#-1,DTUW		
5498	031060	000207			1\$:	RTS	PC		
5499	031062	104412			CI8:	SAVREG			;SAVE R0 - R5
5500	031064	032764	010000	000010		BIT	#BIT12,RMCS2(R4)		;IS 'NED' SET ?
5501						BNE	1\$		;BR IF YES
5502	031072	005001				CLR	R1		
5503	031074	005003				CLR	R3		
5504	031076	105761	026506		1\$:	TSTB	DRVACT(R1)		;DRIVE ACTIVE?
5505						BEQ	5\$		;BRANCH IF NO
5506	031102	001003				BNE	22\$		;BRANCH IF IN ACTIVE
5507	031104	105761	026546			TSTB	DPRQS(R1)		;PORT REQUEST
5508	031110	001443				BEQ	5\$		;BRANCH IF NOT
5509	031112	013702	026556		22\$:	MOV	TRNSWT,R2		;GET THE "TRANSFER WAIT" QUEUE
5510	031116	020137	026630			CMP	R1,DTUW		;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
5511	031122	001402				BEQ	2\$		;BRANCH IF YES

```

5512 031124 004737 034676      JSR    PC,GETREQ      ;GET THE DPB POINTER
5513 031130 005702      2$:   TST    R2          ;QUEUE ENTRY FOR DRIVE ?
5514 031132 001413      BEQ    4$             ;BR IF NOT
5515 031134 032764 010000 000010  BIT    #BIT12,RMCS2(R4) ;'NED' SET ?
5516 031142 001404      BEQ    3$             ;BR IF NOT
5517 031144 012762 100002 000016  MOV    #BIT15!BIT01,16(R2) ;SET 'DRIVE NON-EXISTENT' INDICATOR
5518 031152 000403      BR     4$             ;CONTINUE
5519 031154 012762 102000 000016  3$:   MOV    #BIT15!BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
5520 031162 012763 177777 026610  4$:   JSR    PC,SVRH70     ;SAVE RH70/RMO3 REGISTERS
5521 031170 105061 026506      MOV    #-1,TIMER(R3)  ;STOP THE TIMER
5522 031174 105061 026546      CLRB  DRVACT(R1)      ;SET "DRIVE ACTIVE" TO IDLE
5523 031200 020137 026630      CLRB  DPROS(R1)      ;CLEAR PORT REQUEST FLAG
5524 031204 001005      CMP    R1,DTUW       ;IS THIS DRIVE SETUP FOR A TRANSFER
5525 031206 012737 177777 026630  BNE   5$             ;BR IF NOT
5526 031214 005037 026556      MOV    #-1,DTUW      ;RESET THE INDICATOR
5527 031220 105061 026564      CLR   TRNSWT         ;CLEAR THE TRANSFER QUEUE
5528 031224 032764 010000 000010  5$:   CLRB  ULDFLG(R1)    ;CLEAR UNLOAD FLAG
5529 031232 005201      BIT    #BIT12,RMCS2(R4) ;'NED' SET ?
5530 031234 062703 177770      BNE   6$             ;BR IF YES
5531 031240 001314      INC   R1             ;MOVE TO THE NEXT DRIVE
5532 031244 012737 177777 026630  ADD   #2,R3
5533 031246 005037 026556      BIC   #1C7,R1
5534 031254 004737 034524 000010  BNE   1$             ;BRANCH IF MORE DRIVES
5535 031260 004737 034524 000010  MOV    #-1,DTUW      ;NO DATA TRANSFERS UNDERWAY
5536 031264 012764 000040 000010  CLR   TRNSWT         ;CLEAR THE 'TRANSFER WAIT' QUEUE
5537 031272 000406      JSR    PC,CLRQUE     ;CLEAR ALL OF THE REQUEST QUEUES
5538 031274 004737 034602 6$:   MOV    #BIT05,RMCS2(R4) ;DO A MASSBUS INIT.
5539 031300 105061 026516      BR     7$             ;CONTINUE
5540 031304 105061 026526      JSR    PC,EMPTYQ    ;CLEAR THE DRIVE'S QUEUE
5541 031310 004737 034160 7$:   CLRB  DRVSTA(R1)    ;SET DRIVE TO OFFLINE
5542 031314 104413      CLRB  DRVTP(R1)     ;CLEAR THE DRIVE TYPE INDICATOR
5543 031316 000207      JSR    PC,SET.IE    ;SET "IE" WITHOUT "TRE"
5544 031316 000207      RESREG  PC          ;RESTORE R0 - R5
5545 031316 000207      RTS   PC            ;RETURN
5546
5547 ;LOOK AHEAD ROUTINE
5548 ;CALL
5549 ;
5550 ;
5551 ;
5552 ;
5553 ;
5554 ;
5555 ;
5556 ;
5557 LA:  MOV    RMADR,R4      ;GET RMCS1'S ADDRESS
5558 MOV    R1,RMCS2(R4)  ;SELECT DRIVE
5559 JSR    RO,RD.RM      ;READ DRIVE STATUS
5560 RMDS  4$             ;ERROR RETURN ADDRESS
5561 4$    ;
5562 BIC   #1C020200,(SP) ;ON CYLINDER ?
5563 CMP   #200,(SP)+    ;PIP=0,DRY=1?
5564 BNE   3$             ;NO
5565 INCB  LACNT(R1)      ;INCREMENT THE LOOK AHEAD COUNT
5566 CMPB  LACNT(R1),MXLACT ;EXCEED MAX?
5567 BGT   2$             ;BRANCH IF YES

```

```

5568 031366 116203 000010          MOVB    10(R2),R3      ;GET DESIRED SECTOR ADDRESS AND
5569 031372 000303          SWAB    R3           ;MULT. BY 64--ALIGN WITH
5570 031374 006203          ASR     R3           ;LOOK AHEAD REGISTER
5571 031376 006203          ASR     R3
5572 031400 012737 000340 177776      MOV     #340,R#PS    ;PRIORITY LEVEL "7"
5573 031406 004037 033470 6S:      JSR     RD,RD.RM     ;READ LOOK AHEAD REGISTER
5574 031412 000020          RMLA   4S
5575 031414 031466          4S
5576 031416 021664 000020          CMP     (SP),RMLA(R4) ;CORRECT LA NUMBER ?
5577 031422 001402          BEQ    7S           ;YES
5578 031424 005726          TST    (SP)+        ;NO,CLEAR STACK
5579 031426 000415          BR     3S
5580 031430 162603 7S:      SUB     (SP)+,R3     ;CALCULATE THE DELTA
5581 031432 002002          BGE    1S
5582 031434 062703 004000          ADD     #(<32.*64.>),R3 ;MAKE THE DELTA POSITIVE
5583 031440 023703 026654 1S:      CMP     MXDLTA,R3   ;CHECK THE DELTA TO SEE
5584 031444 002406          BLT    3S           ;IF IT IS WITHIN THE
5585 031446 023703 026656          CMP     MNDLTA,R3  ;WINDOW---IF YES, ZERO
5586 031452 002003          BGE    3S           ;THE LOOK AHEAD COUNT
5587 031454 105061 026574 2S:      CLRB   LACNT(R1)   ;AND TAKE THE I/O EXIT
5588 031460 005720          TST    (R0)+
5589 031462 005720 3S:      TST    (R0)+        ;ADJUST THE RETURN ADDRESS
5590 031464 000402          BR     5S
5591 031466 004737 030762 4S:      JSR     PC,C17      ;PROCESS THE ERROR
5592 031472 000200 5S:      RTS     RD          ;RETURN
5593
5594          ;INTERRUPT SERVICE ROUTINE
5595
5596 031474 112737 000001 026562 ISR:    MOVB    #1,ACTDRV   ;SET "ACTIVE DRIVER" FLAG
5597 031502 104412          SAVREG ;SAVE R0 - R5
5598 031504 013704 026644          MOV     RMADR,R4   ;ADDRESS OF RHSCS1
5599 031510 013701 026630          MOV     DTUW,R1    ;GET "DATA TRANSFER UNDERWAY" INDICATOR
5600 031514 002403          BLT    1S           ;BRANCH IF NO DATA TRANSFER UNDERWAY
5601 031516 004737 031540          JSR     PC,TD       ;CALL TRANSFER DONE
5602 031522 000402          BR     2S
5603 031524 004737 031710 1S:      JSR     PC,SC       ;CALL SPECIAL CONDITIONS
5604 031530 104413 2S:      RESREG ;RESTORE R0 - R5
5605 031532 105037 026562          CLRB   ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
5606 031536 000002          RTI
5607
5608          ;TRANSFER DONE ROUTINE
5609
5610 031540 105061 026506 026630 TD:    CLRB   DRVACT(R1)  ;SET DRIVE ACTIVE INDICATOR TO IDLE
5611 031544 012737 177777          MOV     #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
5612 031552 006301          ASL    R1
5613 031554 012761 177777 026610          MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
5614 031562 006201          ASR    R1
5615 031564 013702 026556          MOV     TRNSWT,R2  ;GET "DPB" ADDRESS FROM THE
5616 031570 005037 026556          CLR    TRNSWT     ;TRANSFER WAIT QUEUE--CLEAR QUEUE
5617 031574 052762 000200 000016          BIS    #BIT07,16(R2) ;SET DONE
5618 031602 010164 000010          MOV     R1,RMCS2(R4) ;SELECT THE DRIVE
5619 031606 004037 033470          JSR     RD,RD.RM   ;TRANSFER ERROR(TRE=1)?
5620 031612 000000          RMCS1
5621 031614 030762          C17
5622 031616 006126          ROL    (SP)+
5623 031620 100417          BMI    3S         ;BR IF YES

```



```

5624 031622 005737 026604          TST    SAVEFG      ;SAVE THE RH70/RMO3 REGISTERS?
5625 031626 100002                    BPL    1$          ;BRANCH IF NO
5626 031630 004737 034042          JSR    PC,SVRH70  ;YES--SAVE THE REGISTERS
5627 031634                    1$:
5628 031634 004737 034676          JSR    PC,GETREQ  ;GET DPB POINTER
5629 031640 005702                    TST    R2          ;ENTRY FOR DRIVE ?
5630 031642 001403                    BEQ    2$          ;BR IF NOT
5631 031644 004737 027702          JSR    PC,OPT     ;CALL OPTIMIZER
5632 031650 000417                    BR     SC          ;CHECK OTHER DRIVES
5633                    ;THE RELEASE DRIVE COMMAND IS FORECD TO ENTER FOR DUAL PORT OPERATION
5634 031652 012714 000113          2$: MOV    #113,(R4) ;RELEASE THE DRIVE
5635 031656 000414                    BR     SC          ;CHECK FOR OTHER DRIVES
5636 031660 052762 100100          000016 3$: BIS    #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
5637 031666 004737 034602          JSR    PC,EMPTYQ  ;EMPTY THE "DRIVE'S WAIT" QUEUE
5638 031672 004737 034042          JSR    PC,SVRH70  ;SAVE THE RH70/RMO3 REGISTERS
5639 031676 012714 040111          MOV    #40111,(R4) ;ISSUE A "DRIVE CLEAR"
5640 031702 012714 000113          MOV    #113,(R4)  ;ISSUE A RELEASE TO THE DRIVE
5641 031706 000400                    BR     SC          ;CHECK FOR OTHER DRIVES
5642
5643
5644
5645                    ;SPECIAL CONDITION ROUTINE
5646
5647 031710 116403 000016          SC:   MOVB   RMAS(R4),R3 ;READ "RMAS"
5648 031714 001014                    BNE   2$          ;BRANCH IF ANY 'ATA' BITS SET
5649 031716 004037 033470          JSR    RO,RD.RM   ;READ CONTROL AND STATUS REGISTER
5650 031722 000000                    RMCS1
5651                    ;
5652 031724 031744                    ;
5653 031726 106126                    1$   IS          ;EXIT IF FAIL TO READ
5654 031730 100405                    ROLB  (SP)+      ;IS "IE"=1?
5655 031732 004037 034766          BMI   1$          ;YES, NO DRIVES TO CHECK
5656 031736 104001                    JSR    RC,ES.SAV  ;SAVE THE ADDRESS IN 'SESCAPE'
5657 031740 004737 034160          ERROR 1          ;REPORT AN ILLEGAL INTERRUPT
5658 031744 000207                    JSR    PC,SET.IE ;SET INTERRUPT ENABLE
5659 031746 005046                    1$: RTS    PC     ;RETURN
5660 031750 110316                    2$: CLR    -(SP)  ;PROCESS ALL DRIVES THAT HAVE
5661 031752 012703 000001          MOV    #1,R3     ;AN "ATA"=1
5662 031756 005001                    CLR    R1
5663 031760 030316                    SC3: BIT    R3,(SP) ;ATA=1?
5664 031762 001005                    BNE   SC5        ;YES--BRANCH
5665 031764 005201                    SC4: INC    R1    ;MOVE TO THE NEXT DRIVE
5666 031766 106303                    ASLB  R3
5667 031770 001373                    BNE   SC3        ;BRANCH IF MORE TO CHECK?
5668 031772 005726                    TST   (SP)+      ;CLEAN OFF THE STACK
5669 031774 000207                    RTS    PC        ;RETURN TO USER
5670 031776 105761 026536          SC5: TSTB   DPINT(R1) ;INITIALIZING THE DRIVE ?
5671 032002 001402                    BEQ   1$          ;BR IF NOT
5672 032004 000137 032720          JMP    SC13      ;PROCESS THE DRIVE
5673 032010 105761 026546          1$: TSTB   DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
5674 032014 001402                    BEQ   2$          ;BR IF NOT
5675 032016 000137 032720          JMP    SC13      ;START THE OUTSTANDING COMMAND
5676 032022 105761 026516          2$: TSTB   DRVSTA(R1) ;CHECK THE DRIVE STATUS
5677 032026 003025                    BGT   5$          ;BRANCH IF ONLINE
5678 032030 105761 026564          TSTB   ULDFLG(R1) ;UNLOAD IN PROGRESS?
5679 032034 003422                    BLE   5$          ;BRANCH IF NOT

```

5680	032036	004737	034676		JSR	PC,GETREQ	:GET DPB POINTER
5681	032042	004737	034042		JSR	PC,SVRH70	:SAVE THE RM70/RM03 REGISTERS
5682	032046	004737	032656		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
5683							:ALSO DO A DRIVE INIT (DRVINT)
5684	032052	105761	026516		TSTB	DRVSTA(R1)	:DID DRIVE COME ONLINE?
5685	032056	003416			BLE	6\$	:NO---BRANCH
5686	032060	032737	040000	026476	BIT	#BIT14,RMERRS	:WAS THERE AN ERROR?
5687	032066	001002			BNE	3\$	:BR IF ERROR
5688	032070	000137	032546		JMP	SC11	:NO ERROR
5689	032074	013705	026500		MOV	RMERRS+2,R5	:YES -- PICKUP RMER1 AND
5690	032100	000504			BR	SC6A	:GO PROCESS THE ERROR
5691	032102	105761	026506		TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
5692	032106	001033			BNE	SC6	:BR IF EITHER
5693	032110	004737	032656		JSR	PC,SC12	:SAVE RMD5, RMER1, RMER2, AND RMMR2
5694							:ALSO DO A DRVINT
5695	032114	105761	026536		TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
5696	032120	001321			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
5697	032122	105761	026516		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
5698	032126	100413			BMI	7\$	:BR IF UNSAFE
5699	032130	032737	020000	026502	BIT	#BIT13,RMERRS+4	:ADDRESS PLUG CHANGED ?
5700	032136	001013			BNE	8\$	:BR IF YES
5701					MOV	#113,-(SP)	:RELEASE COMMAND
5702	032140	012746	000111		MOV	#111,-(SP)	:DRIVE CLEAR
5703	032144	004037	033650		JSR	RD,WRT.RM	:WRITE THE COMMAND INTO RMCS1
5704	032150	000000			RMCS1		:REGISTER INDEX
5705	032152	032516			SCB		:PARITY EXIT ADDRESS
5706	032154	011605			MOV	(SP),R5	:PICKUP (RMS) BEFORE THE ERROR CALL
5707	032156	004037	034766		JSR	RD,ES.SAV	:SAVE THE ADDRESS IN 'SESCAPE'
5708	032162	104002			ERROR	2	:REPORT THE UNEXPECTED ATTENTION
5709	032164	000677			BR	SC4	:GO CHECK FOR MORE ATA'S
5710	032166						
5711	032166	004037	034766		JSR	RD,ES.SAV	:SAVE THE ADDRESS IN 'SESCAPE'
5712	032172	104005			ERROR	5	:REPORT THE ADDRESS PLUG CHANGE
5713	032174	000673			BR	SC4	:CHECK FOR MORE DRIVES
5714	032176	006301			ASL	R1	:SETUP TO ADDRESS WORDS
5715	032200	012761	177777	026610	MOV	#-1,TIMER(R1)	:STOP THE TIMER
5716	032206	006201			ASR	R1	:RESTORE THE DRIVE ADDRESS
5717	032210	004737	034676		JSR	PC,GETREQ	:GET THE DPB POINTER FROM THE QUEUE
5718	032214	010164	000010		MOV	R1,RMCS2(R4)	:SELECT DRIVE
5719	032220	000137	032546		JMP	SC11	:PROCESS THE SEARCH
5720	032224	004037	033470		JSR	RD,RD.RM	:READ THE RM03'S STATUS REG.
5721	032230	000012			RMD5		
5722	032232	032516			SCB		
5723	032234	011605			MOV	(SP),R5	:AND PUT IT IN R5
5724	032236	006126			ROL	(SP)+	:WAS THERE AN ERROR?
5725	032240	100407			BMI	1\$	:BR IF ERROR
5726	032242	105761	026506		TSTB	DRVACT(R1)	:CHECK DRIVE'S STATE
5727	032246	003137			BGT	SC11	:BR IF DRIVE ACTIVE WITH ORDER
5728	032250	052762	100210	000016	BIS	#BIT15!BIT07!BIT03,16(R2)	:INFORM USER OF ERROR RECOVER COMPLETION
5729	032256	000470			BR	SC7	
5730	032260	004037	033470		JSR	RD,RD.RM	:READ ERROR REGISTER #1
5731	032264	000014			RMER1		
5732	032266	032516			SCB		
5733	032270	012605			MOV	(SP)+,R5	:AND SAVE IT IN R5
5734	032272	004737	034042		JSR	PC,SVRH70	:SAVE RM70/RM03 REGISTERS
5735	032276	012746	000111		MOV	#111,-(SP)	:ISSUE A DRIVE CLEAR

5736	032302	004037	033650			JSR	RO,WRT.RM	
5737	032306	000000				RMCS1		
5738	032310	032516				SCB		
5739	032312	006105			SC6A:	ROL	R5	; WAS "UNSAFE" CONDITION =1?
5740	032314	100406				BMI	1\$	; BRANCH IF YES
5741	032316	005702				TST	R2	; ANYTHING IN QUEUE ?
5742	032320	001447				BEG	SC7	; BR IF NOT
5743	032322	052762	100240	000016		BIS	#BIT15!BIT07!BIT05,16(R2)	; INFORM USER OF ERROR
5744	032330	000443				BR	SC7	
5745	032332	004037	033470		1\$:	JSR	RO,RO.RM	; READ DRIVE STATUS REG. #1
5746	032336	000012				RMDS		
5747	032340	032516				SCB		
5748	032342	011605				MOV	(SP),R5	; SAVE RMDS IN R5
5749	032344	006126				ROL	(SP)+	; "ERR"=1?
5750	032346	100011				BPL	2\$	; BR IF NO--UNSAFE CLEARED
5751	032350	112761	177777	026516		MOVB	#-1,DRVSTA(R1)	; DRIVE IS UNSAFE
5752	032356	004737	034042			JSR	PC,SVRH70	; SAVE RM70/RM03 REGISTERS
5753	032362	052762	110000	000016		BIS	#BIT15!BIT12,16(R2)	; INFORM USER OF UNSAFE ERROR
5754	032370	000423				BR	SC7	
5755	032372	032705	010000		2\$:	BIT	#BIT12,R5	; "MOL" = 1 ?
5756	032376	001015				BNE	3\$	; BR IF YES
5757	032400	112761	177777	026506		MOVB	#-1,DRVACT(R1)	; ACTIVE ERROR RECOVER
5758	032406	112761	000001	026516		MOVB	#1,DRVSTA(R1)	; ONLINE
5759	032414	006301				ASL	R1	
5760	032416	012761	072460	026610		MOV	#30000.,TIMER(R1)	; START 30 SECOND TIMER
5761	032424	006201				ASR	R1	
5762	032426	000137	031764			JMP	SC4	
5763	032432	052762	100220	000016	3\$:	BIS	#BIT15!BIT07!BIT04,16(R2)	; INFORM USER OF ERROR
5764	032440	105061	026506		SC7:	CLRB	DRVACT(R1)	; DRIVE IS IDLE
5765						JSR	PC,EMPTYQ	; DUMP THE QUEUE
5766	032444	004737	034720			JSR	PC,POPQUE	; REMOVE THE QUEUE
5767	032450	105761	026564			TSTB	ULDFLG(R1)	; UNLOAD IN PROGRESS OR QUEUE?
5768	032454	003002				BGT	1\$	; BR IF NOT
5769	032456	105061	026564			CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
5770	032462	116164	026632	000016	1\$:	MOVB	ATABIT(R1),RMAS(R4)	; CLEAR ATTENTION BIT
5771	032470	105761	026516			TSTB	DRVSTA(R1)	; IS THE DRIVE UNSAFE ?
5772	032474	100406				BMI	2\$	; BR IF IT IS
5773						MOV	#113,-(SP)	; RELEASE COMMAND
5774	032476	012746	000111			MOV	#111,-(SP)	; DRIVE CLEAR COMMAND
5775	032502	004037	033650			JSR	RO,WRT.RM	; WRITE THE COMMAND INTO RPCS1
5776	032506	000000				RMCS1		; REGISTER INDEX
5777	032510	032516				SCB		; PARITY EXIT ADDRESS
5778	032512	000137	031764		2\$:	JMP	SC4	; CHECK FOR MORE DRIVES
5779	032516	105761	026506		SC8:	TSTB	DRVACT(R1)	; IS DRIVE IDLE?
5780	032522	001405				BEG	1\$	; YES--BRANCH
5781	032524	004737	034676			JSR	PC,GETREQ	; GET DPB POINTER
5782	032530	004737	030762			JSR	PC,CI7	; PROCESS THE PARITY ERROR
5783	032534	000402				BR	2\$	; CONTINUE
5784	032536				1\$:			
5785						JSR	PC,CI7	; PROCESS THE PARITY ERROR
5786	032536	004737	031004			JSR	PC,CI7B	; PROCESS THE UNCORRECTABLE PARITY ERROR
5787	032542	000137	031764		2\$:	JMP	SC4	; CHECK MORE DRIVES
5788	032546	105761	026564		SC11:	TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS"?
5789	032552	003402				BLE	1\$	; BRANCH IF NO
5790	032554	105061	026564			CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
5791	032560	105061	026506		1\$:	CLRB	DRVACT(R1)	; SET DRIVE IDLE

```

5792 032564 136137 026632 026560 BITB ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
5793 ;AN I/O COMMAND?
5794 032572 001012 BNE 2$ ;BRANCH IF YES
5795 032574 004737 034720 JSR PC,POPQUE ;REMOVE REQUEST FROM QUEUE
5796 032600 052762 000200 000016 BIS #BIT07,16(R2) ;SET "DONE" BIT
5797 032606 005737 026604 TST SAVEFG ;SAVE THE REGISTERS?
5798 032612 100002 BPL 2$ ;BRANCH IF NO
5799 032614 004737 034042 JSR PC,SVRH70 ;YES--SAVE ALL OF THE RH70/RM03 REG'S
5800 032620 116164 026632 000016 2$: MOVB ATABIT(R1),RMAS(R4) ;CLEAR ATTENTION BIT
5801 032626 146137 026632 026560 BICB ATABIT(R1),SRCHWT ;CLEAR IMPLIED SEEK SET
5802 032634 006301 ASL R1 ;WORD INDEX
5803 032636 012761 177777 026610 MOV #-1,TIMER(R1) ;STOP CLOCK
5804 032644 006201 ASR R1 ;RESTORE R1
5805 032646 004737 027702 JSR PC,OPT ;START A REQUEST
5806 032652 000137 031764 JMP SC4 ;CHECK FOR MORE DRIVES
5807 032656 010164 000010 SC12: MOV R1,RMCS2(R4) ;SELECT DRIVE
5808 032662 016437 000012 026476 MOV RMD5(R4),RMERRS ;SAVE THE FOUR REGISTERS THAT
5809 032670 016437 000014 026500 RMER1(R4),RMERRS+2 ;WILL TELL US SOMETHING
5810 032676 016437 000042 026502 MOV RMER2(R4),RMERRS+4
5811 032704 016437 000040 026504 MOV RMMR2(R4),RMERRS+6
5812 ; JSR RO,DRVINT ;INIT. THE STATE OF THE DRIVE
5813 ; BR 1$ ;TAKE ERROR EXIT
5814 032712 000207 RTS PC ;RETURN
5815 032714 005726 1$: TST (SP)+ ;POP PC OFF OF THE STACK
5816 032716 000677 BR SC8 ;PROCESS THE PARITY ERROR
5817 032720 006301 SC13: ASL R1 ;SETUP TO ADDRESS WORDS
5818 032722 012761 177777 026610 MOV #-1,TIMER(R1) ;STOP THE TIMER
5819 032730 006201 ASR R1
5820 032732 010164 000010 MOV R1,RMCS2(R4) ;SELECT THE DRIVE
5821 032736 116164 026632 000016 MOVB ATABIT(R1),RMAS(R4) ;CLEAR THE ATTENTION BIT
5822 032744 105761 026536 1$: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
5823 032750 001424 BEQ 2$ ;BR IF NOT
5824 032752 105061 026536 CLRB DPINT(R1) ;CLEAR THE INIT INDICATOR
5825 032756 004037 027074 JSR RO,DRVINT ;GO INIT THE DRIVE
5826 032762 000240 NOP ;DUMMY PARITY ERROR RETURN
5827 032764 105761 026516 TSTB DRVSTA(R1) ;DRIVE ONLINE ?
5828 032770 003014 BGT 2$ ;BR IF YES -- START ORDER
5829 032772 005702 TST R2 ;QUEUE ENTRY FOR THE DRIVE
5830 032774 001426 BEQ 3$ ;BR IF NOT
5831 032776 004737 034676 JSR PC,GETREQ ;GET DPB ADDRESS
5832 033002 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
5833 033010 004737 034042 JSR PC,SVRH70 ;SAVE THE REGISTERS
5834 ; JSR PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
5835 033014 004737 034720 JSR PC,POPQUE ;REMOVE THE QUEUE
5836 033020 000414 BR 3$
5837 033022 032764 004000 000000 2$: BIT #BIT11,RMCS1(R4) ;DVA SET ?
5838 033030 001006 BNE 4$ ;SET THEN CALL OPT
5839 033032 006301 ASL R1
5840 033034 012761 060000 026610 MOV #60000,TIMER(R1)
5841 033042 006201 ASR R1
5842 033044 000402 BR 3$
5843 033046 004737 027702 4$: JSR PC,OPT ;START THE PENDING REQUEST
5844 033052 000137 031764 3$: JMP SC4 ;PROCESS OTHER DRIVES
5845
5846 ;/RM03 TIMER ROUTINE
5847 ;CALL

```

```

5848      ;      MOV      #TIME, -(SP)      ; ELAPSED TIME IN MILLISECONDS ON THE STACK
5849      ;      JSR      PC, RPTMR          ; CALL RMO3 TIME ROUTINE
5850
5851 033056 005737 026562      RPTMR: TST      ACTDRV          ; CHECK "ACTDRV & ACTSTR"
5852 033062 001027              BNE      4$              ; IF NON ZERO EXIT
5853 033064 112737 000001 026563      MOVB     #1, ACTSTR      ; SET "ACTSTR"
5854 033072 104412              SAVREG                    ; SAVE R0 - R5
5855 033074 005001              CLR      R1              ; START WITH DRIVE 0
5856 033076 005003              CLR      R3
5857 033100 005763 026610      1$: TST      TIMER(R3)    ; IS THE TIMER RUNNING?
5858 033104 002406              BLT     2$              ; BRANCH IF NO
5859 033106 166663 000002 026610      SUB     2(SP), TIMER(R3) ; COUNT THE INTERVAL
5860 033114 003002              BGT     2$              ; BR IF NO SOFTWARE TIMEOUT
5861 033116 004737 033146      JSR     PC, STO          ; CALL SOFTWARE TIMEOUT ROUTINE
5862 033122 005201      2$: INC     R1              ; MOVE TO NEXT DRIVE
5863 033124 005723              TST     (R3)+
5864 033126 022701 000010      CMP     #8., R1         ; OUT OF DRIVES?
5865 033132 003362              BGT     1$              ; BRANCH IF NO
5866 033134 104413      3$: RESREG                    ; RESTORE R0 - R5
5867 033136 105037 026563      CLRB   ACTSTR          ; ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
5868 033142 012616      4$: MOV     (SP)+, (SP)    ; ADJUST THE STACK
5869 033144 000207      RTS     PC              ; RETURN
5870
5871      ; SOFTWARE TIMEOUT ROUTINE
5872
5873      ; NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
5874      ; OR GREATER
5875
5876      ; CALL:
5877      ;      MOV     #DRVNUM, R1          ; DRIVE NUMBER
5878      ;      JSR     PC, STO              ; CALL
5879      ;      RETURN
5880
5881 033146 010146      STO:  MOV     R1, -(SP)      ; SAVE R1
5882 033150 010246      MOV     R2, -(SP)      ; SAVE R2
5883 033152 010346      MOV     R3, -(SP)      ; SAVE R3
5884 033154 010446      MOV     R4, -(SP)      ; SAVE R4
5885 033156 013704 026644      MOV     RMADR, R4      ; GET ADDRESS OF "RMCS1"
5886 033162 010164 000010      MOV     R1, RMCS2(R4)  ; SELECT THE DRIVE
5887 033166 004037 033470      JSR     R0, RD.RM      ; READ "DRIVE STATUS REG"
5888 033172 000012
5889      ;
5890 033174 033456      ;
5891 033176 105726      STOS   ST05
5892 033200 100436      ST09   ST09
5893 033202 105761 026536      ST01: TSTB   (SP)+      ; IS "DRY"=1?
5894 033206 001033      BMI    DPINT(R1)      ; BR IF YES
5895 033210 105761 026546      TSTB   ST02           ; TRYING TO INITIALIZE THE DRIVE ?
5896 033214 001030      BNE    DPRQS(R1)      ; BR IF YES
5897 033216 013702 026556      MOV     TRANSW, R2     ; OUTSTANDING PORT REQUEST FOR THE DRIVE ?
5898 033222 020137 026630      MOV     ST02           ; BR IF YES
5899 033226 001404      TRANSW, R2           ; PICKUP TRANSFER WAIT QUEUE
5900 033230 000137 033456      CMP     R1, DTUW       ; TRANSFER UNDERWAY ON THIS DRIVE?
5901 033234 004737 034676      BEQ    1$             ; BRANCH IF YES
5902 033240 052762 101000 000016 1$: JMP     ST09           ; IF NOT DON'T BOTHER DRIVES
5903 033246 004737 034042      JSR     PC, GETREQ     ; GET DPB ADDRESS
5904      BIS     #BIT15!BIT09, 16(R2) ; SET THE ERROR FLAGS
5905      JSR     PC, SVRH70   ; SAVE RH70/RM03 REGISTERS

```

```

5904 ; MOV #BIT05,RMCS2(R4) ;"INIT" THE MASS BUS
5905 033252 105061 026506 CLRB DRVACT(R1) ;DRIVE IS IDLE
5906 033256 105061 026564 CLRB ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
5907 033262 005037 026556 CLR TRNSWT ;CLEAR DPB ADDRESS
5908 033266 012737 177777 026630 MOV #-1,DTUW ;CLEAR THE TRANSFER DRIVE #
5909 033274 000470 ST09 ;DON'T BOTHER OTHER DRIVES
5910 033276 116405 000016 ST02: MOV RMA5(R4),R5 ;READ ATTENTION REG
5911 033302 136105 026632 BITB ATABIT(R1),R5 ;IS ATTENTION FOR THIS DRIVE UP ?
5912 033306 001007 ST03 ;YES--BRANCH
5913 033310 105761 026536 TSTB DPINT(R1) ;TRYING TO INITIALIZE THE DRIVE ?
5914 033314 001021 BNE ST06 ;BR IF YES - DRIVE NOT ONLINE
5915 033316 105761 026546 TSTB DPRQS(R1) ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
5916 033322 001035 BNE ST07 ;BR IF YES - NO RESPONSE TO REQUEST
5917 033324 000454 BR ST09 ;OTHER WISE EXIT
5918 033326 105761 026536 ST03: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
5919 033332 001003 BNE IS ;BR IF INIT PENDING
5920 033334 105761 026546 TSTB DPRQS(R1) ;PORT REQUEST PENDING ?
5921 033340 001446 BEQ ST09 ;BR IF NOT
5922 033342 012763 177777 026610 IS: MOV #-1,TIMER(R3) ;STOP THE TIMER
5923 033350 000442 BR ST09 ;EXIT
5924 033352 004737 031062 ST05: JSR PC,CIB ;GO HANDLE THE PARITY ERROR
5925 033356 000437 BR ST09
5926 033360 105061 026536 ST06: CLRB DPINT(R1) ;CLEAR THE INITIALIZE INDICATOR
5927 033364 105061 026516 CLRB DRVSTA(R1) ;SET UNIT OFFLINE
5928 033370 012763 177777 026610 MOV #-1,TIMER(R3) ;STOP THE TIMER
5929 033376 004737 034676 JSR PC,GETREQ ;GET THE DPB ADDRESS
5930 033402 005702 TST R2 ;REQUEST IN QUEUE ?
5931 033404 001424 BEQ ST09 ;BR IF NOT
5932 033406 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
5933 033414 000414 BR ST08 ;FINISH
5934 033416 012763 177777 026610 ST07: MOV #-1,TIMER(R3) ;STOP THE TIMER
5935 033424 105061 026546 CLRB DPRQS(R1) ;CLEAR PORT REQUEST INDICATOR
5936 033430 004737 034676 JSR PC,GETREQ ;GET DPB ADDRESS
5937 033434 005702 TST R2 ;QUEUE ENTRY FOR DRIVE ?
5938 033436 001407 BEQ ST09 ;BR IF NONE
5939 033440 012762 100004 000016 MOV #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
5940 033446 004737 034602 ST08: JSR PC,EMPTYQ ;CLEAR THE QUEUE FOR THE DRIVE
5941 033452 004737 034042 JSR PC,SVRH70 ;SAVE THE REGISTERS
5942 033456 012604 ST09: MOV (SP)+,R4 ;RESTORE R4
5943 033460 012603 MOV (SP)+,R3 ;RESTORE R3
5944 033462 012602 MOV (SP)+,R2 ;RESTORE R2
5945 033464 012601 MOV (SP)+,R1 ;RESTORE R1
5946 033466 000207 RTS PC ;RETURN
5947
5948 ;ROUTINE TO READ A RH70/RM03 REGISTER
5949
5950 ;CALL
5951 JSR RO,RD,RM ;GO READ A REGISTER
5952 INDEX ;REG. INDEX FROM BASE
5953 ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
5954 ;AT THIS ADDRESS
5955 RETURN ;CONTENTS OF REG. IS ON THE STACK
5956
5957 033470 013737 026642 033636 RD,RM: MOV MCPMX,RD,RM2 ;MAX. RETRYS ALLOWED
5958 033476 011646 MOV (SP)-,(SP) ;SAVE RD FOR RETURN
5959 033500 013737 026644 033514 MOV RMADR,RD,ADR ;FORM THE DESIRED ADDRESS

```

```

5960 033506 062037 033514          ADD      (RO)+,RD.ADR      ; USING THE BASE AND THE INDEX
5961 033512 013727          RD.RM1: MOV      @ (PC)+,(PC)+ ; READ THE DESIRED REGISTER OF THE RMO3
5962 033514 000000          RD.ADR: .WORD    0          ; ADDRESS IS FORMED HERE
5963 033516 000000          RD.WRD: .WORD    0          ; REG. CONTENTS PUT HERE
5964 033520 013766 033516 000002      MOV      RD.WRD,2(SP)      ; RETURN IT TO THE USER
5965 033526 013746 026644      MOV      RMADR,-(SP)      ; PUT THE ADDRESS ON THE STACK
5966 033532 062716 000010      ADD      #RMCS2,(SP)     ; FORM THE ADDRESS OF RMCS2
5967 033536 032736 010000      BIT      #BIT12,@(SP)+   ; CHECK THE 'NED' BIT
5968 033542 001037          BNE      RD.RM3          ; BR IF DRIVE NON-EXISTENT
5969 033544 017746 173074      MOV      @RMADR,-(SP)    ; READ RMCS1
5970 033550 032716 020000      BIT      #BIT13,(SP)    ; DID MCPE SET?
5971 033554 001002          BNE      1$             ; BRANCH IF YES
5972 033556 022620          CMP      (SP)+,(RO)+    ; ADJUST FOR RETURN
5973 033560 000432          BR      RD.RM4          ; EXIT
5974 033562          1$:
5975 033562 004037 034766          JSR      RO,ES.SAV      ; SAVE THE ADDRESS IN '$ESCAPE'
5976 033566 104003          ERROR   3              ; REPORT "MCPE" ERROR
5977 033570 005737 026630      TST      DTUW          ; DATA TRANSFER UNDERWAY?
5978 033574 100405          BMI     2$             ; NO--BRANCH
5979 033576 032716 040000      BIT      #BIT14,(SP)    ; NO--"TRE"=1?
5980 033602 001402          BEQ     2$             ; NO--BRANCH
5981 033604 005726          TST      (SP)+         ; YES--CLEAN OFF THE STACK AND
5982 033606 000415          BR      RD.RM3        ; TAKE THE FATAL ERROR EXIT
5983 033610 052716 040000      2$:      BIS      #BIT14,(SP)  ; CLEAR "MCPE" BY SENDING A "1" TO "TRE"
5984 033614 000316          SWAB    (SP)          ; POSITION BEFORE WRITING
5985 033616 013737 026644 033632      MOV      RMADR,3$      ; FORM ADDRESS OF HIGH BYTE
5986 033624 005237 033632      INC     3$            ;
5987 033630 112637          MOV     (SP)+,@(PC)+   ; WRITE THE HIGH BYTE OF RMCS1
5988 033632 000000          3$:      .WORD    0          ; ADDRESS STORAGE
5989 033634 005327          DEC     (PC)+         ; EXCEEDED MAX. RETRYS
5990 033636 000003          RD.RM2: .WORD    3          ;
5991 033640 002324          BGE     RD.RM1        ; BRANCH IF NO
5992 033642 011000          RD.RM3: MOV      (RO),RO ; FATAL ERROR EXIT
5993 033644 012616          RD.RM4: MOV      (SP)+,(SP) ;
5994 033646 000200          RTS      RD.RM4      ;
5995
5996          ; ROUTINE TO WRITE A REGISTER
5997          ;
5998          ; CALL
5999          ;
6000          ;
6001          ;
6002          ;
6003          ;
6004          ;
6005 033650 013737 026642 034026      WRT.RM: MOV      MCPMX,WRT.R2 ; MAX RETRYS ALLOWED
6006 033656 016637 000002 033736      MOV      2(SP),WRT.WD  ; SAVE THE WORD TO WRITE
6007 033664 012616          MOV      (SP)+,(SP)    ; ADJUST THE STACK
6008 033666 012037 033740      MOV      (RO)+,WRT.AD  ; GET INDEX OF REGISTER TO BE WRITTEN
6009 033672 001015          BNE     1$             ; BRANCH IF NOT RMCS1
6010 033674 122737 000150 033736      CMP     #150,WRT.WD    ; IS THE COMMAND FOR DATA TRANSFERS?
6011 033702 002411          BLT     1$             ; YES--DON'T GET THE OLD A16 & A17, & PSEL
6012 033704 004037 033470      JSR      RO,RD.RM      ; NO---COMBINE A16&A17, & PSEL WITH
6013 033710 000000          RMCS1  ; THE COMMAND BEFORE SENDING IT TO
6014 033712 034032          WRT.R3  ; THE RH70/RMO3
6015 033714 000316          SWAB    (SP)

```

```

6016 033716 042716 177770      BIC      #1C7,(SP)
6017 033722 112637 033737      MOV      (SP)+,WRT.WD+1
6018 033726 063737 026644 033740 1$:      ADD      RMADR,WRT.AD      ;FORM THE ADDRESS OF THE DISK REG.
6019 033734 012737      MOV      (PC)+,(PC)+      ;LOAD THE DESIRED REG.
6020 033736 000000      WRT.WD: .WORD 0          ;WORD TO WRITE GOES HERE
6021 033740 000000      WRT.AD: .WORD 0          ;ADDRESS IS FORMED HERE
6022 033742 013746 026644      MOV      RMADR,-(SP)      ;PUT THE ADDRESS ON THE STACK
6023 033746 062716 000010      ADD      #RMCS2,(SP)     ;FORM THE ADDRESS OF RMCS2
6024 033752 032736 010000      BIT      #BIT12,(SP)+    ;CHECK THE 'NED' BIT
6025 033756 001025      BNE      WRT.R3          ;BR IF DRIVE NON-EXISTENT
6026 033760 004037 033470      JSR      RO,RO.RM        ;CHECK FOR PARITY ERROR ON WRITE
6027 033764 000014      RMER1
6028 033766 034032      WRT.R3
6029 033770 032726 000010      BIT      #BIT03,(SP)+
6030 033774 001420      BEQ      WRT.R4          ;BRANCH IF "PAR=0"
6031 033776 016037 177776 034010      MOV      -2(RO),1$      ;PICKUP THE INDEX
6032 034004 004037 033470      JSR      RO,RO.RM        ;READ THE REG.
6033 034010 000000      1$:      .WORD 0              ;REG. INDEX
6034 034012 034032      WRT.R3
6035 034014 004037 034766      JSR      RO,ES.SAV      ;RETURN TO THIS ADDRESS ON ERROR
6036 034020 104004      ERROR 4                ;SAVE THE ADDRESS IN 'SESCAPE'
6037 034022 005726      TST      (SP)+          ;REPORT THE PARITY ON WRITE ERROR
6038 034024 005327      DEC      (PC)+          ;CLEAR OFF THE STACK
6039 034026 000003      WRT.R2: .WORD 3        ;DECREMENT THE ERROR COUNT
6040 034030 002341      BGE      WRT.R1          ;RETRY COUNTER
6041 034032 011000      WRT.R3: MOV      (RO),RO ;TRY AGAIN IF NOT FINISHED
6042 034034 000401      BR      WRT.R5          ;TAKE THE "PARITY ON WRITE" ERROR EXIT
6043 034036 005720      WRT.R4: TST      (RO)+
6044 034040 000200      WRT.R5: RTS      RO    ;EXIT
6045
6046      ;ROUTINE TO SAVE THE RH70/RP04/5/RM03 REGISTERS AS PER DPB+14
6047
6048      ;CALL
6049
6050      ;
6051      MOV      #DPBNUM,R2      ;DPB POINTER TO R2
6052      JSR      PC,SVRH70      ;SAVE THE DRIVES REG'S
6053
6054      SVRH70: SAVREG
6055      TST      R2            ;SAVE R0 - R5
6056      BEQ      6$          ;QUEUE ENTRY FOR THE DRIVE ?
6057      MOV      RMADR,R4      ;BR IF NONE
6058      MOV      (R2),RMCS2(R4) ;SELECT DRIVE
6059      MOV      14(R2),R3     ;GET THE ERROR TABLE POINTER
6060      BEQ      6$          ;EXIT IF NO ADDRESS
6061      CLR      3$          ;COUNTER & POINTER
6062      CMP      3$,#RMDB      ;REACHED THE BUFFER REGISTER ?
6063      BNE      2$          ;BR IF NOT
6064      BIT      #BIT07,RMCS2(R4) ;'OR' SET ?
6065      BNE      2$          ;BR IF SET
6066      CLR      (R3)+        ;STORE RMDB AS ZEROES
6067      BR      4$          ;CONTINUE
6068      JSR      RO,RO.RM      ;READ THE SELECTED REGISTER
6069      .WORD 0              ;REGISTER INDEX
6070      5$:      ERROR RETURN ADDRESS
6071      MOV      (SP)+,(R3)+   ;STORE THE REGISTER CONTENTS
        CMP      3$,#RMEC2    ;REACHED THE END ?
        BEQ      6$          ;BR IF YES

```



```

6072 034140 062737 000002 034122      ADD    #2,3$      ; INCREMENT THE REGISTER INDEX
6073 034146 000751                BR     1$         ; CONTINUE READING THE REGISTERS
6074 034150 004737 030762      5$:   JSR    PC,C17 ; PROCESS THE UNCORRECTABLE PARITY ERROR
6075 034154 104413                6$:   RESREG ; RESTORE R0 - R5
6076 034156 000207                RTS     PC        ; RETURN
6077
6078 ; ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
6079 ; CALL
6080 ;
6081 ;     MOV    #DRVNUM,R1      ; DRIVE NUMBER TO R1
6082 ;     JSR    PC,SET.IE      ; SET "IE"
6083 ;
6084 SET.IE: MOV    R4,-(SP)      ; SAVE R4
6085         MOV    RMADR,R4    ; PICKUP ADDRESS OF RMCS1
6086         MOV    R1,RMCS2(R4) ; SELECT DRIVE
6087         MOV    (R4),-(SP)   ; READ RMCS1
6088         BIS    #BIT14,(SP)  ; SET THE "TRE" BIT OF THE WORD READ
6089         SWAB  (SP)          ; ADJUST FOR DATO
6090         MOVB  #BIT06,(R4)   ; SET "IE"
6091         BIT   #BIT12,RMCS2(R4) ; IS "NED"=1?
6092         BNE   1$           ; YES--CLEAR "TRE"
6093         TST  (SP)+         ; CLEAN OFF THE STACK
6094         BR    2$
6095         1$:  MOVB  (SP)+,1(R4) ; CLEAR "TRE"
6096         2$:  MOV    (SP)+,R4 ; RESTORE R4
6097         RTS     PC        ; RETURN TO CALLER
6098
6099 ; QUEUE COUNT
6100 QCNT:  .BYTE  0      ; DRIVE 0
6101       .BYTE  0      ; DRIVE 1
6102       .BYTE  0      ; DRIVE 2
6103       .BYTE  0      ; DRIVE 3
6104       .BYTE  0      ; DRIVE 4
6105       .BYTE  0      ; DRIVE 5
6106       .BYTE  0      ; DRIVE 6
6107       .BYTE  0      ; DRIVE 7
6108
6109 ; QUEUE INPUT POINTERS
6110
6111 QINPT: .WORD  QDRV0    ; DRIVE 0
6112       .WORD  QDRV1    ; DRIVE 1
6113       .WORD  QDRV2    ; DRIVE 2
6114       .WORD  QDRV3    ; DRIVE 3
6115       .WORD  QDRV4    ; DRIVE 4
6116       .WORD  QDRV5    ; DRIVE 5
6117       .WORD  QDRV6    ; DRIVE 6
6118       .WORD  QDRV7    ; DRIVE 7
6119
6120 ; QUEUE OUTPUT POINTERS
6121
6122 QOUTPT: .WORD  QDRV0    ; DRIVE 0
6123        .WORD  QDRV1    ; DRIVE 1
6124        .WORD  QDRV2    ; DRIVE 2
6125        .WORD  QDRV3    ; DRIVE 3
6126        .WORD  QDRV4    ; DRIVE 4
6127        .WORD  QDRV5    ; DRIVE 5

```

```

6128 034276 034464          .WORD  QDRV6          ;DRIVE 6
6129 034300 034504          .WORD  QDRV7          ;DRIVE 7
6130
6131 034302 034324          QSTART: .WORD  QDRV0          ;DRIVE 0 START ADDRESS
6132 034304 034344          QSTOP:  .WORD  QDRV1          ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
6133 034306 034364          .WORD  QDRV2          ;STOP DRIVE 1--START DRIVE 2
6134 034310 034404          .WORD  QDRV3          ;STOP DRIVE 2--START DRIVE 3
6135 034312 034424          .WORD  QDRV4          ;STOP DRIVE 3--START DRIVE 4
6136 034314 034444          .WORD  QDRV5          ;STOP DRIVE 4--START DRIVE 5
6137 034316 034464          .WORD  QDRV6          ;STOP DRIVE 5--START DRIVE 6
6138 034320 034504          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
6139 034322 034524          .WORD  QTERM          ;STOP DRIVE 7
6140
6141          ;DRIVE REQUEST QUEUES
6142
6143 034324 00001C          QDRV0:  .BLKW  10
6144 034344 000010          QDRV1:  .BLKW  10
6145 034364 000010          QDRV2:  .BLKW  10
6146 034404 000010          QDRV3:  .BLKW  10
6147 034424 000010          QDRV4:  .BLKW  10
6148 034444 000010          QDRV5:  .BLKW  10
6149 034464 000010          QDRV6:  .BLKW  10
6150 034504 000010          QDRV7:  .BLKW  10
6151          QTERM=.
6152
6153          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
6154          :CALL
6155          :      JSR      PC,CLRQUE
6156          :
6157
6158 034524 104412          CLRQUE: SAVREG          ;SAVE R0 - R5
6159 034526 012702 034232          MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
6160 034532 005022          CLR      (R2)+          ;DRIVES 0 & 1
6161 034534 005022          CLR      (R2)+          ;DRIVES 2 & 3
6162 034536 005022          CLR      (R2)+          ;DRIVES 4 & 5
6163 034540 005022          CLR      (R2)+          ;DRIVES 6 & 7
6164 034542 012703 000010          MOV      #8,R3          ;MOVE THE STARTING
6165 034546 012701 034302          MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
6166 034552 012122          1$:  MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
6167 034554 005303          DEC      R3
6168 034556 001375          BNE      1$
6169 034560 012703 000010          MOV      #8,R3          ;MOVE THE STARTING ADDRESS
6170 034564 012701 034302          MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
6171 034570 012122          2$:  MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
6172 034572 005303          DEC      R3
6173 034574 001375          BNE      2$
6174 034576 104413          RESREG
6175 034600 000207          RTS      PC          ;RESTORE R0 - R5
6176
6177          ;EMPTY THE QUEUE SPECIFIED BY R1
6178          :CALL
6179          :      MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
6180          :      JSR      PC,EMPTYQ
6181          :
6182
6183 034602 105061 034232          EMPTYQ: CLRB      QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE

```

```

6184 034606 006301          ASL      R1
6185 034610 016161 034242 034262  MOV     QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
6186 034616 006201          ASR     R1
6187 034620 000207          RTS     PC
6188
6189          ;ROUTINE TO PUT A REQUEST IN QUEUE
6190          ;CALL
6191          ;
6192          ;
6193          ;
6194          ;
6195          ;
6196          ;
6197          ;
6198 034622 122761 000010 034232  DRVQUE: CMPB   #10,QCNT(R1)      ;IS QUEUE FULL?
6199 034630 001421          BEQ     2$              ;BR IF YES-TAKE RETURN1
6200 034632 105261 034232          INCB   QCNT(R1)        ;INCREMENT QUEUE COUNT
6201 034636 006301          ASL     R1
6202 034640 010271 034242          MOV     R2,QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
6203 034644 062761 000002 034242  ADD     #2,QINPT(R1)    ;UPDATE THE QUEUE POINTER
6204 034652 026161 034242 034304  CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
6205 034660 001003          BNE     1$              ;BRANCH IF NO
6206 034662 016161 034302 034242  MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
6207 034670 006201          1$:   ASR     R1
6208 034672 005720          TST    (R0)+           ;TAKE RETURN 2
6209 034674 000200          2$:   RTS     RD              ;RETURN TO USER
6210
6211          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
6212          ;CALL
6213          ;
6214          ;
6215          ;
6216          ;
6217          ;
6218          ;
6219 034676 005002          GETREQ: CLR     R2
6220 034700 105761 034232          TSTB   QCNT(R1)        ;IS THERE ANY REQUEST IN QUEUE?
6221 034704 001404          BEQ     2$              ;NO---BRANCH
6222 034706 006301          1$:   ASL     R1
6223 034710 017102 034262          MOV     @QOUTPT(R1),R2 ;PICKUP "DPB" POINTER FOR THIS DRIVE
6224 034714 006201          ASR     R1
6225 034716 000207          2$:   RTS     PC              ;RETURN TO USER
6226
6227          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
6228          ;CALL
6229          ;
6230          ;
6231          ;
6232          ;
6233          ;
6234 034720 105361 034232          POPQUE: DECB   QCNT(R1)      ;DECREMENT QUEUE COUNT
6235 034724 006301          ASL     R1
6236 034726 017102 034262          MOV     @QOUTPT(R1),R2 ;GET THE "DPB" POINTER
6237 034732 005071 034262          CLR     @QOUTPT(R1)    ;REMOVE DPB ADDRESS FROM THE QUEUE
6238 034736 062761 000002 034262  ADD     #2,QOUTPT(R1)   ;UPDATE THE QUEUE POINTER
6239 034744 026161 034262 034304  CMP     QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?

```

```

6240 034752 001003
6241 034754 016161 034302 034262 BNE 1$ ;NO--BRANCH TO EXIT
6242 034762 006201 1$: MOV QSTART(R1),QOUTP(R1) ;YES--RESET THE POINTER
6243 034764 000207 ASR R1
6244 RTS PC ;RETURN TO USER
6245 ;ROUTINE TO SAVE THE CONTENTS OF 'SESCAPE' WHEN THE DRIVER
6246 ;REPORTS AN ERROR DIRECTLY.
6247 ;CALL
6248 ;
6249 JSR RO,ES.SAV
6250 ERROR N ;:THE ERROR CALL
6251 RETURN ;:THE RETURN IS PAST THE ERROR CALL
6252
6253 034766 012037 035002 ES.SAV: MOV (RO)+,1$ ;GET THE ERROR CALL
6254 034772 013746 001174 MOV $ESCAPE,-(SP) ;SAVE THE ADDRESS IN 'SESCAPE'
6255 034776 005037 001174 CLR $ESCAPE ;CLEAR THE ESCAPE RETURN
6256 035002 000000 1$: .WORD 0 ;THE ERROR CALL IS MOVED HERE
6257 035004 012637 001174 MOV (SP)+,$ESCAPE ;RESTORE THE ESCAPE ADDRESS
6258 035010 000200 RTS RO ;RETURN
6259
6260 ;:*****
6261 .NLIST BEX
6262

```

.SBTTL TELETYPE MESSAGES

```

035012 047440 043106 044514 UNTOFF: .ASCIZ / OFFLINE/
035023 040 047117 044514 UNTON: .ASCIZ / ONLINE/
035033 040 047516 020124 NOTRM: .ASCIZ @ NOT AN RMO3/RMO2@
035055 040 047516 020124 NOTPRS: .ASCIZ / NOT PRESENT/
035072 052440 051516 043101 NOTSAF: .ASCIZ / UNSAFE/
035102 046522 031060 000 RMO3B: .ASCIZ /RMO2/
135107 122 030120 000065 RPOS: .ASCIZ /RPOS/
J35114 046522 031460 000 RMO3A: .ASCIZ /RMO3/
035121 125 044516 020124 SYSTAT: .ASCIZ /UNIT STATUS/<CR><LF><LF>
035140 005015 051104 053111 MUNIT: .ASCIZ <CR><LF>/DRIVE: /
035152 027440 000040 SLASH: .ASCIZ @ / @
035156 000103 C: .ASCIZ /C/
035160 000124 T: .ASCIZ /T/
035162 000060 MORVD: .ASCIZ /O/
035164 020040 LIN4SP: .ASCIZ / /
035166 020040 000 LINSR: .ASCIZ / /
035171 105 052116 051105 ENTADR: .ASCIZ /ENTER ADDRESS LIMITS:/<CR><LF>
035221 040 042040 044522 MOFFLN: .ASCIZ / DRIVE OFFLINE/<CR><LF>
035243 040 051104 053111 MDRNP: .ASCIZ / DRIVE NOT PRESENT/<CR><LF>
035270 042040 044522 042526 MER11: .ASCIZ / DRIVE NOT AVAILABLE/<CR><LF>
035317 040 051104 053111 MNRMO3: .ASCIZ @ DRIVE NOT AN RMO3@<CR><LF>
035344 042040 044522 042526 MUSDR: .ASCIZ / DRIVE UNSAFE/<CR><LF>
035364 051440 046105 041505 MSEL: .ASCIZ / SELECTED/
035376 005015 051120 043517 MMODE: .ASCIZ <CR><LF>/PROGRAM MODE (C OR F): /
035430 043040 051117 040515 MFORMAT: .ASCIZ / FORMAT & VERIFY/
035451 103 042510 045503 MHECK: .ASCIZ /CHECK ONLY/
035464 047506 046522 052101 MORMAT: .ASCIZ /FORMAT & VERIFY/
035504 005015 047412 042520 MSIZE: .ASCIZ <CR><LF><LF>/OPERATE IN 32 SECTOR MODE (Y OR N) ? /
035555 117 042520 040522 MSEC22: .ASCIZ /OPERATION WILL BE IN 32 SECTOR (16 BIT) MODE/<CR><LF>
035634 050117 051105 052101 MSEC20: .ASCIZ /OPERATION WILL BE IN 30 SECTOR (18 BIT) MODE/<CR><LF>

```

035713	111	053116	046101	BADENT:	.ASCIZ	/INVALID ENTRY/<CR><LF>
035733	105	042116	047111	MADRER:	.ASCII	/ENDING DSK ADRS MUST BE EQUAL TO OR GREATER/<CR><LF>
036010	044124	047101	051440		.ASCIZ	/THAN STARTING ADRS/<CR><LF>
036035	123	046105	041505	MSELP:	.ASCII	/SELECT DATA PATTERN (BY ENTERING 0,1 OR 2)/<CR><LF>
036111	040	030050	020051		.ASCII	/ (0) ZERO'S /<CR><LF>
036127	040	030450	020051		.ASCII	/ (1) ONE'S/<CR><LF>
036143	040	031050	020051		.ASCIZ	/ (2) WORST CASE: /
036165	127	051117	052123	MPATD:	.ASCIZ	/WORST CASE/
036200	005015	051412	040524	MSFOU:	.ASCIZ	<CR><LF><LF>/STARTING FORMAT ON DRIVE /
036235	015	005012	052123	MSCHK:	.ASCIZ	<CR><LF><LF>/STARTING CHECK ON DRIVE /
036271	015	005012	047506	MFCMPT:	.ASCIZ	<CR><LF><LF>/FORMAT COMPLETE, /
036316	005015	041412	042510	MCCMPT:	.ASCIZ	<CR><LF><LF>/CHECK COMPLETE, /
036342	047524	040524	020114	NUMERR:	.ASCIZ	/TOTAL ERRORS DETECTED: /
036372	051120	051505	047105	ADDRIS:	.ASCIZ	/PRESENT ADDRESS IS: /
036417	015	050012	053517	PMSG:	.ASCIZ	<CR><LF>/POWER ON, PROGRAM STARTS AT 200/<LF>
036462	047105	042524	020122	MSG1:	.ASCIZ	/ENTER THE SERIAL NUMBER (MAXIMUM 10 OCTAL DIGITS) /<CR><LF><LF>
036550	005015	051440	020116	SN1:	.ASCIZ	<CR><LF>/ SN # : /
036566	005015	053117	051105	MSG3:	.ASCII	<CR><LF>/OVER 126 BAD SECTORS HAVE BEEN DETECTED/<CR><LF>
036641	120	041501	020113		.ASCIZ	/PACK NOT ACCEPTABLE !/
036667	015	044412	041516	MSG2:	.ASCII	<CR><LF>/INCORRECT MANUFACTURE DEFINED BAD SECTOR/
036741	040	047111	047506		.ASCII	/ INFORMATION/<CR><LF>
036757	015	044412	044516		.ASCII	<CR><LF>/INITIALIZE THE BAD SECTOR FILE ?/
037021	050	047101	053523		.ASCIZ	/(ANSWER: Y OR N <CR>)/
037050	005015	046101	043511	MSG4:	.ASCIZ	<CR><LF>/ALIGNMENT PACK, PROGRAM HALT !/
037111	015	051412	046105	MSGBLK:	.ASCII	<CR><LF>/SELECT ONE OF THE FOLLOWING FUNCTIONS/<CR><LF>
037162	042513	044531	020116		.ASCII	/KEYIN U.S.R. BAD SECTORS .....0/<CR><LF>
037243	120	044522	052116		.ASCII	/PRINT M.F.G. BAD SECTORS.....1/<CR><LF>
037324	051120	047111	020124		.ASCII	/PRINT U.S.R. BAD SECTORS.....2/<CR><LF>
037405	111	044516	044524		.ASCII	/INITIALIZE U.S.R. BAD SECTOR FILE.....3/<CR><LF>
037466	051120	047111	020124		.ASCII	/PRINT A SECTOR OF THE LAST TRACK.....4/<CR><LF>
037547	105	044530	020124		.ASCIZ	/EXIT .....<CR>/<CR><LF>
037631	105	052116	051105	MSG5:	.ASCIZ	/ENTER 16 BIT M.F.G. BAD SECTOR ADDRESS/<CR><LF>
037702	047105	042524	020122	MSG6:	.ASCIZ	/ENTER 18 BIT M.F.G. BAD SECTOR ADDRESS/<CR><LF>
037753	105	052116	051105	MSG7:	.ASCIZ	/ENTER 16 BIT U.S.R. BAD SECTOR ADDRESS/<CR><LF>
040024	047105	042524	020122	MSG8:	.ASCIZ	/ENTER 18 BIT U.S.R. BAD SECTOR ADDRESS/<CR><LF>
040075	061	020066	044502	MSG9:	.ASCIZ	/16 BIT M.F.G. BAD SECTOR FILE/<CR><LF>
040135	061	020070	044502	MSG10:	.ASCIZ	/18 BIT M.F.G. BAD SECTOR FILE/<CR><LF>
040175	061	020066	044502	MSG11:	.ASCIZ	/16 BIT U.S.R. BAD SECTOR FILE/<CR><LF>
040235	061	020070	044502	MSG12:	.ASCIZ	/18 BIT U.S.R. BAD SECTOR FILE/<CR><LF>
040275	105	052116	051105	MSG13:	.ASCIZ	/ENTER SECTOR NUMBER: /
040324	005015	047504	042516	MSG14:	.ASCIZ	<CR><LF>/DONE/
040333	105	052116	051105	MSG15:	.ASCIZ	/ENTER M.F.G. BAD SECTOR ADDRESS/<CR><LF>
040375	127	044522	042524	MSG16:	.ASCIZ	/ILLEGAL BAD SECTOR PACK NOT ACCEPTABLE/<CR><LF>
040443	120	041501	020113	MSG17:	.ASCIZ	/WRITE BAD SECTOR FILE NOT SUCCESSFUL/<CR><LF>
					.ASCIZ	/PACK IS NOT ACCEPTABLE/<CR><LF>

.EVEN

(1)

;;\*\*\*\*\*

.SBTTL ERROR MESSAGES

(1)

;;\*\*\*\*\*

040474 044122 030461 044440 EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RMAS=0)/

040535	125	042516	050130	EM2:	.ASCIZ	/UNEXPECTED ATTENTION OCCURRED/
040573	115	051501	041123	EM3:	.ASCIZ	/MASSBUS PARITY ERROR (MCPE=1)/
040631	115	051501	041123	EM4:	.ASCIZ	/MASSBUS PARITY ERROR (PAR=1)/
040666	042101	051104	051505	EM5:	.ASCIZ	/ADDRESS PLUG CHANGE BIT SET/
040722	044122	030461	042040	EM6:	.ASCIZ	/RH11 DIDN'T RESPOND TO ADDRESSING/
040764	051104	053111	020105	EM10:	.ASCIZ	/DRIVE OFFLINE/
041002	042520	051522	051511	EM11:	.ASCIZ	/PERSISTENT DRIVE UNSAFE ERROR/
041040	047125	047503	051122	EM12:	.ASCIZ	/UNCORRECTABLE MASSBUS PARITY ERROR/
041103	123	043117	053524	EM13:	.ASCIZ	/SOFTWARE TIMEOUT/
041124	051104	053111	020105	EM14:	.ASCIZ	/DRIVE UNSAFE ERROR/
041147	103	047117	051124	EM15:	.ASCIZ	@CONTROLLER/DRIVE ERROR DURING WRITE@
041213	103	047117	051124	EM16:	.ASCIZ	@CONTROLLER/DRIVE ERROR DURING WRITE CHECK@
041265	122	052105	044522	EM17:	.ASCIZ	@RETRIES NOT SUCCESSFUL - SECTOR NOT ACCEPTABLE@
041343	104	052101	020101	EM20:	.ASCIZ	@DATA ERROR DURING WRITE CHECK@
041401	103	047117	051124	EM21:	.ASCIZ	@CONTROLLER/DRIVE ERROR VERIFYING HEADERS@
041452	042510	042101	051105	EM22:	.ASCIZ	@HEADER COMPARE ERROR VERIFYING HEADERS@
041521	103	046131	047111	EM23:	.ASCIZ	@CYLINDER FIELD IN HEADER IS NOT CORRECT@
041571	127	044522	042524	EM24:	.ASCIZ	@WRITE CHECK ERROR@
041613	111	046114	043505	EM25:	.ASCIZ	/ILLEGAL BAD SECTOR,PACK NOT ACCEPTABLE/

(1)

;;\*\*\*\*\*

041662	046522	051501	051011	DH1:	.ASCIZ	/RMAS RMCS1 RMER1 RMER2 RMDS RMDC RMDA/
041730	051104	053111	020105	DH2:	.ASCIZ	/DRIVE RMDS RMER1 RMER2 RMAS RMCS1/
042002	051104	053111	020105	DH3:	.ASCIZ	/DRIVE REG ADR DATA RMCS1 RMER1 RMER2/
042053	104	044522	042526	DH4:	.ASCIZ	/DRIVE REG ADR GOOD BAD RMCS1 RMER1 RMER2/
042136	044122	040440	042104	DH6:	.ASCIZ	/RH ADDRESS/
042151	104	044522	042526	DH10:	.ASCIZ	/DRIVE ERR PC RMCS1 RMCS2 RMDS RMER1 RMER2/
042236	046522	041505	020061	DH10A:	.ASCIZ	/RMEC1 RMEC2 RMWC RMB A RMDA RMAS RMLA/
042323	122	042115	004502	DH10B:	.ASCIZ	/RMDB RMMR1 RMDT RMSN RMOF CYLINDER TRACK/
042400	051104	053111	020105	DH17:	.ASCIZ	/DRIVE ERR PC CYLINDER TRACK SECTOR/
042447	104	044522	042526	DH20:	.ASCIZ	/DRIVE ERR PC CYLINDER TRACK SECTOR/
042516	046522	051503	020061	DH20A:	.ASCIZ	/RMCS1 RMCS2 RMDS RMER1 RMER2 RMMR2 RMEC1 RMEC2/
042613	122	053515	020103	DH20B:	.ASCIZ	/RMWC RMB A RMDA RMAS RMLA RMDB RMMR1 RMDT/
042711	122	051515	004516	DH20C:	.ASCIZ	/RMSN RMOF CYLINDER TRACK/
042744	020040	020040	020040	DH23:	.ASCII	/ EXPT'D/<CR><LF>
042774	051104	053111	020105	DH24:	.ASCIZ	/DRIVE ERR PC CYLNDR ACTUAL HEADER/
043042	020040	020040	020040	DH24:	.ASCII	/
043130	051104	053111	020105	DH25:	.ASCIZ	/DRIVE ERR PC CYLNDR TRACK SECTOR MEMORY DISK/<CR><LF>
043215	104	044522	042526	DH25:	.ASCIZ	/DRIVE CYLNDR TRACK SECTOR DATA DATA/
				.LIST	BEX	

6263  
6264  
6265  
6266  
6267  
6268  
6269  
6270  
6271  
6272  
6273  
6274  
6275  
6276  
6277

043256	001374	005540	005554	DT1:	.WORD	ATTN, RM.REG, RM.REG+14, RM.REG+42, RM.REG+12, RM.REG+34, RM.REG+6
043264	005602	005552	005574			
043272	005546					
043274	001372	026476	026500	DT2:	.WORD	DDRIVE, RMERRS, RMERRS+2, RMERRS+4, ATTN, RMCS1
043302	026502	001374	000000			
043310	001372	033514	033516	DT3:	.WORD	DDRIVE, RD.ADR, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
043316	005540	005554	005602			
043324	001372	033740	033736	DT4:	.WORD	DDRIVE, WRT.AD, WRT.WD, RD.WRD, RM.REG, RM.REG+14, RM.REG+42
043332	033516	005540	005554			
043340	005602					
043342	001272			DT6:	.WORD	\$RMADR
043344	001220	001132	005540	DT10:	.WORD	DRIVE, \$ERRPC, RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42
043352	005550	005552	005554			

6278	043360	005602						
6279	043362	005604	005606	005542		.WORD	RM.REG+44, RM.REG+46, RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20	
6280	043370	005544	005546	005556				
6281	043376	005560						
6282	043400	005562	005564	005566		.WORD	RM.REG+22, RM.REG+24, RM.REG+26, RM.REG+30, RM.REG+32, DS.CYL, DS.TRK	
6283	043406	005570	005572	001366				
6284	043414	001370						
6285	043416	001220	001132	001366	DT17:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC	
6286	043424	001370	001350					
6287	043430	001220	001132	001366	DT20:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC	
6288	043436	001370	001350					
6289	043442	005540	005550	005552		.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42, RM.REG+40, RM.REG+44, RM.RE	
6290	043450	005554	005602	005600				
6291	043456	005604	005606					
6292	043462	005542	005544	005546		.WORD	RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20, RM.REG+22, RM.REG+24, RM.RE	
6293	043470	005556	005560	005562				
6294	043476	005564	005566					
6295	043502	005570	005572	001366		.WORD	RM.REG+30, RM.REG+32, DS.CYL, DS.TRK	
6296	043510	001370						
6297	043512	001220	001132	043746	DT23:	.WORD	DRIVE, \$ERRPC, BUFP, RBUF, RBUF+2	
6298	043520	043736	043740					
6299	043524	001220	001132	001366	DT24:	.WORD	DRIVE, \$ERRPC, DS.CYL, DS.TRK, SAVSEC, \$GDDAT, RM.REG+22	
6300	043532	001370	001350	001140				
6301	043540	005562						
6302	043542	005540	005550	005552		.WORD	RM.REG, RM.REG+10, RM.REG+12, RM.REG+14, RM.REG+42, RM.REG+40, RM.REG+44, RM.RE	
6303	043550	005554	005602	005600				
6304	043556	005604	005606					
6305	043562	005542	005544	005546		.WORD	RM.REG+2, RM.REG+4, RM.REG+6, RM.REG+16, RM.REG+20, RM.REG+22, RM.REG+24, RM.RE	
6306	043570	005556	005560	005562				
6307	043576	005564	005566					
6308	043602	005570	005572	001366		.WORD	RM.REG+30, RM.REG+32, DS.CYL, DS.TRK	
6309	043610	001370						
6310								
6311	043612	001220	005622	001370	DT25:	.WORD	DRIVE, FMTDPB+12, DS.TRK, SAVSEC	
6312	043620	001350						
6313	043622	000001						
6314	043624	007	000		DF1:	.WORD	1	
6315	043626	000001				.BYTE	7,0	
6316	043630	006	000		DF2:	.WORD	1	
6317	043632	000001				.BYTE	6,0	
6318	043634	006	000		DF3:	.WORD	1	
6319	043636	000001				.BYTE	6,0	
6320	043640	007	000		DF4:	.WORD	1	
6321	043642	000001				.BYTE	7,0	
6322	043644	001	000		DF6:	.WORD	1	
6323	043646	000003				.BYTE	1,0	
6324	043650	007	000		DF10:	.WORD	3	
6325	043652	042236				.BYTE	7,0	
6326	043654	007	000			.WORD	DH10A	
6327	043656	042323				.BYTE	7,0	
6328	043660	007	140			.WORD	DH10B	
6329	043662	000001				.BYTE	7,140	
6330	043664	005	034		DF17:	.WORD	1	
6331	043666	000004				.BYTE	5,34	
6332	043670	005	034		DF20:	.WORD	4	
6333	043672	042516				.BYTE	5,34	
						.WORD	DH20A	

```

6334 043674 010 000
6335 043676 042613
6336 043700 010 000
6337 043702 042711
6338 043704 004 014
6339 043706 000001 DF23:
6340 043710 005 000 DF24:
6341 043712 000004
6342 043714 007 034
6343 043716 042516
6344 043720 010 000
6345 043722 042613
6346 043724 010 000
6347 043726 042711
6348 043730 004 014
6349
6350 043732 000001 DF25:
6351 043734 004 000
6352
6353
6354
6355
6356
6357
6358 043736 000000 000000 000000 RBUF:
6359 043744 000000
6360
6361 043746
6362
6363
6364
6365 043746 005015 041412 051132
6366 043762 046522 031460 051057
6367 044011 120 047522 051107
6368 044046 005015 047524 023440 LOADRV:
6369 044134 040520 045503 047440
6370 044232 047101 020104 042522
6371
6372
6373
6374
6375
6376
6377
6378 044264 005737 001400
6379 044270 001434
6380 044272 005037 001400
6381 044276 012700 001272
6382 044302 104401 044400

```

```

.BYTE 8,0
.WORD DH20B
.BYTE 8,0
.WORD DH20C
.BYTE 4,14
.WORD 1
.BYTE 5,0
.WORD 4
.BYTE 7,34
.WORD DH20A
.BYTE 8,0
.WORD DH20B
.BYTE 8,0
.WORD DH20C
.BYTE 4,14
.WORD 1
.BYTE 4,0
;*****
;BUFFER STARTS HERE
;*****
RBUF: .WORD 0,0,0,0 ;BUFFER FOR HEADER CHECK
BUF: ;FORMAT AND CHECK BUFFER STARTS HERE
;AND WILL OVERLAY ALL REMAINING CODE IN PROGRAM
.NLIST BEX
TITLE: .ASCII <CR><LF><LF>/CZRMACD/<CR><LF>
.ASCII @RMO3/RMO2 FORMATTER @<CR><LF><LF>
.ASCII /PROGRAM NEEDS 20 K MEMORY/<CR><LF><LF>
LOADRV: .ASCII <CR><LF>/TO 'FORMAT' OR 'CHECK' DRIVE 0, REPLACE THE 'XXDP'/<CR><LF>
.ASCII /PACK ON DRIVE 0 WITH ANOTHER PACK, CLEAR MEMORY LOCATION 40,/<CR><LF>
.ASCII /AND RESTART THE PROGRAM/<CR><LF>
.LIST BEX

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF
;REQUIRED.
;NOTE: THIS ROUTINE DESTROYS R0-R4
;CALL
;
; JSR PC,BUSADR
; RETURN
;
BUSADR: TST CHGADR ;INPUT FROM TTY REQUESTED?
BEQ 3$ ;NO--BRANCH
CLR CHGADR ;YES--CLEAR THE REQUEST FLAG
1$: MOV #RAMADR,R0 ;FIRST ADDRESS
TYPE ,RAMCS1 ;"RAMCS1="

```



6383	044306	011046			MOV	(R0),-(SP)	:PRESENT RMCS1 ADDRESS
6384	044310	104402			TYPOC		:TYPE IT
6385	044312	104401	035166		TYPE	,LINS	:2 SPACES
6386	044316	104411			RDLIN		:GET THE ENTRY
6387	044320	012601			MOV	(SP)+,R1	:ADDRESS OF ASCII TEXT
6388	044322	004537	044422		JSR	R5,CK.NUM	:ENTER AND STORE NEW ADDRESS
6389	044326	000763			BR	1\$	:ERROR RET
6390	044330	012700	001274	2\$:	MOV	#\$RMVEC,R0	:CHECK VERC
6391	044334	104401	044411		TYPE	1,RHVEC	:RHVEC="
6392	044340	011046			MOV	(R0),-(SP)	:PRESENT RH11 VECTOR ADDRESS ON THE STACK
6393	044342	104402			TYPOC		:TYPE IT
6394	044344	104401	035166		TYPE	,LINS	:2 SPACES
6395	044350	104411			RDLIN		:READ THE ENTRY
6396	044352	012601			MOV	(SP)+,R1	:ASCII TEXT ADDRESS
6397	044354	004537	044422		JSR	R5,CK.NUM	:ENTER AND STORE NEW ADDRESS
6398	044360	000763			BR	2\$	:ERROR RET
6399	044362	012700	001272	3\$:	MOV	#\$RMADR,R0	:FIRST ADDRESS OF NEW PARAMETERS
6400	044366	012701	026644		MOV	#\$RMADR,R1	:FIRST ADDRESS OF WHERE TO PUT THEM
6401	044372	012021			MOV	(R0)+,(R1)+	:BUS ADDRESS
6402	044374	012021			MOV	(R0)+,(R1)+	:VECTOR ADDRESS
6403	044376	000207			RTS	PC	:RETURN

6405	044400	046522	051503	020061	MRMCS1: .ASCIZ	ARMCS1 = 2
6406	044406	020075	000			
6407	044411	122	053110	041505	MRHVEC: .ASCIZ	ARMVEC = 2
6408	044416	036440	000040			

```

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

```

6422	044422	010246			CK.NUM: MOV	R2, -(SP)	:SAVE R2
6423	044424	010346			MOV	R3, -(SP)	:SAVE R3
6424	044426	010446			MOV	R4, -(SP)	:SAVE R4
6425	044430	012703	000006		MOV	#6,R3	:MAX OCTAL DIGITS IN THE NUMBER
6426	044434	005002			CLR	R2	:FINAL OCTAL VALUE
6427	044436	112104		1\$:	MOVB	(R1)+,R4	:GET CURRENT POINTED BYTE
6428	044440	001424			BEQ	3\$	:BRANCH, IF TERMINATOR DETECTED
6429	044442	120427	000060		CMPB	R4, #'0	:SMALLER THAN ASCII-0 ?
6430	044446	103425			BLO	5\$	:YES, ERROR EXIT
6431	044450	120427	000067		CMPB	R4, #'7	:LARGER THAN ASCII-7 ?
6432	044454	101022			BHI	5\$	:YES, ERROR EXIT
6433	044456	006302			ASL	R2	:SHIFT LEFT
6434	044460	103420			BCS	5\$	
6435	044462	006302			ASL	R2	:ONE
6436	044464	103416			BCS	5\$	
6437	044466	006302			ASL	R2	:OCTAL DIGIT
6438	044470	103414			BCS	5\$	:ERROR IF CARRY BIT SET

```

6439 044472 042704 177770
6440 044476 060402
6441 044500 005303
6442 044502 001401
6443 044504 000754
6444 044506 112104
6445 044510 001004
6446 044512 005702
6447 044514 001401
6448 044516 010210
6449 044520 005725
6450 044522 012604
6451 044524 012603
6452 044526 012602
6453 044530 000205
6454
6455
6456
6457
6458
6459
6460

```

000001

.END

```

BIC #177770,R4 ;CHOP OFF HIGHER BITS
ADD R4,R2 ;APPENDING CURRENT DIGIT
DEC R3 ;DECREMENT BYTE COUNT
BEQ 2$ ;BRANCH, IF LAST DIGIT
BR 1$ ;LOOPING BACK
2$: MOVB (R1)+,R4 ;CHECK TERMINATOR
BNE 5$ ;BRANCH IF NOT FOUND
3$: TST R2 ;FINAL VALUE = 0 ?
BEQ 4$ ;YES
MOV R2,(R0) ;REPLACE THE ORIGINAL VALUE
4$: TST (R5)+ ;ADJUST FOR NORMAL RET
5$: MOV (SP)+,R4 ;RESTORE 4
MOV (SP)+,R3 ;RESTORE R3
MOV (SP)+,R2 ;RESTORE R2
RTS R5 ;EXIT

```

CZRMACD RM03/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY11 30(1046) 22-NOV-77 08:16 PAGE 124  
CROSS REFERENCE TABLE -- USER SYMBOLS

ABASE = 176700	1101#	1412	1453											
ACDW1 = 000000	1412#	1455												
ACDW2 = 000000	1412#	1456												
ACK = 000123	1349#													
ACPUOP = 000000	1412#	1427												
ACTDRV 026562	4946#	5219*	5270*	5596*	5605*	5851								
ACTSTR 026563	4952#	5853*	5867*											
ADDRIS 036372	3309#	6262#												
ADDW0 = 000000	1412#													
ADDW1 = 000000	1412#													
ADDW10 = 000000	1412#													
ADDW11 = 000000	1412#													
ADDW12 = 000000	1412#													
ADDW13 = 000000	1412#													
ADDW14 = 000000	1412#													
ADDW15 = 000000	1412#													
ADDW2 = 000000	1412#													
ADDW3 = 000000	1412#													
ADDW4 = 000000	1412#													
ADDW5 = 000000	1412#													
ADDW6 = 000000	1412#													
ADDW7 = 000000	1412#													
ADDW8 = 000000	1412#													
ADDW9 = 000000	1412#													
ADEVCT = 000000	1412#	1418												
ADEVN = 000000	1412#	1454												
ADRTBL 006132	1628#	2741	2812	3008	3740									
AENV = 000000	1412#	1423												
AENVN = 000000	1412#	1424												
AFATAL = 000000	1412#	1415												
AMADR1 = 000000	1412#	1440												
AMADR2 = 000000	1412#	1444												
AMADR3 = 000000	1412#	1447												
AMADR4 = 000000	1412#	1450												
AMAMS1 = 000000	1412#	1434												
AMAMS2 = 000000	1412#	1442												
AMAMS3 = 000000	1412#	1445												
AMAMS4 = 000000	1412#	1448												
AMSGA0 = 000000	1412#	1420												
AMSGLG = 000000	1412#	1421												
AMSGTY = 000000	1412#	1414												
AMTYP1 = 000000	1412#	1435												
AMTYP2 = 000000	1412#	1443												
AMTYP3 = 000000	1412#	1446												
AMTYP4 = 000000	1412#	1449												
AOE = 001000	1251#													
APASS = 000000	1412#	1417												
APRIOR = 000000	1412#													
APT 007726	2066#													
APTCSU = 000040	4086	4193#												
APTENV = 000001	3967	4079	4149	4191#										
APTSIZ = 000200	1970	4190#												
APTSPO = 000100	4081	4151	4192#											
ASWREG = 000000	1412	1425												
ATA = 100000	1238#													
ATABIT 026632	2049	5018#	5186	5282	5388	5770	5792	5800	5801	5821	5911			







			1835	1842	1849	1856	1863	1870	1891	1898	6276#							
DT10	043344	1828	1835															
DT17	043416	1877	6285#															
DT2	043274	1786	6268#															
DT20	043430	1884	6287#															
DT23	043512	1905	6297#															
DT24	043524	1912	6299#															
DT25	043612	1919	6311#															
DT3	043310	1793	6270#															
DT4	043324	1800	6272#															
DT6	043342	1814	6275#															
DULP	027076	5145#	5156															
DVA	= 004000	1225#																
ECH	= 000100	1248#																
ECI	= 004000	1316#																
EDIT	001410	1505#	2068*	2528*	2531*	2961												
EMPTYQ	034602	5489	5540	5637	5940	6183#												
EMTVEC=	000030	1096#	1945*	1946*	1976*													
EM1	040474	1777	6262#															
EM10	040764	1826	6262#															
EM11	041002	1833	6262#															
EM12	041040	1840	6262#															
EM13	041103	1847	6262#															
EM14	041124	1854	6262#															
EM15	041147	1861	6262#															
EM16	041213	1868	6262#															
EM17	041265	1875	6262#															
EM2	040535	1784	6262#															
EM20	041343	1882	6262#															
EM21	041401	1889	6262#															
EM22	041452	1896	6262#															
EM23	041521	1903	6262#															
EM24	041571	1910	6262#															
EM25	041613	1917	6262#															
EM3	040573	1791	6262#															
EM4	040631	1798	6262#															
EM5	040666	6262#																
EM6	040722	1812	6262#															
ENDCYL	001320	1472#	1586	2078*	2218*	2221*	2233	2505	2515	2517	2926	2935						
ENDTRK	001324	1474#	1587	2079*	2169*	2237	2507	2522										
ENTADR	035171	2228	6262#															
ERINDX	016144	2316	2752	2784	2905	2988	3088#											
ERR	= 040000	1237#	2804															
ERRVEC=	000004	1089#	1955	1956*	1967*	3244*	3245*	3255*	3264*									
ES.SAV	034766	5655	5707	5711	5975	6035	6253#											
FER	= 000020	1246#	2793															
FMTDPB	005610	1553#	2171*	2215	2217*	2291*	2292*	2293*	2294*	2295*	2296*	2297*	2299*	2300*				
		2302	2304	2306*	2310	2312	2330	2562*	2565*	2566*	2568	2570	2598	2600				
		2660*	2661*	2662*	2672*	2680	2693*	2695	2697	2717	2722*	2732	2737*	2739				
		2741*	2742*	2746	2748	2769*	2773	2775	2810*	2812*	2820	2825*	2834	2891*				
		2892*	2893*	2894*	2895*	2899	2901	2911	2932	2938*	2939*	2940*	2950*	2968*				
		2969*	2971	2973	2975*	2976*	2977*	2978*	2979*	2980*	2982	2984	3000	3005*				
		3008*	3009*	3010*	3071	3090	3093	3096	3099	3102	3121	3131*	3132*	3133*				
		3134*	3135*	3136*	3137*	3138*	3151*	3152	3154	3162*	3163*	3164	3171*	3172*				
		3173*	3184*	3185*	3186*	3311	3317	3372	3374	3376	3382	3383	3510	3512				
		3516	3524	3525	3945	3946	6311											
FMTX	005630	1570#	3519*	3520*	3521*	3522*	3523*	3524*	3525*	3531	3533							











CZRMACD RM03/2 FORMATTER  
CZRMAC.P11 21-NOV-77 15:13

MACY11 30(1046) 22-NOV-77 08:16 PAGE 133  
CROSS REFERENCE TABLE -- USER SYMBOLS

RMINIT	026662	2017	2200	5087#											
RMLA	= 000020	5061#	5574	5576											
RMFR1	= 000024	5063#													
RMFR2	= 000040	5069#	5811												
RNOF	= 000032	5066#	5179	5410	5414	5434	5438								
RMR	= 000004	1244#													
RMSN	= 000030	5065#													
RMVEC	026646	2001#	2011#	5036#	5104	5106	5218								
RMWC	= 000002	5054#													
RM.REG	005540	1529#	1562	2778	2790	2792	2804	2913	2915	3002	3124	3947	3949	6265	
		6270	6272	6276	6279	6282	6289	6292	6295	6299	6302	6305	6308		
RM03	027410	2301	2309	2567	2745	2772	2898	2970	2981	3530	5217#				
RM03A	035114	2057	6262#												
RM03B	035102	2051	6262#												
RNOP	= 000101	1341#													
RPTMR	033056	3286	5851#												
RPOS	035107	2054	6262#												
RTC	= 000117	1347#													
SAVEFG	026604	2018#	2201#	4986#	5463	5624	5797								
SAVREG	= 104412	3993	4696	4834#	5087	5220	5280	5499	5597	5854	6052	6158			
SAVSEC	001350	1484#	1592	1597	1601	2502#	2778#	2780#	2781#	2782#	2809	2810	2862*	3002*	
		3004#	3005	3006	3384	3517	3523	3551	3559	3567*	3574	3584*	3591	3612*	
		3619	3629*	3636	3756*	3763	3765	3768	3770	3775*	3778*	3780	3785*	6285	
		6287	6299	6311											
SAVMC	001352	1485#	2824*	3184											
SC	031710	5603	5632	5635	5641	5647#									
SCAWC	016576	2836	3184#												
SC01	= 000100	1301#													
SC02	= 000200	1302#													
SC04	= 000400	1303#													
SC10	= 001000	1304#													
SC11	032546	5688	5719	5727	5788#										
SC12	032656	5682	5693	5807#											
SC13	032720	5672	5675	5817#											
SC20	= 002000	1305#													
SC3	031760	5663#	5667												
SC4	031764	5665#	5696	5709	5713	5762	5778	5787	5806	5844					
SC5	031776	5664	5670#												
SC6	032176	5692	5714#												
SC6A	032312	5690	5739#												
SC7	032440	5729	5742	5744	5754	5764#									
SC8	032516	5705	5722	5732	5738	5747	5777	5779#	5816						
SEARCH	= 000131	1350#													
SECCU	005434	1522#	2575	2594	2622	2623	2640	2642*	2651	2653	2673	2856			
SEC30	001362	1489#	2085*	2094*	2158*	2164*	2393	2563	2589	3201	3447	3449*	3453*	3479	
		3485	3527	3603	3667										
SEEK	= 000105	1342#													
SEEKFG	026606	2202#	4994#	5092	5312										
SELDIV	= 000145	1353#													
SERNL	012020	2414#													
SETFMT	= 000143	1352#	2299	2566	2969										
SETHDR	016624	2561	2889	2945	3170	3194#	3452								
SETPAT	012642	2510	2530	2535#											
SETTBL	016312	2560	2888	3131#											
SETVEC	007464	1998	2003	2016#											
SET.IE	034160	5121	5152	5255	5329	5543	5657	6084#							







SCNTLU	025620	4471	4564	4590#											
SCPUOP	001234	1427#													
SCRLF	001203	1402#	2025	2060	2064	2285	2419	3022	3066	3070	3075	3320	3386	3647	
		3655	3672	3679	3742	3965	3986	4003	4008	4018	4035	4048	4100	4135	
		4482	4569	4589											
SDBLK	024374	4297	4331	4339#											
SDB2D	026122	4677	4696#												
SDECVL	026302	4698	4744#												
SDEVCT	001216	1418#	2114#	3013*	3064*										
SDEVN	001264	1454#	2069	2071											
SDDAGN	015752	3033	3038	3044#											
SDTBL	024364	4300	4335#												
SENDAD	015742	1132	3040#												
SENDCT	015726	3035#													
SENV	001226	1423#	1989	2002	3967	4079	4149	4173							
SENM	001227	1424#	1970	4081	4086	4151									
SEOP	015324	2075	2117	2880	2928	2960#	3126								
SEOPCT	015720	3032#	3036												
SERFLG	001117	1372#	3952*	3986											
SERMAX	001131	1378#	1952*												
SERROR	022516	1945	3941#												
SERRPC	001132	1379#	3959*	3960*	3961	3986	6276	6285	6287	6297	6299				
SERRTB	006526	1774#	4000												
SERTTL	001126	1376#	2177*	3020	3958*	3986									
SESCAP	001174	1399#	1951*	2315*	2328*	2751*	2764*	2783*	2798*	2803*	2844*	2904*	2918*	2987*	
		3120*	3981	3983	3986	5655	5707	5711	5975	6035	6245	6254	6255*	6257*	
SETABL	001226	1422#													
SETEND	001272	1162	1457#												
SFATAL	001210	1415#	4177*												
SFFLG	023730	4140*	4143*	4171	4180*	4188#									
SFILLC	001172	1397#	4104	4135											
SFILLS	001171	1396#	4135												
SGADR	001134	1380#													
SGDDAT	001140	1382#	3951*	6299											
SGET42	015732	3037#													
SGTSMR	025000	4447#	4829												
SHD	= 000000	991													
SHIBTS	001100	1157#													
SICNT	001120	1373#													
SILLUP	026020	4601	4617	4636#											
SINTAG	001151	1387#	4444*	4463	4483	4596									
SITEMB	001130	1377#	3961*	3969	3986	3995									
SLF	001204	1403#	3986	4135	4579	4589									
SLFLG	023727	4181*	4187#												
SLKCSB	001300	1464#	3251*												
SLKCSR	001276	1463#	3246	3252*											
SLKS	001306	1466#	3256	3261*											
SLVEC	001310	1467#	3257												
SLPADR	001122	1374#													
SLPERR	001124	1375#	2307*	2743*	2770*	2896*	3119	3980							
SLPVEC	001302	1465#	3247												
SMADR1	001240	1440#													
SMADR2	001244	1444#													
SMADR3	001250	1447#													
SMADR4	001254	1450#													
SMAIL	001206	1158	1162	1413#	1935	1969	1989	3967	4079						









.SAPTY	981#	4137
.SCATC	981#	1116
.SCMTA	981#	1362
.SDB2D	981#	4683
.SEOP	981#	2953
.SERR0	981#	3927
.SERRT	981#	
.SPOWE	981#	4597
.SREAD	981#	4341
.SSAVE	981#	4746
.SSB2D	981#	4664
.SSUPR	981#	4640
.STRAP	981#	4792
.STYPD	981#	4273
.STYPE	981#	4056
.STYPO	981#	4195

. ABS. 044532 000

ERRORS DETECTED: 0

RMD3:CZRMAC,RMD3:CZRMAC.SEQ/NL:MD:MC:CND:TOC/LI:ME/DOC/SOL/CRF=RMD3:RMDRV5.P11,RMD3:CZRMAC.P11

RUN-TIME: 29 19 1 SECONDS

RUN-TIME RATIO: 439/50=8.6

CORE USED: 39K (77 PAGES)

DOCUMENT PAGES: 139