

# RP04

VERIFICATION PROGRAM  
MD-11-DERPM-B

EP-DERPM-B-DL-A

NOV 1976

COPYRIGHT © 1976

**digital**

FICHE 1 OF 1

MADE IN USA

The microfiche card contains a grid of 120 frames arranged in 10 rows and 12 columns. The frames on the left side of the card contain various types of data, including text, tables, and diagrams. The right side of the card is mostly blank.







CONTENTS

1. ABSTRACT
2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PROGRAM STORAGE
  - 2.3 PRELIMINARY PROGRAMS
3. LOADING PROCEDURES
4. STARTING PROCEDURES
  - 4.1 CONROL SWITCH SETTINGS
  - 4.2 STARTING ADDRESSES
  - 4.3 PROGRAM & OPERATOR ACTION
    - 4.3.1 DRIVE SELECTION
    - 4.3.2 RH70 - UNIBUS ADDRESSES
5. OPERATIONAL SWITCH SETTINGS
6. ERRORS
  - 6.1 TEST ERRORS
  - 6.2 ENTRY ERRORS
  - 6.3 SUBSYSTEM/DEVICE ERROR MESSAGES
7. RESTRICTIONS
8. PROGRAM DESCRIPTION
  - 8.1 VERIFICATION MODE
  - 8.2 ALIGNMENT MODE
  - 8.3 RANDOM SEEK UTILITY
9. PROGRAM LISTING

4  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82



001  
002  
003  
004  
005  
006  
007  
008  
009  
010  
011  
012  
013  
014  
015  
016  
017  
018  
019  
020  
021  
022  
023  
024  
025  
026  
027  
028  
029  
030  
031  
032  
033  
034  
035  
036  
037  
038

1. ABSTRACT

THIS PROGRAM CHECKS RPO4 DISK DRIVE HEAD ALIGNMENT OR ALLOWS THE OPERATOR TO ALIGN HEADS WITH THE DDU OR DEDU TEST BOX. THE PROGRAM ALSO CONTAINS A UTILITY ROUTINE WHICH PERFORMS 5,000 RANDOM SEEKS WITHOUT DATA TRANSFER ON THE DRIVE BEING CHECKED. THE RANDOM SEEK UTILITY ALLOWS THE OPERATOR TO EXERCISE THE DRIVE WITH THE ALIGNMENT PACK IN PLACE AND THEN RE-VERIFY HEAD ALIGNMENT.

2. REQUIREMENTS

2.1 PDP-11 COMPUTER WITH A TELETYPE, AN RPO4 DISK DRIVE, AN ALIGNMENT DISK PACK ('CE' PACK), AND A DDU OR DEDU TEST BOX.

2.2 PROGRAM STORAGE

THE PROGRAM REQUIRES APPROXIMATELY 3K OF STORAGE.

2.3 PRELIMINARY PROGRAMS

NONE

3. LOADING PROCEDURE

THIS PROGRAM IS LOADED USING THE ABSOLUTE LOADER OR BY USING THE APPROPRIATE 'XXDP' SYSTEM.

4. STARTING PROCEDURES

4.1 CONTROL SWITCH SETTINGS

(SEE SECTION 5)

4.2 STARTING ADDRESSES

THE PROGRAM IS STARTED AT LOCATION 200.  
RESTART ADDRESS IS LOCATION 204.

4.3 PROGRAM & OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3)

2. LOAD ADDRESS 200

3. SET THE SWITCHES AND PRESS START

4. PLACE AN ALIGNMENT DISK PACK ON THE DRIVE TO BE CHECKED AND CONNECT THE DDU OR DEDU TEST BOX. PUT THE DISK DRIVE



139  
140  
141  
142

IN WRITE PROTECT.

5. CYCLE UP THE DRIVE TO BE CHECKED. ALLOW SUFFICIENT TIME FOR THE ALIGNMENT PACK TO REACH OPERATING TEMPERATURE AND



143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198

TO STABILIZE. REFER TO APPLICABLE MAINTENANCE PROCEDURES FOR THE AMOUNT OF TIME.

6. SELECT THE DRIVE TO BE CHECKED IN RESPONSE TO THE TYPEOUT ON THE TELETYPE.
7. THE PROGRAM WILL ASK FOR THE MODE OF OPERATION: ENTER AN 'A' IF THE PROGRAM IS TO BE USED FOR HEAD ALIGNMENT; ENTER A 'V' IF HEAD ALIGNMENT IS TO BE CHECKED; OR ENTER AN 'E' IF THE RANDOM SEEK UTILITY IS TO BE RUN.
8. IF AN 'A' IS ENTERED IN RESPONSE TO THE PROGRAM MODE MESSAGE, THE PROGRAM WILL POSITION THE RPD4 AT CYLINDER 245. THE PROGRAM WILL THEN ASK THE OPERATOR TO ENTER A HEAD. THE HEAD ENTERED WILL BE SELECTED AND THE PROGRAM WILL REQUEST ANOTHER HEAD. UNTIL A NEW HEAD IS ENTERED, THE LAST ENTERED HEAD WILL REMAIN SELECTED. THE ALIGNMENT SEQUENCE MAY BE TERMINATED BY ENTERING A 'CONTROL C'; THE PROGRAM WILL THEN RETURN TO THE DRIVE SELECTION ROUTINE.
9. IF SW<02> IS SET, THE PROGRAM WILL ASK FOR HEAD AND CYLINDER ADDRESSES. (HEAD AND CYLINDER ADDRESSES ARE DECIMAL.) WITH SW<02> SET, ONLY THE ALIGNMENT OF THE HEAD SPECIFIED BY THE OPERATOR WILL BE CHECKED.
10. THE PROGRAM WILL BEGIN CHECKING HEAD ALIGNMENT ON THE SPECIFIED DRIVE AS SOON AS A DRIVE NUMBER IS ENTERED, (SW<02> = 0).
11. WHEN THE PROGRAM HAS COMPLETED THE HEAD ALIGNMENT CHECK ON THE PRESENT DRIVE, CYCLE THE DRIVE DOWN, AND TRANSFER THE DDU AND THE ALIGNMENT PACK TO THE NEXT DRIVE TO BE CHECKED. WHEN THE PACK HAS STABILIZED, THE NEW DRIVE CAN BE SELECTED. IT IS NOT NECESSARY TO RESTART THE PROGRAM.

#### 4.3.1 DRIVE SELECTION

ENTER A DRIVE NUMBER IN RESPONSE TO THE 'ENTER DRIVE NUMBER:' TYPEOUT FROM THE PROGRAM. ONLY VALID DRIVE NUMBERS (0 - 7) WILL BE ACCEPTED BY THE PROGRAM. NOTE THAT THE DRIVE SETUP (DDU CONNECTION AND ALIGNMENT PACK STABILIZATION) MUST HAVE BEEN COMPLETED BEFORE A DRIVE IS SELECTED.

#### 4.3.2 RH70 - UNIBUS ADDRESSES

THE PROGRAM ASSUMES THAT THE RH70 ADDRESSES START AT LOCATION 176700 AND THAT THE VECTOR ADDRESS IS 254. THESE ADDRESSES MAY BE CHANGED WHEN THE PROGRAM IS STARTED AT LOCATION 200. ENTER THE RH70 ADDRESS INTO LOCATION 1200 AND THE VECTOR ADDRESS INTO LOCATION 1202.

#### 5. OPERATIONAL SWITCH SETTINGS

-----  
SW<15>=1...HALT ON ERROR  
SW<13>=1...INHIBIT ERROR TYPEOUTS



GO1

MAINDEC-11-DERPM-B  
DERPMB.P11

MACY11 27(732) 05-OCT-76 15:01 PAGE 7

199  
200  
201  
202

SW<09>=1...LOOP ON ERROR  
SW<02>=1...CHECK ALIGNMENT OF THE SPECIFIED HEAD  
SW<01>=1...LOOP ON THE CURRENT HEAD  
SW<00>=1...TYPEOUT ALL TRACK CENTER VALUES



203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258

6. ERRORS  
-----

6.1 TEST ERRORS

THE PROGRAM CHECKS THAT EACH HEAD IS WITHIN + OR - 150 MICRO-INCHES FROM THE TRACK CENTERLINE FOR CYLINDER 245 AND IS WITHIN + OR - 350 MICRO-INCHES FOR CYLINDERS 4 AND 400. IF A HEAD IS FOUND THAT IS NOT WITHIN THESE TOLERANCES, AN ERROR MESSAGE IS TYPED. THE ERROR MESSAGE IDENTIFIES THE CYLINDER, HEAD, AND THE ACTUAL POSITION OF THE HEAD RELATIVE TO THE CENTERLINE. (IF SW<00>=1, THE ACTUAL ALIGNMENT OF EACH HEAD WILL BE TYPED OUT; HEADS OUT OF TOLERANCE WILL NOT BE IDENTIFIED.)

6.2 ENTRY ERRORS

THE PROGRAM WILL NOT ACCEPT DRIVE NUMBERS GREATER THAN 7. IF AN INVALID DRIVE NUMBER IS ENTERED IN RESPONSE TO THE DRIVE NUMBER REQUEST, THE PROGRAM WILL TYPE A '?'; THE OPERATOR MAY ENTER A VALID NUMBER.

THE PROGRAM WILL NOT ACCEPT HEAD ADDRESSES GREATER THAN 18 (DECIMAL) OR CYLINDER ADDRESSES OTHER THAN 4, 245, OR 400. IF AN INVALID VALUE IS ENTERED FOR EITHER THE HEAD OR THE CYLINDER, THE PROGRAM WILL REJECT THE ENTRY AND RETURN TO THE DRIVE SELECTION ROUTINE.

THE PROGRAM WILL NOT ALLOW A DRIVE TO BE ASSIGNED IF IT IS NOT 'WRITE PROTECTED'. IF THIS IS ATTEMPTED, THE OPERATOR WILL BE NOTIFIED.

DRIVES ASSIGNED ARE CHECKED TO ENSURE THAT THE DRIVE ASSIGNED IS PRESENT AND ONLINE. THE PROGRAM WILL REJECT ASSIGNMENTS FOR DRIVES WHICH ARE NOT PRESENT OR ARE NOT ONLINE.

6.3 SUBSYSTEM/DEVICE ERROR MESSAGES

1. 'ILLEGAL RH70 INTERRUPT (SC=0 OR RHAS=0)' - A SUBSYSTEM INTERRUPT OCCURED, BUT THE 'SC' BIT IN RHCS1 IS NOT SET OR THE ATTENTION REGISTER (RHAS) IS ZERO.
2. 'UNEXPECTED ATTENTION' - THE INDICATED DRIVE INTERRUPTED BUT NO INTERRUPT WAS EXPECTED.
3. 'CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH70' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH70 WHEN THE INDICATED REGISTER WAS READ.
4. 'CONTROL BUS PARITY ERROR WAS DETECTED BY THE RPO4' - THE ADDRESSED DRIVE DETECTED A CONTROL BUS PARITY ERROR WHEN THE CONTROLLER ATTEMPTED TO WRITE INTO THE INDICATED DRIVE REGISTER.
5. 'ATTENTION FROM AN UNASSIGNED DRIVE' - THE ATTENTION REGISTER HAS AN ATTENTION BIT SET FOR A DRIVE WHICH IS NOT CURRENTLY ACTIVE IN AN OPERATION. THIS CAN BE CAUSED BY CYCLING A DRIVE



259  
260  
261  
262

UP OR DOWN WHILE ANOTHER DRIVE IS BEING CHECKED.

- 6. 'DEVICE ERROR' - THE INDICATED DRIVE HAS EITHER DETECTED AN UNSAFE CONDITION WHICH WAS CLEARED BY A 'DRIVE CLEAR' COMMAND



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  
318

OR A DCL RELATED ERROR WAS DETECTED. REFER TO THE DISPLAYED REGISTERS FOR EXPLANATORY INFORMATION.

7. 'DEVICE UNSAFE ERROR' - THE INDICATED DRIVE SIGNALLED AN UNSAFE CONDITION WHICH COULD NOT BE CLEARED BY A 'DRIVE CLEAR' COMMAND. TESTING OF THAT DEVICE WILL BE TERMINATED AND THE PROGRAM WILL REQUEST ANOTHER DEVICE.
8. 'FATAL MASSBUS PARITY ERROR' - THE PROGRAM DETECTED A MASSBUS PARITY ERROR WHILE ISSUING A 'DRIVE CLEAR' TO RESET A PREVIOUSLY DETECTED MASSBUS PARITY ERROR. THE PROGRAM WILL HALT IF THIS CONDITION IS DETECTED.
9. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM HAS TRIED 3 TIMES TO READ OR WRITE A REGISTER AND A PARITY ERROR HAS OCCURED EACH TIME.

7. RESTRICTIONS

BECAUSE AN ALIGNMENT DISK PACK IS USED, THE PROGRAM WILL NOT CHECK HEAD ALIGNMENT ON A DRIVE UNLESS THE DRIVE IS WRITE PROTECTED.

8. PROGRAM DESCRIPTION

8.1 VERIFICATION MODE

THE PROGRAM CHECKS HEAD ALIGNMENT AT CYLINDER 245, HEADS 0 - 18, AT CYLINDERS 400 AND 4, HEADS 0 AND 18, AND REVERFIES ALIGNMENT AT CYLINDER 245, HEADS 0 - 18. THE OPERATOR WILL BE NOTIFIED IF ANY HEAD IS OUT OF ALIGNMENT BY MORE THAN THE SPECIFIED AMOUNT.

HEAD ALIGNMENT IS CHECKED IN THE FOLLOWING MANNER:

1. OFFSET THE POSITIONER TO +1200 MICRO-INCHES.
2. STORE THE SIGN CHANGE BIT.
3. MOVE THE POSITIONER IN THE OPPOSITE DIRECTION IN 25 MICRO-INCH INCREMENTS UNTIL THE SIGN CHANGE BIT CHANGES VALUE. STORE THE OFFSET VALUE.
4. OFFSET THE POSITIONER TO -1200 MICRO-INCHES AND REPEAT STEPS 2 AND 3 ABOVE.
5. AVERAGE THE TWO SIGN CHANGE OFFSET VALUES AND REPORT IF THE SELECTED HEAD IS MISALIGNED BY MORE THAN + OR - 150 MICRO-INCHES FOR CYLINDER 245 OR + OR - 350 MICRO-INCHES FOR CYLINDERS 4 AND 400.

REPEAT THE ABOVE SEQUENCE FOR ALL HEADS AT CYLINDER 245 AND FOR HEADS 0 AND 18 AT CYLINDERS 4 AND 400.



K01

MAINDEC-11-DERPM-B  
DERPMB.P11

MACY11 27(732) 05-OCT-76 15:01 PAGE 11

319  
320  
321  
322

8.2

ALIGNMENT MODE  
THE PROGRAM MAY ALSO BE USED TO PROVIDE HEAD SELECTION FOR DDU  
CONTROLLED HEAD ALIGNMENT. WHEN THIS MODE IS SELECTED, THE  
PROGRAM WILL PERFORM NO ALIGNMENT CHECKING - ONLY THE REQUESTED



323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
40  
41  
42  
43  
44  
45  
46

HEAD IS SELECTED IN THE DCL. THE ACTUAL ALIGNMENT MUST BE PERFORMED USING THE ALIGNMENT METER ON THE DDU (OR DEDU). WHEN USED IN THE ALIGNMENT MODE, THE PROGRAM PROVIDES HEAD SELECTION WHICH IS NOT PRESENT IN THE DDU.

8.3 RANDOM SEEK UTILITY

THE PROGRAM CONTAINS A UTILITY ROUTINE WHICH PERFORMS 20,000 (10) RANDOM SEEK OPERATIONS ON THE DRIVE BEING CHECKED. THIS UTILITY ALLOWS THE OPERATOR TO EXERCISE THE HEAD ASSEMBLY ON THE DRIVE AFTER HEAD ALIGNMENT HAS BEEN PERFORMED.

THE UTILITY ROUTINE IS NORMALLY USED AFTER HEADS HAVE BEEN ALIGNED: THE RANDOM SEEK UTILITY IS CALLED AND HEAD ALIGNMENT IS RE-VERIFIED AT THE COMPLETION OF THE RANDOM SEEKS. USE OF THIS ROUTINE DOES NOT REQUIRE CYCLING DOWN THE DRIVE AND REPLACING THE ALIGNMENT PACK WITH A WORK PACK FOR THE HEAD SHAKE DOWN.

9. PROGRAM LISTING  
-----

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

DOCUMENT  
\*\*\*\*\*  
MAINDEC-11-DERPM-B  
\*\*\*\*\*

COPYRIGHT 1975  
DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASS. 01754



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  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441

TABLE OF CONTENTS  
\*\*\*\*\*

13	OPERATIONAL SWITCH SETTINGS
24	BASIC DEFINITIONS
129	TRAP CATCHER
136	STARTING ADDRESS(ES)
145	COMMON TAGS
192	RPO4 ADDRESS & VECTOR LOCATIONS
201	ERROR POINTER TABLE
313	RPO4 DRIVER COMMANDS
341	PROGRAM START AND INITIALIZATION ROUTINES
421	SETUP TO CHECK ONLY THE SPECIFIED HEAD
460	MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGNMENT CYLINDERS
498	ROUTINE TO FIND THE TRACK CENTER
596	ROUTINE TO SELECT HEAD FOR DJU CONTROLLED HEAD ALIGNMENT
623	ROUTINE TO PERFORM 5,000 RANDOM SEEKS
650	COMMON ENTRY TO THE RPO4 DRIVER
685	SUBROUTINES
891	MACRO ROUTINES
895	RANDOM NUMBER GENERATOR ROUTINE
942	ERROR HANDLER ROUTINE
974	ERROR MESSAGE TYPEOUT ROUTINE
1031	TYPE ROUTINE
1078	BINARY TO OCTAL (ASCII) AND TYPE





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

2 COPYRIGHT (C) 1973,1974,1975  
DIGITAL EQUIPMENT CORP.  
MAYNARD, MASS. 01754

PROGRAM BY C. HESS

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

13

\*\*\*\*\*  
OPERATIONAL SWITCH SETTINGS  
\*\*\*\*\*

14

SWITCH	USE
15	HALT ON ERROR
13	INHIBIT ERROR TYPEOUTS
9	LOOP ON ERROR
2	CHECK ONLY THE SPECIFIED HEAD
1	LOOP ON THE CURRENT HEAD
0	TYPE ALL TRACK CENTER VALUES

24

\*\*\*\*\*  
BASIC DEFINITIONS  
\*\*\*\*\*

- 26 INITIAL ADDRESS OF THE STACK POINTER \*\*\* 1100 \*\*\*
- 37 GENERAL PURPOSE REGISTER DEFINITIONS
- 49 PRIORITY LEVEL DEFINITIONS
- 59 "SWITCH REGISTER" SWITCH DEFINITIONS
- 87 DATA BIT DEFINITIONS (BIT00 TO BIT15)
- 115 BASIC "CPU" TRAP VECTOR ADDRESSES

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

```

*****
129 TRAP CATCHER
*****

132 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

*****
136 STARTING ADDRESS(ES)
*****

143 *****

*****
145 COMMON TAGS
*****

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

190 *****

*****
192 RPO4 ADDRESS & VECTOR LOCATIONS
*****

194 *****

199 *****

*****
201 ERROR POINTER TABLE
*****

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

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

311 *****

```



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

```
313 *****  
RPO4 DRIVER COMMANDS  
*****  
315 *****  
339 *****  
341 *****  
PROGRAM START AND INITIALIZATION ROUTINES  
*****  
346 *****  
419 *****  
421 *****  
SETUP TO CHECK ONLY THE SPECIFIED HEAD  
*****  
423 *****  
458 *****  
460 *****  
MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGNMENT CYLINDERS  
*****  
462 *****  
496 *****  
498 *****  
ROUTINE TO FIND THE TRACK CENTER  
*****  
500 *****  
594 *****
```

594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645

```
*****  
596 ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT  
*****  
598 *****  
621 *****  
623 *****  
ROUTINE TO PERFORM 5,000 RANDOM SEEKS  
*****  
625 *****  
648 *****  
650 *****  
COMMON ENTRY TO THE RPO4 DRIVER  
*****  
652 *****  
683 *****  
685 *****  
SUBROUTINES  
*****  
687 *****  
889 *****  
891 *****  
MACRO ROUTINES  
*****  
893 *****  
895 *****  
RANDOM NUMBER GENERATOR ROUTINE  
*****  
897 THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATO  
WITH A RANGE OF 0 TO 2(+33)-1.  
CALL:  
JSR PC,$RAND ;;CALL THE ROUTINE  
901 RETURN ;;RETURN HERE THE RANDOM  
;;NUMBER WILL BE IN  
;;SHINUM,SLONUM
```



646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695

```
940 *****
*****
942 ERROR HANDLER ROUTINE
*****
944 THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
      SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
      AND GO TO SERRTYP ON ERROR
      THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
      SW15=1 HALT ON ERROR
      SW13=1 INHIBIT ERROR TYPEOUTS
      CALL      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
972 *****
*****
974 ERROR MESSAGE TYPEOUT ROUTINE
*****
976 THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE
      ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE"
      AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
1029 *****
*****
1031 TYPE ROUTINE
*****
1033 ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 B
      THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE
      NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CH
      NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED
      NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.

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

      2) USING A JSR INSTRUCTION
          MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON
          JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
          MESADDR      ;;FIRST ADDRESS OF MESSAGE
```

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

```

1076 *****
1078 *****
      BINARY TO OCTAL (ASCII) AND TYPE
      *****
1080 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIG
      OCTAL (ASCII) NUMBER AND TYPE IT.
      $TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS
      CALL:
              MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
              TYPOS   ;;CALL FOR TYPEOUT
              .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS
              .BYTE   M              ;;M=1 OR 0
                                      ;;1=TYPE LEADING ZEROS
                                      ;;0=SUPPRESS LEADING ZER
      $TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE
      $TYPOS OR $TYPOC
      CALL:
              MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
              TYPON   ;;CALL FOR TYPEOUT
      $TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
      CALL:
              MOV     NUM,-(SP)      ;;NUMBER TO BE TYPED
              TYPOC   ;;CALL FOR TYPEOUT
1154 *****
1156 *****
      CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
      *****
1158 THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIG
      SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER
      NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE T
      BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS
      REPLACED WITH SPACES.
      CALL:
              MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE S
              TYPDS   ;;GO TO THE ROUTINE
1222 *****

```



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

1224 \*\*\*\*\*  
TRAP DECODER  
\*\*\*\*\*

1226 THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTIO  
AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDR  
OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL  
GO TO THAT ROUTINE.

1239 \*\*\*\*\*  
TRAP TABLE  
\*\*\*\*\*

1241 THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLE  
BY THE "TRAP" INSTRUCTION.

1258 \*\*\*\*\*  
RH11/RP04 DRIVER (REV. 0.9)  
\*\*\*\*\*

1260 COPYRIGHT (C) 1974  
DIGITAL EQUIPMENT CORP.  
MAYNARD, MA 01754  
AUTHOR: JIM LACEY

1266 STORAGE FOR RHDS1, RHER1, RHER2, AND RHER3 ON AN ERROR "2" OR "5  
RPERRS = RHDS1  
RPERRS+2 = RHER1  
RPERRS+4 = RHER2  
RPERRS+6 = RHER3

1277 TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)  
DRVACT=0 IMPLIES DRIVE IS IDLE

1279 DRVACT>0 IMPLIES DRIVE IS ACTIVE WITH A COMMAND  
DRVACT<0 IMPLIES DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATIO

1291 TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)  
DRVSTA=0 IMPLIES DRIVE IS OFFLINE  
DRVSTA>0 IMPLIES DRIVE IS ONLINE  
DRVSTA<0 IMPLIES DRIVE IS UNSAFE OR NONEXISTENT

1305 TABLE OF DRIVE TYPES (DRVSTYP=8 WORDS)  
DRVSTYP WILL CONTAIN THE DRIVE TYPE OF ALL ONLINE, OFFLINE, AND  
UNSAFE DRIVES. IF A DRIVE IS NONEXISTENT DRVSTYP WILL BE ZERO.

1318 TRANSFER WAIT FLAG (TRNSWT=1 WORD)  
THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF  
"DPB" OF THE I/O OPERATION.

793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843

- 1324 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.
- 1332 RPO4 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
- 1333 ACTDRV=0 IMPLIES DRIVER IS INACTIVE  
ACTDRV>0 IMPLIES DRIVER IS ACTIVE
- 1338 SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)  
ACTSTR=0 IMPLIES SOFTWARE TIMER ROUTINE IS INACTIVE  
ACTSTR>0 IMPLIES SOFTWARE TIMER ROUTINE IS ACTIVE
- 1344 UNLOAD FLAG (ULDFLG=8 BYTES)  
ULDFLG=0 IMPLIES NO UNLOAD COMMAND  
ULDFLG>0 IMPLIES UNLOAD COMMAND IN PROGRESS  
ULDFLG<0 IMPLIES UNLOAD COMMAND IN WAIT QUEUE
- 1358 LOOK AHEAD COUNT (LACNT=8 BYTES)  
LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
- 1370 SAVE REGISTERS FLAG (SAVEFG =1 WORD)  
SAVEFG <0 IMPLIES SAVE THE RH11/RPO4 REGISTERS WHEN THE  
OPERATION IS COMPLETED AS PER (DPB+14).  
SAVEFG=0 IMPLIES SAVE THE RH11/RPO4 REGISTERS, AS PER  
(DPB+14), AFTER AN ERROR.
- 1378 SEEK FLAG (SEEKFG=1 WORD)  
SEEKFG=0 IMPLIES WHEN THE DISK ADDRESS ISN'T IN THE WINDOW  
FOR A DATA TRANSFER START A SEARCH COMMAND  
SEEKFG<0 IMPLIES DATA TRANSFER WILL DO IMPLIED SEEKS,  
DISREGARD THE WINDOW
- 1386 TIMEOUT TABLE (TIMER=8 WORDS)
- 1387 THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
- 1398 DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)  
DTUW<0 IMPLIES NO DATA TRANSFER UNDERWAY  
DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE
- 1404 ATTENTION BITS TABLE (ATABIT=8 BYTES)  
THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES  
ATTENTION BIT
- 1417 RPO4 TO RH11 "MASS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEF  
CALLING IT FATAL (MCPMX=1 WORD)





900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951

```

1830 LOOK AHEAD ROUTINE
CALL
      MOV     #DRVNUM,R1      ;DRIVE NUMBER
      MOV     #DPB,R2        ;POINT TO DPB
      JSR     RO,LA          ;GO CHECK THE WINDOW
      RETURN1
      RETURN2
      RETURN3
      ;ERROR RETURN
      ;START A SEARCH
      ;START A DATA TRANSFER
      ;CI2=SEARCH REQUESTED BY DATA XF
      ;CI4=NOT DATA TRANSFER

1873 INTERRUPT SERVICE ROUTINE

1887 TRANSFER DONE ROUTINE

1914 SPECIAL CONDITION ROUTINE

2078 RPO4 TIMER ROUTINE
CALL
      MOV     #TIME,-(SP)    ;ELAPSED TIME IN MILLISECONDS ON
      JSR     RO,RPTMR      ;CALL RPO4 TIME ROUTINE

2104 SOFTWARE TIMEOUT ROUTINE
CALL:  STO
      MOV     #DRVNUM,R1    ;DRIVE NUMBER
      JSR     RO,STO        ;CALL--DRVACT MUST BE NONZERO
      RETURN

2169 ROUTINE TO READ A RH11/RPO4 REGISTER
CALL
      JSR     RO,RO.RP      ;GO READ A REGISTER
      INDEX  ERRADR        ;REG. INDEX FROM BASE
      ;ERROR ADDRESS--PROCESS ERROR ST
      ;AT THIS ADDRESS
      RETURN               ;CONTENTS OF REG. IS ON THE STAC

2212 ROUTINE TO WRITE A RH11/RPO4 REGISTER
CALL
      MOV     DATA,-(SP)   ;DATA TO BE LOADED ON THE STACK
      JSR     RO,WRT.RP    ;CALL THE ROUTINE TO LOAD(WRITE)
      INDEX  ERRADR        ;INDEX OF THE REGISTER TO BE LOA
      ;ADDRESS TO RETURN TO ON AN ERRO
      RETURN               ;ERROR FREE RETURN

2256 ROUTINE TO SAVE THE RH11/RPO4 REGISTERS AS PER DPB+14
CALL
      MOV     #DPBNUM,R2   ;DPB POINTER TO R2
      JSR     PC,SVRH11    ;SAVE THE DRIVES REG'S

```



952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007

```

2284 ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
CALL
      MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
      JSR    PC,SET.IE      ;SET "IE"
      RETURN

2305 QUEUE COUNT

2359 ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
CALL
      JSR    PC,CLRQUE

2383 EMPTY THE QUEUE SPECIFIED BY R1
CALL
      MOV     DRVNUM,R1      ;DRIVE NUMBER TO R1
      JSR    PC,EMPTYQ

2395 ROUTINE TO PUT A REQUEST IN QUEUE
CALL
      MOV     #DRVNUM,R1      ;DRIVE NUMBER
      MOV     #DPB,R2        ;ADDRESS OF PARAMETER BLOCK
      JSR    RO,DRVQUE       ;GO PUT REQUEST IN QUEUE
      RETURN1                ;RETURN HERE IF QUEUE IS FULL
      RETURN2                ;RETURN HERE IF REQUEST IS IN QU

2417 ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
CALL
      MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
      JSR    PC,GETREQ       ;GO GET THE REQUEST
      RETURN                  ;R2="DPB" ADDRESS OF THE REQUEST
                              ;R2=0 IF NO REQUEST IN QUEUE

2433 ROUTINE TO "POP" THE REQUEST FROM QUEUE
CALL
      MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
      JSR    PC,POPQUE       ;CALL TO REMOVE REQUEST
      RETURN                  ;R2=ADDRESS OF DPB REMOVED

2450 ROUTINES TO SAVE R0-R5 AND R1-R5
CALL: SAVR05
      JSR    RO,SAVR05
      RETURN                  ;R0-R5 IS ON THE STACK

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

UPON RETURN FROM SAVR05 AND SAVR15 THE STACK WILL LOOK LIKE:
+12   R0
+10   R1
+06   R2
+04   R3
    
```

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  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048

```
+02      R4
2467     TOP      R5
2478     ROUTINES TO RESTORE R0-R5 AND R1-R5
        CALL:    GETR05
                JSR      R0,GETR05
                RETURN   ;R0-R5 HAVE BEEN RESTORED
        CALL:    GETR15
                JSR      R0,GETR15
                RETURN   ;R1-R5 HAVE BEEN RESTORED
2498     *****
2500     *****
        DATA PARAMETER BLOCK
        *****
2502     *****
2557     *****
2559     *****
        HEAD CODE TABLE
        *****
2561     *****
2591     *****
2593     *****
        OFFSET CODE TABLE
        *****
2595     *****
2698     *****
```



1049  
1050  
1051  
1052  
1053  
1054

2700

\*\*\*\*\*  
MESSAGES  
\*\*\*\*\*

2702

\*\*\*\*\*

1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110

000001

001100

177776

177774

177772

177570

177570

000000

000001

000002

000003

000004

000005

000006

000007

000000

000040

000100

000140

000200

000240

```

:
:TITLE MAINDEC-11-DERPM-B
:*COPYRIGHT (C) 1973,1974,1975
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY C. HESS
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-A2).
:*
$TN=1

.SBTTL OPERATIONAL SWITCH SETTINGS
:*
:*      SWITCH              USE
:*      -----            -
:*      15                  HALT ON ERROR
:*      13                  INHIBIT ERROR TYPEOUTS
:*      9                   LOOP ON ERROR
:*      2                   CHECK ONLY THE SPECIFIED HEAD
:*      1                   LOOP ON THE CURRENT HEAD
:*      0                   TYPE ALL TRACK CENTER VALUES

.SBTTL BASIC DEFINITIONS

:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV EMT,ERROR          ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE         ;;BASIC DEFINITION OF SCOPE CALL
PS= 177776                ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
STKLMT= 177774           ;;STACK LIMIT REGISTER
PIRQ= 177772            ;;PROGRAM INTERRUPT REQUEST REGISTER
SWR= 177570             ;;SWITCH REGISTER
DISPLAY=SWR

:*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
.EQUIV R6,SP             ;;STACK POINTER
.EQUIV R7,PC             ;;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

```



1111 000300 RP6= 300 ;;PRIORITY LEVEL 6  
1112 000340 PR7= 340 ;;PRIORITY LEVEL 7

.\*"SWITCH REGISTER" SWITCH DEFINITIONS

1114  
1115 100000 SW15= 100000  
1116 040000 SW14= 40000  
1117 020000 SW13= 20000  
1118 010000 SW12= 10000  
1119 004000 SW11= 4000  
1120 002000 SW10= 2000  
1121 001000 SW09= 1000  
1122 000400 SW08= 400  
1123 000200 SW07= 200  
1124 000100 SW06= 100  
1125 000040 SW05= 40  
1126 000020 SW04= 20  
1127 000010 SW03= 10  
1128 000004 SW02= 4  
1129 000002 SW01= 2  
1130 000001 SW00= 1  
1131 .EQUIV SW09, SW9  
1132 .EQUIV SW08, SW8  
1133 .EQUIV SW07, SW7  
1134 .EQUIV SW06, SW6  
1135 .EQUIV SW05, SW5  
1136 .EQUIV SW04, SW4  
1137 .EQUIV SW03, SW3  
1138 .EQUIV SW02, SW2  
1139 .EQUIV SW01, SW1  
1140 .EQUIV SW00, SW0

.\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

1142  
1143 100000 BIT15= 100000  
1144 040000 BIT14= 40000  
1145 020000 BIT13= 20000  
1146 010000 BIT12= 10000  
1147 004000 BIT11= 4000  
1148 002000 BIT10= 2000  
1149 001000 BIT09= 1000  
1150 000400 BIT08= 400  
1151 000200 BIT07= 200  
1152 000100 BIT06= 100  
1153 000040 BIT05= 40  
1154 000020 BIT04= 20  
1155 000010 BIT03= 10  
1156 000004 BIT02= 4  
1157 000002 BIT01= 2  
1158 000001 BIT00= 1  
1159 .EQUIV BIT09, BIT9  
1160 .EQUIV BIT08, BIT8  
1161 .EQUIV BIT07, BIT7  
1162 .EQUIV BIT06, BIT6  
1163 .EQUIV BIT05, BIT5  
1164 .EQUIV BIT04, BIT4  
1165 .EQUIV BIT03, BIT3  
1166 .EQUIV BIT02, BIT2

```

1167      .EQUIV BIT01,BIT1
1168      .EQUIV BIT00,BIT0
1169
1170      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1171      000004      ERRVEC= 4      ;; TIME OUT AND OTHER ERRORS
1172      000010      RESVEC= 10     ;; RESERVED AND ILLEGAL INSTRUCTIONS
1173      000014      TBITVEC=14     ;; "T" BIT
1174      000014      TRTVEC= 14     ;; TRACE TRAP
1175      000014      BPTVEC= 14     ;; BREAKPOINT TRAP (BPT)
1176      000020      IOTVEC= 20     ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
1177      000024      PWRVEC= 24     ;; POWER FAIL
1178      000030      EMTVEC= 30     ;; EMULATOR TRAP (EMT) **ERROR**
1179      000034      TRAPVEC=34     ;; "TRAP" TRAP
1180      000060      TKVEC= 60      ;; TTY KEYBOARD VECTOR
1181      000064      TPVEC= 64      ;; TTY PRINTER VECTOR
1182      000240      PIRQVEC=240    ;; PROGRAM INTERRUPT REQUEST VECTOR
1183
1184      .SBTTL TRAP CATCHER
1185
1186      000000      .=0
1187      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1188      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1189      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1190
1191      .SBTTL STARTING ADDRESS(ES)
1192      000200      .=200
1193
1194      000200      000137      001356      JMP      @#START      ;; JUMP TO STARTING ADDRESS OF PROGRAM
1195      000204      000137      001356      JMP      @#START      ;; RESTART THE PROGRAM - ADDRESS CHANGES TO
1196                                          ;; 'INIT' AFTER PROGRAM STARTED INITIALLY
1197

```



```

1198
1199
1200
1201
1202
1203
1204
1205      001100
1206
1207      001100
1208      001100  000000
1209      001102    000
1210      001103    000
1211      001104  000000
1212      001106  000000
1213      001110  000000
1214      001112  000000
1215      001114    000
1216      001115    001
1217      001116  000000
1218      001120  000000
1219      001122  000000
1220      001124  000000
1221      001126  000000
1222      001130  000000  000000  000000
1223      001136  177560
1224      001140  177562
1225      001142  177564
1226      001144  177566
1227      001146    000
1228      001147    002
1229      001150    012
1230      001151    000
1231      001152    077
1232      001153    015
1233      001154  000012
1234      001156  000000
1235      001160  000000
1236      001162  000000
1237      001164  000000
1238      001166  000000
1239      001170  000000
1240      001172  000000
1241      001174  000000
1242      001176  000000
1243      001200  000000
1244
1245
1246
1247
1248
1249
1250
1251      001202  176700
1252      001204  000254
1253

```

\*\*\*\*\*

.SBTTL COMMON TAGS

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

.=1100

```

$CMTAG:
$PASS:  .WORD  0
$STNM:  .BYTE  0
$ERFLG: .BYTE  0
$ICNT:  .WORD  0
$LPADR: .WORD  0
$LPERR: .WORD  0
$ERTTL: .WORD  0
$ITEMB: .BYTE  0
$ERMAX: .BYTE  1
$ERRPC: .WORD  0
$GDADR: .WORD  0
$BDADR: .WORD  0
$GDDAT: .WORD  0
$BDDAT: .WORD  0,0,0
$TKS:   177560
$TKB:   177562
$TPS:   177564
$TPB:   177566
$NULL:  .BYTE  0
$FILLS: .BYTE  2
$FILLC: .BYTE 12
$TPFLG: .BYTE  0
$QUES:  .ASCII  /?/
$CRLF:  .ASCII  <15>
$LF:    .ASCII  <12>
$DRIVE: .WORD  0
$ATTN:  .WORD  0
$SHEAD: .WORD  0
$TOPLN: .WORD  0
$OFFDIR: .WORD  0
$BITIND: .WORD  0
$SINCNG: .WORD  0
$PLUS:  .WORD  0
$INREG: .WORD  0
$TOLER: .WORD  0

```

```

:: START OF COMMON TAGS
:: CONTAINS PASS COUNT
:: CONTAINS THE TEST NUMBER
:: CONTAINS ERROR FLAG
:: CONTAINS SUBTEST ITERATION COUNT
:: CONTAINS SCOPE LOOP
:: CONTAINS SCOPE RETURN FOR ERRORS
:: CONTAINS TOTAL ERRORS DETECTED
:: CONTAINS ITEM CONTROL BYTE
:: CONTAINS MAX. ERRORS PER TEST
:: CONTAINS PC OF LAST ERROR INSTRUCTION
:: CONTAINS OF 'GOOD' DATA
:: CONTAINS OF 'BAD' DATA
:: CONTAINS 'GOOD' DATA
:: CONTAINS 'BAD' DATA
:: RESERVED--NOT TO BE USED
:: TTY KBD STATUS
:: TTY KBD BUFFER
:: TTY PRINTER STATUS REG.
:: TTY PRINTER BUFFER REG.
:: CONTAINS NULL CHARACTER FOR FILLS
:: CONTAINS # OF FILLER CHARACTERS REQUIRED
:: INSERT FILL CHARS. AFTER A "LINE FEED"
:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
:: QUESTION MARK
:: CARRIAGE RETURN
:: LINE FEED
:: DRIVE NUM STORAGE FOR DRIVER ERR CALLS
:: ATTENTION REGISTER STORAGE FOR ERROR MESSAGES
:: HEAD ADDRESS STORED FOR ERROR MESSAGES
:: 'TYPE THE HEADER' INDICATOR
:: NEG OFFSET PASS IND
:: SIGN CHANGE BIT STORED INDICATOR
:: SIGN CHANGE BIT STORAGE
:: STORE THE POS SIGN CHANGE CODE
:: CONTAINS THE VALUE READ FROM THE REGISTER
:: ALIGNMENT TOLERANCE FOR PRESENT CYLINDER

```

\*\*\*\*\*

.SBTTL RPO4 ADDRESS & VECTOR LOCATIONS

\*\*\*\*\*

```

$RPADR: .WORD  176700      ;RH11/RPO4 UNIBUS ADDRESS
$RPVEC: .WORD  254        ;RH11/RPO4 VECTOR ADDRESS

```

```

1254 ;*****
1255
1256 .SBTTL ERROR POINTER TABLE
1257
1258 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
1259 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
1260 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
1261 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
1262 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
1263
1264 ;* EM ;:POINTS TO THE ERROR MESSAGE
1265 ;* DH ;:POINTS TO THE DATA HEADER
1266 ;* DT ;:POINTS TO THE DATA
1267 ;* DF ;:POINTS TO THE DATA FORMAT
1268
1269
1270 001206 SERRTB:
1271
1272 ;ERROR 1
1273
1274 001206 015200 EM1 ;ILLEGAL RH11 INTERRUPT
1275 001210 000000 0
1276 001212 000000 0
1277 001214 000000 0
1278
1279 ;ERROR 2
1280
1281 001216 015250 EM2 ;UNEXPECTED ATTENTION DETECTED
1282 001220 015736 DH2 ;DRV RHAS
1283 001222 016266 DT2 ;DATA POINTER
1284 001224 016362 DF2 ;DRV-DEC, REG=OCTAL
1285
1286 ;ERROR 3
1287
1288 001226 015275 EM3 ;MASSBUS CONTROL BUS PARITY ERROR (SEEN BY THE RH11)
1289 001230 016013 DH3 ;DRV ADDR OF REG CONTENTS
1290 001232 016304 DT3 ;DATA POINTER
1291 001234 016370 DF3 ;ALL OCTAL EXCEPT DRV
1292
1293 ;ERROR 4
1294
1295 001236 015353 EM4 ;CONTROL BUS PARITY ERROR (SEEN BY THE RPO4)
1296 001240 016041 DH4 ;DRV REG ADDR WRITTEN READ
1297 001242 016314 DT4 ;DATA POINTER
1298 001244 016373 DF4 ;OCTAL EXCEPT DRV
1299
1300 ;ERROR 5
1301
1302 001246 015431 EM5 ;ATTEN FROM AN OFFLINE OR UNAVAIL DRIVE
1303 001250 015736 DH2 ;DRV RHAS
1304 001252 016266 DT2 ;DATA POINTER
1305 001254 016362 DF2 ;OCTAL
1306
1307 ;ERROR 6 ;RESERVED
1308
1309 001256 000000 0

```



1310	001260	000000	0	
1311	001262	000000	0	
1312	001264	000000	0	
1313				
1314				;ERROR 7 ;RESERVED
1315				
1316	001266	000000	0	
1317	001270	000000	0	
1318	001272	000000	0	
1319	001274	000000	0	
1320				
1321				;ERROR 10
1322				
1323	001276	015470	EM10	;DEVICE ERROR
1324	001300	016100	DH10	;CS1 CS2 DS1 ER1,ER2,ER3
1325	001302	016326	DT10	;DATA POINTER
1326	001304	016377	DF10	;ALL ARE OCTAL
1327				
1328				;ERROR 11
1329				
1330	001306	015505	EM11	;DEVICE UNSAFE
1331	001310	016100	DH10	;CS1 CS2 DS1 ER1,ER2,ER3
1332	001312	016326	DT10	;DATA POINTER
1333	001314	016377	DF10	;ALL ARE OCTAL
1334				
1335				;ERROR 12
1336				
1337	001316	015531	EM12	;HEAD OUT OF ALIGNMENT
1338	001320	016156	DH12	;HEAD TRK CENTER IN U INCHES CYL
1339	001322	016344	DT12	;DATA POINTER
1340	001324	016405	DF12	;BOTH ARE DECIMAL
1341				
1342				;ERROR 13
1343				
1344	001326	015557	EM13	;HEAD OUT OF ALIGNMENT BY MORE THAN + OR - 1200 U INCHES
1345	001330	016247	DH13	;HEAD CYL
1346	001332	016354	DT13	;DATA POINTER
1347	001334	016410	DF13	;DECIMAL
1348				
1349				;ERROR 14
1350				
1351	001336	015640	EM14	;FATAL MASSBUS PARITY ERROR
1352	001340	000000	0	
1353	001342	000000	0	
1354	001344	000000	0	
1355				
1356				;ERROR 15
1357				
1358	001346	015673	EM15	;UNCORRECTABLE MASSBUS PARITY ERROR
1359	001350	000000	0	
1360	001352	000000	0	
1361	001354	000000	0	
1362				
1363				
1364				
1365				

1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421

000101  
000103  
000105  
000107  
000111  
000113  
000115  
000117  
000121  
000123  
000131  
000141  
000143  
000145  
000151  
000153  
000161  
000163  
000171  
000173

\*\*\*\*\*

.SBTTL RPO4 DRIVER COMMANDS

\*\*\*\*\*

RNOP= 101 ; NO OPERATION  
UNLOAD= 103 ; UNLOAD  
SEEK= 105 ; SEEK  
RECAL= 107 ; RECALIBRATE  
DRVCLR= 111 ; DRIVE CLEAR  
REL= 113 ; RELEASE  
OFFSET= 115 ; OFFSET  
RTC= 117 ; RETURN TO CENTER LINE  
PRESET= 121 ; READ IN PRESET  
ACK= 123 ; PACK ACKNOWLEDGE  
SEARCH= 131 ; SEARCH  
GETREG= 141 ; GET REGISTERS  
SETFMT= 143 ; SET FORMAT (& ECI OR HCI)  
SELDRV= 145 ; SELECT DRIVE  
WCHKD= 151 ; WRITE CHECK DATA  
WCHKHD= 153 ; WRITE CHECK HEADER & DATA  
WRTDAT= 161 ; WRITE DATA  
WRTHD= 163 ; WRITE HEADER & DATA  
RDDAT= 171 ; READ DATA  
RDHD= 173 ; READ HEADER & DATA

\*\*\*\*\*

.SBTTL PROGRAM START AND INITIALIZATION ROUTINES

STARTING ADDRESS = 200  
RESTART ADDRESS = 204

\*\*\*\*\*

START:

MOV #340, @#PS ; LOCK OUT ALL INTERRUPTS  
MOV #SCMTAG, R6 ; FIRST LOCATION TO BE CLEARED  
CLR (R6)+ ; CLEAR MEMORY LOCATION  
CMP #STKS, R6 ; DONE?  
BNE .-6 ; LOOP BACK IF NO  
MOV #STACK, SP ; SETUP THE STACK POINTER  
MOV #ERROR, @#EMTVEC ; EMT VECTOR FOR ERROR ROUTINE  
MOV #340, @#EMTVEC+2 ; LEVEL 7  
MOV #STRAP, @#TRAPVEC ; TRAP VECTOR FOR TRAP CALLS  
MOV #340, @#TRAPVEC+2 ; LEVEL 7  
MOV #123456, \$LONUM ; INIT RANDOM NUMBER GENERATOR  
MOV #176543, \$SHINUM ; INIT RANDOM NUMBER GENERATOR  
1\$: BR 2\$ ; TYPE SETUP INFO  
BR ; BYPASS SETUP INFO  
2\$: TYPE , TITLE ; TYPE PROGRAM TITLE  
TYPE , ODU ; TYPE 'DDU' SETUP MESSAGE  
TYPE , IXEGND ; GROUND 'IXE' MESSAGE  
TYPE, \$CRLF ; CR-LF

001356  
001356 012737 000340 177776  
001364 012706 001100  
001370 005026  
001372 022706 001136  
001376 001374  
001400 012706 001100  
001404 012737 004434 000030  
001412 012737 000340 000032  
001420 012737 005472 000034  
001426 012737 000340 000036  
001434 012737 123456 004432  
001442 012737 176543 004430  
000401  
000424  
104400 013420  
104400 013522  
104400 014000  
104400 001153



1422	001474	013737	001202	005672		MOV	\$RPADR,RPADR		;MOVE RPO4 ADDRESS TO DRIVER
1423	001502	013737	001204	005674		MOV	\$RPVEC,RPVEC		;VECTOR ADDRESS
1424	001510	012737	001524	000206		MOV	#INIT,206		;SETUP RESTART ADDRESS
1425	001516	012737	000240	001450		MOV	#NOP,1\$		;BYPASS SWITCH FOR SETUP MESSAGES
1426	001524	005037	177776		INIT:	CLR	PS		;SET PROCESSOR PRIORITY TO ZERO
1427	001530	005037	001164			CLR	TOPLN		;CLEAR HEADER INDICATOR
1428	001534	104400	014366			TYPE	,ENTERD		;ASK FOR DRIVE ASSIGNMENT
1429	001540	004737	003642			JSR	PC,GETDEC		;GET DRIVE NUMBER
1430	001544	000406				BR	5\$		; 'CR' ENTERED
1431	001546	112637	013102			MOVB	(SP)+,DPB		;DRIVE NUMBER TO PAR BLOCK
1432	001552	123727	013102	000010		CMPB	DPB,#8.		;CHECK DRIVE NUMBER
1433	001560	103403				BLO	6\$		;BR IF DRIVE NUMBER OK
1434	001562	104400	001152		5\$:	TYPE	,SQUES		;TYPE QUESTION MARK
1435	001566	000756				BR	INIT		;TRY AGAIN
1436	001570	004737	005710		6\$:	JSR	PC,RPINIT		;INITIALIZE THE DRIVER
1437	001574	012737	177777	005634		MOV	#-1,SEEKFG		;SET SEEK ALLOW FLAG
1438	001602	012737	177777	005632		MOV	#-1,SAVEFG		;SAVE RPO4 REGISTERS
1439	001610	005000				CLR	RD		;CLEAR RD FOR THE DRIVE NUMBER
1440	001612	153700	013102			BISB	DPB,RD		;USE DRIVE NUMBER AS AN INDEX
1441	001616	105760	005554			TSTB	DRVSTA(RD)		;SEE IF DRIVE AVAILABLE
1442	001622	003004				BGT	1\$		;BRANCH IF IT IS
1443	001624	104400	014335			TYPE	,NODRV		; 'DRIVE NOT AVAILABLE'
1444	001630	000137	001524			JMP	INIT		;TRY IT AGAIN
1445	001634	112737	000012	013112	1\$:	MOVB	#RHDS1,DPB+SEC		;ADDR OF RHDS1
1446	001642	112737	000012	013113		MOVB	#RHDS1,DPB+TRK		;ADDR OF RHDS1
1447	001650	012737	001176	013110		MOV	#INREG,DPB+BUF		;READ REG HERE
1448	001656	004537	003252			JSR	R5,DRIVER		;READ THE REGISTER
1449	001662	000141				.WORD	GETREG		;DRIVER CODE
1450	001664	032737	004000	001176		BIT	#BIT11,INREG		;SEE IF WRITE LOCK SET
1451	001672	001003				BNE	2\$		;BR IF WRITE LOCKED
1452	001674	104400	014415			TYPE	,WLOCK		;REPORT NOT WRITE LOCKED
1453	001700	000711				BR	INIT		;TRY IT AGAIN
1454	001702	104400	014555		2\$:	TYPE	,MODE		;SEE IF ALIGN, VERIFY, OR EXERCISE
1455	001706	104416				RDCHR			;GET ENTRY
1456	001710	012637	002010			MOV	(SP)+,4\$		;SAVE THE ENTRY
1457	001714	122737	000126	002010		CMPB	#'V',4\$		;WAS A 'V' ENTERED ?
1458	001722	001422				BEQ	3\$		;BR IF IT WAS
1459	001724	022737	000101	002010		CMP	#'A',4\$		;WAS AN 'A' ENTERED ?
1460	001732	001002				BNE	+.6		;BR IF NOT
1461	001734	000137	003050			JMP	DDUALN		;HEADS TO BE ALIGNED WITH DDU
1462	001740	022737	000105	002010		CMP	#'E',4\$		;SEE IF EXERCISE
1463	001746	001002				BNE	+.6		;BR IF 'E' NOT ENTERED
1464	001750	000137	003150			JMP	RANDOM		;GO TO EXERCISE ROUTINE
1465	001754	104400	002010			TYPE	,4\$		;ECHO THE INVALID CHARACTER
1466	001760	104400	001152			TYPE	,SQUES		;TYPE A QUESTION MARK
1467	001764	000137	001524			JMP	INIT		;GO ALL THE WAY BACK TO DRIVE ENTRY
1468	001770	104400	014621		3\$:	TYPE	,VERIFY		;TYPE 'VERIFY'
1469	001774	032737	000004	177570		BIT	#SW2,SWR		;SWITCH 2 SET ?
1470	002002	001500				BEQ	CYLDER		;BR IF NOT SET - CHECK ALL HEADS
1471	002004	000137	002012			JMP	ONEHD		;CHECK ONLY THE SPECIFIED HEAD
1472	002010	000000			4\$:	.WORD	0		;OPERATOR ENTRY GOES HERE
1473									
1474									
1475									
1476									
1477									

\*\*\*\*\*  
.SBTTL SETUP TO CHECK ONLY THE SPECIFIED HEAD

```

1478 ;*****
1479
1480 002012 104400 015047 ONEHD: TYPE ENT CYL ;ASK FOR CYLINDER ADDRESS
1481 002016 004737 003642 JSR PC,GETDEC ;GET THE CYLINDER ADDRESS
1482 002022 000427 BR 1$ ;'CR' ENTERED
1483 002024 012637 013114 MOV (SP)+,DPB+CYL ;MOVE IT TO PARAMETER BLOCK
1484 002030 012737 000536 001200 MOV #350.,TOLER ;SET INITIAL ALIGNMENT TOLERANCE TO 350 U INCHES
1485 002036 022737 000004 013114 CMP #4,DPB+CYL ;SEE IF CYLINDER = 004
1486 002044 001423 BEQ 2$ ;BR IF IT IS
1487 002046 022737 000620 013114 CMP #400.,DPB+CYL ;CYLINDER = 400
1488 002054 001417 BEQ 2$ ;BR IF 400
1489 002056 012737 000226 001200 MOV #150.,TOLER ;CHANGE TOLERANCE TO 150 U INCHES
1490 002064 022737 000365 013114 CMP #245.,DPB+CYL ;SEE IF CYLINDER < 240
1491 002072 001410 BEQ 2$ ;BR IF CYLINDER = 245
1492 002074 104400 015131 TYPE ,BADCYL ;REPORT INVALID CYLINDER
1493 002100 000744 BR ONEHD ;TRY IT AGAIN
1494 002102 104400 014716 1$: TYPE ,CYL245 ;CYLINDER 245 DEFAULT MESSAGE
1495 002106 012737 000365 013114 MOV #245.,DPB+CYL ;CYLINDER ADDRESS
1496 002114 104400 015030 2$: TYPE ,ENTHD ;ASK FOR HEAD NUMBER
1497 002120 004737 003642 JSR PC,GETDEC ;GET THE HEAD NUMBER
1498 002124 000411 BR 3$ ;'CR' ENTERED
1499 002126 012637 001162 MOV (SP)+,$HEAD ;SAVE HEAD NUMBER
1500 002132 022737 000022 001162 CMP #18.,$HEAD ;SEE IF VALID HEAD ADDRESS
1501 002140 002007 BGE 4$ ;BR IF IT IS
1502 002142 104400 015102 TYPE ,BADTRK ;TYPE INVALID HEAD ADDRESS MESSAGE
1503 002146 000762 BR 2$ ;TRY AGAIN
1504 002150 104400 014660 3$: TYPE ,HDZERO ;HEAD 0 DEFAULT MESSAGE
1505 002154 005037 001162 CLR $HEAD ;CLEAR HEAD STORAGE
1506 002160 004537 003252 4$: JSR R5,DRIVER ;DO A RECALIBRATE
1507 002164 000107 .WORD RECAL ;DRIVER CODE FOR RECALIBRATE
1508 002166 004537 003252 JSR R5,DRIVER ;SEEK TO THE SPECIFIED CYLINDER
1509 002172 000105 .WORD SEEK ;DRIVER SEEK CODE
1510 002174 004737 002376 JSR PC,CHECK ;CHECK THE HEAD
1511 002200 000137 001524 JMP INIT ;GET NEXT DRIVE/HEAD
1512
1513 ;*****
1514
1515 .SBTTL MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGNMENT CYLINDERS
1516
1517 ;*****
1518
1519 002204 004537 003252 CYLDER: JSR R5,DRIVER ;RECALIBRATE TO RESET THE
1520 002210 000107 .WORD RECAL ;CURRENT CYLINDER REGISTER
1521 002212 012737 000365 013114 1$: MOV #245.,DPB+CYL ;START AT CYLINDER 245
1522 002220 012737 000226 001200 MOV #150.,TOLER ;ALIGNMENT TOLERANCE = 150 U IN
1523 002226 004537 003252 JSR R5,DRIVER ;SEEK TO CYLINDER 245
1524 002232 000105 .WORD SEEK ;DRIVER SEEK CODE
1525 002234 012704 013172 MOV #HEAD1,R4 ;ADDRESS OF 0 - 18 HEAD TABLE
1526 002240 004737 002360 JSR PC,2$ ;CHECK ALIGNMENT ALL HEADS, CYL 245
1527 002244 012737 000620 013114 MOV #400.,DPB+CYL ;CHANGE CYLINDER TO 400
1528 002252 012737 000536 001200 MOV #350.,TOLER ;CHANGE ALIGN TOLER TO 350 U INCHES
1529 002260 004537 003252 JSR R5,DRIVER ;SEEK TO CYLINDER 400
1530 002264 000105 .WORD SEEK ;DRIVER SEEK CODE
1531 002266 012704 013242 MOV #HEAD2,R4 ;CHANGE HEAD TABLE ADDRESS
1532 002272 004737 002360 JSR PC,2$ ;CHECK ALIGNMENT AT CYL 400
1533 002276 012737 000004 013114 MOV #4,DPB+12 ;CHANGE CYLINDER TO 004

```



1534	002304	004537	003252		JSR	R5, DRIVER	; SEEK TO CYLINDER 4
1535	002310	000105			.WORD	SEEK	; DRIVER SEEK CODE
1536	002312	004737	002360		JSR	PC, 2\$	; CHECK ALIGNMENT AT CYL 004
1537	002316	012737	000365	013114	MOV	#245., DPB+CYL	; GO BACK TO CYLINDER 245
1538	002324	012737	000226	001200	MOV	#150., TOLER	; CHANGE ALIGN TOLER BACK TO 150 U INCHES
1539	002332	004537	003252		JSR	R5, DRIVER	; SEEK
1540	002336	000105			.WORD	SEEK	; DRIVER SEEK CODE
1541	002340	012704	013172		MOV	#HEAD1, R4	; ADDRESS OF 0 - 18 HEAD ADDRESS TABLE
1542	002344	004737	002360		JSR	PC, 2\$	; RECHECK ALIGNMENT AT CYLINDER 245
1543	002350	104400	015164		TYPE	ENDMSG	; TYPE 'DONE'
1544	002354	000137	001524		JMP	INIT	; GET NEXT DRIVE OR HEAD
1545	002360	012437	001162	2\$:	MOV	(R4)+, \$HEAD	; MOVE HEAD ADDRESS FROM TABLE
1546	002364	100403			BMI	3\$	; BR IF END OF TABLE
1547	002366	004737	002376		JSR	PC, CHECK	; CHECK THE ALIGNMENT
1548	002372	000772			BR	2\$	; GET NEXT HEAD VALUE
1549	002374	000207		3\$:	RTS	PC	; RETURN

;\*\*\*\*\*

.SBTTL ROUTINE TO FIND THE TRACK CENTER

;\*\*\*\*\*

1557	002376	005037	001166	CHECK:	CLR	OFFDIR	; CLEAR THE OFFSET DIRECTION INDICATOR
1558	002402	005037	001170		CLR	BITIND	; CLEAR THE SIGN BIT INDICATOR
1559	002406	012702	002260		MOV	#1200., R2	; MAXIMUM POSITIVE OFFSET
1560	002412	012703	000141		MOV	#97., R3	; MAXIMUM NUMBER OF OFFSETS
1561	002416	012701	013256		MOV	#OFFTBL, R1	; OFFSET VALUE TABLE ADDRESS
1562	002422	013746	001162		MOV	\$HEAD, -(SP)	; HEAD VALUE TO STACK
1563	002426	000316			SWAB	(SP)	; SWITCH FOR RHDA LOAD
1564	002430	004037	011776		JSR	RO, WRT.RP	; LOAD RHDA WITH HEAD ADDRESS
1565	002434	000006			RHDA		; REGISTER ADDRESS
1566	002436	001524			INIT		; UNCORRECTABLE PARITY ERROR RETURN
1567	002440	112137	013103	1\$:	MOVB	(R1)+, DPB+CODE	; OFFSET VALUE TO PARAMETER BLOCK
1568	002444	162702	000031		SUB	#25., R2	; DECREMENT OFFSET VALUE
1569	002450	000404			BR	21\$	; BYPASS REVERSE DIR SETUP
1570	002452	114137	013103	2\$:	MOVB	-(R1), DPB+CODE	; REVERSE OFFSET VALUE TO PAR BLOCK
1571	002456	062702	000031		ADD	#25., R2	; INCREMENT OFFSET VALUE
1572	002462	004537	003252	21\$:	JSR	R5, DRIVER	; OFFSET THE DRIVE
1573	002466	000115			.WORD	OFFSET	; OFFSET OP CODE
1574	002470	005737	001170		TST	BITIND	; INITIAL SIGN VALUE STORED ?
1575	002474	001554			BEQ	10\$	; BRANCH IF NOT
1576	002476	042737	077777	013154	BIC	#77777, REG+RHOF	; KEEP ONLY THE SIGN VALUE
1577	002504	023737	013154	001172	CMP	REG+RHOF, SINCNG	; SIGN CHANGED ?
1578	002512	001006			BNE	4\$	; BRANCH IF IT DID
1579	002514	005303		3\$:	DEC	R3	; DECREMENT THE OFFSET ATTEMPT COUNTER
1580	002516	001510			BEQ	8\$	; BRANCH IF OFFSETS COMPLETED
1581	002520	005737	001166		TST	OFFDIR	; SEE WHICH DIRECTION
1582	002524	001745			BEQ	1\$	; BRANCH IF FORWARD
1583	002526	000751			BR	2\$	; REVERSE DIRECTION
1584	002530	005737	001166	4\$:	TST	OFFDIR	; FINISHED WITH REVERSE ?
1585	002534	001015			BNE	5\$	; BRANCH IF YES
1586	002536	010237	001174		MOV	R2, PLUS	; SAVE THE SIGN CHANGE VALUE
1587	002542	012702	175520		MOV	#-1200., R2	; MAXIMUM NEGATIVE OFFSET VALUE
1588	002546	012703	000141		MOV	#97., R3	; MAXIMUM NUMBER OF OFFSETS
1589	002552	012701	013417		MOV	#ENDTBL, R1	; BEGINNING OF NEG DIRECTION VALUES



```

1590 002556 005137 001166      CUM      OFFDIR      ;SET REVERSE DIR INDICATOR
1591 002562 005037 001170      CLR      BITIND      ;RESET SIGN BIT INDICATOR
1592 002566 000731                BR        2$          ;START REVERSE OFFSETS
1593 002570 063702 001174      5$:     ADD      PLUS,R2 ;ADD THE TWO SIGN CHANGE VALUES
1594 002574 006202                ASR      R2          ;'DIVIDE' BY 2
1595 002576 010237 001174      MOV      R2,PLUS     ;SAVE THE RESULT
1596 002602 032737 000001 177570    BIT      #SW0,SWR    ;SEE IF SWRO SET
1597 002610 001010                BNE      6$          ;BRANCH IF IT IS
1598 002612 005702                TST      R2          ;SEE IF VALUE NEGATIVE
1599 002614 100001                BPL      .+4         ;BRANCH IF POSITIVE
1600 002616 005402                NEG      R2          ;RECOMPLEMENT THE VALUE
1601 002620 023702 001200    CMP      TOLER,R2   ;SEE IF TRACK CENTERLINE WITHIN TOLERANCE
1602 002624 002036                BGE      7$          ;BRANCH IF VALUE OK
1603 002626 104012                ERROR    12         ;REPORT THE ERROR
1604 002630 000470                BR        9$          ;CHECK FOR LOOP ON ERROR
1605 002632 010246                6$:     MOV      R2,-(SP) ;SAVE TRACK CENTER VALUE
1606 002634 005737 001164    TST      TOPLN      ;TYPE HEADER ?
1607 002640 001005                BNE      61$         ;BR IF NOT
1608 002642 104400 014463    TYPE     ,HEADER    ;TYPE THE HEADER
1609 002646 012737 177777 001164    MOV      #-1,TOPLN ;CLEAR THE INDICATOR
1610 002654                61$:
1611 002654 013746 001162    MOV      $HEAD,-(SP) ;:SAVE $HEAD FOR TYPEOUT
1612 002660 104410                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
1613 002662 013746 013114    MOV      DPB+CYL,-(SP) ;:SAVE DPB+CYL FOR TYPEOUT
1614 002666 104410                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
1615 002670 013746 001174    MOV      PLUS,-(SP) ;:SAVE PLUS FOR TYPEOUT
1616 002674 104410                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
1617 002676 104400 001153    TYPE     ,%CRLF     ;
1618 002702 023726 001200    CMP      TOLER,(SP)+ ;SEE IF HEAD IN TOLERANCE
1619 002706 002005                BGE      7$          ;BR IF IN TOLERANCE
1620 002710 032737 100000 177570    BIT      #SW15,SWR  ;HALT IF ERROR ?
1621 002716 001401                BEQ      7$          ;BR IF NOT
1622 002720 000000                HALT     ;HEAD JUST TYPED OUT OF TOLERANCE
1623 002722 032737 000002 177570    7$:     BIT      #SW1,SWR  ;SWITCH 1 SET ?
1624 002730 001001                BNE      .+4         ;BR IF SET - LOOP ON HEAD
1625 002732 000207                RTS      PC          ;RETURN
1626 002734 000137 002376    JMP      CHECK       ;DO THE HEAD AGAIN
1627 002740 032737 000001 177570    8$:     BIT      #SW0,SWR  ;CHECK SWRO
1628 002746 001002                BNE      11$         ;BR IF SET
1629 002750 104013                ERROR    13         ;REPORT > 1200 ERROR
1630 002752 000417                BR        9$          ;SEE IF LOOP ON ERROR
1631 002754                11$:
1632 002754 013746 001162    MOV      $HEAD,-(SP) ;:SAVE $HEAD FOR TYPEOUT
1633 002760 104410                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
1634 002762 013746 013114    MOV      DPB+CYL,-(SP) ;:SAVE DPB+CYL FOR TYPEOUT
1635 002766 104410                TYPDS    ;:GO TYPE--DECIMAL ASCII WITH SIGN
1636 002770 104400 014454    TYPE     ,TOOLRG    ; '>1200 U INCHES' MESSAGE
1637 002774 104400 001153    TYPE     ,%CRLF     ;CR-LF
1638 003000 032737 100000 177570    BIT      #SW15,SWR  ;HALT IF ERROR ?
1639 003006 001401                BEQ      9$          ;BR IF NOT
1640 003010 000000                HALT     ;HEAD JUST TYPED OUT OF TOLERANCE
1641 003012 032737 001000 177570    9$:     BIT      #SW9,SWR  ;SEE IF SWITCH 9 SET
1642 003020 001740                BEQ      7$          ;IF NOT SET, GET NEXT HEAD
1643 003022 000137 002376    JMP      CHECK       ;SET, DO THE HEAD AGAIN
1644 003026 005137 001170    10$:    COM      BITIND     ;SET THE SIGN BIT INDICATOR
1645 003032 042737 077777 013154    BIC      #77777,REG+RHOF ;LEAVE THE SIGN BIT

```



```

1646 003040 013737 013154 001172      MOV      REG+RHOF,SINCNG      ;STORE THE SIGN CHANGE BIT
1647 003046 000622                      BR          3$
1648
1649      ;*****
1650
1651      .SBTTL  ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT
1652
1653      ;*****
1654
1655 003050 104400 014633      DDUALN:  TYPE      ,ALIGN      ;TYPE 'ALIGN'
1656 003054 104400 014762      TYPE      ,ALN245      ;'POSITIONED AT 245' MESSAGE
1657 003060 012737 000365 013114      MOV      #245,DPB+CYL      ;LOAD CYLINDER ADDRESS
1658 003066 004537 003252      JSR      R5,DRIVER      ;DO A RECALIBRATE
1659 003072 000107      .WORD      RECAL      ;DRIVER CODE FOR RECALIBRATE
1660 003074 004537 003252      JSR      R5,DRIVER      ;SEEK TO CYLINDER 245
1661 003100 000105      .WORD      SEEK      ;DRIVER CODE FOR SEEK
1662 003102 104400 015030      1$:      TYPE      ,ENTHD      ;ASK FOR HEAD NUMBER
1663 003106 004737 003642      JSR      PC,GETDEC      ;GET THE ENTRY
1664 003112 000403      BR          2$      ;'CR' ENTERED
1665 003114 021627 000022      CMP      (SP),#18.      ;ENTRY CAN'T BE GREATER THAN 18
1666 003120 101403      BLOS     3$      ;BR IF LESS OR EQUAL
1667 003122 104400 001152      2$:      TYPE      ,QUES      ;TYPE A QUESTION MARK
1668 003126 000765      BR          1$      ;TRY AGAIN
1669 003130 011637 001162      3$:      MOV      (SP),SHEAD      ;HEAD NUMBER FOR TYPEOUT
1670 003134 000316      SWAB     (SP)      ;PUT HEAD NUMBER INTO UPPER BYTE
1671 003136 004037 011776      JSR      R0,WRT.RP      ;LOAD RHDA
1672 003142 000006      RHDA     ;REGISTER ADDRESS
1673 003144 001524      INIT     ;UNCORRECTABLE PARITY ERROR RETURN
1674 003146 000755      BR          1$      ;WAIT FOR NEXT HEAD ENTRY
1675
1676      ;*****
1677
1678      .SBTTL  ROUTINE TO PERFORM 5,000 RANDOM SEEKS
1679
1680      ;*****
1681
1682 003150 104400 014644      RANDOM:  TYPE      ,EXER      ;TYPE 'EXERCISE'
1683 003154 012737 011610 003250      MOV      #5000,SEKCNT      ;NUMBER OF SEEKS
1684 003162 004737 004304      1$:      JSR      PC,$RAND      ;CYCLE THE RANDOM NUMBER GENERATOR
1685 003166 013746 004432      MOV      $LONUM,-(SP)      ;PUT LOW DIVIDEND ON THE STACK
1686 003172 013746 004430      MOV      $HINUM,-(SP)      ;PUT HIGH DIVIDEND ON THE STACK
1687 003176 012746 000632      MOV      #410,-(SP)      ;PUT DIVISOR ON STACK
1688 003202 004737 003372      JSR      PC,LINKDV      ;GET REMAINDER (RANDOM CYLINDER ADDR)
1689 003206 021637 013114      CMP      (SP),DPB+CYL      ;SEE IF SAME RANDOM CYLINDER AS LAST
1690 003212 001003      BNE     2$      ;BR IF DIFFERENT CYLINDER
1691 003214 062706 000004      ADD      #4,SP      ;ADJUST THE STACK POINTER
1692 003220 000760      BR          1$      ;TRY AGAIN
1693 003222 012637 013114      2$:      MOV      (SP)+,DPB+CYL      ;PUT CYL ADDR IN THE DPB
1694 003226 005726      TST     (SP)+      ;CORRECT THE STACK POINTER
1695 003230 004537 003252      JSR      R5,DRIVER      ;DO THE SEEK
1696 003234 000105      .WORD      SEEK      ;SEEK CODE
1697 003236 005337 003250      DEC      SEKCNT      ;DECREMENT THE SEEK COUNT
1698 003242 001347      BNE     1$      ;BR IF NOT DONE
1699 003244 000137 001524      JMP      INIT      ;RETURN TO COMMAND ROUTINE
1700
1701 003250 000000      SEKCNT:  .WORD      0      ;SEEK COUNTER

```



1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757

003252 012537 013104  
003256 004037 006300  
003262 013102  
003264 000240  
003266 005737 013120  
003272 001775  
003274 100401  
003276 000205  
003300 032737 000200 013120  
003306 001021  
003310 032737 070000 013120  
003316 001013  
003320 032737 004000 013120  
003326 001005  
003330 104014  
003332 012706 001100  
003336 000137 001524  
003342 104015  
003344 000403  
003346 104011  
003350 000770  
003352 104010  
003354 032737 001000 177570  
003362 001745  
003364 162705 000006  
003370 000742

```
*****  
.SBTTL COMMON ENTRY TO THE RPO4 DRIVER  
*****  
DRIVER: MOV (R5)+,DPB+COMND ;COMMAND CODE  
1$: JSR R0,RPO4 ;DRIVER ENTRY  
DPB ;DATA PARAMETER BLOCK ADDRESS  
NOP ;REQUEST NOT ACCEPTED  
2$: TST DPB+STATUS ;SEE IF ORDER COMPLETE  
BEQ 2$ ;BR IF NOT COMPLETE  
BMI 5$ ;BRANCH IF ERROR  
3$: RTS R5 ;RETURN  
5$: BIT #BIT7,DPB+STATUS ;DID THE ORDER TERMINATE  
BNE 9$ ;BRANCH IF IT DID  
BIT #70000,DPB+STATUS ;SEE WHICH FATAL ERROR  
BNE 8$ ;BRANCH IF DRIVE UNSAFE  
BIT #BIT11,DPB+STATUS ;SEE IF UNCORRECTABLE PARITY ERROR  
BNE 7$ ;BRANCH IF UNCORRECTABLE  
ERROR 14 ;FATAL MASSBUS PARITY ERROR  
6$: MOV #STACK,SP ;RESET THE STACK POINTER  
JMP INIT ;RETURN TO THE START  
7$: ERROR 15 ;UNCORRECTABLE MASSBUS PARITY ERROR  
BR 10$ ;CHECK LOOP ON ERROR SWITCH  
8$: ERROR 11 ;REPORT DRIVE UNSAFE ERROR  
BR 6$ ;RETURN TO START  
9$: ERROR 10 ;DRIVE ERROR OR DATA ERROR  
10$: BIT #SW9,SWR ;LOOP ON ERROR ?  
BEQ 3$ ;BRANCH IF NOT  
SUB #6,R5 ;CORRECT RETURN FOR LOOP  
BR 3$ ;RETURN
```

```
*****  
.SBTTL SUBROUTINES  
*****  
;LINK ROUTINE TO THE DIVISION UTILITY SUBROUTINE  
; THIS ROUTINE ALLOWS THE 'SYSMAC' DIVIDE ROUTINE  
; CALLING SEQUENCE TO BE USED
```

```
LINKDV:  
MOV R5,-(SP) ;:PUSH R5 ON STACK  
MOV R4,-(SP) ;:PUSH R4 ON STACK  
MOV R3,-(SP) ;:PUSH R3 ON STACK  
MOV R2,-(SP) ;:PUSH R2 ON STACK  
MOV R1,-(SP) ;:PUSH R1 ON STACK  
MOV R0,-(SP) ;:PUSH R0 ON STACK  
MOV 16(SP),R5 ;:DIVISOR  
CLR R4 ;:OTHER DIVISOR WORD  
MOV 20(SP),R2 ;:UPPER DIVIDEND WORD
```



```

1758 003420 016603 000022      MOV      22(SP),R3      ;LOWER DIVIDEND WORD
1759 003424 005000              CLR      R0            ;CLEAR OTHER DIVIDEND REGISTERS
1760 003426 005001              CLR      R1
1761 003430 004737 003464      JSR      PC,M.DPID    ;GO TO THE DIVIDE ROUTINE
1762 003434 010166 000020      MOV      R1,20(SP)    ;REMAINDER ON THE STACK
1763 003440 010366 000022      MOV      R3,22(SP)    ;QUOTIENT ON THE STACK
1764 003444 012600              MOV      (SP)+,R0     ;POP STACK INTO R0
1765 003446 012601              MOV      (SP)+,R1     ;POP STACK INTO R1
1766 003450 012602              MOV      (SP)+,R2     ;POP STACK INTO R2
1767 003452 012603              MOV      (SP)+,R3     ;POP STACK INTO R3
1768 003454 012604              MOV      (SP)+,R4     ;POP STACK INTO R4
1769 003456 012605              MOV      (SP)+,R5     ;POP STACK INTO R5
1770 003460 012616              MOV      (SP)+,(SP)   ;MOVE RETURN UP THE STACK
1771 003462 000207      RTS      PC
1772
1773      :
1774      :
1775      :
1776      :
1777      :
1778      :
1779      :
1780 003464 012746 000040      M.DPID: MOV      #40,-(SP)  ;COUNTER FOR DIVISION CYCLES
1781 003470 010446              MOV      R4,-(SP)    ;HIGH ORDER
1782 003472 010546              MOV      R5,-(SP)    ;LOW ORDER DIVISOR TO THE STACK
1783 003474 005466 000002      NEG      2(SP)        ;FORM NEGATIVE
1784 003500 005416              NEG      @SP          ;VERSION OF THE DIVISOR
1785 003502 005666 000002      SBC      2(SP)
1786 003506 061601              ADD      @SP,R1
1787 003510 005500              ADC      R0            ;PERFORM THE INITIAL SUBTRACTION
1788 003512 066600 000002      ADD      2(SP),R0
1789 003516 103445              BCS      M.DP50      ;IF CARRY THEN OVERFLOW HAS OCCURRED
1790 003520 005046              CLR      -(SP)        ;THIS IS A LONGER LASTING CARRY BIT
1791 003522 006103              M.DP40: ROL      R3
1792 003524 006102              ROL      R2
1793 003526 006101              ROL      R1
1794 003530 006100              ROL      R0
1795 003532 005716              TST      @SP          ;TEST "CARRY" INDICATOR
1796 003534 001410              BEQ      M.DP41      ;IF NO "CARRY" THEN ADD ELSE SUBTRACT
1797 003536 005016              CLR      @SP          ;CLEAR UP FOR NEXT TIME
1798 003540 066601 000002      ADD      2(SP),R1
1799 003544 005500              ADC      R0            ;ADD -(DIVISOR)
1800 003546 005516              ADC      @SP          ;SET "CARRY"
1801 003550 066600 000004      ADD      4(SP),R0; <- I
1802 003554 000404              BR       M.DP42
1803 003556 060501              M.DP41: ADD      R5,R1
1804 003560 005500              ADC      R0            ;ADD +(DIVISOR)
1805 003562 005516              ADC      @SP          ;SET "CARRY"
1806 003564 060400              ADD      R4,R0 ; <- I
1807 003566 005516              M.DP42: ADC      @SP          ;SET "CARRY"
1808 003570 005716              TST      @SP          ;TEST THE UPDATE INDICATOR
1809 003572 001401              BEQ      .+4 ;-> ;IF ZERO FORGET IT
1810 003574 005203              INC      R3 ; <- I ;NO CARRY POSSIBLE HERE
1811 003576 005366 000006      DEC      6(SP) ; <- ;DECREMENT COUNTER
1812 003602 003347              BGT      M.DP40      ;BRANCH IF MORE TO DO
1813 003604 006003              ROR      R3

```

```

1814 003606 103404          BCS      M.DP44
1815 003610 060501          ADD      R5,R1
1816 003612 005500          ADC      RO
1817 003614 060400          ADD      R4,RO
1818 003616 000241          CLC
1819 003620 006103          M.DP44: ROL      R3
1820 003622 062706 000010          ADD      #10,SP          ;ADJUST STACK BY 4 WORDS
1821 003626 000242          CLV
1822 003630 000207          RTS      PC
1823 003632 062706 000006          M.DP50: ADD      #6,SP
1824 003636 000262          SEV
1825 003640 000207          RTS      PC
1826
1827          ;ROUTINE TO GET DECIMAL ENTRIES
1828          ;RETURN IF CARRIAGE RETURN ENTERED
1829          ;RETURN +2 IF OTHER CHARACTER ENTERED
1830          ;GO TO 'INIT' IF CONTROL C ENTERED
1831
1832 003642 011646          GETDEC: MOV      (SP),-(SP)          ;PROVIDE SPACE FOR FIRST CHAR
1833 003644 104420          1$:  RDLIN          ;READ AN ASCIZ LINE
1834 003646 012600          MOV      (SP)+,RO          ;ADDRESS OF 1ST CHAR.
1835 003650 010037 003756          MOV      RO,6$          ;SAVE IN CASE OF BAD INPUT
1836 003654 105710          TSTB     (RO)          ;FIRST CHARACTER A 'CR' ?
1837 003656 001443          BEQ      7$          ;BR IF IT IS
1838 003660 122710 000003          CMPB     #3,(RO)          ;CONTROL C ?
1839 003664 001442          BEQ      9$          ;BR IF IT IS
1840 003666 005046          CLR      -(SP)          ;CLEAR DATA WORD
1841 003670 112001          2$:  MOVB     (RO)+,R1          ;PICKUP THIS CHARACTER
1842 003672 001421          BEQ      4$          ;GET OUT IF ZERO
1843 003674 122701 000060          CMPB     #'0,R1          ;MAKE SURE THIS CHARACTER
1844 003700 003023          BGT      5$          ;IS A DIGIT BETWEEN 0 & 9
1845 003702 122701 000071          CMPB     #'9,R1
1846 003706 002420          BLT      5$
1847 003710 006316          ASL      (SP)          ;#2
1848 003712 011646          MOV      (SP),-(SP)          ;SAVE FOR LATER
1849 003714 006316          ASL      (SP)          ;#4
1850 003716 006316          ASL      (SP)          ;#8
1851 003720 062616          ADD      (SP)+,(SP)          ;#10
1852 003722 102412          BVS      5$          ;OVERFLOW ISN'T ALLOWED
1853 003724 162701 000060          SUB      #'0,R1          ;STRIP AWAY THE ASCII JUNK
1854 003730 060116          ADD      R1,(SP)          ;ADD IN THIS DIGIT
1855 003732 102406          BVS      5$          ;OVERFLOW ISN'T ALLOWED
1856 003734 000755          BR       2$          ;LOOP
1857 003736 012666 000002          4$:  MOV      (SP)+,2(SP)          ;SAVE THE RESULT
1858 003742 062716 000002          ADD      #2,(SP)          ;INCREMENT THE RETURN ADDRESS
1859 003746 000410          BR       8$          ;EXIT
1860 003750 005726          5$:  TST      (SP)+          ;CLEAN PARTIAL NUMBER FROM STACK
1861 003752 105010          CLRB     (RO)          ;SET A TERMINATOR
1862 003754 104400          TYPE          ;TYPE THE INPUT UP TO BAD CHAR.
1863 003756 000000          6$:  .WORD     0          ;POINTER GOES HERE
1864 003760 104400 001152          TYPE          ;"?" "CR" & "LF"
1865 003764 000727          BR       1$          ;TRY AGAIN
1866 003766 012616          7$:  MOV      (SP)+,(SP)          ;RESTORE THE PC
1867 003770 000207          8$:  RTS      PC          ;RETURN
1868 003772 000137 001524          9$:  JMP      INIT          ;CONTROL C ENTERED
1869

```



```

1870      ; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1871      ; CALL:
1872      ; RDCHR          ; INPUT A SINGLE CHARACTER FROM THE TTY
1873      ; RETURN HERE   ; CHARACTER IS ON THE STACK
1874      ;
1875
1876      003776 011646      SRDCHR: MOV      (SP), -(SP)      ; PUSH DOWN THE PC
1877      004000 016666 000004 000002  MOV      4(SP), 2(SP)      ; SAVE THE PS
1878      004006 105777 175124      1S:   TSTB     @STKS          ; WAIT FOR
1879      004012 100375          BPL      1S              ; A CHARACTER
1880      004014 117766 175120 000004  MOVB     @STKB, 4(SP)      ; READ THE TTY
1881      004022 042766 177600 000004  BIC      @C<177>, 4(SP)    ; GET RID OF JUNK IF ANY
1882      004030 000002          RTI                      ; GO BACK TO USER
1883
1884      ; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1885      ; CALL:
1886      ; RDLIN          ; INPUT A STRING FROM THE TTY
1887      ; RETURN HERE   ; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1888      ;                ; TERMINATOR WILL BE A BYTE OF ALL 0'S
1889
1890      004032 010346      SRDLIN: MOV      R3, -(SP)      ; SAVE R3
1891      004034 005046      CLR      -(SP)          ; CLEAR THE RUBOUT KEY
1892      004036 012703 004300      1S:   MOV      @STTYIN, R3      ; GET ADDRESS
1893      004042 022703 004304      2S:   CMP      @STTYIN+4, R3    ; BUFFER FULL?
1894      004046 101450          BLOS     4S              ; BR IF YES
1895      004050 104416          RDCHR    ; GO READ ONE CHARACTER FROM THE TTY
1896      004052 112613      MOVB     (SP)+, (R3)      ; GET CHARACTER
1897      004054 122713 000177      CMPB     @177, (R3)      ; IS IT A RUBOUT
1898      004060 001022      BNE      5S              ; BR IF NO
1899      004062 005716          TST      (SP)            ; IS THIS THE FIRST RUBOUT?
1900      004064 001007      BNE      6S              ; BR IF NO
1901      004066 112737 000134 004264  MOVB     #' \, 9S        ; TYPE A BACK SLASH
1902      004074 104400 004264      TYPE     9S
1903      004100 012716 177777          MOV      1-1, (SP)      ; SET THE RUBOUT KEY
1904      004104 005303      6S:   DEC      R3              ; BACKUP BY ONE
1905      004106 020327 004300      CMP      R3, @STTYIN     ; STACK EMPTY?
1906      004112 103426          BLO      4S              ; BR IF YES
1907      004114 111337 004264      MOVB     (R3), 9S        ; SETUP TO TYPEOUT THE DELETED CHAR.
1908      004120 104400 004264      TYPE     9S
1909      004124 000746          BR       2S              ; GO READ ANOTHER CHAR.
1910      004126 005716      5S:   TST      (SP)            ; RUBOUT KEY SET?
1911      004130 001406          BEQ      7S              ; BR IF NO
1912      004132 112737 000134 004264  MOVB     #' \, 9S        ; TYPE A BACK SLASH
1913      004140 104400 004264      TYPE     9S
1914      004144 005016          CLR      (SP)            ; CLEAR THE RUBOUT KEY
1915      004146 122713 000003      7S:   CMPB     @3, (R3) ; IS CHARACTER A CTRL C ?
1916      004152 001425          BEQ      8S              ; BR IF IT IS
1917      004154 122713 000025      CMPB     @25, (R3)      ; IS CHARACTER A CTRL U?
1918      004160 001006          BNE      3S              ; BR IF NO
1919      004162 104400 004266          TYPE     SCNTLU         ; TYPE A CONTROL "U"
1920      004166 000723          BR       1S              ; GO START OVER
1921      004170 104400 001152      4S:   TYPE     SQUES         ; TYPE A '?'
1922      004174 000720          BR       1S              ; CLEAR THE BUFFER AND LOOP
1923      004176 111337 004264      3S:   MOVB     (R3), 9S        ; ECHO THE CHARACTER
1924      004202 104400 004264          TYPE     9S
1925      004206 122723 000015      CMPB     @15, (R3)+     ; CHECK FOR RETURN

```

```

1926 004212 001313          BNE      25          ;LOOP IF NOT RETURN
1927 004214 105063 177777  CLRB     -1(R3)     ;CLEAR RETURN (THE 15)
1928 004220 104400 001154  TYPE     $LF        ;TYPE A LINE FEED
1929 004224 000405          BR       105
1930 004226 104400 004273 8$:      TYPE     .SCNTLC ;TYPE A CONTROL C
1931 004232 112737 000003 004300  MOVB     #3,$TTYIN  ;FORCE 1ST CHARACTER IN BUF TO CTRL C
1932 004240 005726          10$:     TST     (SP)+    ;CLEAN RUBOUT KEY FROM THE STACK
1933 004242 012603          MOV      (SP)+,R3   ;RESTORE R3
1934 004244 011646          MOV      (SP)-,(SP) ;ADJUST THE STACK AND PUT ADDRESS OF THE
1935 004246 016666 000004 000002  MOV      4(SP),2(SP) ;FIRST ASCII CHARACTER ON IT
1936 004254 012766 004300 000004  MOV      #$TTYIN,4(SP)
1937 004262 000002          RTI              ;RETURN
1938 004264          000          9$:      .BYTE     0          ;STORAGE FOR ASCII CHAR. TO TYPE
1939 004265          000          .BYTE     0          ;TERMINATOR
1940 004266 052536 005015 000  SCNTLU:  .ASCIZ  /↑U/<15><12> ;CONTROL "U"
1941 004273          136 006503 000012 SCNTLC:  .ASCIZ  /↑C/<15><12> ;CONTROL C
1942 004300 000004          $TTYIN:  .BLKB   4          ;RESERVE 4 BYTES FOR TTY INPUT
1943          .EVEN
1944          ;*****
1945          .SBTTL  MACRO ROUTINES
1946          ;*****
1947          .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
1948          ;*****
1949          ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
1950          ;*WITH A RANGE OF 0 TO 2(+33)-1.
1951          ;*CALL:
1952          ;*      JSR      PC,$RAND      ;:CALL THE ROUTINE
1953          ;*      RETURN                    ;:RETURN HERE THE RANDOM
1954          ;*                                  ;:NUMBER WILL BE IN
1955          ;*                                  ;:SHINUM,$LONUM
1956          ;*
1957          ;*
1958          ;*
1959          ;*
1960 004304          SRAND:
1961 004304 010046          MOV      R0,-(SP)   ;:PUSH R0 ON STACK
1962 004306 010146          MOV      R1,-(SP)   ;:PUSH R1 ON STACK
1963 004310 010246          MOV      R2,-(SP)   ;:PUSH R2 ON STACK
1964 004312 010346          MOV      R3,-(SP)   ;:PUSH R3 ON STACK
1965 004314 013700 004432  MOV      $LONUM,R0  ;:SET R0 WITH LOW
1966 004320 013701 004430  MOV      $SHINUM,R1 ;:SET R1 WITH HIGH
1967 004324 012703 177771  MOV      #-7,R3     ;:SET SHIFT COUNT
1968 004330 005002          CLR      R2        ;:ZERO R2
1969 004332 006300          1$:      ASL      R0        ;:SHIFT R0 LEFT AND
1970 004334 006101          ROL     R1        ;:ROTATE CARRY INTO R1 AND
1971 004336 006102          ROL     R2        ;:ROTATE CARRY INTO R2
1972 004340 005203          INC     R3        ;:CHECK FOR DONE
1973 004342 001373          BNE     1$        ;:CONTINUE SHIFT LOOP
1974 004344 063700 004432  ADD     $LONUM,R0  ;:ADD NUMBER TO MAKE X 129
1975 004350 005501          ADC     R1        ;:PROPOGATE CARRY
1976 004352 063701 004430  ADD     $SHINUM,R1 ;:ADD NUMBER TO MAKE X 129
1977 004356 005502          ADC     R2        ;:PROPOGATE CARRY
1978 004360 062700 001057  ADD     #1057,R0   ;:ADD LOW CONSTANT
1979 004364 005501          ADC     R1        ;:PROPOGATE CARRY
1980 004366 005502          ADC     R2        ;:PROPOGATE CARRY
1981 004370 062701 047401  ADD     #47401,R1  ;:ADD HIGH CONSTANT

```



```

1982 004374 005502
1983 004376 062702 000006
1984 004402 060200
1985 004404 005501
1986 004406 010037 004432
1987 004412 010137 004430
1988 004416 012603
1989 004420 012602
1990 004422 012601
1991 004424 012600
1992 004426 000207
1993 004430 176543
1994 004432 123456

```

```

ADC R2 ;; PROPOGATE CARRY
ADD #6,R2 ;; ADD HIGHEST CONSTART
ADD R2,R0 ;; REPRIME R0 WITH HIGHEST DIGIT
ADC R1 ;; PROPOGATE CARRY
MOV R0,$LONUM ;; SAVE R0
MOV R1,$SHINUM ;; SAVE R1
MOV (SP)+,R3 ;; POP STACK INTO R3
MOV (SP)+,R2 ;; POP STACK INTO R2
MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,R0 ;; POP STACK INTO R0
RTS PC ;; RETURN

```

```

$SHINUM: .WORD 176543
$LONUM: .WORD 123456

```

\*\*\*\*\*

.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 SERRTYP ON ERROR
; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; *SW15=1 HALT ON ERROR
; *SW13=1 INHIBIT ERROR TYPEOUTS
; *CALL
; * ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

```

2007
2008 004434
2009 004434 010137 001156
2010 004440 010537 001160
2011 004444 105237 001103
2012 004450 001775
2013 004452 013737 001102 177570
2014 004460 005237 001112
2015 004464 011637 001116
2016 004470 162737 000002 001116
2017 004476 117737 174414 001114
2018 004504 032737 020000 177570
2019 004512 001004
2020 004514 004737 004536
2021 004520 104400 001153
2022 004524 005737 177570
2023 004530 100001
2024 004532 000000
2025 004534
2026 004534 000002

```

```

SERROR:
MOV R1,DRIVE
MOV R5,ATTN
7$: INCB SERFLG ;; SET THE ERROR FLAG
BEQ 7$ ;; DON'T LET THE FLAG GO TO ZERO
MOV $TSTNM,$@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
INC SERRTL ;; INC THE ERROR COUNT
MOV (SP),SERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
SUB #2,SERRPC
MOVB @SERRPC,$ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
BIT #BIT13,$@SWR ;; SKIP TYPEOUT IF SET
BNE 2$ ;; SKIP TYPEOUTS
JSR PC,$@SERRTYP ;; GO TO USER ERROR ROUTINE
TYPE $SCLF
2$: TST $@SWR ;; HALT ON ERROR
BPL 3$ ;; SKIP IF CONTINUE
HALT ;; HALT ON ERROR!
3$: RTI ;; RETURN

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

; *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
; *ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" (SERRTB),
; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

```

2035 004536
2036 004536 104400 001153
2037 004542 010046

```

```

SERRTYP:
TYPE $SCLF ;; "CARRIAGE RETURN" & "LINE FEED"
MOV R0,-(SP) ;; SAVE R0

```

```

2038 004544 005000          CLR      RO          ;; PICKUP THE ITEM INDEX
2039 004546 153700 001114  BISB     @#SITEMB,RO
2040 004552 001004          BNE     1$          ;; IF ITEM NUMBER IS ZERO, JUST
2041                                ;; TYPE THE PC OF THE ERROR
2042 004554 013746 001116  MOV      $ERRPC,-(SP) ;; SAVE $ERRPC FOR TYPEOUT
2043                                ;; ERROR ADDRESS
2044 004560 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2045 004562 000445          BR      10$        ;; GET OUT
2046 004564 005300          1$: DEC     RO          ;; ADJUST THE INDEX SO THAT IT WILL
2047 004566 006300          ASL     RO          ;; WORK FOR THE ERROR TABLE
2048 004570 006300          ASL     RO
2049 004572 006300          ASL     RO
2050 004574 062700 001206  ADD     #$ERRTB,RO  ;; FORM TABLE POINTER
2051 004600 012037 004610  MOV     (RO)+,2$    ;; PICKUP "ERROR MESSAGE" POINTER
2052 004604 001404          BEQ     3$          ;; SKIP TYPEOUT IF NO POINTER
2053 004606 104400          TYPE          ;; TYPE THE "ERROR MESSAGE"
2054 004610 000000          2$: .WORD 0          ;; "ERROR MESSAGE" POINTER GOES HERE
2055 004612 104400 001153  TYPE    $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
2056 004616 012037 004626  3$: MOV     (RO)+,4$ ;; PICKUP "DATA HEADER" POINTER
2057 004622 001404          BEQ     5$          ;; SKIP TYPEOUT IF 0
2058 004624 104400          TYPE          ;; TYPE THE "DATA HEADER"
2059 004626 000000          4$: .WORD 0          ;; "DATA HEADER" POINTER GOES HERE
2060 004630 104400 001153  TYPE    $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
2061 004634 010146          5$: MOV     R1,-(SP) ;; SAVE R1
2062 004636 012001          MOV     (RO)+,R1   ;; PICKUP "DATA TABLE" POINTER
2063 004640 001415          BEQ     9$          ;; BR IF NO DATA TO BE TYPED
2064 004642 012000          MOV     (RO)+,RO   ;; PICKUP "DATA FORMAT" POINTER
2065 004644 105720          6$: TSTB   (RO)+    ;; "OCTAL" OR "DECIMAL"
2066 004646 001003          BNE     7$          ;; BR IF DECIMAL
2067 004650 013146          MOV     @ (R1)+,-(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
2068 004652 104402          TYPOC          ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2069 004654 000402          BR      8$
2070 004656          7$:
2071 004656 013146          MOV     @ (R1)+,-(SP) ;; SAVE @ (R1)+ FOR TYPEOUT
2072 004660 104410          TYPDS          ;; GO TYPE--DECIMAL ASCII WITH SIGN
2073 004662 005711          8$: TST     (R1)     ;; IS THERE ANOTHER NUMBER?
2074 004664 001403          BEQ     9$          ;; BR IF NO
2075 004666 104400 004706  TYPE    ,11$      ;; TYPE TWO(2) SPACES
2076 004672 000764          BR      6$          ;; LOOP
2077
2078 004674 012601          9$: MOV     (SP)+,R1  ;; RESTORE R1
2079 004676 012600          10$: MOV    (SP)+,RO ;; RESTORE RO
2080 004700 104400 001153  TYPE    $CRLF      ;; "CARRIAGE RETURN" & "LINE FEED"
2081 004704 000207          RTS     PC         ;; RETURN
2082 004706 020040 000          11$: .ASCIZ / /      ;; TWO(2) SPACES
2083                                .EVEN
2084                                ;*****
2085

```

.SBTTL TYPE ROUTINE

```

2088 ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2089 ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2090 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2091 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2092 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2093 ;*

```



```

2094      ;*CALL:
2095      ;*1) USING A TRAP INSTRUCTION
2096      ;*      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2097      ;*OR
2098      ;*      TYPE
2099      ;*      MESADR
2100      ;*
2101      ;*2) USING A JSR INSTRUCTION
2102      ;*      MOV      PS,-(SP)      ;;PUSH PROCESSOR STATUS WORD ON THE STACK
2103      ;*      JSR      PC,$TYPE      ;;CALL TYPE ROUTINE
2104      ;*      MESADDR      ;;FIRST ADDRESS OF MESSAGE
2105
2106      004712 105737 001151      $TYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
2107      004716 100002              BPL          1$          ;; BR IF YES
2108      004720 000000              HALT          ;; HALT HERE IF NO TERMINAL
2109      004722 000407              BR           3$          ;; LEAVE
2110      004724 010046      1$:     MOV      RO,-(SP)      ;; SAVE RO
2111      004726 017600      2$:     MOV      @2(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
2112      004732 112046      3$:     MOVB     (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2113      004734 001005              BNE          4$          ;; BR IF IT ISN'T THE TERMINATOR
2114      004736 005726              TST         (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
2115      004740 012600      4$:     MOV      (SP)+,RO      ;; RESTORE RO
2116      004742 062716 000002      5$:     ADD      #2,(SP)      ;; ADJUST RETURN PC
2117      004746 000002              RTI          ;; RETURN
2118      004750 004737 005002      6$:     JSR      PC,$TYPEPC      ;; GO TYPE THIS CHARACTER
2119      004754 123726 001150      7$:     CMPB     $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
2120      004760 001364              BNE          2$          ;; IF NO GO GET NEXT CHAR.
2121      004762 013746 001146      8$:     MOV      $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
2122      ;*      AND THE NULL CHAR.
2123      004766 105366 000001      9$:     DECB     1(SP)      ;; DOES A NULL NEED TO BE TYPED?
2124      004772 002770              BLT          5$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
2125      004774 004737 005002      10$:    JSR      PC,$TYPEPC      ;; GO TYPE A NULL
2126      005000 000772              BR           6$          ;; LOOP
2127      005002 105777 174134      $TYPEPC: TSTB     @2$TPS      ;; WAIT UNTIL PRINTER IS READY
2128      005006 100375              BPL          $TYPEPC
2129      005010 116677 000002 174126  MOVB     2(SP),@2$TPB      ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2130      005016 000207              RTS          PC
2131      ;*****
2132
2133      .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2134
2135      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2136      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
2137      ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2138      ;*CALL:
2139      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2140      ;*      TYPOS      ;;CALL FOR TYPEOUT
2141      ;*      .BYTE   N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2142      ;*      .BYTE   M      ;;M=1 OR 0
2143      ;*
2144      ;*      ;;1=TYPE LEADING ZEROS
2145      ;*      ;;0=SUPPRESS LEADING ZEROS
2146      ;*$STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2147      ;*$TYPOS OR $TYPOC
2148      ;*CALL:
2149      ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
  
```

```

2150          ;*      TYPON          ;;CALL FOR TYPEOUT
2151          ;*
2152          ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2153          ;*CALL:
2154          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2155          ;*      TYPOC          ;;CALL FOR TYPEOUT
2156
2157 005020 017646 000000          $TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
2158 005024 116637 000001 005243 MOVVB   1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
2159 005032 112637 005245          MOVVB   (SP)+,$OMODE+1      ;;NUMBER OF DIGITS TO TYPE
2160 005036 062716 000002          ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
2161 005042 000406
2162 005044 112737 000001 005243 $TYPOC: MOVVB   #1,$OFILL      ;;SET THE ZERO FILL SWITCH
2163 005052 112737 000006 005245 MOVVB   #6,$OMODE+1      ;;SET FOR SIX(6) DIGITS
2164 005060 112737 000005 005242 $TYPON: MOVVB   #5,$OCNT      ;;SET THE ITERATION COUNT
2165 005066 010346          MOV      R3,-(SP)      ;;SAVE R3
2166 005070 010446          MOV      R4,-(SP)      ;;SAVE R4
2167 005072 010546          MOV      R5,-(SP)      ;;SAVE R5
2168 005074 113704 005245          MOVVB   $OMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2169 005100 005404          NEG      R4
2170 005102 062704 000006          ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
2171 005106 110437 005244          MOVVB   R4,$OMODE      ;;SAVE IT FOR USE
2172 005112 113704 005243          MOVVB   $OFILL,R4      ;;GET THE ZERO FILL SWITCH
2173 005116 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
2174 005122 005003          CLR      R3      ;;CLEAR THE OUTPUT WORD
2175 005124 006105          1$: ROL      R5      ;;ROTATE MSB INTO "C"
2176 005126 000404          BR      3$      ;;GO DO MSB
2177 005130 006105          2$: ROL      R5      ;;FORM THIS DIGIT
2178 005132 006105
2179 005134 006105
2180 005136 010503          MOV      R5,R3
2181 005140 006103          3$: ROL      R3      ;;GET LSB OF THIS DIGIT
2182 005142 105337 005244          DECB    $OMODE      ;;TYPE THIS DIGIT?
2183 005146 100016          BPL      7$      ;;BR IF NO
2184 005150 042703 177770          BIC     #177770,R3      ;;GET RID OF JUNK
2185 005154 001002          BNE     4$      ;;TEST FOR 0
2186 005156 005704          TST     R4      ;;SUPPRESS THIS 0?
2187 005160 001403          BEQ     5$      ;;BR IF YES
2188 005162 005204          4$: INC     R4      ;;DON'T SUPPRESS ANYMORE 0'S
2189 005164 052703 000060          BIS     #'0,R3      ;;MAKE THIS DIGIT ASCII
2190 005170 052703 000040          5$: BIS     #' ,R3      ;;MAKE ASCII IF NOT ALREADY
2191 005174 110337 005240          MOVVB   R3,8$      ;;SAVE FOR TYPING
2192 005200 104400 005240          TYPE    8$      ;;GO TYPE THIS DIGIT
2193 005204 105337 005242          7$: DECB    $OCNT      ;;COUNT BY 1
2194 005210 003347          BGT     2$      ;;BR IF MORE TO DO
2195 005212 002402          BLT     6$      ;;BR IF DONE
2196 005214 005204          INC     R4      ;;INSURE LAST DIGIT ISN'T A BLANK
2197 005216 000744          BR      2$      ;;GO DO THE LAST DIGIT
2198 005220 012605          6$: MOV     (SP)+,R5      ;;RESTORE R5
2199 005222 012604          MOV     (SP)+,R4      ;;RESTORE R4
2200 005224 012603          MOV     (SP)+,R3      ;;RESTORE R3
2201 005226 016666 000002 000004 MOV     2(SP),4(SP)      ;;SET THE STACK FOR RETURNING
2202 005234 012616          MOV     (SP)+,(SP)
2203 005236 000002          RTI
2204 005240          8$: .BYTE 0      ;;RETURN
2205 005241          .BYTE 0      ;;STORAGE FOR ASCII DIGIT
                ;;TERMINATOR FOR TYPE ROUTINE

```



```

2206 005242 000 $OCNT: .BYTE 0 ;;OCTAL DIGIT COUNTER
2207 005243 000 $OFILL: .BYTE 0 ;;ZERO FILL SWITCH
2208 005244 000000 $OMODE: .WORD 0 ;;NUMBER OF DIGITS TO TYPE
2209 ;*****
2210
2211 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2212
2213 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2214 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2215 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2216 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2217 ;*REPLACED WITH SPACES.
2218 ;*CALL:
2219 ;* MOV NUM,-(SP) ;;PUT THE BINARY NUMBER ON THE STACK
2220 ;* TYPDS ;;GO TO THE ROUTINE
2221
2222 $TYPDS:
2223 005246 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
2224 005250 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
2225 005252 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
2226 005254 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
2227 005256 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
2228 005260 012746 020200 MOV #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
2229 005264 0166C5 000020 MOV 20(SP),R5 ;;GET THE INPUT NUMBER
2230 005270 100004 BPL 1$ ;;BR IF INPUT IS POS.
2231 005272 005405 NEG R5 ;;MAKE THE BINARY NUMBER POS.
2232 005274 112766 000055 000001 MOVB #'-,1(SP) ;;MAKE THE ASCII NUMBER NEG.
2233 005302 005000 1$: CLR R0 ;;ZERO THE CONSTANTS INDEX
2234 005304 012703 005462 MOV #5DBLK,R3 ;;SETUP THE OUTPUT POINTER
2235 005310 112723 000040 MOVB #' ,(R3)+ ;;SET THE FIRST CHARACTER TO A BLANK
2236 005314 005002 2$: CLR R2 ;;CLEAR THE BCD NUMBER
2237 005316 016001 005452 MOV $DTBL(R0),R1 ;;GET THE CONSTANT
2238 005322 160105 3$: SUB R1,R5 ;;FORM THIS BCD DIGIT
2239 005324 002402 BLT 4$ ;;BR IF DONE
2240 005326 005202 INC R2 ;;INCREASE THE BCD DIGIT BY 1
2241 005330 000774 BR 3$
2242 005332 060105 4$: ADD R1,R5 ;;ADD BACK THE CONSTANT
2243 005334 005702 TST R2 ;;CHECK IF BCD DIGIT=0
2244 005336 001002 BNE 5$ ;;FALL THROUGH IF 0
2245 005340 105716 TSTB (SP) ;;STILL DOING LEADING 0'S?
2246 005342 100407 BMI 7$ ;;BR IF YES
2247 005344 106316 5$: ASLB (SP) ;;MSD?
2248 005346 103003 BCC 6$ ;;BR IF NO
2249 005350 116663 000001 177777 MOVB 1(SP),-1(R3) ;;YES--SET THE SIGN
2250 005356 052702 000060 6$: BIS #'0,R2 ;;MAKE THE BCD DIGIT ASCII
2251 005362 052702 000040 7$: BIS #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
2252 005366 110223 MOVB R2,(R3)+ ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
2253 005370 005720 TST (R0)+ ;;JUST INCREMENTING
2254 005372 020027 000010 CMP R0,#10 ;;CHECK THE TABLE INDEX
2255 005376 002746 BLT 2$ ;;GO DO THE NEXT DIGIT
2256 005400 003002 BGT 8$ ;;GO TO EXIT
2257 005402 010502 MOV R5,R2 ;;GET THE LSD
2258 005404 000764 BR 6$ ;;GO CHANGE TO ASCII
2259 005406 105726 8$: TSTB (SP)+ ;;WAS THE LSD THE FIRST NON-ZERO?
2260 005410 100003 BPL 9$ ;;BR IF NO
2261 005412 116663 177777 177776 MOVB -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
  
```





```
2318 ;*AUTHOR: JIM LACEY
2319
2320
2321 ;*STORAGE FOR RHDS1, RHER1, RHER2, AND RHER3 ON AN ERROR "2" OR "5"
2322 ;*RPERRS = RHDS1
2323 ;*RPERRS+2 = RHER1
2324 ;*RPERRS+4 = RHER2
2325 ;*RPERRS+6 = RHER3
2326
2327 005534 000000 RPERRS: .WORD 0
2328 005536 000000 .WORD 0
2329 005540 000000 .WORD 0
2330 005542 000000 .WORD 0
2331
2332 ;*TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
2333 ;*DRVACT=0 IMPLIES DRIVE IS IDLE
2334 ;*DRVACT>0 IMPLIES DRIVE IS ACTIVE WITH A COMMAND
2335 ;*DRVACT<0 IMPLIES DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
2336
2337 005544 000 DRVACT: .BYTE 0 ;DRIVE 0
2338 005545 000 .BYTE 0 ;DRIVE 1
2339 005546 000 .BYTE 0 ;DRIVE 2
2340 005547 000 .BYTE 0 ;DRIVE 3
2341 005550 000 .BYTE 0 ;DRIVE 4
2342 005551 000 .BYTE 0 ;DRIVE 5
2343 005552 000 .BYTE 0 ;DRIVE 6
2344 005553 000 .BYTE 0 ;DRIVE 7
2345
2346 ;*TABLE OF DRIVE STATUS INDICATORS (DRVSTA=8 BYTES)
2347 ;*DRVSTA=0 IMPLIES DRIVE IS OFFLINE
2348 ;*DRVSTA>0 IMPLIES DRIVE IS ONLINE
2349 ;*DRVSTA<0 IMPLIES DRIVE IS UNSAFE OR NONEXISTENT
2350
2351 005554 000 DRVSTA: .BYTE 0 ;DRIVE 0
2352 005555 000 .BYTE 0 ;DRIVE 1
2353 005556 000 .BYTE 0 ;DRIVE 2
2354 005557 000 .BYTE 0 ;DRIVE 3
2355 005560 000 .BYTE 0 ;DRIVE 4
2356 005561 000 .BYTE 0 ;DRIVE 5
2357 005562 000 .BYTE 0 ;DRIVE 6
2358 005563 000 .BYTE 0 ;DRIVE 7
2359
2360 ;*TABLE OF DRIVE TYPES (DRV TYP=8 WORDS)
2361 ;*DRV TYP WILL CONTAIN THE DRIVE TYPE OF ALL ONLINE, OFFLINE, AND
2362 ;*UNSAFE DRIVES. IF A DRIVE IS NONEXISTENT DRV TYP WILL BE ZERO.
2363
2364 005564 000000 DRV TYP: .WORD 0 ;DRIVE 0
2365 005566 000000 .WORD 0 ;DRIVE 1
2366 005570 000000 .WORD 0 ;DRIVE 2
2367 005572 000000 .WORD 0 ;DRIVE 3
2368 005574 000000 .WORD 0 ;DRIVE 4
2369 005576 000000 .WORD 0 ;DRIVE 5
2370 005600 000000 .WORD 0 ;DRIVE 6
2371 005602 000000 .WORD 0 ;DRIVE 7
2372
2373 ;*TRANSFER WAIT FLAG (TRNSWT=1 WORD)
```





```

2430
2431 005632 000000 SAVEFG: .WORD 0
2432
2433 ;*SEEK FLAG (SEEKFG=1 WORD)
2434 ;*SEEKFG=0 IMPLIES WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
2435 ;*FOR A DATA TRANSFER START A SEARCH COMMAND
2436 ;*SEEKFG<0 IMPLIES DATA TRANSFER WILL DO IMPLIED SEEKS,
2437 ;*DISREGARD THE WINDOW
2438
2439 005634 000000 SEEKFG: .WORD 0
2440
2441 ;*TIMEOUT TABLE (TIMER=8 WORDS)
2442 ;*THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
2443
2444 005636 177777 TIMER: .WORD -1 ;DRIVE 0
2445 005640 177777 .WORD -1 ;DRIVE 1
2446 005642 177777 .WORD -1 ;DRIVE 2
2447 005644 177777 .WORD -1 ;DRIVE 3
2448 005646 177777 .WORD -1 ;DRIVE 4
2449 005650 177777 .WORD -1 ;DRIVE 5
2450 005652 177777 .WORD -1 ;DRIVE 6
2451 005654 177777 .WORD -1 ;DRIVE 7
2452
2453 ;*DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
2454 ;*DTUW<0 IMPLIES NO DATA TRANSFER UNDERWAY
2455 ;*DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
2456
2457 005656 177777 DTUW: .WORD -1
2458
2459 ;*ATTENTION BITS TABLE (ATABIT=8 BYTES)
2460 ;*THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
2461 ;*ATTENTION BIT
2462
2463 005660 001 ATABIT: .BYTE 1 ;DRIVE 0
2464 005661 002 .BYTE 2 ;DRIVE 1
2465 005662 004 .BYTE 4 ;DRIVE 2
2466 005663 010 .BYTE 10 ;DRIVE 3
2467 005664 020 .BYTE 20 ;DRIVE 4
2468 005665 040 .BYTE 40 ;DRIVE 5
2469 005666 100 .BYTE 100 ;DRIVE 6
2470 005667 200 .BYTE 200 ;DRIVE 7
2471
2472 ;*RP04 TO RH11 "MASS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
2473 ;*CALLING IT FATAL (MCPEMX=1 WORD)
2474
2475 005670 000003 MCPEMX: .WORD 3
2476
2477 ;*STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RP04),
2478 ;*RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).
2479 005672 176700 RPADR: .WORD 176700
2480 005674 000254 000240 RPVEC: .WORD 254,5*32.
2481
2482 ;*MAXIMUM NUMBER OF LOOK AHEADS ALLOWED IS 4 (MXLACT=1 WORD)
2483 005700 000004 MXLACT: .WORD 4
2484 ;*MAXIMUM DELTA DELAY IS 8 SECTORS (MXDLTA=1 WORD)
2485 005702 001000 MXDLTA: .WORD 8.*64.
  
```



```

2486 ;*MINIMUM DELTA DELAY IS 2 SECTORS (MNDLTA=1 WORD)
2487 005704 000200 MNDLTA: .WORD 2*64.
2488 ;*MAXIMUM SEARCH FOR I/O WINDOW IS 5 SECTORS (MXWNDW=1 WORD)
2489 005706 000005 MXWNDW: .WORD 5
2490
2491 ;*DEFINITIONS OF THE RH11/RP04 ADDRESS INDEXES
2492
2493 000000 RHCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)
2494 000002 RHWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)
2495 000004 RHBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)
2496 000006 RHDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)
2497 000010 RHCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)
2498 000012 RHDS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)
2499 000014 RHER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)
2500 000016 RHAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)
2501 000020 RHLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)
2502 000022 RHDB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)
2503 000024 RHMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)
2504 000026 RHDT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)
2505 000030 RMSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)
2506 000032 RHOF=32 ;OFFSET REGISTER (DRIVE REG. 11)
2507 000034 RHCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)
2508 000036 RHCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)
2509 000040 RHER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)
2510 000042 RHER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)
2511 000044 RHEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)
2512 000046 RHEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)
2513
2514 ;*RH11/RP04 DRIVER INIT. CODE
2515 ;*THIS ROUTINE WILL DETERMINE WHICH RP04 DRIVES ARE
2516 ;*AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR
2517 ;*TO THE PROPER STATE FOR EACH DRIVE.
2518 ;*NOTE: THIS ROUTINE CALLS DRVINT
2519
2520 ;*CALL
2521
2522 ;* JSR PC,RPINIT
2523 ;* RETURN
2524
2525 005710 013746 177776 RPINIT: MOV 2#PS, -(SP) ;SAVE PROCESSOR STATUS
2526 005714 013737 005676 177776 MOV RPVEC+2, 2#PS ;SET PROCESSOR STATUS TO RH11/RP04 LEVEL
2527 005722 004037 013040 JSR R0, SAVR15 ;GO SAVE R1-R5
2528 005726 004737 012576 JSR PC, CLRQUE ;CLEAR ALL REQUEST QUEUES
2529 005732 012701 005534 MOV #RPERRS, R1 ;FIRST ADDRESS TO BE CLEARED
2530 005736 012702 005634 MOV #SEEKFG, R2 ;LAST ADDRESS TO BE CLEARED
2531 005742 005021 1$: CLR (R1)+ ;CLEAR
2532 005744 020102 CMP R1, R2 ;ARE WE DONE?
2533 005746 101775 BLOS 1$ ;BRANCH IF NO
2534 005750 012702 005656 MOV #DTUW, R2 ;LAST ADDRESS
2535 005754 012721 177777 2$: MOV #-1, (R1)+ ;INITIALIZE
2536 005760 020102 CMP R1, R2 ;DONE?
2537 005762 101774 BLOS 2$ ;LOOP IF NO
2538 005764 005001 CLR R1 ;DRIVE NUMBER WILL BE IN R1
2539 005766 013704 005672 MOV RPADR, R4 ;FIRST ADDRESS OF RH11/RP04
2540 005772 012764 000040 000010 MOV #BIT05, RHCS2(R4) ;MASSBUS INIT.
2541 006000 004037 006052 3$: JSR R0, DRVINT ;GO INIT. A DRIVE

```



```

2542 006004 000417          BR      7$
2543 006006 005201          4$:    INC      R1          ;MOVE TO THE NEXT DRIVE
2544 006010 042701 177770    BIC      #1C7,R1        ;DON'T LET THE DRIVE NUMBER GET TO BIG
2545 006014 001371          BNE      3$            ;BRANCH IF MORE DRIVES TO INIT.
2546 006016 013703 005674    MOV      RPVEC,R3      ;SETUP THE RH11/RP04 VECTOR
2547 006022 012723 010062    MOV      #ISR,(R3)+
2548 006026 013713 005676    MOV      RPVEC+2,(R3)
2549 006032 004037 013060    JSR      RO,GETR15     ;RESTORE R1-R5
2550 006036 012637 177776    MOV      (SP)+,2#PS    ;RESTORE THE PROCESSOR STATUS
2551 006042 000207          RTS      PC            ;BYE-BYE
2552 006044 105061 005554    7$:    CLR      DRVSTA(R1) ;SET THE DRIVE STATUS TO OFFLINE
2553 006050 000756          BR      4$            ;GO DO THE NEXT DRIVE
2554
2555          ;*DRIVE INIT. ROUTINE
2556          ;*THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
2557          ;*AN RPO4. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
2558          ;*IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
2559          ;*INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
2560          ;*DRVSTA IS SET TO THE PROPER CONDITION.
2561          ;*CALL
2562          ;*
2563          ;*      MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
2564          ;*      MOV      RPADR,R4        ;UNIBUS ADDRESS OF RH11/RP04 (RHCS1)
2565          ;*      JSR      RO,DRVINT      ;CALLED BY A JSR
2566          ;*      RETURN1      ;ERROR OCCURRED (PARITY)
2567          ;*      RETURN2      ;NORMAL RETURN
2568 006052 004037 013040    DRVINT: JSR      RO,SAVR15   ;SAVE R1-R5
2569 006056 010164 000010    MOV      R1,RHCS2(R4) ;SELECT A DRIVE
2570 006062 010103          MOV      R1,R3        ;COPY THE DRIVE NUMBER INTO
2571 006064 006303          ASL      R3           ;R3 AND POSITION IT FOR TABLE INDEXING
2572 006066 112761 177777 005554    MOV      #-1,DRVSTA(R1) ;START DRIVE STATUS AS NONEXISTENT
2573 006074 005063 005564    CLR      DRVSTYP(R3)  ;CLEAR THE DRIVE TYPE INDICATOR
2574 006100 112777 000111 177564    MOV      #111,2RPADR  ;DO A "DRIVE CLEAR" COMMAND
2575 006106 032764 010000 000010    BIT      #BIT12,RHCS2(R4) ;NONEXISTENT DRIVE?
2576 006114 001403          BEQ      1$          ;NO---BRANCH
2577 006116 004737 012232    JSR      PC,SET.IE    ;GO SET "IE" WITHOUT A "TRE"
2578 006122 000462          BR      6$          ;LEAVE THIS ROUTINE
2579 006124 004037 011640    1$:    JSR      RO,RO.RP    ;READ THE DRIVE TYPE REG.
2580 006130 000026          RHDT
2581 006132 006272          7$
2582 006134 012605          MOV      (SP)+,R5     ;ERROR RETURN ADDRESS
2583 006136 010563 005564    MOV      R5,DRVSTYP(R3) ;PUT DRIVE TYPE IN R5
2584 006142 022705 020020    CMP      #20020,R5    ;SAVE THE DRIVE TYPE
2585 006146 001403          BEQ      2$          ;IS IT A SINGLE PORT RPO4?
2586 006150 022705 024020    CMP      #24020,R5    ;BRANCH IF YES
2587 006154 001043          BNE      5$          ;IS IT A DUAL PORT RPO4?
2588 006156 012746 000121    2$:    MOV      #121,-(SP)  ;BRANCH IF NO
2589 006162 004037 011776    JSR      RO,WRT.RP    ;DO A "READ-IN PRESET"
2590 006166 000000          RHCS1
2591 006170 006272          7$
2592 006172 012746 010000    MOV      #BIT12,-(SP) ;SET FMT22=1
2593 006176 004037 011776    JSR      RO,WRT.RP
2594 006202 000032          RHOF
2595 006204 006272          7$
2596 006206 004037 011640    JSR      RO,RO.RP    ;READ RHDS1
2597 006212 000012          RHDS1
  
```

```

2598 006214 006272          7$
2599 006216 012605          MOV      (SP)+,R5          ;AND SAVE IT IN R5
2600 006220 100011          BPL      4$              ;BRANCH IF ATA=0
2601 006222 116164 005660 000016  MOVB     ATABIT(R1),RHAS(R4) ;CLEAR ATTENTION BIT
2602 006230 004037 011640          JSR      RO,RO.RP        ;FIND OUT WHY ATA=1
2603 006234 000014          RHER1
2604 006236 006272          7$
2605 006240 006126          ROL      (SP)+          ;IS IT UNSAFE?
2606 006242 100412          BMI      6$              ;BRANCH IF YES
2607 006244 005105          4$:     COM      R5          ;CHECK MOL, DPR, DRY, AND VV
2608 006246 042705 167077          BIC      #1C<BIT12!BIT08!BIT07!BIT06>,R5
2609 006252 001004          BNE      5$              ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
2610 006254 112761 000001 005554  MOVB     #1,DRVSTA(R1)    ;SET DRIVE STATUS TO ONLINE
2611 006262 000402          BR       6$
2612 006264 105061 005554          5$:     CLRB     DRVSTA(R1)  ;DRIVE STATUS = OFFLINE
2613 006270 005720          6$:     TST      (R0)+      ;STEP OVER THE ERROR RETURN
2614 006272 004037 013060          7$:     JSR      RO,GETR15  ;RESTORE R1-R5
2615 006276 000200          RTS      RO              ;RETURN
2616
2617          ;*REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
2618          ;
2619          ;*CALL
2620          ;
2621          ;*   JSR      RO,#RP04          ;CALL THE RP04 DRIVER
2622          ;*   PNTADR          ;ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK
2623          ;*   RETURN1         ;RETURN HERE IF QUEUE IS FULL
2624          ;*   RETURN2         ;RETURN HERE IF REQUEST IS IN QUEUE OR THERE
2625          ;*                   ;IS AN ERROR CONDITION
2626
2627 006300 013746 177776          RP04:   MOV      #PS,-(SP)        ;SAVE THE CALLING STATUS
2628 006304 013737 005676 177776  MOV      RPVEC+2,#PS      ;DON'T ALLOW ANY RP04 INTERRUPTS
2629 006312 112737 000001 005610  MOVB     #1,ACTDRV        ;SET "ACTIVE DRIVER" FLAG
2630 006320 004037 013040          JSR      RO,SAVR15        ;SAVE R1-R5
2631 006324 012002          MOV      (R0)+,R2        ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
2632 006326 005062 000016          CLR      16(R2)          ;CLEAR THE STATUS/ERROR INDICATOR
2633 006332 111201          MOVB     (R2),R1         ;PICKUP THE DRIVE NUMBER
2634 006334 105761 005554          TSTB     DRVSTA(R1)      ;CHECK DRIVES STATUS
2635 006340 003017          BGT      1$              ;BRANCH IF ONLINE
2636 006342 105761 005612          TSTB     ULDFLG(R1)      ;UNLOAD COMMAND IN QUEUE?
2637 006346 001034          BNE      3$              ;BRANCH IF YES
2638 006350 013704 005672          MOV      RPADR,R4        ;UNIBUS ADDRESS OF RHCS1
2639 006354 004037 006052          JSR      RO,DRVINT        ;GO INIT. THE DRIVE
2640 006360 000442          BR       6$              ;ERROR RETURN
2641 006362 105761 005554          TSTB     DRVSTA(R1)      ;IS DRIVE STATUS ONLINE?
2642 006366 003004          BGT      1$              ;BRANCH IF YES
2643 006370 052762 140000 000016  BIS      #BIT15!BIT14,16(R2) ;ERROR--DRIVE IS OFFLINE OR NONEXISTENT
2644 006376 000423          BR       4$              ;GO TO EXIT
2645 006400 004037 012700          1$:     JSR      RO,DRVQUE    ;PUT THIS REQUEST IN QUEUE
2646 006404 000421          BR       5$              ;QUEUE IS FULL
2647 006406 122762 000103 000002  CMPB     #103,2(R2)      ;IS THIS REQ. FOR AN UNLOAD?
2648 006414 001003          BNE      2$              ;BR IF NO
2649 006416 112761 177777 005612  MOVB     #-1,ULDFLG(R1)  ;SET THE "UNLOAD IN QUEUE" FLAG
2650 006424 105761 005544          2$:     TSTB     DRVACT(R1)  ;IS THIS DRIVE ACTIVE?
2651 006430 001006          BNE      4$              ;BR IS YES
2652 006432 004737 006474          JSR      PC,OPT          ;CALL THE OPTIMIZER
2653 006436 000403          BR       4$

```



```

2654 006440 052762 120000 000016 3$: BIS #BIT15:BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
2655 006446 005720 4$: TST (R0)+
2656 006450 004037 013060 5$: JSR R0,GETR15 ;RESTORE R1-R5
2657 006454 105037 005610 CLRB ACTDRV ;CLEAR "ACTIVE DRIVER" FLAG
2658 006460 012637 177776 MOV (SP)+,J#PS ;RETURN "PS" TO USER LEVEL
2659 006464 000200 RTS R0 ;RETURN TO CALLER
2660 006466 004737 007476 6$: JSR PC,C17 ;GO HANDLE THE PARITY ERROR
2661 006472 000765 BR 4$
2662
2663 ;*OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
2664
2665 ;*CALL
2666 ;* MOV #DRVNUM,R1 ;DRIVE NUMBER TO R1
2667 ;* JSR PC,OPT ;SETUP A COMMAND
2668
2669 OPT: JSR R0,SAVR15 ;SAVE R1-R5
2670 006500 013746 177776 MOV J#PS,-(SP) ;SAVE PROC. STATUS
2671 006504 146137 005660 005606 BICB ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
2672 006512 004737 012754 JSR PC,GETREQ ;GET "DPB" POINTER OF REQUEST
2673 006516 005702 TST R2 ;IS THERE A REQUEST IN QUEUE?
2674 006520 001444 BEQ 6$ ;NO--BRANCH TO EXIT
2675 006522 105761 005554 TSTB DRVSTA(R1) ;IS DRIVE ONLINE?
2676 006526 003006 BGT 1$ ;YES--BRANCH
2677 006530 004737 012776 JSR PC,POPQUE ;NO--REMOVE REQUEST FROM QUEUE
2678 006534 052762 140000 000016 BIS #BIT15:BIT14,16(R2) ;SET ERROR BIT OF STATUS/ERROR INDICATOR
2679 006542 000426 BR 5$ ;BRANCH TO EXIT
2680 006544 122762 000150 000002 1$: CMPB #150,2(R2) ;IS THE REQUEST FOR I/O?
2681 006552 002403 BLT 2$ ;YES--BRANCH
2682 006554 004737 007054 JSR PC,C14 ;CALL THE COMMAND INITIATOR
2683 006560 000417 BR 5$ ;BRANCH TO EXIT
2684 006562 005737 005656 2$: TST DTUW ;DATA TRANSFER UNDERWAY?
2685 006566 002012 BGE 4$ ;YES--GO START A SEARCH
2686 006570 005737 005634 TST SEEKFG ;DO IMPLIED SEEKS?
2687 006574 100404 BMI 3$ ;YES---BRANCH
2688 006576 004037 007724 JSR R0,LA ;NO--DO LOOK AHEAD
2689 006602 000406 BR 5$ ;RETURN HERE ON A PARITY ERROR
2690 006604 000403 BR 4$ ;GO START A SEARCH
2691 006606 004737 006640 3$: JSR PC,C11 ;START A DATA TRANSFER
2692 006612 000402 BR 5$
2693 006614 004737 006746 4$: JSR PC,C13 ;START A SEARCH
2694 006620 012637 177776 5$: MOV (SP)+,J#PS ;RESTORE PROC. STATUS
2695 006624 004037 013060 JSR R0,GETR15 ;RESTORE R1-R5
2696 006630 000207 RTS PC
2697 006632 004737 012232 6$: JSR PC,SET.IE ;SET "IE" WITHOUT A "TRE"
2698 006636 000770 BR 5$ ;EXIT THE ROUTINE
2699
2700 ;*COMMAND INITIATOR
2701
2702 ;*CALL
2703 ;* MOV #DRVNUM,R1 ;DRIVE NUMBER
2704 ;* MOV #DPB,R2 ;ADDRESS OF DPB
2705 ;* JSR PC,C1? ;C1?= C11,C13, OR C14
2706 ;* ;WHERE:
2707 ;* ;C11=DATA TRANSFER
2708 ;* ;C12=SEARCH REQUESTED BY DATA XFER
2709 ;* ;C14=NOT DATA TRANSFER

```

2710									
2711	006640	004737	012776		CI1:	JSR	PC,POPQUE		;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
2712	006644	010237	005604			MOV	R2,TRNSWT		;PUT REQ. IN TRANSFER WAIT QUEUE
2713	006650	010203			CI2:	MOV	R2,R3		;DPB ADDRESS TO R3
2714	006652	013704	005672			MOV	RPADR,R4		;RHCS1 ADDRESS
2715	006656	010164	000010			MOV	R1,RHCS2(R4)		;SELECT DRIVE
2716	006662	062703	000004			ADD	#4,R3		;DESIRED WORD COUNT
2717	006666	062704	000002			ADD	#2,R4		;RHWG ADDRESS
2718	006672	012324				MOV	(R3)+,(R4)+		;LOAD WORD COUNT
2719	006674	012324				MOV	(R3)+,(R4)+		;LOAD BUFFER ADDRESS
2720	006676	012346				MOV	(R3)+,-(SP)		;LOAD SECTOR AND TRACK
2721	006700	004037	011776			JSR	RO,WRT.RP		;CALL THE LOAD(WRITE) ROUTINE
2722	006704	000006				RHDA			;INDEX OF REGISTER TO LOAD
2723	006706	007476				CI7			;ERROR RETURN ADDRESS
2724	006710	012346				MOV	(R3)+,-(SP)		;LOAD CYLINDER ADDRESS
2725	006712	004037	011776			JSR	RO,WRT.RP		
2726	006716	000034				RHCA			
2727	006720	007476				CI7			
2728	006722	016246	000002			MOV	2(R2),-(SP)		;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"
2729	006726	004037	011776			JSR	RO,WRT.RP		
2730	006732	000000				RHCS1			
2731	006734	007476				CI7			
2732	006736	010137	005656			MOV	R1,DTUW		;SET "DATA TRANSFER UNDERWAY"
2733	006742	000137	007432			JMP	CI5		
2734	006746	013704	005672		CI3:	MOV	RPADR,R4		;RHCS1 ADDRESS
2735	006752	010164	000010			MOV	R1,RHCS2(R4)		;SELECT DRIVE
2736	006756	016246	000012			MOV	12(R2),-(SP)		;DESIRED CYLINDER ADDRESS
2737	006762	004037	011776			JSR	RO,WRT.RP		
2738	006766	000034				RHCA			
2739	006770	007476				CI7			
2740	006772	116203	000010			MOVB	10(R2),R3		;PICKUP SECTOR ADDRESS
2741	006776	163703	005706			SUB	MXWINDW,R3		;BACKUP BY MAX. SEARCH FOR I/O WINDOW
2742	007002	002002				BGE	1\$		
2743	007004	062703	000026			ADD	#22,R3		
2744	007010	010346			1\$:	MOV	R3,-(SP)		;COMBINE THE ADJUSTED SECTOR WITH
2745	007012	116266	000011	000001		MOVB	11(R2),1(SP)		;THE DESIRED TRACK
2746	007020	004037	011776			JSR	RO,WRT.RP		;LOAD DESIRED TRACK & SECTOR
2747	007024	000006				RHDA			
2748	007026	007476				CI7			
2749	007030	012746	000131			MOV	#131,-(SP)		;START A SEARCH
2750	007034	004037	011776			JSR	RO,WRT.RP		
2751	007040	000000				RHCS1			
2752	007042	007476				CI7			
2753	007044	156137	005660	005606		BISB	ATABIT(R1),SRCHWT		;SET "SEARCH WAIT" KEY
2754	007052	000567				BR	CI5		
2755	007054	013704	005672		CI4:	MOV	RPADR,R4		;RHCS1 ADDRESS
2756	007060	010164	000010			MOV	R1,RHCS2(R4)		;SELECT DRIVE
2757	007064	116203	000002			MOVB	2(R2),R3		;PICKUP THE REQUESTED COMMAND
2758	007070	122703	000131			CMPB	#131,R3		;IS IT A SEARCH COMMAND?
2759	007074	001007				BNE	1\$		;BRANCH IF NO
2760	007076	016246	000010			MOV	10(R2),-(SP)		;LOAD DESIRED TRACK & SECTOR
2761	007102	004037	011776			JSR	RO,WRT.RP		
2762	007106	000006				RHDA			
2763	007110	007476				CI7			
2764	007112	000403				BR	2\$		;GO LOAD CYLINDER
2765	007114	122703	000105		1\$:	CMPB	#105,R3		;IS IT A SEEK COMMAND



2766	007120	001007			BNE	3\$		:BRANCH IF NO
2767	007122	016246	000012	2\$:	MOV	12(R2),-(SP)		:LOAD DESIRED CYLINDER
2768	007126	004037	011776		JSR	RD,WRT.RP		
2769	007132	000034			RHCA			
2770	007134	007476			CI7			
2771	007136	000546			BR	CI6		
2772	007140	122703	000115	3\$:	CMPB	#115,R3		:IS IT AN "OFFSET" COMMAND?
2773	007144	001013			BNE	4\$		:BR IF NO
2774	007146	004037	011640		JSR	RD,RD.RP		:MERGE THE OFFSET VALUE INTO RHOF
2775	007152	000032			RHOF			:BUT DON'T CHANGE THE UPPER
2776	007154	007476			CI7			
2777	007156	116216	000001		MOVB	1(R2),(SP)		:BYTE WHEN LOADING THE
2778	007162	004037	011776		JSR	RD,WRT.RP		:REGISTER (RHOF)
2779	007166	000032			RHOF			
2780	007170	007476			CI7			
2781	007172	000530			BR	CI6		:GO START THE COMMAND
2782	007174	122703	000107	4\$:	CMPB	#107,R3		:IS IT A "RECALIBRATE" COMMAND?
2783	007200	001525			BEQ	CI6		:BRANCH IF YES
2784	007202	122703	000117		CMPB	#117,R3		:IS IT A RETURN TO CENTER?
2785	007206	001522			BEQ	CI6		:BRANCH IF YES
2786	007210	122703	000103		CMPB	#103,R3		:IS IT AN "UNLOAD" COMMAND?
2787	007214	001016			BNE	5\$		:BRANCH IF NO
2788	007216	112761	000001	005544	MOVB	#1,DRVACT(R1)		:SET THE DRIVE ACTIVE INDICATOR
2789	007224	105061	005554		CLRB	DRVSTA(R1)		:PUT DRIVE STATUS TO OFFLINE
2790	007230	112761	000001	005612	MOVB	#1,ULDFLG(R1)		:SET "UNLOAD IN PROGRESS" FLAG
2791	007236	010346			MOV	R3,-(SP)		:START THE "UNLOAD" COMMAND
2792	007240	004037	011776		JSR	RD,WRT.RP		
2793	007244	000000			RHCS1			
2794	007246	007476			CI7			
2795	007250	000207			RTS	PC		:RETURN TO USER
2796	007252	122703	000143	5\$:	CMPB	#143,R3		:IS IT A "SET FORMAT" COMMAND?
2797	007256	001014			BNE	6\$		:BRANCH IF NO
2798	007260	004037	011640		JSR	RD,RD.RP		:READ THE OFFSET REGISTER
2799	007264	000032			RHOF			
2800	007266	007476			CI7			
2801	007270	116266	000001	000001	MOVB	1(R2),1(SP)		:COMBINE "FMT22" "ECI" AND "HCI"
2802	007276	004037	011776		JSR	RD,WRT.RP		:LOAD "FMT22", "ECI", AND/OR "HCI".
2803	007302	000032			RHOF			
2804	007304	007476			CI7			
2805	007306	000436			BR	12\$		
2806	007310	122703	000141	6\$:	CMPB	#141,R3		:IS IT A "GET REGISTER" COMMAND?
2807	007314	001023			BNE	10\$		:BRANCH IF NO
2808	007316	016203	000006	7\$:	MOV	6(R2),R3		:POINTS TO 1ST ADDRESS OF WHERE
2809								:TO PUT THE REGISTER(S)
2810	007322	116237	000010	007340	MOVB	10(R2),9\$		:INIT. THE INDEX FOR THE FIRST REG.
2811	007330	116205	000011		MOVB	11(R2),R5		:INDEX OF LAST REG. TO MOVE
2812	007334	004037	011640	8\$:	JSR	RD,RD.RP		:READ RP04 REGISTER
2813	007340	000000		9\$:	RHCS1			:INDEX OF REG. TO READ
2814	007342	007476			CI7			
2815	007344	012623			MOV	(SP)+,(R3)+		:GET THE CONTENTS OF RH11/RP04 REG.
2816	007346	023705	007340		CMP	9\$,R5		:LAST REG. BEEN READ?
2817	007352	001414			BEQ	12\$		:GET OUT IF YES
2818	007354	062737	000002	007340	ADD	#2,9\$		:INCREASE THE INDEX BY 2
2819	007362	000764			BR	8\$		:LOOP--MORE TO READ
2820	007364	122703	000145	10\$:	CMPB	#145,R3		:IS IT A "SELECT DRIVE" COMMAND?
2821	007370	001405			BEQ	12\$		:BRANCH IF YES

```

2822 007372 010346          11$:  MOV      R3,-(SP)          ;LOAD THE COMMAND
2823 007374 004037 011776    JSR      RD,WRT.RP
2824 007400 000000          RHCS1
2825 007402 007476          CI7
2826 007404 004737 012776    12$:  JSR      PC,POPQUE        ;REMOVE REQ. FROM QUEUE
2827 007410 052762 000200 000016  BIS      #BIT07,16(R2)    ;SET THE "DONE" BIT
2828 007416 005737 005632    TST      SAVEFG          ;SAVE THE RH11/RP04 REGISTERS?
2829 007422 100002          BPL      13$             ;BRANCH IF NO
2830 007424 004737 012140    JSR      PC,SVRH11       ;YES--GO SAVE THE REGISTERS
2831 007430 000207          13$:  RTS      PC              ;RETURN TO USER
2832 007432 006301          CIS:  ASL      R1
2833 007434 012761 001750 005636  MOV      #1000.,TIMER(R1) ;SET A ONE SECOND TIMER
2834 007442 006201          ASR      R1
2835 007444 112761 000001 005544  MOV      #1,DRVACT(R1)   ;SET THE DRIVE ACTIVE
2836 007452 000207          RTS      PC              ;RETURN TO THE USER
2837 007454 010346          CI6:  MOV      R3,-(SP)          ;LOAD THE COMMAND
2838 007456 004037 011776    JSR      RD,WRT.RP
2839 007462 000000          RHCS1
2840 007464 007476          CI7
2841 007466 000761          BR      CIS
2842 007470 105761 005544          CI7A: TSTB     DRVACT(R1)      ;IS THE DRIVE ACTIVE?
2843 007474 001405          BEQ      CI7B           ;BRANCH IF NO
2844 007476 012762 104000 000016  CI7:  MOV      #BIT15:BIT11,16(R2) ;SET "PARITY" ERROR INDICATOR
2845 007504 004737 012140    JSR      PC,SVRH11       ;GO SAVE THE RH11/RP04 REGISTERS
2846 007510 012746 000111          CI7B: MOV      #111,-(SP)    ;DO A "DRIVE CLEAR"
2847 007514 004037 011776    JSR      RD,WRT.RP
2848 007520 000000          RHCS1
2849 007522 007562          CI8
2850 007524 004737 012660    JSR      PC,EMPTYQ      ;EMPTY THE QUEUE
2851 007530 105061 005612    CLRB    ULDFLG(R1)      ;CLEAR THE UNLOAD IN QUEUE FLAG
2852 007534 105061 005544    CLRB    DRVACT(R1)      ;DRIVE IS IDLE
2853 007540 020137 005656    CMP     R1,DTUW         ;IF THIS DRIVE HAD AN I/O REQUEST
2854 007544 001005          BNE     1$              ;IN PROGRESS CLEAR ALL OF THE FLAGS
2855 007546 005037 005604    CLR     TRANSW
2856 007552 012737 177777 005656  MOV     #-1,DTUW
2857 007560 000207          1$:  RTS      PC
2858 007562 004037 013040          CI8:  JSR      RD,SAVR15     ;SAVE R1-R5
2859 007566 013704 005672    MOV     RPADR,R4        ;PICKUP THE ADDRESS OF THE FIRST REGISTER
2860 007572 005001          CLR     R1
2861 007574 005003          CLR     R3
2862 007576 105761 005544          1$:  TSTB     DRVACT(R1)     ;DRIVE ACTIVE?
2863 007602 001421          BEQ     3$              ;BRANCH IF NO
2864 007604 013702 005604    MOV     TRANSW,R2       ;GET THE "TRANSFER WAIT" QUEUE
2865 007610 020137 005656    CMP     R1,DTUW         ;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
2866 007614 001402          BEQ     2$              ;BRANCH IF YES
2867 007616 004737 012754    JSR     PC,GETREQ       ;GET THE DPB POINTER
2868 007622 004737 012140    JSR     PC,SVRH11       ;SAVE RH11/RP04 REGISTERS
2869 007626 052762 102000 000016  BIS     #BIT15:BIT10,16(R2) ;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
2870 007634 012763 177777 005636  MOV     #-1,TIMER(R3)   ;STOP THE TIMER
2871 007642 105061 005544          CLR     DRVACT(R1)     ;SET "DRIVE ACTIVE" TO IDLE
2872 007646 105061 005612          3$:  CLRB    ULDFLG(R1)     ;CLEAR UNLOAD FLAG
2873 007652 005201          INC     R1              ;MOVE TO THE NEXT DRIVE
2874 007654 062703 000002    ADD     #2,R3
2875 007660 042701 177770    BIC     #1<7,R1
2876 007664 001344          BNE     1$              ;BRANCH IF MORE DRIVES
2877 007666 012737 177777 005656  MOV     #-1,DTUW        ;NO DATA TRANSFERS UNDERWAY

```



```

2878 007674 005037 005604 CLR TRNSWT ;CLEAR THE "TRANSFER WAIT" QUEUE
2879 007700 004737 012576 JSR PC,CLRQUE ;CLEAR ALL OF THE REQUEST QUEUES
2880 007704 012764 000040 000010 MOV #BIT05,RHCS2(R4) ;DO A MASSBUS INIT.
2881 007712 004737 012232 JSR PC,SET.IE ;SET "IE" WITHOUT "IRE"
2882 007716 004037 013060 JSR RO,GETR15 ;RESTORE THE REGISTERS
2883 007722 000207 RTS PC ;RETURN
2884
2885 ;*LOOK AHEAD ROUTINE
2886
2887 ;*CALL
2888
2889 ;* MOV #DRVNUM,R1 ;DRIVE NUMBER
2890 ;* MOV #DPB,R2 ;POINT TO DPB
2891 ;* JSR RO,LA ;GO CHECK THE WINDOW
2892 ;* RETURN1 ;ERROR RETURN
2893 ;* RETURN2 ;START A SEARCH
2894 ;* RETURN3 ;START A DATA TRANSFER
2895
2896 LA: MOV RPADR,R4 ;GET RHCS1'S ADDRESS
2897 MOV R1,RHCS2(R4) ;SELECT DRIVE
2898 JSR RO,RD.RP ;READ CURRENT CYLINDER
2899 RHCC
2900 4$ ;ERROR RETURN ADDRESS
2901 CMP (SP)+,12(R2) ;IS CURRENT CYLINDER=DESIRED
2902 ;CYLINDER?
2903 BNE 3$ ;EXIT IF NO
2904 INCB LACNT(R1) ;INCREMENT THE LOOK AHEAD COUNT
2905 CMPB LACNT(R1),MXLACT ;EXCEED MAX?
2906 BGT 2$ ;BRANCH IF YES
2907 MOVB 10(R2),R3 ;GET DESIRED SECTOR ADDRESS AND
2908 SWAB R3 ;MULT. BY 64--ALIGN WITH
2909 ASR R3 ;LOOK AHEAD REGISTER
2910 ASR R3
2911 MOV #340,0#PS ;PRIORITY LEVEL "7"
2912 JSR RO,RD.RP ;READ LOOK AHEAD REGISTER
2913 RHLA
2914 4$
2915 SUB (SP)+,R3 ;CALCULATE THE DELTA
2916 BGE 1$
2917 ADD #(<22.*64.>),R3 ;MAKE THE DELTA POSITIVE
2918 CMP MXDLTA,R3 ;CHECK THE DELTA TO SEE
2919 BLT 3$ ;IF IT IS WITHIN THE
2920 CMP MNDLTA,R3 ;WINDOW---IF YES, ZERO
2921 BGE 3$ ;THE LOOK AHEAD COUNT
2922 CLRB LACNT(R1) ;AND TAKE THE I/O EXIT
2923 TST (RO)+
2924 3$: TST (RO)+ ;ADJUST THE RETURN ADDRESS
2925 RTS RO
2926 4$: JSR PC,C17 ;PROCESS THE ERROR
2927 RTS RO ;TAKE ERROR RETURN
2928
2929 ;*INTERRUPT SERVICE ROUTINE
2930 010062 112737 000001 005610 ISR: MOVB #1,ACTDRV ;SET "ACTIVE DRIVER" FLAG
2931 010070 004037 013040 JSR RO,SAVR15 ;SAVE R1-R5
2932 010074 013704 005672 MOV RPADR,R4 ;ADDRESS OF RHCS1
2933 010100 013701 005656 MOV DTUW,R1 ;GET "DATA TRANSFER UNDERWAY" INDICATOR
  
```

```

2934 010104 002403          BLT      1$          ;BRANCH IF NO DATA TRANSFER UNDERWAY
2935 010106 004737 010132    JSR      PC,TD      ;CALL TRANSFER DONE
2936 010112 000402          BR       2$          ;EXIT
2937 010114 004737 010260    1$: JSR      PC,SC    ;CALL SPECIAL CONDITIONS
2938 010120 004037 013060    2$: JSR      RD,GETR15 ;RESTORE R1-R5
2939 010124 105037 005610    CLR     ACTDRV      ;CLEAR "ACTIVE DRIVER" FLAG
2940 010130 000002          RTI              ;RETURN
2941
2942          ;*TRANSFER DONE ROUTINE
2943
2944 010132 105061 005544    TD:  CLR     DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
2945 010136 012737 177777 005656    MOV     #-1,DTUW    ;NO DATA TRANSFERS UNDERWAY
2946 010144 006301          ASL     R1
2947 010146 012761 177777 005636    MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
2948 010154 006201          ASR     R1
2949 010156 013702 005604    MOV     TRNSWT,R2   ;GET "DPB" ADDRESS FROM THE
2950 010162 005037 005604    CLR     TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
2951 010166 052762 000200 000016    BIS     #BIT07,16(R2) ;SET DONE
2952 010174 010164 000010    MOV     R1,RHCS2(R4) ;SELECT THE DRIVE
2953 010200 004037 011640    JSR     RD,RD.RP    ;TRANSFER ERROR(TRE=1)?
2954 010204 000000          RHCS1
2955 010206 007476          CI7
2956 010210 006126          ROL     (SP)+
2957 010212 100410          BMI     2$          ;BR IF YES
2958 010214 005737 005632    TST     SAVEFG      ;SAVE THE RH11/RP04 REGISTERS?
2959 010220 100002          BPL     1$          ;BRANCH IF NO
2960 010222 004737 012140    JSR     PC,SVRH11   ;YES--SAVE THE REGISTERS
2961 010226 004737 006474    1$: JSR     PC,OPT    ;CALL OPTIMIZER
2962 010232 000456          BR     SC2          ;SPECIAL CONDITION (ENTRY #2)
2963 010234 052762 100100 000016 2$: BIS     #BIT15:BIT06,16(R2) ;SET DATA ERROR FLAG
2964 010242 004737 012660    JSR     PC,EMPTYQ   ;EMPTY THE "DRIVES WAIT" QUEUE
2965 010246 004737 012140    JSR     PC,SVRH11   ;SAVE THE RH11/RP04 REGISTERS
2966 010252 012714 040111    MOV     #40111,(R4) ;ISSUE A "DRIVE CLEAR"
2967 010256 000444          BR     SC2          ;SPECIAL CONDITION (ENTRY #2)
2968
2969          ;*SPECIAL CONDITION ROUTINE
2970
2971 010260 005001    SC:  CLR     R1          ;START WITH DRIVE 0
2972 010262 012702 000001    MOV     #1,R2
2973 010266 105761 005554    1$:  TSTB   DRVSTA(R1) ;NONEXISTENT?
2974 010272 002016          BGE     2$          ;NO--BRANCH
2975 010274 005201          INC     R1          ;YES--MOVE TO THE NEXT DRIVE
2976 010276 106302          ASLB   R2          ;MORE DRIVES?
2977 010300 103372          BCC     1$          ;YES--BRANCH
2978 010302 013746 005634    MOV     SEEKFG,-(SP) ;SAVE THE "SEEK FLAG"
2979 010306 013746 005632    MOV     SAVEFG,-(SP) ;SAVE THE "SAVE FLAG"
2980 010312 004737 005710    JSR     PC,RPINIT   ;GO INIT. THE SUBSYSTEM
2981 010316 012637 005632    MOV     (SP)+,SAVEFG ;RESTORE THE "SAVE FLAG"
2982 010322 012637 005634    MOV     (SP)+,SEEKFG ;RESTORE THE "SEEK FLAG"
2983 010326 000405          BR     SC1          ;TAKE ERROR EXIT
2984 010330 010164 000010    2$:  MOV     R1,RHCS2(R4) ;SELECT DRIVE
2985 010334 116405 000001    MOVB   1(R4),R5    ;IS "SC"=1?
2986 010340 100413          BMI     SC2          ;BRANCH IF YES
2987 010342 012701 000010    SC1: MOV     #8.,R1   ;DO EIGHT DRIVES
2988 010346 005301    1$:  DEC     R1          ;NEXT DRIVE
2989 010350 002743          BLT     SC          ;BRANCH IF OUT OF DRIVES

```



2990	010352	105761	005544		TSTB	DRVACT(R1)		; IS THIS DRIVE IDLE?
2991	010356	001373			BNE	1\$		; BRANCH IF NO
2992	010360	104001			ERROR	1		; REPORT THE ERROR
2993	010362	004737	012232		JSR	PC,SET.IE		; GO SET INTERRUPT ENABLE
2994	010366	000207			RTS	PC		
2995	010370	012701	000003	SC2:	MOV	#3,R1		; READ RHAS UP TO THREE TIMES
2996	010374	116403	000016	1\$:	MOVB	RHAS(R4),R3		; READ "RHAS"
2997	010400	001011			BNE	2\$		; BRANCH IF ANY ATA BITS = 1
2998	010402	005301			DEC	R1		; COUNT THIS READ
2999	010404	003373			BGT	1\$		; LOOP IF MORE READS ALLOWED
3000	010406	004037	011640		JSR	RD,RD.RP		; READ CONTROL AND STATUS REGISTER
3001	010412	000000			RHCS1			
3002	010414	007562			CIB			
3003	010416	106126			ROLB	(SP)+		; IS "IE"=1?
3004	010420	100350			BPL	SC1		; NO--TAKE ERROR EXIT
3005	010422	000207			RTS	PC		; YES--RETURN
3006	010424	005046		2\$:	CLR	-(SP)		; PROCESS ALL DRIVES THAT HAVE
3007	010426	110316			MOVB	R3,(SP)		; AN "ATA"=1
3008	010430	012703	000001		MOV	#1,R3		
3009	010434	005001			CLR	R1		
3010	010436	030316		SC4:	BIT	R3,(SP)		; ATA=1?
3011	010440	001005			BNE	SC5		; YES--BRANCH
3012	010442	005201		SC3:	INC	R1		; MOVE TO THE NEXT DRIVE
3013	010444	106303			ASLB	R3		
3014	010446	001373			BNE	SC4		; BRANCH IF MORE TO CHECK?
3015	010450	005726			TST	(SP)+		; CLEAN OFF THE STACK
3016	010452	000207			RTS	PC		; RETURN TO USER
3017	010454	023701	005656	SC5:	CMP	DTUW,R1		; IS THIS DRIVE SETUP FOR I/O?
3018	010460	001002			BNE	1\$		; NO---BRANCH
3019	010462	005726			TST	(SP)+		; YES---CLEAN OFF THE STACK (RHAS)
3020	010464	000622			BR	TD		; BRANCH TO "TRANSFER DONE"
3021	010466	105761	005554	1\$:	TSTB	DRVSTA(R1)		; CHECK THE DRIVE STATUS
3022	010472	003030			BGT	SC6		; BRANCH IF ONLINE
3023	010474	105761	005612		TSTB	ULDFLG(R1)		; UNLOAD IN PROGRESS?
3024	010500	003420			BLE	2\$		; BRANCH IF NO
3025	010502	004737	012754		JSR	PC,GETREQ		; GET DPB POINTER
3026	010506	004737	012140		JSR	PC,SVRH11		; SAVE THE RH11/RP04 REGISTERS
3027	010512	004737	011170		JSR	PC,SC12		; SAVE RHDS1, RHER1, RHER2, AND RHER3
3028								; ALSO DO A DRIVE INIT (DRVINT)
3029	010516	105761	005554		TSTB	DRVSTA(R1)		; DID DRIVE COME ONLINE?
3030	010522	003411			BLE	3\$		; NO---BRANCH
3031	010524	032737	040000	005534	BIT	#BIT14,RPERRS		; WAS THERE AN ERROR?
3032	010532	001565			BEQ	SC11		; NO -- BRANCH
3033	010534	013705	005536		MOV	RPERRS+2,R5		; YES -- PICKUP RHER1 AND
3034	010540	000447			BR	SC6A		; GO PROCESS THE ERROR
3035	010542	004737	011170	2\$:	JSR	PC,SC12		; SAVE RHDS1, RHER1, RHER2, AND RHER3
3036								; ALSO DO A DRVINT
3037	010546	011605		3\$:	MOV	(SP),R5		; PICKUP (RHAS) BEFORE THE ERROR CALL
3038	010550	104005			ERROR	5		; REPORT THE ERROR
3039	010552	000733			BR	SC3		; GO CHECK FOR MORE ATA'S
3040	010554	006301		SC6:	ASL	R1		
3041	010556	012761	177777	005636	MOV	#-1,TIMER(R1)		; STOP THE TIMER
3042	010564	006201			ASR	R1		
3043	010566	004737	012754		JSR	PC,GETREQ		; GET THE DPB POINTER FROM THE QUEUE
3044	010572	010364	000016		MOV	R3,RHAS(R4)		; CLEAR ATTENTION
3045	010576	010164	000010		MOV	R1,RHCS2(R4)		; SELECT DRIVE



```

3046 010602 004037 011640 JSR RO, RD.RP ;READ THE RP04'S STATUS REG.
3047 010606 000012 RHDS1
3048 010610 011026 SC8
3049 010612 011605 MOV (SP), R5 ;AND PUT IT IN R5
3050 010614 006126 ROL (SP)+ ;WAS THERE AN ERROR?
3051 010616 100115 BPL SC9 ;BR IF NO
3052 010620 105761 005544 TSTB DRVACT(R1) ;CHECK THE DRIVE ACTIVE INDICATOR
3053 010624 001522 BEQ SC10 ;BRANCH IF IDLE
3054 010626 004037 011640 JSR RO, RD.RP ;READ ERROR REGISTER #1
3055 010632 000014 RHER1
3056 010634 011026 SC8
3057 010636 012605 MOV (SP)+, R5 ;AND SAVE IT IN R5
3058 010640 004737 012140 JSR PC, SVRH11 ;SAVE RH11/RP04 REGISTERS
3059 010644 012746 000111 MOV #111, -(SP) ;ISSUE A DRIVE CLEAR
3060 010650 004037 011776 JSR RO, WRT.RP
3061 010654 000000 RHCS1
3062 010656 011026 SC8
3063 010660 006105 SC6A: ROL R5 ;WAS "UNSAFE" CONDITION =1?
3064 010662 100404 BMI 1$ ;BRANCH IF YES
3065 010664 052762 100240 000016 BIS #BIT15:BIT07:BIT05, 16(R2) ;INFORM USER OF ERROR
3066 010672 000443 BR SC7
3067 010674 004037 011640 1$: JSR RO, RD.RP ;READ DRIVE STATUS REG. #1
3068 010700 000012 RHDS1
3069 010702 011026 SC8
3070 010704 011605 MOV (SP), R5 ;SAVE RHDS1 IN R5
3071 010706 006126 ROL (SP)+ ;"ERR"=1?
3072 010710 100013 BPL 2$ ;BR IF NO--UNSAFE CLEARED
3073 010712 112761 177777 005554 MOVB #-1, DRVSTA(R1) ;DRIVE IS UNSAFE
3074 010720 004737 012140 JSR PC, SVRH11 ;SAVE RH11/RP04 REGISTERS
3075 010724 010364 000016 MOV R3, RHAS(R4) ;CLEAR ATTENTION
3076 010730 052762 110000 000016 BIS #BIT15:BIT12, 16(R2) ;INFORM USER OF UNSAFE ERROR
3077 010736 000421 BR SC7
3078 010740 105705 2$: TSTB R5 ;"DRY" =1?
3079 010742 100414 BMI 3$ ;BRANCH IF YES
3080 010744 112761 177777 005544 MOVB #-1, DRVACT(R1) ;ACTIVE ERROR RECOVER
3081 010752 112761 000001 005554 MOVB #1, DRVSTA(R1) ;ONLINE
3082 010760 006301 ASL R1
3083 010762 012761 072460 005636 MOV #30000., TIMER(R1) ;START 30 SECOND TIMER
3084 010770 006201 ASR R1
3085 010772 000623 BR SC3
3086 010774 052762 100220 000016 3$: BIS #BIT15:BIT07:BIT04, 16(R2) ;INFORM USER OF ERROR
3087 011002 105061 005544 SC7: CLRB DRVACT(R1) ;DRIVE IS IDLE
3088 011006 004737 012660 JSR PC, EMPTYQ ;DUMP THE QUEUE
3089 011012 105761 005612 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS OR QUEUE?
3090 011016 001611 BEQ SC3 ;BR IF NO
3091 011020 105061 005612 CLRB ULDFLG(R1) ;CLEAR UNLOAD FLAG
3092 011024 000606 BR SC3
3093 011026 005726 SC8: TST (SP)+ ;REMOVE (RHAS) FROM THE STACK
3094 011030 105761 005544 TSTB DRVACT(R1) ;IS DRIVE IDLE?
3095 011034 001404 BEQ 1$ ;YES--BRANCH
3096 011036 004737 012754 JSR PC, GETREQ ;GET DPB POINTER
3097 011042 000137 007476 JMP CI7 ;PROCESS THE PARITY ERROR
3098 011046 000137 007510 1$: JMP CI7B ;PROCESS THE PARITY ERROR
3099 011052 105761 005544 SC9: TSTB DRVACT(R1) ;TEST DRIVE ACTIVE
3100 011056 003013 BGT SC11 ;BRANCH IF DRIVE IS ACTIVE
3101 011060 001404 BEQ SC10 ;BRANCH IF DRIVE IS IDLE

```



```

3102 011062 052762 100210 000016      BIS      #BIT15!BIT07!BIT03,16(R2) ;INFORM USER OF ERROR RECOVER COMPLETION
3103 011070 000744                      BR      SC7
3104 011072 011605                      MOV      (SP),R5 ;PUT (RHAS) IN R5
3105 011074 004737 011170      SC10:   JSR      PC,SC12      ;SAVE RHDS1, RHER1, RHER2, AND RHER3
3106 011100 104002                      ERROR    2      ;REPORT THE ERROR
3107 011102 000137 010442                      JMP      SC3      ;GO CHECK FOR MORE ATA'S
3108 011106 105761 005612      SC11:   TSTB     ULDFLG(R1) ;"UNLOAD IN PROGRESS"?
3109 011112 003402                      BLE     1$      ;BRANCH IF NO
3110 011114 105061 005612                      CLRB    ULDFLG(R1) ;CLEAR UNLOAD FLAG
3111 011120 105061 005544      1$:     CLRB    DRVACT(R1) ;SET DRIVE IDLE
3112 011124 136137 005660 005606      BITB    ATABIT(R1),SRCHWT ;DOING A SEARCH OPERATION FOR
3113                                     ;AN I/O COMMAND?
3114 011132 001012                      BNE     2$      ;BRANCH IF YES
3115 011134 004737 012776                      JSR      PC,POPQUE ;REMOVE REQUEST FROM QUEUE
3116 011140 052762 000200 000016      BIS     #BIT07,16(R2) ;SET "DONE" BIT
3117 011146 005737 005632                      TST     SAVEFG ;SAVE THE RH11/RP04 REGISTERS?
3118 011152 100002                      BPL     2$      ;BRANCH IF NO
3119 011154 004737 012140                      JSR      PC,SVRH11 ;YES--SAVE ALL OF THE RH11/RP04 REG'S
3120 011160 004737 006474      2$:     JSR      PC,OPT ;START A REQUEST
3121 011164 000137 010442                      JMP      SC3
3122 011170 010164 000010      SC12:   MOV      R1,RHCS2(R4) ;SELECT DRIVE
3123 011174 016437 000012 005534      MOV      RHDS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
3124 011202 016437 000014 005536      MOV      RHER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
3125 011210 016437 000040 005540      MOV      RHER2(R4),RPERRS+4
3126 011216 016437 000042 005542      MOV      RHER3(R4),RPERRS+6
3127 011224 004037 006052                      JSR      RO,DRVINT ;INIT. THE STATE OF THE DRIVE
3128 011230 000401                      BR      1$      ;TAKE ERROR EXIT
3129 011232 000207                      RTS     PC      ;RETURN
3130 011234 005726      1$:     TST     (SP)+ ;POP PC OFF OF THE STACK
3131 011236 000673                      BR      SC8      ;PROCESS THE PARITY ERROR
3132
3133      ;*RP04 TIMER ROUTINE
3134      ;*CALL
3135      ;*
3136      ;*
3137
3138 011240 005737 005610      RPTMR: TST     ACTDRV ;CHECK "ACTDRV & ACTSTR"
3139 011244 001032                      BNE     4$      ;IF NON ZERO EXIT
3140 011246 112737 000001 005611      MOVB    #1,ACTSTR ;SET "ACTSTR"
3141 011254 004037 013040                      JSR      RO,SAVR15 ;SAVE R1-R5
3142 011260 005001                      CLR     R1      ;START WITH DRIVE 0
3143 011262 005003                      CLR     R3
3144 011264 005763 005636      1$:     TST     TIMER(R3) ;IS THE TIMER RUNNING?
3145 011270 002407                      BLT     2$      ;BRANCH IF NO
3146 011272 166663 000016 005636      SUB     16(SP),TIMER(R3) ;COUNT THE INTERVAL
3147 011300 003003                      BGT     2$      ;BR IF NO SOFTWARE TIMEOUT
3148 011302 004037 011336                      JSR      RO,STO ;CALL SOFTWARE TIMEOUT ROUTINE
3149 011306 000405                      BR      3$      ;GO TO THE EXIT
3150 011310 005201      2$:     INC     R1 ;MOVE TO NEXT DRIVE
3151 011312 005723                      TST     (R3)+
3152 011314 022701 000010                      CMP     #8.,R1 ;OUT OF DRIVES?
3153 011320 003361                      BGT     1$      ;BRANCH IF NO
3154 011322 004037 013060      3$:     JSR      RO,GETR15 ;RESTORE R1-R5
3155 011326 105037 005611                      CLRB    ACTSTR ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
3156 011332 012616      4$:     MOV     (SP)+,(SP) ;ADJUST THE STACK
3157 011334 000200                      RTS     RO ;RETURN

```



```

3158
3159
3160
3161
3162
3163
3164
3165
3166 011336 013746 177776
3167 011342 013737 005676 177776
3168 011350 004037 013040
3169 011354 013704 005672
3170 011360 010164 000010
3171 011364 004037 011640
3172 011370 000012
3173 011372 011632
3174 011374 105726
3175 011376 100471
3176 011400 013702 005604
3177 011404 020137 005656
3178 011410 001402
3179 011412 004737 012754
3180 011416 052762 101000 000016 15:
3181 011424 004737 012140
3182 011430 012764 000040 000010
3183 011436 105061 005544
3184 011442 105061 005612
3185 011446 005001
3186 011450 005003
3187 011452 004037 006052 25:
3188 011456 000465
3189 011460 105761 005544
3190 011464 001414
3191 011466 013702 005604
3192 011472 023701 005656
3193 011476 001402
3194 011500 004737 012754
3195 011504 052762 100400 000016 35:
3196 011512 105061 005544
3197 011516 105061 005612 45:
3198 011522 012763 177777 005636
3199 011530 005723
3200 011532 005201
3201 011534 022701 000010
3202 011540 003344
3203 011542 012737 177777 005656
3204 011550 005037 005604
3205 011554 004737 012576
3206 011560 000417
3207 011562 016405 000016
3208 011566 136105 005660
3209 011572 001011
3210 011574 020137 005656
3211 011600 001277
3212 011602 004037 011640
3213 011606 000000

; *SOFTWARE TIMEOUT ROUTINE
; *
; *CALL: STO
; *      MOV      #DRVNUM,R1      ;DRIVE NUMBER
; *      JSR      RO,STO          ;CALL--DRVACT MUST BE NONZERO
; *      RETURN
; *
STO:  MOV      2#PS, -(SP)          ;SAVE THE PROCESSOR STATUS
      MOV      RPVEC+2, 2#PS      ;SET THE "PS" TO RP04 BR LEVEL
      JSR      RO, SAVR15         ;SAVE R1-R5
      MOV      RPADR, R4          ;GET ADDRESS OF "RHCS1"
      MOV      R1, RHCS2(R4)      ;SELECT THE DRIVE
      JSR      RO, RD.RP          ;READ "DRIVE STATUS REG"
      RHDS1
      STOS
      TSTB     (SP)+
      BMI     ST02
      MOV      TRANSW, R2
      CMP      R1, DTUW
      BEQ     15
      JSR      PC, GETREQ
      BIS      #BIT15:BIT09, 16(R2) ;SET THE ERROR FLAGS
      JSR      PC, SVRH11
      MOV      #BIT05, RHCS2(R4) ;"INIT" THE MASS BUS
      CLRB    DRVACT(R1)
      CLRB    ULDFLG(R1)
      CLR     R1
      CLR     R3
      JSR      RO, DRVINT
      BR      ST05
      TSTB    DRVACT(R1)
      BEQ     45
      MOV      TRANSW, R2
      CMP      DTUW, R1
      BEQ     35
      JSR      PC, GETREQ
      BIS      #BIT15:BIT08, 16(R2) ;INFORM USER OF INIT.
      CLRB    DRVACT(R1)
      CLRB    ULDFLG(R1)
      MOV      #-1, TIMER(R3)
      TST     (R3)+
      INC     R1
      CMP     #8., R1
      BGT     25
      MOV      #-1, DTUW
      CLR     TRANSW
      JSR      PC, CLARQUE
      BR      ST04
      MOV      RHAS(R4), R5
      BITB    ATABIT(R1), R5
      BNE     ST03
      CMP     R1, DTUW
      BNE     ST01
      JSR      RO, RD.RP
      RHCS1

; IS "DRY"=1?
; BR IF YES
; PICKUP TRANSFER WAIT QUEUE
; TRANSFER UNDERWAY ON THIS DRIVE?
; BRANCH IF YES
; GET DPB ADDRESS
; SET THE ERROR FLAGS
; SAVE RH11/RP04 REGISTERS
; "INIT" THE MASS BUS
; DRIVE IS IDLE
; CLEAR THE UNLOAD FLAG
; START WITH DRIVE 0
; INIT. THIS DRIVE
; PARITY ERROR RETURN
; DRIVE IDLE BEFORE THE INIT.?
; YES--BRANCH
; GET TRANSFER WAIT QUEUE
; WAS THERE I/O ON THIS DRIVE?
; YES--BRANCH
; GET THE DPB POINTER FROM QUEUE
; SET DRIVE ACTIVE TO IDLE
; NO UNLOAD
; STOP THE TIMER
; UPDATE THE INDEX
; INCREMENT THE DRIVE NUMBER
; LAST DRIVE BEEN CHECKED?
; NO--LOOP
; NO DATA TRANSFERS UNDERWAY
; CLEAR TRANSFER WAIT QUEUE
; CLEAR ALL REQUEST QUEUES
; EXIT
; IS ATTENTION FOR THIS
; DRIVE UP?
; YES--BRANCH
; DATA TRANSFER UNDERWAY FOR THIS DRIVE
; BR IF NO
; YES--CHECK "RDY"
    
```



```

3214 011610 011632          ST05
3215 011612 105726          TSTB      (SP)+
3216 011614 100271          BPL      ST01      ;BR IF "RDY"=0
3217 011616 005720          ST03: TST      (RO)+ ;ADJUST FOR THE PROPER RETURN
3218 011620 004037 013060   ST04: JSR      RD,GETR15 ;RESTORE R1-R5
3219 011624 012637 177776   MOV      (SP)+,2#PS. ;RESTORE PROCESSOR STATUS
3220 011630 000200          RTS      RO        ;RETURN
3221 011632 004737 007562   ST05: JSR      PC,C18 ;GO HANDLE THE PARITY ERROR
3222 011636 000770          BR       ST04
3223
3224          ;*ROUTINE TO READ A RH11/RP04 REGISTER
3225
3226          ;*CALL
3227          ;*
3228          ;* JSR      RO,RD.RP ;GO READ A REGISTER
3229          ;* INDEX ;REG. INDEX FROM BASE
3230          ;* ERRADR ;ERROR ADDRESS--PROCESS ERROR STARTING
3231          ;* ;AT THIS ADDRESS
3232          ;* RETURN ;CONTENTS OF REG. IS ON THE STACK
3233 011640 013737 005670 011764 RD.RP: MOV      MCPEMX,RD.RP2 ;MAX. RETRYS ALLOWED
3234 011646 011646          MOV      (SP)-,(SP) ;SAVE RO FOR RETURN
3235 011650 013737 005672 011664          MOV      RPADR,RD.ADR ;FORM THE DESIRED ADDRESS
3236 011656 062037 011664          ADD      (RO)+,RD.ADR ;USING THE BASE AND THE INDEX
3237 011662 013727          RD.RP1: MOV     2(PC)+,(PC)+ ;READ THE DESIRED REGISTER OF THE RP04
3238 011664 000000          RD.ADR: .WORD 0 ;ADDRESS IS FORMED HERE
3239 011666 000000          RD.WRD: .WORD 0 ;REG. CONTENTS PUT HERE
3240 011670 013766 011666 000002          MOV      RD.WRD,2(SP) ;RETURN IT TO THE USER
3241 011676 017746          MOV      2RPADR,-(SP) ;READ RHCSI
3242 011702 032716 020000          BIT      #BIT13,(SP) ;DID MCPE SET?
3243 011706 001002          BNE     1$ ;BRANCH IF YES
3244 011710 022620          CMP     (SP)+,(RO)+ ;ADJUST FOR RETURN
3245 011712 000200          RTS     RO        ;RETURN IF NO
3246 011714
3247 011714 104003          1$: ERROR 3 ;REPORT THE ERROR
3248 011716 005737 005656          TST     DTUM ;DATA TRANSFER UNDERWAY?
3249 011722 100405          BMI     2$ ;NO--BRANCH
3250 011724 032716 040000          BIT     #BIT14,(SP) ;NO--"TRE"=1?
3251 011730 001402          BEQ     2$ ;NO--BRANCH
3252 011732 005726          TST     (SP)+ ;YES--CLEAN OFF THE STACK AND
3253 011734 000415          BR      RD.RP3 ;TAKE THE FATAL ERROR EXIT
3254 011736 052716 040000          2$: BIS     #BIT14,(SP) ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
3255 011742 000316          SWAB   (SP) ;POSITION BEFORE WRITING
3256 011744 013737 005672 011760          MOV     RPADR,3$ ;FORM ADDRESS OF HIGH BYTE
3257 011752 005237 011760          INC     3$
3258 011756 112637          MOVB   (SP)+,2(PC)+ ;WRITE THE HIGH BYTE OF RHCSI
3259 011760 000000          3$: .WORD 0 ;ADDRESS STORAGE
3260 011762 005327          DEC     (PC)+ ;EXCEEDED MAX. RETRYS
3261 011764 000003          RD.RP2: .WORD 3
3262 011766 002335          BGE     RD.RP1 ;BRANCH IF NO
3263 011770 011000          RD.RP3: MOV     (RO),RO ;FATAL ERROR EXIT
3264 011772 012616          MOV     (SP)+,(SP)
3265 011774 000200          RTS     RO
3266
3267          ;*ROUTINE TO WRITE A RH11/RP04 REGISTER
3268
3269          ;*CALL

```

```

3270          :*      MOV      DATA, -(SP)      ;DATA TO BE LOADED ON THE STACK
3271          :*      JSR      RD, WRT.RP        ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.
3272          :*      INDEX   ;INDEX OF THE REGISTER TO BE LOADED
3273          :*      ERRADR  ;ADDRESS TO RETURN TO ON AN ERROR
3274          :*      RETURN ;ERROR FREE RETURN
3275
3276 011776 016637 000002 012064 WRT.RP: MOV      2(SP), WRT.WD ;SAVE THE WORD TO WRITE
3277 012004 012616          MOV      (SP)+, (SP) ;ADJUST THE STACK
3278 012006 012037 012066          MOV      (R0)+, WRT.AD ;GET INDEX OF REGISTER TO BE WRITTEN
3279 012012 001020          BNE      2$ ;BRANCH IF NOT RHCS1
3280 012014 122737 000150 012064          CMPB   #150, WRT.WD ;IS THE COMMAND FOR DATA TRANSFERS?
3281 012022 002414          BLT      2$ ;YES--DON'T GET THE OLD A16&A17, & PSEL
3282 012024 004037 011640          JSR      RD, RD.RP ;NO---COMBINE A16&A17, & PSEL WITH
3283 012030 000000          RHCS1 ;THE COMMAND BEFORE SENDING IT TO
3284 012032 012050          1$ ;THE RH11/RP04
3285 012034 000316          SWAB   (SP)
3286 012036 042716 177770          BIC   #1C7, (SP)
3287 012042 112637 012065          MOVB  (SP)+, WRT.WD+1
3288 012046 000402          BR     2$
3289 012050 011000          1$: MOV   (R0), R0 ;TAKE THE ERROR EXIT
3290 012052 000200          RTS   R0
3291 012054 063737 005672 012066 2$: ADD   RPADR, WRT.AD ;FORM THE ADDRESS OF THE DISK REG.
3292 012062 012737          MOV   (PC)+, 2(PC)+ ;LOAD THE DESIRED REG.
3293 012064 000000          WRT.WD: .WORD 0 ;WORD TO WRITE GOES HERE
3294 012066 000000          WRT.AD: .WORD 0 ;ADDRESS IS FORMED HERE
3295 012070 004037 011640          JSR   RD, RD.RP ;CHECK FOR PARITY ERROR ON WRITE
3296 012074 000014          RHER1
3297 012076 012130          2$
3298 012100 032726 000010          BIT   #BIT03, (SP)+
3299 012104 001413          BEQ   3$ ;BRANCH IF "PAR=0"
3300 012106 016037 177776 012120          MOV   -2(R0), 1$ ;PICKUP THE INDEX
3301 012114 004037 011640          JSR   RD, RD.RP ;READ THE REG.
3302 012120 000000          1$: .WORD 0 ;REG. INDEX
3303 012122 012130          2$ ;RETURN TO THIS ADDRESS ON ERROR
3304 012124 104004          ERROR 4 ;REPORT THE ERROR
3305 012126 005726          TST   (SP)+ ;CLEAN OFF THE STACK
3306 012130 011000          2$: MOV   (R0), R0 ;TAKE THE "PARITY ON WRITE" ERROR EXIT
3307 012132 000200          RTS   R0
3308 012134 005720          3$: TST   (R0)+ ;ADJUST FOR ERROR FREE EXIT
3309 012136 000200          RTS   R0
3310
3311          ;*ROUTINE TO SAVE THE RH11/RP04 REGISTERS AS PER DPB+14
3312          ;*CALL
3313          ;*
3314          ;*      MOV      #DPBNUM, R2      ;DPB POINTER TO R2
3315          ;*      JSR      PC, SVRH11      ;SAVE THE DRIVES REG'S
3316
3317 012140 004037 013040 SVRH11: JSR   R0, SAVR15 ;SAVE REG.'S R1-R5
3318 012144 013704          MOV   RPADR, R4
3319 012150 111264 000010          MOVB (R2), RHCS2(R4) ;SELECT DRIVE
3320 012154 016202 000014          MOV   14(R2), R2 ;GET THE ERROR TABLE POINTER
3321 012160 001421          BEQ   4$ ;EXIT IF 0
3322 012162 005003          CLR   R3 ;COUNTER & POINTER
3323 012164 012705 000022          MOV   #RHDB, R5 ;PROBLEM REGISTER
3324 012170 020305          1$: CMP   R3, R5 ;REACHED RHDB?
3325 012172 001005          BNE  2$ ;BR IF NO

```



```

3326 012174 105764 000010      TSTB   RHCS2(R4)      ;CHECK "OR"
3327 012200 100402              BMI     2$           ;BRANCH IF RHDB CAN BE READ
3328 012202 005022              CLR     (R2)+        ;ELSE SAVE IT AS 0'S
3329 012204 000403              BR      3$
3330 012206 010446      2$:   MOV     R4,-(SP)      ;FORM RH11/RP04 ADDRESS THAT IS
3331 012210 060316              ADD     R3,(SP)      ;TO BE READ
3332 012212 013622              MOV     2(SP)+,(R2)+ ;AND READ IT
3333 012214 005723      3$:   TST     (R3)+        ;MOVE TO NEXT REG INDEX
3334 012216 020327 000046      CMP     R3,#RHEC2    ;DONE?
3335 012222 003762              BLE     1$           ;BRANCH IF NO
3336 012224 004037 013060      4$:   JSR     RD,GETR15   ;GET REGISTERS R1-R5
3337 012230 000207              RTS     PC           ;RETURN TO USER
3338
3339      ;*ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
3340      ;*CALL
3341      ;*   MOV     #DRVNUM,R1      ;DRIVE NUMBER TO R1
3342      ;*   JSR     PC,SET.IE      ;SET "IE"
3343      ;*   RETURN
3344
3345 012232 010446      SET.IE: MOV     R4,-(SP)      ;SAVE R4
3346 012234 013704 005672      MOV     RPADR,R4     ;PICKUP ADDRESS OF RHCS1
3347 012240 010164 000010      MOV     R1,RHCS2(R4) ;SELECT DRIVE
3348 012244 011446      MOV     (R4)-,(SP)   ;READ RHCS1
3349 012246 052716 040000      BIS     #BIT14,(SP)  ;SET THE "TRE" BIT OF THE WORD READ
3350 012252 000316      SWAB   (SP)         ;ADJUST FOR DAT0
3351 012254 112714 000100      MOVB   #BIT06,(R4)   ;SET "IE"
3352 012260 032764 010000 000010  BIT     #BIT12,RHCS2(R4) ;IS "NED"=1?
3353 012266 001002      BNE     1$           ;YES--CLEAR "TRE"
3354 012270 005726      TST     (SP)+        ;CLEAN OFF THE STACK
3355 012272 000402      BR      2$
3356 012274 112664 000001      1$:   MOVB   (SP)+,1(R4) ;CLEAR "TRE"
3357 012300 012604      2$:   MOV     (SP)+,R4    ;RESTORE R4
3358 012302 000207      RTS     PC           ;RETURN TO CALLER
3359
3360      ;*QUEUE COUNT
3361 012304      000      QCNT: .BYTE 0          ;DRIVE 0
3362 012305      000      .BYTE 0          ;DRIVE 1
3363 012306      000      .BYTE 0          ;DRIVE 2
3364 012307      000      .BYTE 0          ;DRIVE 3
3365 012310      000      .BYTE 0          ;DRIVE 4
3366 012311      000      .BYTE 0          ;DRIVE 5
3367 012312      000      .BYTE 0          ;DRIVE 6
3368 012313      000      .BYTE 0          ;DRIVE 7
3369
3370      ;QUEUE INPUT POINTERS
3371
3372 012314 012376      QINPT: .WORD  QDRV0      ;DRIVE 0
3373 012316 012416      .WORD  QDRV1      ;DRIVE 1
3374 012320 012436      .WORD  QDRV2      ;DRIVE 2
3375 012322 012456      .WORD  QDRV3      ;DRIVE 3
3376 012324 012476      .WORD  QDRV4      ;DRIVE 4
3377 012326 012516      .WORD  QDRV5      ;DRIVE 5
3378 012330 012536      .WORD  QDRV6      ;DRIVE 6
3379 012332 012556      .WORD  QDRV7      ;DRIVE 7
3380
3381      ;QUEUE OUTPUT POINTERS

```

```

3382
3383 012334 012376          QOUTPT: .WORD  QDRV0          ;DRIVE 0
3384 012336 012416          .WORD  QDRV1          ;DRIVE 1
3385 012340 012436          .WORD  QDRV2          ;DRIVE 2
3386 012342 012456          .WORD  QDRV3          ;DRIVE 3
3387 012344 012476          .WORD  QDRV4          ;DRIVE 4
3388 012346 012516          .WORD  QDRV5          ;DRIVE 5
3389 012350 012536          .WORD  QDRV6          ;DRIVE 6
3390 012352 012556          .WORD  QDRV7          ;DRIVE 7
3391
3392 012354 012376          QSTART: .WORD  QDRV0          ;DRIVE 0 START ADDRESS
3393 012356 012416          QSTOP:  .WORD  QDRV1          ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
3394 012360 012436          .WORD  QDRV2          ;STOP DRIVE 1--START DRIVE 2
3395 012362 012456          .WORD  QDRV3          ;STOP DRIVE 2--START DRIVE 3
3396 012364 012476          .WORD  QDRV4          ;STOP DRIVE 3--START DRIVE 4
3397 012366 012516          .WORD  QDRV5          ;STOP DRIVE 4--START DRIVE 5
3398 012370 012536          .WORD  QDRV6          ;STOP DRIVE 5--START DRIVE 6
3399 012372 012556          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
3400 012374 012576          .WORD  QTERM          ;STOP DRIVE 7
3401
3402          ;DRIVE REQUEST QUEUES
3403
3404 012376 000010          QDRV0:  .BLKW  10
3405 012416 000010          QDRV1:  .BLKW  10
3406 012436 000010          QDRV2:  .BLKW  10
3407 012456 000010          QDRV3:  .BLKW  10
3408 012476 000010          QDRV4:  .BLKW  10
3409 012516 000010          QDRV5:  .BLKW  10
3410 012536 000010          QDRV6:  .BLKW  10
3411 012556 000010          QDRV7:  .BLKW  10
3412          QTERM=.
3413
3414          ;*ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
3415          ;*
3416          ;*CALL
3417          ;*      JSR      PC,CLRQUE
3418
3419 012576 004037 013040          CLRQUE: JSR      RD,SAVR15          ;SAVE R1-R5
3420 012602 012702 012304          MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
3421 012606 005022          CLR      (R2)+          ;DRIVES 0 & 1
3422 012610 005022          CLR      (R2)+          ;DRIVES 2 & 3
3423 012612 005022          CLR      (R2)+          ;DRIVES 4 & 5
3424 012614 005022          CLR      (R2)+          ;DRIVES 6 & 7
3425 012616 012703 000010          MOV      #8,R3          ;MOVE THE STARTING
3426 012622 012701 012354          MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
3427 012626 012122          1$:  MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
3428 012630 005303          DEC      R3
3429 012632 001375          BNE     1$
3430 012634 012703 000010          MOV      #8,R3          ;MOVE THE STARTING ADDRESS
3431 012640 012701 012354          MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
3432 012644 012122          2$:  MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
3433 012646 005303          DEC      R3
3434 012650 001375          BNE     2$
3435 012652 004037 013060          JSR      RD,GETR15          ;RESTORE R1-R5
3436 012656 000207          RTS      PC
3437

```



```

3438 ;*EMPTY THE QUEUE SPECIFIED BY R1
3439 ;
3440 ;*CALL
3441 ;*   MOV   DRVNUM,R1   ;DRIVE NUMBER TO R1
3442 ;*   JSR   PC,EMPTYQ
3443 ;
3444 012660 105061 012304   EMPTYQ: CLRB   QCNT(R1)   ;CLEAR NUMBER OF ITEMS IN QUEUE
3445 012664 006301         ASL   R1
3446 012666 016161 012314 012334   MOV   QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
3447 012674 006201         ASR   R1
3448 012676 000207         RTS   PC
3449 ;
3450 ;*ROUTINE TO PUT A REQUEST IN QUEUE
3451 ;
3452 ;*CALL
3453 ;*   MOV   #DRVNUM,R1   ;DRIVE NUMBER
3454 ;*   MOV   #DPB,R2     ;ADDRESS OF PARAMETER BLOCK
3455 ;*   JSR   RD,DRVQUE   ;GO PUT REQUEST IN QUEUE
3456 ;*   RETURN1          ;RETURN HERE IF QUEUE IS FULL
3457 ;*   RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
3458 ;
3459 012700 122761 000010 012304   DRVQUE: CMPB   #10,QCNT(R1)   ;IS QUEUE FULL?
3460 012706 001421         BEQ   2$   ;BR IF YES-TAKE RETURN1
3461 012710 105261 012304         INCB   QCNT(R1)   ;INCREMENT QUEUE COUNT
3462 012714 006301         ASL   R1
3463 012716 010271 012314         MOV   R2,QINPT(R1)   ;PUT THIS REQUEST IN QUEUE
3464 012722 062761 000002 012314   ADD   #2,QINPT(R1)   ;UPDATE THE QUEUE POINTER
3465 012730 026161 012314 012356   CMP   QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
3466 012736 001003         BNE   1$   ;BRANCH IF NO
3467 012740 016161 012354 012314   MOV   QSTART(R1),QINPT(R1) ;YES--RESET POINTER
3468 012746 006201         1$:  ASR   R1
3469 012750 005720         TST   (R0)+   ;TAKE RETURN 2
3470 012752 000200         2$:  RTS   R0   ;RETURN TO USER
3471 ;
3472 ;*ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
3473 ;
3474 ;*CALL
3475 ;*   MOV   #DRVNUM,R1   ;DRIVE NUMBER TO R1
3476 ;*   JSR   PC,GETREQ   ;GO GET THE REQUEST
3477 ;*   RETURN          ;R2="DPB" ADDRESS OF THE REQUEST
3478 ;*   ;R2=0 IF NO REQUEST IN QUEUE
3479 ;
3480 012754 005002         GETREQ: CLR   R2
3481 012756 105761 012304         TSTB  QCNT(R1)   ;IS THERE ANY REQUEST IN QUEUE?
3482 012762 001404         BEQ   2$   ;NO---BRANCH
3483 012764 006301         1$:  ASL   R1
3484 012766 017102 012334         MOV   QOUTPT(R1),R2   ;PICKUP "DPB" POINTER FOR THIS DRIVE
3485 012772 006201         ASR   R1
3486 012774 000207         2$:  RTS   PC   ;RETURN TO USER
3487 ;
3488 ;*ROUTINE TO "POP" THE REQUEST FROM QUEUE
3489 ;
3490 ;*CALL
3491 ;*   MOV   #DRVNUM,R1   ;DRIVE NUMBER TO R1
3492 ;*   JSR   PC,POPQUE   ;CALL TO REMOVE REQUEST
3493 ;*   RETURN          ;R2=ADDRESS OF DPB REMOVED

```

```

3494
3495 012776 105361 012304 POPQUE: DECB QCNT(R1) ;DECREMENT QUEUE COUNT
3496 013002 006301 ASL R1
3497 013004 017102 012334 MOV QOUTPT(R1),R2 ;GET THE "DPB" POINTER
3498 013010 062761 000002 012334 ADD #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
3499 013016 026161 012334 012356 CMP QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
3500 013024 001003 BNE IS ;NO--BRANCH TO EXIT
3501 013026 016161 012354 012334 MOV QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
3502 013034 006201 IS: ASR R1
3503 013036 000207 RTS PC ;RETURN TO USER
3504
3505 ;*ROUTINES TO SAVE R0-R5 AND R1-R5
3506 ;*
3507 ;*CALL: SAVR05
3508 ;* JSR RO,SAVR05
3509 ;* RETURN ;RO-R5 IS ON THE STACK
3510 ;*
3511 ;*CALL: SAVR15
3512 ;* JSR RO,SAVR15
3513 ;* RETURN ;R1-R5 IS ON THE STACK
3514 ;*
3515 ;*UPON RETURN FROM SAVR05 AND SAVR15 THE STACK WILL LOOK LIKE:
3516 ;*
3517 ;*+12 RO
3518 ;*+10 R1
3519 ;*+06 R2
3520 ;*+04 R3
3521 ;*+02 R4
3522 ;*TOP R5
3523
3524 013040 SAVR05: ;SAVE R0 BY THE JSR INST.
3525 013040 010146 SAVR15: MOV R1,-(SP) ;SAVE R1
3526 013042 010246 MOV R2,-(SP) ;SAVE R2
3527 013044 010346 MOV R3,-(SP) ;SAVE R3
3528 013046 010446 MOV R4,-(SP) ;SAVE R4
3529 013050 010546 MOV R5,-(SP) ;SAVE R5
3530 013052 016646 000012 MOV 12(SP),-(SP) ;GET R0
3531 013056 000200 RTS RO
3532
3533 ;*ROUTINES TO RESTORE R0-R5 AND R1-R5
3534 ;*
3535 ;*CALL: GETR05
3536 ;* JSR RO,GETR05
3537 ;* RETURN ;RO-R5 HAVE BEEN RESTORED
3538 ;*
3539 ;*CALL: GETR15
3540 ;* JSR RO,GETR15
3541 ;* RETURN ;R1-R5 HAVE BEEN RESTORED
3542 ;*
3543 013060 011666 000014 GETR15: MOV (SP),14(SP) ;POSITION R0
3544 013064 005726 GETR05: TST (SP)+ ;POP R0 OF THE JSR OFF OF THE STACK
3545 013066 012605 MOV (SP)+,R5 ;RESTORE R5
3546 013070 012604 MOV (SP)+,R4 ;RESTORE R4
3547 013072 012603 MOV (SP)+,R3 ;RESTORE R3
3548 013074 012602 MOV (SP)+,R2 ;RESTORE R2
3549 013076 012601 MOV (SP)+,R1 ;RESTORE R1
  
```



```

3550 013100 000200
3551
3552
3553
3554
3555
3556
3557
3558
3559 013102 000
3560 013103 000
3561 013104 000
3562 013105 000
3563 013106 000000
3564 013110 001176
3565
3566 013112 000
3567
3568 013113 000
3569
3570 013114 000000
3571 013116 013122
3572
3573
3574
3575
3576
3577 013120 000000
3578
3579
3580
3581
3582
3583 013122 000000
3584 013124 000000
3585 013126 000000
3586 013130 000000
3587 013132 000000
3588 013134 000000
3589 013136 000000
3590 013140 000000
3591 013142 000000
3592 013144 000000
3593 013146 000000
3594 013150 000000
3595 013152 000000
3596 013154 000000
3597 013156 000000
3598 013160 000000
3599 013162 000000
3600 013164 000000
3601 013166 000000
3602 013170 000000
3603
3604 000001
3605 000002

```

RYS RO ;RESTORE RO

```

;*****
.SBTTL DATA PARAMETER BLOCK
;*****

```

```

DPB: .BYTE 0 ;DRIVE NUMBER
      .BYTE 0 ;OFFSET VALUE OR FMT22, ECI, AND HCI
      .BYTE 0 ;COMMAND
      .BYTE 0 ;PSEL AND A17 AND A16
      .WORD 0 ;WORD COUNT (MUST BE NEG)
      .WORD INREG ;BUFFER ADDRESS OR
                ;REGISTER TABLE POINTER
      .BYTE 0 ;SECTOR ADDRESS OR
                ;FIRST REGISTER INDEX
      .BYTE 0 ;TRACK ADDRESS OR
                ;LAST REGISTER INDEX
      .WORD 0 ;CYLINDER ADDRESS
      .WORD REG ;ERROR TABLE POINTER
                ;POINTS TO THE FIRST OF TWENTY
                ;LOCATIONS WHERE THE DRIVER IS
                ;TO STORE THE RH11/RP04 REGISTERS
                ;ON AN ERROR. IF ZERO, REGISTERS
                ;ARE NOT SAVED.
      .WORD 0 ;STATUS/ERROR INDICATOR
                ;BIT15 = 1 => ERROR OCCURED
                ;BIT07 = 1 => DONE
                ;BIT 14 - BIT10 AND BIT06 - BIT03
                ;INDICATE TYPE OF ERROR

```

```

REG: .WORD 0 ;STORE RP04 REGISTERS HERE
      .WORD 00 ;RHMC
      .WORD 00 ;RHBA
      .WORD 00 ;RHDA
      .WORD 00 ;RHCS2
      .WORD 00 ;RHDS1
      .WORD 00 ;RHER1
      .WORD 00 ;RHAS
      .WORD 00 ;RHLA
      .WORD 00 ;RHDB
      .WORD 00 ;RHMR
      .WORD 00 ;RHDT
      .WORD 00 ;RHSN
      .WORD 00 ;RHOF
      .WORD 00 ;RHCA
      .WORD 00 ;RHCC
      .WORD 00 ;RHER2
      .WORD 00 ;RHER3
      .WORD 00 ;RHEC1
      .WORD 00 ;RHEC2

```

```

CODE=1 ;DPB INDEX EQUATES
COMND=2

```

3606 000006  
3607 000010  
3608 000011  
3609 000012  
3610 000016

BUF=6  
SEC=10  
TRK=11  
CYL=12  
STATUS=16

\*\*\*\*\*

.SBTTL HEAD CODE TABLE

\*\*\*\*\*

3618 013172 000000  
3619 013174 000001  
3620 013176 000002  
3621 013200 000003  
3622 013202 000004  
3623 013204 000005  
3624 013206 000006  
3625 013210 000007  
3626 013212 000010  
3627 013214 000011  
3628 013216 000012  
3629 013220 000013  
3630 013222 000014  
3631 013224 000015  
3632 013226 000016  
3633 013230 000017  
3634 013232 000020  
3635 013234 000021  
3636 013236 000022  
3637 013240 100000

HEAD1: .WORD 0  
.WORD 1  
.WORD 2  
.WORD 3  
.WORD 4  
.WORD 5  
.WORD 6  
.WORD 7  
.WORD 10  
.WORD 11  
.WORD 12  
.WORD 13  
.WORD 14  
.WORD 15  
.WORD 16  
.WORD 17  
.WORD 20  
.WORD 21  
.WORD 22  
.WORD 100000

;HEAD ADDRESSES FOR CYL 245

;TABLE TERMINATOR

3638 013242 000000  
3639 013244 000022  
3640 013246 100000  
3641 013250 000000  
3642 013252 000022  
3643 013254 100000

HEAD2: .WORD 0  
.WORD 22  
.WORD 100000  
.WORD 0  
.WORD 22  
.WORD 100000

;HEAD ADDRESSES FOR CYLS 400 & 004

;TERMINATOR

\*\*\*\*\*

.SBTTL OFFSET CODE TABLE

\*\*\*\*\*

3651 013256  
3652 013256 060  
3653 013256 057  
3654 013257 056  
3655 013260 055  
3656 013261 054  
3657 013262 053  
3658 013263 052  
3659 013264 051  
3660 013265 050  
3661 013266

OFFTBL: .BYTE 60  
.BYTE 57  
.BYTE 56  
.BYTE 55  
.BYTE 54  
.BYTE 53  
.BYTE 52  
.BYTE 51  
.BYTE 50



3662	013267	047	.BYTE	47
3663	013270	046	.BYTE	46
3664	013271	045	.BYTE	45
3665	013272	044	.BYTE	44
3666	013273	043	.BYTE	43
3667	013274	042	.BYTE	42
3668	013275	041	.BYTE	41
3669	013276	040	.BYTE	40
3670	013277	037	.BYTE	37
3671	013300	036	.BYTE	36
3672	013301	035	.BYTE	35
3673	013302	034	.BYTE	34
3674	013303	033	.BYTE	33
3675	013304	032	.BYTE	32
3676	013305	031	.BYTE	31
3677	013306	030	.BYTE	30
3678	013307	027	.BYTE	27
3679	013310	026	.BYTE	26
3680	013311	025	.BYTE	25
3681	013312	024	.BYTE	24
3682	013313	023	.BYTE	23
3683	013314	022	.BYTE	22
3684	013315	021	.BYTE	21
3685	013316	020	.BYTE	20
3686	013317	017	.BYTE	17
3687	013320	016	.BYTE	16
3688	013321	015	.BYTE	15
3689	013322	014	.BYTE	14
3690	013323	013	.BYTE	13
3691	013324	012	.BYTE	12
3692	013325	011	.BYTE	11
3693	013326	010	.BYTE	10
3694	013327	007	.BYTE	7
3695	013330	006	.BYTE	6
3696	013331	005	.BYTE	5
3697	013332	004	.BYTE	4
3698	013333	003	.BYTE	3
3699	013334	002	.BYTE	2
3700	013335	001	.BYTE	1
3701	013336	000	.BYTE	0
3702	013337	201	.BYTE	201
3703	013340	202	.BYTE	202
3704	013341	203	.BYTE	203
3705	013342	204	.BYTE	204
3706	013343	205	.BYTE	205
3707	013344	206	.BYTE	206
3708	013345	207	.BYTE	207
3709	013346	210	.BYTE	210
3710	013347	211	.BYTE	211
3711	013350	212	.BYTE	212
3712	013351	213	.BYTE	213
3713	013352	214	.BYTE	214
3714	013353	215	.BYTE	215
3715	013354	216	.BYTE	216
3716	013355	217	.BYTE	217
3717	013356	220	.BYTE	220

3718	013357	221	.BYTE	221
3719	013360	222	.BYTE	222
3720	013361	223	.BYTE	223
3721	013362	224	.BYTE	224
3722	013363	225	.BYTE	225
3723	013364	226	.BYTE	226
3724	013365	227	.BYTE	227
3725	013366	230	.BYTE	230
3726	013367	231	.BYTE	231
3727	013370	232	.BYTE	232
3728	013371	233	.BYTE	233
3729	013372	234	.BYTE	234
3730	013373	235	.BYTE	235
3731	013374	236	.BYTE	236
3732	013375	237	.BYTE	237
3733	013376	240	.BYTE	240
3734	013377	241	.BYTE	241
3735	013400	242	.BYTE	242
3736	013401	243	.BYTE	243
3737	013402	244	.BYTE	244
3738	013403	245	.BYTE	245
3739	013404	246	.BYTE	246
3740	013405	247	.BYTE	247
3741	013406	250	.BYTE	250
3742	013407	251	.BYTE	251
3743	013410	252	.BYTE	252
3744	013411	253	.BYTE	253
3745	013412	254	.BYTE	254
3746	013413	255	.BYTE	255
3747	013414	256	.BYTE	256
3748	013415	257	.BYTE	257
3749	013416	260	.BYTE	260
3750	013417			

ENDTBL: .EVEN

;\*\*\*\*\*

.SBTTL MESSAGES

;\*\*\*\*\*

3759	013420	005015	050122	032060	TITLE: .ASCII <15><12>/RPO4 HEAD ALIGNMENT VERIFICATION PROGRAM/<15><12>
3760	013426	044040	040505	020104	
3761	013434	046101	043511	046516	
3762	013442	047105	020124	042526	
3763	013450	044522	044506	040503	
3764	013456	044524	047117	050040	
3765	013464	047522	051107	046501	
3766	013472	005015			
3767	013474	040515	047111	042504	.ASCIZ /MAINDEC-11-DERPM-B/<15><12><12>
3768	013502	026503	030461	042055	
3769	013510	051105	046520	041055	
3770	013516	005015	000012		
3771					
3772	013522	047504	047040	052117	DDU: .ASCII /DO NOT SELECT DRIVE UNTIL DDU (OR DEDU) HAS BEEN CONNECTED/<15><12>
3773	013530	051440	046105	041505	



3774	013536	020124	051104	053111
3775	013544	020105	047125	044524
3776	013552	020114	042104	020125
3777	013560	047450	020122	042504
3778	013566	052504	020051	040510
3779	013574	020123	042502	047105
3780	013602	041440	047117	042516
3781	013610	052103	042105	005015
3782	013616	020040	020040	042040
3783	013624	052504	024040	051117
3784	013632	042040	042105	024525
3785	013640	050040	052514	051507
3786	013646	044440	052116	020117
3787	013654	051104	053111	020105
3788	013662	046123	052117	020123
3789	013670	034501	023040	040440
3790	013676	030061	005015	
3791	013702	020040	020040	051440
3792	013710	052105	042040	052504
3793	013716	024040	051117	042040
3794	013724	042105	024525	043040
3795	013732	047125	052103	047511
3796	013740	020116	053523	052111
3797	013746	044103	052040	020117
3798	013754	044047	040505	020104
3799	013762	046101	043511	046516
3800	013770	047105	023524	005015
3801	013776	000012		
3802				
3803	014000	042504	042520	042116
3804	014006	047111	020107	047117
3805	014014	052040	042510	044440
3806	014022	042116	053111	042111
3807	014030	040525	020114	046101
3808	014036	043511	046516	047105
3809	014044	020124	040520	045503
3810	014052	052440	042523	026104
3811	014060	041440	047117	052123
3812	014066	047101	006524	012
3813	014073	040	020040	020040
3814	014100	047111	042504	020130
3815	014106	051105	047522	051522
3816	014114	046440	054501	047440
3817	014122	041503	051125	020056
3818	014130	052040	020117	047111
3819	014136	044510	044502	020124
3820	014144	044124	020105	047111
3821	014152	042504	020130	051105
3822	014160	047522	051522	006454
3823	014166	012		
3824	014167	040	020040	020040
3825	014174	052506	041516	044524
3826	014202	047117	023440	052053
3827	014210	047120	042504	042530
3828	014216	051122	020047	052515
3829	014224	052123	041040	020105

.ASCII / DDU (OR DEDU) PLUGS INTO DRIVE SLOTS A9 & A10/<15><12>

.ASCIZ / SET DDU (OR DEDU) FUNCTION SWITCH TO 'HEAD ALIGNMENT'/<15><12><12>

IXEGND: .ASCII /DEPENDING ON THE INDIVIDUAL ALIGNMENT PACK USED, CONSTANT/<15><12>

.ASCII / INDEX ERRORS MAY OCCUR. TO INHIBIT THE INDEX ERRORS,/<15><12>

.ASCII / FUNCTION '+TPNDEXERR' MUST BE DISABLED IN THE DRIVE /<15><12>



3830	014232	044504	040523	046102
3831	014240	042105	044440	020116
3832	014246	044124	020105	051104
3833	014254	053111	020105	005015
3834	014262	020040	020040	041040
3835	014270	044505	043516	041440
3836	014276	042510	045503	042105
3837	014304	026440	043440	047522
3838	014312	047125	020104	044520
3839	014320	020116	030501	030101
3840	014326	030065	006470	005012
3841	014334	000		
3842				
3843	014335	015	042012	044522
3844	014342	042526	047040	052117
3845	014350	040440	040526	046111
3846	014356	041101	042514	005015
3847	014364	000012		
3848				
3849	014366	005015	047105	042524
3850	014374	020122	051104	053111
3851	014402	020105	052516	041115
3852	014410	051105	020072	000
3853				
3854	014415	015	042012	044522
3855	014422	042526	047040	052117
3856	014430	053440	044522	042524
3857	014436	050040	047522	042524
3858	014444	052103	042105	005015
3859	014452	000012		
3860				
3861	014454	037040	031061	030060
3862	014462	000		
3863				
3864	014463	015	020012	020040
3865	014470	020040	020040	020040
3866	014476	020040	020040	052040
3867	014504	045522	041440	047105
3868	014512	042524	006522	012
3869	014517	040	020040	042510
3870	014524	042101	020040	054503
3871	014532	020114	044450	020116
3872	014540	020125	047111	044103
3873	014546	051505	006451	005012
3874	014554	000		
3875				
3876	014555	101	046050	043511
3877	014562	024516	020054	024126
3878	014570	051105	043111	024531
3879	014576	020054	051117	042440
3880	014604	054050	051105	044503
3881	014612	042523	020051	020077
3882	014620	000		
3883				
3884	014621	126	051105	043111
3885	014626	006531	005012	000

.ASCIZ / BEING CHECKED - GROUND PIN A1A0508/<15><12><12>

NODRV: .ASCIZ <15><12>/DRIVE NOT AVAILABLE/<15><12><12>

ENTERD: .ASCIZ <15><12>/ENTER DRIVE NUMBER: /

WLOCK: .ASCIZ <15><12>/DRIVE NOT WRITE PROTECTED/<15><12><12>

TOOLRG: .ASCIZ / >1200/

HEADER: .ASCII <15><12>/ TRK CENTER/<15><12>

.ASCIZ / HEAD CYL (IN U INCHES)/<15><12><12>

MODE: .ASCIZ /A(LIGN), V(ERIFY), OR E(XERCISE) ? /

VERIFY: .ASCIZ /VERIFY/<15><12><12>



3886					
3887	014633	101	044514	047107	ALIGN: .ASCIZ /ALIGN/<15><12><12>
3888	014640	005015	000012		
3889					
3890	014644	054105	051105	044503	EXER: .ASCIZ /EXERCISE/<15><12><12>
3891	014652	042523	005015	000012	
3892					
3893	014660	042510	042101	030040	HDZERO: .ASCIZ /HEAD 0 SELECTED BY DEFAULT/<15><12><12>
3894	014666	051440	046105	041505	
3895	014674	042524	020104	054502	
3896	014702	042040	043105	052501	
3897	014710	052114	005015	000012	
3898					
3899	014716	054503	044514	042116	CYL245: .ASCIZ /CYLINDER 245 SELECTED BY DEFAULT/<15><12><12>
3900	014724	051105	031040	032464	
3901	014732	051440	046105	041505	
3902	014740	042524	020104	054502	
3903	014746	042040	043105	052501	
3904	014754	052114	005015	000012	
3905					
3906	014762	044504	045523	044440	ALN245: .ASCIZ /DISK IS POSITIONED AT CYLINDER 245/<15><12><12>
3907	014770	020123	047520	044523	
3908	014776	044524	047117	042105	
3909	015004	040440	020124	054503	
3910	015012	044514	042116	051105	
3911	015020	031040	032464	005015	
3912	015026	000012			
3913					
3914	015030	005015	047105	042524	ENTHD: .ASCIZ <15><12>/ENTER HEAD: /
3915	015036	020122	042510	042101	
3916	015044	020072	000		
3917					
3918	015047	015	042412	052116	ENTCYL: .ASCIZ <15><12>/ENTER CYLINDER ADDRESS: /
3919	015054	051105	041440	046131	
3920	015062	047111	042504	020122	
3921	015070	042101	051104	051505	
3922	015076	035123	000040		
3923					
3924	015102	047111	040526	044514	BADTRK: .ASCIZ /INVALID HEAD ADDRESS/<15><12>
3925	015110	020104	042510	042101	
3926	015116	040440	042104	042522	
3927	015124	051523	005015	000	
3928					
3929	015131	111	053116	046101	BADCYL: .ASCIZ /INVALID CYLINDER ADDRESS/<15><12>
3930	015136	042111	041440	046131	
3931	015144	047111	042504	020122	
3932	015152	042101	051104	051505	
3933	015160	006523	000012		
3934					
3935	015164	005015	042012	047117	ENDMSG: .ASCIZ <15><12><12>/DONE/<15><12><12>
3936	015172	006505	005012	000	
3937					
3938					
3939	015200				.EVEN
3940					
3941	015200	046111	042514	040507	EM1: .ASCIZ /ILLEGAL RH11 INTERRUPT (SC=0 OR RHAS=0)/

## MESSAGES

3942	015206	020114	044122	030461	
3943	015214	044440	052116	051105	
3944	015222	052522	052120	024040	
3945	015230	041523	030075	047440	
3946	015236	020122	044122	051501	
3947	015244	030075	000051		
3948					
3949	015250	047125	054105	042520	EM2: .ASCIZ /UNEXPECTED ATTENTION/
3950	015256	052103	042105	040440	
3951	015264	052124	047105	044524	
3952	015272	047117	000		
3953					
3954	015275	103	047117	051124	EM3: .ASCIZ /CONTROL BUS PARITY ERROR DETECTED BY THE RH11/
3955	015302	046117	041040	051525	
3956	015310	050040	051101	052111	
3957	015316	020131	051105	047522	
3958	015324	020122	042504	042524	
3959	015332	052103	042105	041040	
3960	015340	020131	044124	020105	
3961	015346	044122	030461	000	
3962					
3963	015353	103	047117	051124	EM4: .ASCIZ /CONTROL BUS PARITY ERROR DETECTED BY THE RPO4/
3964	015360	046117	041040	051525	
3965	015366	050040	051101	052111	
3966	015374	020131	051105	047522	
3967	015402	020122	042504	042524	
3968	015410	052103	042105	041040	
3969	015416	020131	044124	020105	
3970	015424	050122	032060	000	
3971					
3972	015431	101	052124	047105	EM5: .ASCIZ /ATTENTION FROM AN OFFLINE UNIT/
3973	015436	044524	047117	043040	
3974	015444	047522	020115	047101	
3975	015452	047440	043106	044514	
3976	015460	042516	052440	044516	
3977	015466	000124			
3978					
3979	015470	042504	044526	042503	EM10: .ASCIZ /DEVICE ERROR/
3980	015476	042440	051122	051117	
3981	015504	000			
3982					
3983	015505	104	053105	041511	EM11: .ASCIZ /DEVICE UNSAFE ERROR/
3984	015512	020105	047125	040523	
3985	015520	042506	042440	051122	
3986	015526	051117	000		
3987					
3988	015531	110	040505	020104	EM12: .ASCIZ /HEAD OUT OF ALIGNMENT/
3989	015536	052517	020124	043117	
3990	015544	040440	044514	047107	
3991	015552	042515	052116	000	
3992					
3993	015557	110	040505	020104	EM13: .ASCIZ /HEAD OUT OF ALIGNMENT BY MORE THAN 1200 U INCHES/
3994	015564	052517	020124	043117	
3995	015572	040440	044514	047107	
3996	015600	042515	052116	041040	
3997	015606	020131	047515	042522	





4054	016220	020104	020040	041440		
4055	016226	046131	024040	047111		
4056	016234	052440	044440	041516		
4057	016242	042510	024523	000		
4058						
4059	016247	040	020040	042510	DH13:	.ASCIZ / HEAD CYL/
4060	016254	042101	020040	020040		
4061	016262	054503	000114			
4062						
4063						.EVEN
4064						
4065	016266	001156	005534	005536	DT2:	.WORD DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0
4066	016274	005540	005542	001160		
4067	016302	000000				
4068						
4069	016304	001156	011664	011666	DT3:	.WORD DRIVE,RD.ADR,RD.WRD,0
4070	016312	000000				
4071						
4072	016314	001156	012066	012064	DT4:	.WORD DRIVE,WRT.AD,WRT.WD,RD.WRD,0
4073	016322	011666	000000			
4074						
4075	016326	013122	013132	013134	DT10:	.WORD REG,REG+RHCS2,REG+RHDS1,REG+RHER1,REG+RHER2,REG+RHER3,0
4076	016334	013136	013162	013164		
4077	016342	000000				
4078						
4079	016344	001162	013114	001174	DT12:	.WORD SHEAD,DPB+CYL,PLUS,0
4080	016352	000000				
4081						
4082	016354	001162	013114	000000	DT13:	.WORD SHEAD,DPB+CYL,0
4083						
4084						
4085						
4086	016362	001	000	000	DF2:	.BYTE 1,0,0,0,0,0
4087	016365	000	000	000		
4088						
4089	016370	001	000	000	DF3:	.BYTE 1,0,0
4090						
4091	016373	001	000	000	DF4:	.BYTE 1,0,0,0
4092	016376	000				
4093						
4094	016377	000	000	000	DF10:	.BYTE 0,0,0,0,0,0
4095	016402	000	000	000		
4096						
4097	016405	001	001	001	DF12:	.BYTE 1,1,1
4098						
4099	016410	001	001		DF13:	.BYTE 1,1
4100						
4101						
4102		000001				.END















RHDT = 000026	2504#	2580												
RHEC1 = 000044	2511#													
RHEC2 = 000046	2512#	3334												
RHER1 = 000014	2499#	2603	3055	3124	3296	4075								
RHER2 = 000040	2509#	3125	4075											
RHER3 = 000042	2510#	3126	4075											
RHLA = 000020	2501#	2912												
RHMR = 000024	2503#													
RHOF = 000032	1576*	1577	1645*	1646	2506#	2594	2775	2779	2799	2803				
RHSN = 000030	2505#													
RHMC = 000002	2494#													
RNCP = 000101	1372#													
RPADR 005672	1422*	2479#	2539	2574*	2638	2714	2734	2755	2859	2895	2932	3169	3235	
	3241	3256	3291	3318	3346									
RPERRS 005534	2327#	2529	3031	3033	3123*	3124*	3125*	3126*	4065					
RPINIT 005710	1436	2525#	2980											
RPTMR 011240	3138#													
RPVEC 005674	1423*	2480#	2526	2546	2548	2628	3167							
RPO4 006300	1710	2627#												
RP6 = 000300	1111#													
RTC = 000117	1379#													
RO = %000000	1093#	1439*	1440*	1441	1564*	1671*	1710*	1754	1759*	1764*	1787*	1788*	1794*	
	1799*	1801*	1804*	1806*	1816*	1817*	1834*	1835	1836	1838	1841	1861*	1961	
	1965*	1969*	1974*	1978*	1984*	1986	1991*	2037	2038*	2039*	2046*	2047*	2048*	
	2049*	2050*	2051	2056	2062	2064*	2065	2079*	2110	2111*	2112	2115*	2223	
	2233*	2237	2253	2254	2267*	2286	2287*	2288	2289*	2290*	2291*	2527*	2541*	
	2549*	2568*	2579*	2589*	2593*	2596*	2602*	2613	2614*	2615*	2630*	2631	2639*	
	2645*	2655	2656*	2659*	2669*	2688*	2695*	2721*	2725*	2729*	2737*	2746*	2750*	
	2761*	2768*	2774*	2778*	2792*	2798*	2802*	2812*	2823*	2838*	2847*	2858*	2882*	
	2897*	2911*	2922	2923	2924*	2926*	2931*	2938*	2953*	3000*	3046*	3054*	3060*	
	3067*	3127*	3141*	3148*	3154*	3157*	3168*	3171*	3187*	3212*	3217	3218*	3220*	
	3236	3244	3245*	3263*	3265*	3278	3282*	3289*	3290*	3295*	3300	3301*	3306*	
	3307*	3308	3309*	3317*	3336*	3419*	3435*	3469	3470*	3531*	3550*			
R1 = %000001	1094#	1561*	1567	1570	1589*	1753	1760*	1762	1765*	1786*	1793*	1798*	1803*	
	1815*	1841*	1843	1845	1853*	1854	1962	1966*	1970*	1975*	1976*	1979*	1981*	
	1985*	1987	1990*	2009	2061	2062*	2067	2071	2073	2078*	2224	2237*	2238	
	2242	2266*	2529*	2531*	2532	2535*	2536	2538*	2543*	2544*	2552*	2569	2570	
	2572*	2601	2610*	2612*	2633*	2634	2636	2641	2649*	2650	2671	2675	2715	
	2732	2735	2753	2756	2788*	2789*	2790*	2832*	2833*	2834*	2835*	2842	2851*	
	2852*	2853	2860*	2862	2865	2871*	2872*	2873*	2875*	2896	2903*	2904	2921*	
	2933*	2944*	2946*	2947*	2948*	2952	2971*	2973	2975*	2984	2987*	2988*	2990	
	2995*	2998*	3009*	3012*	3017	3021	3023	3029	3040*	3041*	3042*	3045	3052	
	3073*	3080*	3081*	3082*	3083*	3084*	3087*	3089	3091*	3094	3099	3108	3110*	
	3111*	3112	3122	3142*	3150*	3152	3170	3177	3183*	3184*	3185*	3189	3192	
	3196*	3197*	3200*	3201	3208	3210	3347	3426*	3427	3431*	3432	3444*	3445*	
	3446*	3447*	3459	3461*	3462*	3463*	3464*	3465	3467*	3468*	3481	3483*	3484	
	3485*	3495*	3496*	3497	3498*	3499	3501*	3502*	3525	3549*				
R2 = %000002	1095#	1559*	1568*	1571*	1586	1587*	1593*	1594*	1595	1598	1600*	1601	1605	
	1752	1757*	1766*	1792*	1963	1968*	1971*	1977*	1980*	1982*	1983*	1984	1989*	
	2225	2236*	2240*	2243	2250*	2251*	2252	2257*	2265*	2530*	2532	2534*	2536	
	2631*	2632*	2633	2643*	2647	2654*	2673	2678*	2680	2712	2713	2728	2736	
	2740	2745	2757	2760	2767	2777	2801	2808	2810	2811	2827*	2844*	2864*	
	2869*	2900	2906	2949*	2951*	2963*	2972*	2976*	3065*	3076*	3086*	3102*	3116*	
	3176*	3180*	3191*	3195*	3319	3320*	3328*	3332*	3420*	3421*	3422*	3423*	3424*	
	3427*	3432*	3463	3480*	3484*	3497*	3526	3548*						
R3 = %000003	1096#	1560*	1579*	1588*	1751	1758*	1763	1767*	1791*	1810*	1813*	1819*	1890	

		1892*	1893	1896*	1897	1904*	1905	1907	1915	1917	1923	1925	1927*	1933*
		1964	1967*	1972*	1988*	2165	2174*	2180*	2181*	2184*	2189*	2190*	2191	2200*
		2226	2234*	2235*	2249*	2252*	2261*	2262*	2264*	2546*	2547*	2548*	2570*	2571*
		2573*	2583*	2713*	2716*	2718	2719	2720	2724	2740*	2741*	2743*	2744	2757*
		2758	2765	2772	2782	2784	2786	2791	2796	2806	2808*	2815*	2820	2822
		2837	2861*	2870*	2874*	2906*	2907*	2908*	2909*	2914*	2916*	2917	2919	2996*
		3007	3008*	3010	3013*	3044	3075	3143*	3144	3146*	3151	3186*	3198*	3199
		3322*	3324	3331	3333	3334	3425*	3428*	3430*	3433*	3527	3547*		
R4	=%000004	1097#	1525*	1531*	1541*	1545	1750	1756*	1768*	1781	1806	1817	2166	2168*
		2169*	2170*	2171	2172*	2186	2188*	2196*	2199*	2539*	2540*	2569*	2575	2601*
		2638*	2714*	2715*	2717*	2718*	2719*	2734*	2735*	2755*	2756*	2859*	2880*	2895*
		2896*	2932*	2952*	2966*	2984*	2985	2996	3044*	3045*	3075*	3122*	3123	3124
		3125	3126	3169*	3170*	3182*	3207	3318*	3319*	3326	3330	3345	3346*	3347*
		3348	3351*	3352	3356*	3357*	3528	3546*						
R5	=%000005	1098#	1448*	1506*	1508*	1519*	1523*	1529*	1534*	1539*	1572*	1658*	1660*	1695*
		1709	1716*	1733*	1749	1755*	1769*	1782	1803	1815	2010	2167	2173*	2175*
		2177*	2178*	2179*	2180	2198*	2227	2229*	2231*	2238*	2242*	2257	2263*	2582*
		2583	2584	2586	2599*	2607*	2608*	2811*	2816	2985*	3033*	3037*	3049*	3057*
		3063*	3070*	3078	3104*	3207*	3208	3323*	3324	3529	3545*			
R6	=%000006	1099#	1101	1405*	1406*	1407								
R7	=%000007	1100#	1102											
SAVEFG	005632	1438*	2431#	2828	2958	2979	2981*	3117						
SAVR05	013040	3524#												
SAVR15	013040	2527	2568	2630	2669	2858	2931	3141	3168	3317	3419	3525#		
SC	010260	2937	2971#	2989										
SC1	010342	2983	2987#	3004										
SC10	011072	3053	3101	3104#										
SC11	011106	3032	3100	3108#										
SC12	011170	3027	3035	3105	3122#									
SC2	010370	2962	2967	2986	2995#									
SC3	010442	3012#	3039	3085	3090	3092	3107	3121						
SC4	010436	3010#	3014											
SC5	010454	3011	3017#											
SC6	010554	3022	3040#											
SC6A	010660	3034	3063#											
SC7	011002	3066	3077	3087#	3103									
SC8	011026	3048	3056	3062	3069	3093#	3131							
SC9	011052	3051	3099#											
SEARCH=	000131	1382#												
SEC =	000010	1445*	3607#											
SEEK =	000105	1374#	1509	1524	1530	1535	1540	1661	1696					
SEEKFG	005634	1437*	2439#	2530	2686	2978	2982*							
SEKCNT	003250	1683*	1697*	1701#										
SELDRV=	000145	1385#												
SETFMT=	000143	1384#												
SET. IE	012232	2577	2697	2881	2993	3345#								
SINCNG	001172	1240#	1577	1646*										
SP =%	000006	1101#	1409*	1431	1456	1483	1499	1562*	1563*	1605*	1611*	1613*	1615*	1618
		1632*	1634*	1665	1669	1670*	1685*	1686*	1687*	1689	1691*	1693	1694	1724*
		1749*	1750*	1751*	1752*	1753*	1754*	1755	1757	1758	1762*	1763*	1764	1765
		1766	1767	1768	1769	1770*	1780*	1781*	1782*	1783*	1784*	1785*	1786	1788
		1790*	1795	1797*	1798	1800*	1801	1805*	1807*	1808	1811*	1820*	1823*	1832*
		1834	1840*	1847*	1848*	1849*	1850*	1851*	1854*	1857*	1858*	1860	1866*	1876*
		1877*	1880*	1881*	1890*	1891*	1896	1899	1903*	1910	1914*	1932	1933	1934*
		1935*	1936*	1961*	1962*	1963*	1964*	1988	1989	1990	1991	2015	2037*	2042*
		2061*	2067*	2071*	2078	2079	2110*	2111	2112*	2114	2115	2116*	2119	2121*















.SSCOP	18		
.SSIZE	18		
.SSUPR	18		
.STRAP	18	1056#	2277
.STYPB	18		
.STYPD	18	1056#	2209
.STYPE	18	1056#	2084
.STYPO	18	1056#	2131

ADC	1787	1799	1800	1804	1805	1807	1816	1975	1977	1979	1980	1982	1985		
ADD	1571	1593	1691	1736	1788	1798	1801	1803	1806	1815	1817	1820	1823	1851	1854
	1858	1974	1976	1978	1981	1983	1984	2050	2116	2160	2170	2242	2716	2717	2743
ASL	2818	2874	2916	3236	3291	3331	3464	3498							
	1847	1849	1850	1969	2047	2048	2049	2571	2832	2946	3040	3082	3445	3462	3483
ASLB	3496														
ASR	2247	2976	3013												
BCC	1594	2834	2908	2909	2948	3042	3084	3447	3458	3485	3502				
BCS	2248	2977													
BEQ	1789	1814													
	1458	1470	1486	1488	1491	1575	1580	1582	1621	1639	1642	1714	1732	1796	1809
	1837	1839	1842	1911	1916	2012	2052	2057	2063	2074	2187	2576	2585	2674	2783
	2785	2817	2821	2843	2863	2866	3032	3053	3090	3095	3101	3178	3190	3193	3251
	3299	3321	3460	3482											
BGE	1501	1602	1619	2685	2742	2915	2920	2974	3262						
BGT	1442	1812	1844	2194	2256	2635	2642	2676	2905	2999	3022	3100	3147	3153	3202
BIC	1576	1645	1881	2184	2544	2608	2875	3286							
BICB	2671														
BIS	2189	2190	2250	2251	2643	2654	2678	2827	2869	2951	2963	3065	3076	3086	3102
	3116	3180	3195	3254	3349										
BISB	1440	2039	2753												
BIT	1450	1469	1596	1620	1623	1627	1638	1641	1717	1719	1721	1731	2018	2575	3010
	3031	3242	3250	3298	3352										
BITB	3112	3208													
BLE	3024	3030	3109	3335											
BLO	1433	1906													
BLOS	1666	1894	2533	2537											
BLT	1846	2124	2195	2239	2255	2681	2918	2934	2989	3145	3281				
BMI	1546	1715	2246	2606	2687	2957	2986	3064	3079	3175	3249	3327			
BNE	1408	1451	1460	1463	1578	1585	1597	1607	1624	1628	1690	1698	1718	1720	1722
	1898	1900	1918	1926	1973	2019	2040	2066	2113	2120	2185	2244	2545	2587	2609
	2637	2648	2651	2759	2766	2773	2787	2797	2807	2854	2876	2902	2991	2997	3011
	3014	3018	3114	3139	3209	3211	3243	3279	3325	3353	3429	3434	3466	3500	
BPL	1599	1879	2023	2107	2128	2183	2230	2260	2600	2829	2959	3004	3051	3072	3118
	3216														
BR	1416	1417	1430	1435	1453	1482	1493	1498	1503	1548	1569	1583	1592	1604	1630
	1647	1664	1668	1674	1692	1727	1729	1734	1802	1856	1859	1865	1909	1920	1922
	1929	2045	2069	2076	2109	2126	2161	2176	2197	2241	2258	2542	2553	2578	2611
	2640	2644	2646	2653	2661	2679	2683	2689	2690	2692	2698	2754	2764	2771	2781
	2805	2819	2841	2936	2962	2967	2983	3020	3034	3039	3066	3077	3085	3092	3103
	3128	3131	3149	3188	3206	3222	3253	3288	3329	3355					
BVS	1852	1855													
CLC	1818														
CLR	1406	1426	1427	1439	1505	1557	1558	1591	1756	1759	1760	1790	1797	1840	1891
	1914	1968	2038	2174	2233	2236	2531	2538	2573	2632	2855	2860	2861	2878	2950
	2971	3006	3009	3142	3143	3185	3186	3204	3322	3328	3421	3422	3423	3424	3480
CLRB	1861	1927	2262	2552	2612	2657	2789	2851	2852	2871	2872	2921	2939	2944	3087
	3091	3110	3111	3155	3183	3184	3196	3197	3444						
CLV	1821														
CMP	1407	1459	1462	1485	1487	1490	1500	1577	1601	1618	1665	1689	1893	1905	2254
	2532	2536	2584	2586	2816	2853	2865	2900	2917	2919	3017	3152	3177	3192	3201
	3210	3244	3324	3334	3465	3499									
CMPB	1432	1457	1838	1843	1845	1897	1915	1917	1925	2119	2647	2680	2758	2765	2772
	2782	2784	2786	2796	2806	2820	2904	3280	3459						
COM	1590	1644	2607												
DEC	1579	1697	1811	1904	2046	2988	2998	3260	3428	3433					





SEV	1824														
SUB	1568	1733	1853	2016	2238	2741	2914	3146							
SWAB	1563	1670	2907	3255	3285	3350									
TRAP	2293	2303	2304	2305	2306	2307	2308	2309	2310						
TST	1574	1581	1584	1598	1606	1694	1713	1795	1808	1860	1899	1910	1932	2022	2073
	2114	2186	2243	2253	2288	2613	2655	2673	2684	2686	2828	2922	2923	2958	3015
	3019	3093	3117	3130	3138	3144	3151	3199	3217	3248	3252	3305	3308	3333	3354
	3469	3544													
TSTB	1441	1836	1878	2065	2106	2127	2245	2259	2634	2636	2641	2650	2675	2842	2862
	2973	2990	3021	3023	3029	3052	3078	3089	3094	3099	3108	3174	3189	3215	3326
	3481														
.ASCII	1231	1232	3759	3772	3782	3803	3813	3824	3864	4048					
.ASCIZ	1233	1940	1941	2082	3767	3791	3834	3843	3849	3854	3861	3869	3876	3884	3887
	3890	3893	3899	3906	3914	3918	3924	3929	3935	3941	3949	3954	3963	3972	3979
	3983	3988	3993	4003	4009	4018	4027	4032	4039	4053	4059				
.BLKB	1942														
.BLKW	2276	3404	3405	3406	3407	3408	3409	3410	3411						
.BYTE	1209	1210	1215	1216	1227	1228	1229	1230	1938	1939	2204	2205	2206	2207	2337
	2338	2339	2340	2341	2342	2343	2344	2351	2352	2353	2354	2355	2356	2357	2358
	2391	2397	2404	2405	2406	2407	2408	2409	2410	2411	2416	2417	2418	2419	2420
	2421	2422	2423	2463	2464	2465	2466	2467	2468	2469	2470	3361	3362	3363	3364
	3365	3366	3367	3368	3559	3560	3561	3562	3566	3568	3653	3654	3655	3656	3657
	3658	3659	3660	3661	3662	3663	3664	3665	3666	3667	3668	3669	3670	3671	3672
	3673	3674	3675	3676	3677	3678	3679	3680	3681	3682	3683	3684	3685	3686	3687
	3688	3689	3690	3691	3692	3693	3694	3695	3696	3697	3698	3699	3700	3701	3702
	3703	3704	3705	3706	3707	3708	3709	3710	3711	3712	3713	3714	3715	3716	3717
	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727	3728	3729	3730	3731	3732
	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743	3744	3745	3746	3747
	3748	3749	4086	4089	4091	4094	4097	4099							
.ENABL	1	1056													
.END	4102														
.ENDC	1061	1074	1075	1083	1169	1183	1195	1199	1204	1206	1231	1246	1250	1254	1255
	1367	1371	1395	1402	1403	1409	1410	1412	1414	1475	1479	1514	1518	1552	1556
	1650	1654	1677	1681	1704	1708	1739	1743	1945	1949	1996	2002	2011	2015	2021
	2022	2026	2027	2028	2046	2084	2085	2132	2210	2278	2287	2290	2292	2302	2303
	2304	2305	2306	2307	2308	2309	2310	2312	3554	3558	3613	3617	3647	3651	3754
	3758														
.EQUIV	1083	1084	1086	1101	1102	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140
	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168					
.EVEN	1943	2083	3751	3939	4063										
.IF	1057	1074	1075	1081	1141	1169	1190	1198	1204	1205	1231	1234	1245	1249	1254
	1366	1370	1394	1401	1403	1404	1409	1410	1412	1414	1474	1478	1513	1517	1551
	1555	1649	1653	1676	1680	1703	1707	1738	1742	1944	1948	1995	2001	2009	2014
	2018	2020	2021	2022	2025	2026	2027	2045	2061	2084	2131	2209	2277	2286	2290
	2292	2293	2303	2304	2305	2306	2307	2308	2309	2310	2312	3553	3557	3612	3616
	3646	3650	3753	3757											
.IFF	1074	1075	1081	1199	1205	1231	1246	1250	1255	1367	1371	1395	1402	1409	1475
	1479	1514	1518	1552	1556	1650	1654	1677	1681	1704	1708	1739	1743	1945	1949
	1996	2001	2015	2025	2026	2027	2028	2045	2061	2085	2132	2210	2278	2287	3554
	3558	3613	3617	3647	3651	3754	3758								
.IFT	2021														
.IFTF	2020														
.IIF	1056	1061	1066	1067	1072	1073	1074	1075	1076	1077	1190	1234	1410	1414	1612
	1614	1616	1633	1635	2002	2003	2004	2005	2027	2043	2068	2072	2131	2302	2303
	2304	2305	2306	2307	2308	2309	2310								
.IRP	1234	1403	1749	1764	1961	1988	2009	2223	2263	3653	3654	3655	3656	3657	3658



	3659	3660	3661	3662	3663	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673
	3674	3675	3676	3677	3678	3679	3680	3681	3682	3683	3684	3685	3686	3687	3688
	3689	3690	3691	3692	3693	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703
	3704	3705	3706	3707	3708	3709	3710	3711	3712	3713	3714	3715	3716	3717	3718
	3719	3720	3721	3722	3723	3724	3725	3726	3727	3728	3729	3730	3731	3732	3733
	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743	3744	3745	3746	3747	3748
	3749														
.LIST	1	1056	1183	1190	1231	1403	2292	2293	2302	2303	2304	2305	2306	2307	2308
	2309	2310	2311	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678
	3679	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693
	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708
	3709	3710	3711	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723
	3724	3725	3726	3727	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738
	3739	3740	3741	3742	3743	3744	3745	3746	3747	3748	3749	3750			
.MACRO	1	1056	1075	1198	2293	2312									
.MCALL	1056	1183													
.NLIST	1	1056	1183	1190	1231	1403	2292	2293	2302	2303	2304	2305	2306	2307	2308
	2309	2310	2311	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663
	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678
	3679	3680	3681	3682	3683	3684	3685	3686	3687	3688	3689	3690	3691	3692	3693
	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708
	3709	3710	3711	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723
	3724	3725	3726	3727	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738
	3739	3740	3741	3742	3743	3744	3745	3746	3747	3748	3749	3750			
.PAGE	1198	1254													
.REM	1														
.REPT	1190	3653	3702												
.SBTTL	1068	1079	1184	1191	1200	1247	1256	1368	1396	1476	1515	1553	1651	1678	1705
	1740	1946	1950	1997	2029	2086	2133	2211	2279	2294	2313	3555	3614	3648	3755
.TITLE	1056														
.WORD	1190	1208	1211	1212	1213	1214	1217	1218	1219	1220	1221	1222	1234	1235	1236
	1237	1238	1239	1240	1241	1242	1243	1251	1252	1449	1472	1507	1509	1520	1524
	1530	1535	1540	1573	1659	1661	1696	1701	1863	1993	1994	2054	2059	2208	2327
	2328	2329	2330	2364	2365	2366	2367	2368	2369	2370	2371	2377	2385	2431	2439
	2444	2445	2446	2447	2448	2449	2450	2451	2457	2475	2479	2480	2483	2485	2487
	2489	3238	3239	3259	3261	3293	3294	3302	3372	3373	3374	3375	3376	3377	3378
	3379	3383	3384	3385	3386	3387	3388	3389	3390	3392	3393	3394	3395	3396	3397
	3398	3399	3400	3563	3564	3570	3571	3577	3583	3584	3585	3586	3587	3588	3589
	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599	3600	3601	3602	3618	3619
	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631	3632	3633	3634
	3635	3636	3637	3639	3640	3641	3642	3643	3644	4065	4069	4072	4075	4079	4082

ERRORS DETECTED: 0  
 DEFAULT GLOBALS GENERATED: 0

\* ,DERPMB.SEQ/SOL/CRF/PAGNUM/NL:TOC/NL:MC:MD:CND=DERPMB.SML,DERPMB.P11  
 RUN-TIME: 34 47 6 SECONDS  
 RUN-TIME RATIO: 275/88=3.1  
 CORE USED: 34K (67 PAGES)

