

# RP04/5/6

## HEAD ALIGNMENT CZRJCBO

AH-9192B-MC

COPYRIGHT © 74-77

FICHE 1 OF 1

JAN 1978

**digital**

MADE IN USA



E0F1CZR000S0411

00010000 780105

IDENTIFICATION

HORICZRJCBSEQ

00010000

780105  
SEQ 0001

PRODUCT CODE: AC-9190B-MC  
PRODUCT NAME: CZRJCBO RPO4/5/6 HEAD ALIGNMENT PROGRAM  
PRODUCT DATE: DECEMBER 1977  
MAINTAINER: DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of Digital's copyright notice) only for use in such system, except as may otherwise be provided in writing by Digital.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright (C) 1974, 1977 Digital Equipment Corporation



CZRJCBO, RPO4/5/6 HEAD ALNMT MACY11 30(1046) 04-NOV-77 17:31  
 CZRJC.B.P11 04-NOV-77 12:49 TABLE OF CONTENTS

SEQ 0002

483	OPERATIONAL SWITCH SETTINGS
563	BASIC DEFINITIONS
694	RH11 REGISTERS
760	RPO4/5/6 REGISTERS
963	TRAP CATCHER
977	PROGRAM STARTING ADDRESS = 200
1040	COMMON TAGS
1197	RH11/RPO4/5/6 ADDRESS & VECTOR LOCATIONS
1215	ERROR POINTER TABLE
1342	RPO4/5/6 DRIVER COMMANDS
1390	PROGRAM START AND INITIALIZATION ROUTINES
1408	INITIALIZE THE COMMON TAGS
1520	GET VALUE FOR SOFTWARE SWITCH REGISTER
1629	SETUP TO CHECK ONLY THE SPECIFIED HEAD
1690	MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGNMENT CYLNDERS
1746	ROUTINE TO FIND THE TRACK CENTER
1875	ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT
1924	ROUTINE TO PERFORM 5,000 RANDOM SEEKS
1969	COMMON ENTRY TO THE RPO4/5/6 DRIVER
2032	SUBROUTINES
2141	MACRO ROUTINES
2143	ERROR HANDLER ROUTINE
2303	ERROR MESSAGE TIMEOUT ROUTINE
2390	TYPE ROUTINE
2504	BINARY TO OCTAL (ASCII) AND TYPE
2591	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2668	TTY INPUT ROUTINE
3096	RANDOM NUMBER GENERATOR ROUTINE
3142	INTEGER DIVIDE ROUTINE
3237	SAVE AND RESTORE R0-R5 ROUTINES
3292	TRAP DECODER
3373	TRAP TABLE
3532	SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)
5012	DATA PARAMETER BLOCK
5091	HEAD CODE TABLE
5158	OFFSET CODE TABLES
5335	MESSAGES
734	BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
8813	CK.NUM - CHECK NUMBER (OCTAL)



CONTENTS  
-----

- 1. ABSTRACT
- 2. REQUIREMENTS
  - 2.1 EQUIPMENT
  - 2.2 PRELIMINARY PROGRAMS
- 3. LOADING PROCEDURES
- 4. STARTING PROCEDURES
  - 4.1 STARTING ADDRESSES
  - 4.2 PROGRAM & OPERATOR ACTION
  - 4.3 DRIVE SELECTION
  - 4.4 RH11 - RH70 UNIBUS ADDRESSES
  - 4.5 OTHER UNIBUS ADDRESSES
- 5. SWITCH REGISTER 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
    - 8.1.1 RPO4/5 OPERATION
    - 8.1.2 RPO6 OPERATION
  - 8.2 ALIGNMENT MODE
  - 8.3 RANDOM SEEK UTILITY
- 9. PROGRAM LISTING

50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90



90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145

1. ABSTRACT

THE RPO4/5/6 HEAD ALIGNMENT VERIFICATION PROGRAM CHECKS RPO4, RPO5, OR RPO6 DISK DRIVE HEAD ALIGNMENT OR ALLOWS THE OPERATOR TO ALIGN HEADS WITH THE APPROPRIATE DRIVE'S TEST BOX. THE PROGRAM ALSO CONTAINS A UTILITY ROUTINE WHICH PERFORMS 5,000 RANDOM SEEKS WITHOUT DATA TRANSFERS. 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 EQUIPMENT

PDP-11 PROCESSOR  
8K MEMORY  
TELETYPE  
PROGRAM LOAD DEVICE  
KW11-L OR KW11-P CLOCK  
RH11 OR RH70 WITH RPO4'S, RPO5'S, OR RPO6'S  
ALIGNMENT PACK ('CE' PACK)  
'DDU' OR 'DEDU' FOR RPO4'S  
'PERCH' FOR RPO5/6'S

2.2 PRELIMINARY PROGRAMS

THE RPO4/5/6 DISK SUBSYSTEM MUST BE OPERATION AND ALL OTHER PROGRAMS IN THE SET FOR THE SUBSYSTEM MUST RUN SUCCESSFULLY.

3. LOADING PROCEDURE

THE PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR IT MAY BE LOADED FROM THE APPROPRIATE 'XXDP' MEDIA USING THE ASSOCIATED LOADER. THE PROGRAM MAY NOT BE INCLUDED IN A 'XXDP' CHAIN.

4. STARTING PROCEDURES

4.1 STARTING ADDRESSES

THE PROGRAM IS STARTED FROM LOCATION 200(8) IF THE ADDRESS OF THE RH11 OR RH70 WILL NOT BE CHANGED.

THE PROGRAM IS STARTED FROM LOCATION 204(8) IF THE ADDRESS OF THE RH11 OR RH70 IS TO BE CHANGED FROM THE PRELOADED VALUE. (SEE SECTION 4.4)



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  
199  
000  
001

## 4.2 PROGRAM &amp; OPERATION ACTION

1. LOAD THE PROGRAM INTO MEMORY (SEE SECTION 3)
2. LOAD ADDRESS 200 OR 204.
3. SET THE SWITCHES AND PRESS 'START'
4. PLACE AN ALIGNMENT DISK PACK ON THE DRIVE TO BE CHECKED AND SET THE DRIVE IN WRITE PROTECT MODE.
5. CYCLE UP THE DRIVE TO BE CHECKED. ALLOW SUFFICIENT TIME FOR THE ALIGNMENT PACK TO REACH OPERATING TEMPERATURE AND TO STABILIZE. REFER TO APPLICABLE MAINTENANCE PROCEDURES FOR THE REQUIRED 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 DRIVE TO THE ALIGNMENT CYLINDER: RPO4/5 - CYLINDER 245; RPO6 - CYLINDER 496. THE PROGRAM WILL THEN ASK FOR A HEAD ADDRESS. THE HEAD ENTERED WILL BE SELECTED AND THE PROGRAM WILL REQUEST ANOTHER HEAD. UNTIL A NEW HEAD ADDRESS IS ENTERED, THE LAST ENTERED HEAD ADDRESS WILL REMAIN SELECTED. THE ALIGNMENT SEQUENCE MAY BE TERMINATED BY TYPING A 'CONTROL C'; THE PROGRAM WILL RETURN TO THE DRIVE SELECTION ROUTINE.
9. IF THE PROGRAM IS BEING RUN IN 'VERIFICATION' MODE AND SW<02> IS SET, THE PROGRAM WILL ASK FOR HEAD AND CYLINDER ADDRESSES. (HEAD AND CYLINDER ADDRESSES ARE ENTERED IN DECIMAL.) WITH SW<02>, ONLY THE ALIGNMENT OF THE SPECIFIED HEAD WILL BE CHECKED.
10. WHEN THE OPERATOR HAS COMPLETED THE PRESENT DRIVE, CYCLE THE DRIVE DOWN, AND TRANSFER THE TEST BOX AND ALIGNMENT PACK TO THE NEXT DRIVE TO BE CHECKED. WHEN THE ALIGNMENT PACK HAS STABILIZED, THE NEW DRIVE CAN BE SELECTED. IT IS NOT NECESSARY TO RESTART THE PROGRAM.

## 4.3 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 (TEST BOX CONNECTION AND ALIGNMENT PACK STABILIZATION) MUST HAVE BEEN COMPLETED BEFORE A DRIVE IS SELECTED.

## 4.4 RH11 - RH70 UNIBUS ADDRESSES



02  
03  
04  
05  
06  
07  
08  
09  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
00

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

4.5 OTHER UNIBUS ADDRESSES

LOC	TAG	CONTENTS	FUNCTION
1144	\$TKS	177560	TTY KEYBOARD STATUS REGISTER
1146	\$TKB	177562	TTY KEYBOARD BUFFER REGISTER
1150	\$TPS	177564	TTY PRINTER STATUS REGISTER
1152	\$TPB	177566	TTY PRINTER BUFFER REGISTER
1244	PKV	104	KW11-P CLOCK VECTOR ADDRESS
1246	PKCS	172540	KW11-P CONTROL REGISTER
1250	PKB	172542	KW11-P COUNT SET REGISTER
1252	PKC	172544	KW11-P COUNTER REGISTER
1254	LKV	100	KW11-L CLOCK VECTOR ADDRESS
1256	LKS	177546	KW11-L STATUS REGISTER

5. SWITCH REGISTER SETTINGS

- SW<15>=1...HALT ON ERROR
- SW<13>=1...INHIBIT ERROR TIMEOUTS
- 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...TIMEOUT ALL TRACK CENTER VALUES

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE 'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RPO4/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNN NEW ='

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

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



2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
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

BE FOLLOWED.

6. ERRORS  
-----

6.1 TEST ERRORS

RPO4/5:

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.)

RPO6:

THE PROGRAM CHECKS THAT EACH HEAD IS WITHIN + OR - 75 MICRO-INCHES FROM THE TRACK CENTERLINE FOR CYLINDER 496 AND IS WITHIN + OR - 175 MICRO-INCHES FOR CYLINDERS 8 AND 800. 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 245(10) FOR RPO4/5'S OR 496(10) FOR RPO6'S. IF AN INVALID CYLINDER OR HEAD ADDRESS IS ENTERED, 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, ONLINE, AND IS AN RPO4, RPO5, OR RPO6. THE PROGRAM WILL REJECT ASSIGNMENTS FOR DRIVES WHICH ARE NOT PRESENT OR ARE NOT ONLINE.

6.3 SUBSYSTEM/DEVICE ERROR MESSAGES

1. 'RH11 INTERRUPT OCCURRED (RPAS=0) - AN INTERRUPT



314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360

- 2. 'UNEXPECTED ATTENTION OCCURRED' - THE INDICATED DRIVE INTERRUPTED, BUT NO INTERRUPT WAS EXPECTED FROM THE INDICATED DRIVE.
- 3. 'MASSBUS PARITY ERROR (MCPE=1)' - A CONTROL BUS PARITY ERROR WAS DETECTED BY THE RH11 WHEN THE INDICATED REGISTER WAS READ.
- 4. 'MASSBUS PARITY ERROR (PAR=1)' - A CONTROL BUS PARITY ERROR OCCURRED WHEN THE INDICATED REGISTER WAS WRITTEN.
- 5. 'ADDRESS PLUG CHANGE BIT SET' - THE PROGRAM FOUND THE 'OPE' BIT SET FOR THE INDICATED DRIVE.
- 6. 'RH11 DIDN'T RESPOND TO ADDRESSING' - THE PROGRAM ACCESSED THE RH11 AT THE INDICATED ADDRESS AND RECEIVED NO RESPONSE.
- 7. 'DRIVE OR DATA ERROR' - THE INDICATED DRIVE COMPLETED AN OPERATION WITH THE INDICATED ERROR CONDITIONS SET.
- 8. 'DRIVE UNSAFE ERROR' - THE INDICATED DRIVE SIGNALLED AN UNSAFE CONDITION.
- 9. 'HEAD OUT OF ALIGNMENT' - THE INDICATED HEAD OF NOT WITHIN TOLERANCE.
- 10. 'HEAD TOO FAR OUT OF ALIGNMENT' - THE INDICATED HEAD IS TOO FAR OUT OF ALIGNMENT FOR THE PROGRAM TO FIND THE TRACK CENTERLINE FOR THAT HEAD.
- 11. 'SOFTWARE TIMEOUT' - THE INDICATED DID NOT COMPLETE THE OPERATION WITH 1 SECOND.
- 12. 'UNCORRECTABLE MASSBUS PARITY ERROR' - THE PROGRAM ATTEMPTED TO PERFORM AN OPERATION AND DETECTED 3 SUCESSIVE MASSBUS PARITY ERRORS OR THE PROGRAM ATTEMPTED TO CLEAR A 'PAR' ERROR IN THE DRIVE AND A PARITY ERROR OCCURRED.
- 13. 'DRIVE IS UNAVAILABLE' - THE INDICATED DRIVE HAS GONE OFFLINE OR HAS BECOME NON-EXISITENT.

7. RESTRICTIONS

BECAUSE AN ALIGNMENT DISK PACK IS USED ON DRIVES TESTED BY THIS PROGRAM, NO DATA TRANSFER OPERATIONS ARE PERFORMED.

8. PROGRAM DESCRIPTION

8.1 VERIFICATION MODE

8.1.1 RPO4/5 OPERATIONS



370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425

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.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

8.1.2 RPO6 OPERATIONS

THE PROGRAM CHECKS HEAD ALIGNMENT AT CYLINDER 496, HEADS 0 - 18, AT CYLINDERS 800 AND 8, HEADS 0, 1, 10, 17, AND 18, AND RE-VERFIES ALIGNMENT AT CYLINDER 496, HEADS 0 - 18. THE OPERATOR WILL BE NOTIFIED IF A HEAD IS OUT OF ALIGNMENT BY MORE THAN THE SPECIFIED AMOUNT.

HEAD ALIGNMENT IS CHECKED IN THE FOLLOWING MANNER:

1. OFFSET THE POSITIONER TO +600 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 -600 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 - 75 MICRO-INCHES FOR CYLINDER 496 OR + OR - 175 MICRO-INCHES FOR CYLINDERS 8 AND 800.

REPEAT THE ABOVE SEQUENCE FOR ALL HEADS AT CYLINDER 496 AND FOR



1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162

HEADS 0, 1, 10, 17, 18 AT CYLINDERS 8 AND 800.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

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

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

8.3

RANDOM SEEK UTILITY

THE PROGRAM CONTAINS A UTILITY ROUTINE WHICH PERFORMS 5,000 (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 SCRATCH PACK FOR THE HEAD SHAKE DOWN.

THE OPERATOR MAY TERMINATE THE OPERATION AT ANY TIME BY TYPING A 'CONTROL C'.

9.

PROGRAM LISTING

-----



464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519

001  
  
  
  
  
  
  
  
  
  
000001  
  
  
  
  
  
  
  
  
  
001  
  
002  
  
001  
000  
  
001  
  
  
  
  
  
  
002  
  
001  
000  
001  
  
  
  
002  
  
001  
000  
001

```

.IIF NB <CZRJCBO, RPO4/5/6 HEAD ALNMT >, .TITLE CZRJCBO, RPO4/5/6 HEAD ALNMT
.IF NB <1976, 1977>
.*COPYRIGHT (C) 1976, 1977
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*
ENDC
.IIF NB <C. HESS>, ;*PROGRAM BY C. HESS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
.*
.IIF NDF SHD, SHD=0
.IIF NDF STN, SHD=SHD!1
.IIF NDF $SWR, SHD=SHD!2
.IIF NE SHD&1, STN=1
.IIF NE SHD&2, $SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
.SBTTL OPERATIONAL SWITCH SETTINGS
.*
.*      SWITCH      USE
.*      -----
.IIF NE 100000&$SWR, ;: 15      HALT ON ERROR
.IIF NE 040000&$SWR, ;: 14      LOOP ON TEST
.IIF NE 020000&$SWR, ;: 13      INHIBIT ERROR TYPE0UTS
.IF NE 010000&$SWR
.*      12      INHIBIT TRACE TRAP
.IFF
.IF NB <>
.*      12
ENDC
ENDC
.IIF NE 004000&$SWR, ;: 11      INHIBIT ITERATIONS
.IF NE 002000&$SWR
.*      10      BELL ON ERROR
.IFF
.IF NB <>
.*      10
ENDC
ENDC
.IF NE 001000&$SWR
.*      9      LOOP ON ERROR
.IFF
.IF NB <LOOP ON ERROR>
.*      9      LOOP ON ERROR
ENDC
ENDC
.IF NE 000400&$SWR
.NLIST
.IIF NDF $SWRMK, $SWRMK=0
.LIST
.IIF EQ <$SWRMK&377>, ;: 8      LOOP ON TEST IN SWR<7:0>
.IIF EQ <$SWRMK&377>-200, ;: 8      LOOP ON TEST IN SWR<6:0>
.IIF EQ <$SWRMK&377>-300, ;: 8      LOOP ON TEST IN SWR<5:0>
.IIF EQ <$SWRMK&377>-340, ;: 8      LOOP ON TEST IN SWR<4:0>

```



520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575

002  
001  
000

001100  
000

```

.IIF EQ <$$SWRMK&377>-360,;*           8           LOOP ON TEST IN SWR<3:0>
.IIF EQ <$$SWRMK&377>-370,;*           8           LOOP ON TEST IN SWR<2:0>
.IIF EQ <$$SWRMK&377>-374,;*           8           LOOP ON TEST IN SWR<1:0>
.IIF EQ <$$SWRMK&377>-376,;*           8           LOOP ON TEST IN SWR<0>
.IFF
.IF NB <>
;*           8
.ENDC
.ENDC
.MACRO .SWALO S07,S06,S05,S04,S03,S02,S01,S00
.IIF NB <S07>,*           8           SWALO
.IIF NB <S06>,*           8           SWALO
.IIF NB <S05>,*           8           SWALO
.IIF NB <S04>,*           8           SWALO
.IIF NB <S03>,*           8           SWALO
.IIF NB <S02>,*           8           SWALO
.IIF NB <S01>,*           8           SWALO
.IIF NB <S00>,*           8           SWALO
.ENDM
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <CHECK ONLY THE SPECIFIED HEAD>,;*           2           CHECK ONLY THE SPECIFIED
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <LOOP ON THE CURRENT HEAD>,;*           1           LOOP ON THE CURRENT HEAD
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <>,*           8           SWALO
.IIF NB <TYPE ALL TRACK CENTER VALUES>,;*           0           TYPE ALL TRACK CENTER VA
.SBTL BASIC DEFINITIONS

.IF NB
;*INITIAL ADDRESS OF THE STACK POINTER *** ***
STACK=
.IFF
;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100 ;
.ENDC
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

```



576	000011	HT=	11	::	CODE FOR HORIZONTAL TAB
577	000012	LF=	12	::	CODE FOR LINE FEED
578	000015	CR=	15	::	CODE FOR CARRIAGE RETURN
579	000200	CRLF=	200	::	CODE FOR CARRIAGE RETURN-LINE FEED
580	177776	PS=	177776	::	PROCESSOR STATUS WORD
581		.EQUIV	PS, PSW		
582	177774	STKLMT=	177774	::	STACK LIMIT REGISTER
583	177772	PIRQ=	177772	::	PROGRAM INTERRUPT REQUEST REGISTER
584	177570	DSWR=	177570	::	HARDWARE SWITCH REGISTER
585	177570	DDISP=	177570	::	HARDWARE DISPLAY REGISTER

::\*GENERAL PURPOSE REGISTER DEFINITIONS

588	000000	R0=	%0	::	GENERAL REGISTER
589	000001	R1=	%1	::	GENERAL REGISTER
590	000002	R2=	%2	::	GENERAL REGISTER
591	000003	R3=	%3	::	GENERAL REGISTER
592	000004	R4=	%4	::	GENERAL REGISTER
593	000005	R5=	%5	::	GENERAL REGISTER
594	000006	R6=	%6	::	GENERAL REGISTER
595	000007	R7=	%7	::	GENERAL REGISTER
596	000006	SP=	%6	::	STACK POINTER
597	000007	PC=	%7	::	PROGRAM COUNTER

::\*PRIORITY LEVEL DEFINITIONS

600	000000	PR0=	0	::	PRIORITY LEVEL 0
601	000040	PR1=	40	::	PRIORITY LEVEL 1
602	000100	PR2=	100	::	PRIORITY LEVEL 2
603	000140	PR3=	140	::	PRIORITY LEVEL 3
604	000200	PR4=	200	::	PRIORITY LEVEL 4
605	000240	PR5=	240	::	PRIORITY LEVEL 5
606	000300	PR6=	300	::	PRIORITY LEVEL 6
607	000340	PR7=	340	::	PRIORITY LEVEL 7

::\*"SWITCH REGISTER" SWITCH DEFINITIONS

610	100000	SW15=	100000		
611	040000	SW14=	40000		
612	020000	SW13=	20000		
613	010000	SW12=	10000		
614	004000	SW11=	4000		
615	002000	SW10=	2000		
616	001000	SW09=	1000		
617	000400	SW08=	400		
618	000200	SW07=	200		
619	000100	SW06=	100		
620	000040	SW05=	40		
621	000020	SW04=	20		
622	000010	SW03=	10		
623	000004	SW02=	4		
624	000002	SW01=	2		
625	000001	SW00=	1		
626		.EQUIV	SW09, SW9		
627		.EQUIV	SW08, SW8		
628		.EQUIV	SW07, SW7		
629		.EQUIV	SW06, SW6		
630		.EQUIV	SW05, SW5		
631		.EQUIV	SW04, SW4		



632  
633  
634  
635  
636 001  
637  
638  
639 100000  
640 040000  
641 020000  
642 010000  
643 004000  
644 002000  
645 001000  
646 000400  
647 000200  
648 000100  
649 000040  
650 000020  
651 000010  
652 000004  
653 000002  
654 000001  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665 000  
666 001  
667  
668  
669 000004  
670 000010  
671 000014  
672 000014  
673 000014  
674 000020  
675 000024  
676 000030  
677 000034  
678 000060  
679 000064  
680 000240  
681 000  
682 001  
683  
684  
685  
686  
687

```
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0  
.IF B (<)  
  
: *DATA BIT DEFINITIONS (BIT00 TO BIT15)  
BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
BIT01= 2  
BIT00= 1  
.EQUIV BIT09,BIT9  
.EQUIV BIT08,BIT8  
.EQUIV BIT07,BIT7  
.EQUIV BIT06,BIT6  
.EQUIV BIT05,BIT5  
.EQUIV BIT04,BIT4  
.EQUIV BIT03,BIT3  
.EQUIV BIT02,BIT2  
.EQUIV BIT01,BIT1  
.EQUIV BIT00,BIT0  
.ENDC  
.IF B (<)  
  
: *BASIC "CPU" TRAP VECTOR ADDRESSES  
ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS  
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS  
TBITVEC= 14 ;: "T" BIT  
TRTVEC= 14 ;: TRACE TRAP  
BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)  
IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**  
PWRVEC= 24 ;: POWER FAIL  
EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**  
TRAPVEC= 34 ;: "TRAP" TRAP  
TKVEC= 60 ;: TTY KEYBOARD VECTOR  
TPVEC= 64 ;: TTY PRINTER VECTOR  
PIRQVEC= 240 ;: PROGRAM INTERRUPT REQUEST VECTOR  
.ENDC  
.IF B  
;: *****  
.IFF  
.NLIST  
.REPT  
.LIST
```



```

688 ;*****
689 .NLIST
690 .ENDR
691 .LIST
692 000 .ENDC
693
694 .SBTTL RH11 REGISTERS
695
696 001 .IF B
697 ;*****
698 .IFF
699 .NLIST
700 .REPT
701 .LIST
702 ;*****
703 .NLIST
704 .ENDR
705 .LIST
706 000 .ENDC
707
708 ;CONTROL AND STATUS REGISTER 1 (RPCS1)
709
710 000100 IE= 100 ;INTERRUPT ENABLE (BIT #6)
711 000200 RDY= 200 ;READY (BIT #7)
712 000400 A16= 400 ;HIGH ORDER BUS ADDRESS BIT (BIT #8)
713 001000 A17= 1000 ;HIGH ORDER BUS ADDRESS BIT (BIT #9)
714 002000 PSEL= 2000 ;PORT SELECT (BIT #10)
715 020000 MCPE= 20000 ;MASSBUS PARITY ERROR (BIT #13)
716 040000 TRE= 40000 ;TRANSFER ERROR (BIT #14)
717 ;SC= 100000 ;SPECIAL CONDITION (BIT #15)
718
719 ;WORD COUNT REGISTER (RPWC)
720 ;(EACH BIT IS CALLED BY BIT NUMBER)
721
722 ;BUS ADDRESS REGISTER (RPBA)
723 ;(EACH BIT IS CALLED BY BIT NUMBER)
724
725 ;CONTROL AND STATUS REGISTER 2 (RPCS2)
726
727 000001 US1= 1 ;UNIT SELECT (BIT #0)
728 000002 US2= 2 ;UNIT SELECT (BIT #1)
729 000004 US4= 4 ;UNIT SELECT (BIT #2)
730 000010 BAI= 10 ;BUS ADDRESS INCREMENT INHIBIT (BIT #3)
731 000020 PAT= 20 ;MASSBUS PARITY TEST (BIT #4)
732 000040 CLR= 40 ;CLEAR (BIT #5)
733 000100 IR= 100 ;INPUT READY (BIT #6)
734 000200 OR= 200 ;OUTPUT READY (BIT #7)
735 000400 MPE= 400 ;MASS BUS PARITY ERROR (BIT #8)
736 001000 MXF= 1000 ;MISSED TRANSFER ERROR (BIT #9)
737 002000 PGE= 2000 ;PROGRAM ERROR (BIT #10)
738 004000 NEM= 4000 ;NON EXISTENT MEMORY (BIT #11)
739 010000 NED= 10000 ;NON EXISTENT DRIVE (BIT #12)
740 020000 UPE= 20000 ;UNIBUS PARITY ERROR (BIT #13)
741 040000 WCE= 40000 ;WRITE CHECK ERROR (BIT #14)
742 100000 DLT= 100000 ;DATA LATE (BIT #15)
743

```



```

744 ;DATA BUFFER REGISTER (RPDB)
745 ;(EACH BIT IS CALLED BY BIT NUMBER)
746
747
748 001 .IF B
749 ;*****
750 .IFF
751 .NLIST
752 .REPT
753 .LIST
754 ;*****
755 .NLIST
756 .ENDR
757 .LIST
758 000 .ENDC
759
760 .SBTTL RPO4/5/6 REGISTERS
761
762 001 .IF B
763 ;*****
764 .IFF
765 .NLIST
766 .REPT
767 .LIST
768 ;*****
769 .NLIST
770 .ENDR
771 .LIST
772 000 .ENDC
773
774 ;CONTROL AND STATUS 1 REGISTER. (#00)
775
776 000001 GO= 1 ;GO BIT (BIT #0)
777 000002 F1= 2 ;FUNCTION CODE BIT #1
778 000004 F2= 4 ;FUNCTION CODE BIT #2
779 000010 F3= 10 ;FUNCTION CODE BIT #3
780 000020 F4= 20 ;FUNCTION CODE BIT #4
781 000040 F5= 40 ;FUNCTION CODE BIT #5
782 004000 DVA= 4000 ;DEVICE AVAILABLE (BIT #11)
783
784 ;DRIVE STATUS REGISTER (RPDS1) (#01)
785
786 000002 :DF5= 1 DRIVE FORWARD 5"/SEC. (BIT #0)
787 000004 :DF20= 2 ;DRIVE FORWARD 20"/SEC. (BIT #1)
788 000004 DIGB= 4 ;DRIVE TO INNER GUARD BAND (BIT #2)
789 000010 GRV= 10 ;GO REVERSE (BIT #3)
790 000020 DL64= 20 ;DIFFERENCE LESS THAN 64 (BIT #4)
791 000040 DE1= 40 ;DIFFERENCE EQUALS 1 (BIT #5)
792 000100 VV= 100 ;VOLUME VALID (BIT #6)
793 000200 DRY= 200 ;DRIVE READY (BIT #7)
794 000400 DPR= 400 ;DRIVE PRESENT (BIT #8)
795 001000 PGM= 1000 ;PROGRAMABLE (BIT #9)
796 002000 LST= 2000 ;LAST SECTOR TRANSFERRED (BIT #10)
797 004000 WRL= 4000 ;WRITE LOCK (BIT #11)
798 010000 MOL= 10000 ;MEDIUM ON-LINE (BIT #12)
799 020000 PIP= 20000 ;POSITIONING OPERATION IN PROGRESS (BIT #13)

```



```

800          040000      ERR=   40000      ; COMPOSITE ERROR (BIT #14)
801          100000      ATA=   100000      ; ATTENTION ACTIVE (BIT #15)
802
803          ; ERROR REGISTER #01 (RPER1) (#02)
804
805          000001      ILF=    1          ; ILLEGAL FUNCTION (BIT #0)
806          000002      ILR=    2          ; ILLEGAL REGISTER (BIT #1)
807          000004      RMR=    4          ; REGISTER MODIFICATION REFUSED (BIT #2)
808          000010      PAR=   10          ; PARITY ERROR (BIT #3)
809          000020      FER=   20          ; FORMAT ERROR (BIT #4)
810          000040      WCF=   40          ; WRITE CLOCK FAIL (BIT #5)
811          000100      ECH=  100          ; ECC HARD ERROR (BIT #6)
812          000200      HCE=  200          ; HEADER COMPARE ERROR (BIT #7)
813          000400      HCRC=  400          ; HEADER CRC ERROR (BIT #8)
814          001000      AOE=  1000          ; ADDRESS OVERFLOW ERROR (BIT #9)
815          002000      IAE=  2000          ; INVALID ADDRESS ERROR (BIT #10)
816          004000      WLE=  4000          ; WRITE LOCK ERROR (BIT #11)
817          010000      DTE=  10000         ; DRIVE TIMING ERROR (BIT #12)
818          020000      OPT=  20000         ; OPERATION INCOMPLETE (BIT #13)
819          040000      UNS=  40000         ; DRIVE UNSAFE (BIT #14)
820          100000      DCK=  100000        ; DATA CHECK ERROR (BIT 15)
821
822          ; MAINTAINABILITY REGISTER (RPMR) (#03)
823
824          000001      DMD=    1          ; DIAGINOSTIC MODE (BIT #0)
825          000002      MCLK=   2          ; MAINTAINABILITY CLOCK (BIT #1)
826          000004      MINX=   4          ; MAINTAINABILITY INDEX (BIT #2)
827          000010      MSTCK=  10          ; MAINTAINABILITY SECTOR CLOCK (BIT #3)
828          000020      MRD=   20          ; MAINTAINABILITY READ (BIT #4)
829          000040      MWR=   40          ; MAINTAINABILITY WRITE (BIT #5)
830          000200      DTSY=  200          ; MAINTAINABILITY SYNC DETECTED (BIT #7)
831
832          ; ATTENTION SUMMARY PSEUDO-REGISTER (RPAS) (#04)
833
834          000001      AT0=    1          ; DEVICE 0 (BIT #0)
835          000002      AT1=    2          ; DEVICE 1 (BIT #1)
836          000004      AT2=    4          ; DEVICE 2 (BIT #2)
837          000010      AT3=   10          ; DEVICE 3 (BIT #3)
838          000020      AT4=   20          ; DEVICE 4 (BIT #4)
839          000040      AT5=   40          ; DEVICE 5 (BIT #5)
840          000100      AT6=  100          ; DEVICE 6 (BIT #6)
841          000200      AT7=  200          ; DEVICE 7 (BIT #7)
842
843          ; DESIRED SECTOR/TRACK ADDRESS REGISTER (RPDA) (#05)
844          ; (EACH BIT IS CALLED BY BIT NUMBER)
845
846          ; DRIVE TYPE REGISTER (RPDT) (#06)
847
848          000001      DT00=   1          ; DRIVE TYPE NUMBER BIT 1
849          000002      DT01=   2          ; DRIVE TYPE NUMBER BIT 2
850          000004      DT02=   4          ; DRIVE TYPE NUMBER BIT 3
851          000010      DT03=  10          ; DRIVE TYPE NUMBER BIT 4
852          000020      DT04=  20          ; DRIVE TYPE NUMBER BIT 5
853          000040      DT05=  40          ; DRIVE TYPE NUMBER BIT 6
854          000100      DT06=  100          ; DRIVE TYPE NUMBER BIT 7
855          000200      DT07=  200          ; DRIVE TYPE NUMBER BIT 8

```

856	000400	DT08=	400	;DRIVE TYPE NUMBER BIT 9
857	004000	DRQ=	4000	;DRIVE REQUEST REQUIRED (BIT #11)
858	020000	MOH=	20000	;MOVING HEAD (BIT #13)
859	040000	TAP=	40000	;TAPE DRIVE (BIT #14)
860	100000	NBA=	100000	;NOT BLOCK ADDRESSED (BIT #15)
861				
862				
863				
864	000001	;LOOK-AHEAD REGISTER (RPLA) (#07)		
865	000002	EXT1=	1	;EXTENSION 1 (BIT #0)
866	000004	EXT2=	2	;EXTENSION 2 (BIT #1)
867	000010	EXT4=	4	;EXTENSION 3 (BIT #2)
868	000020	EXT10=	10	;EXTENSION 4 (BIT #3)
869	000040	EXT20=	20	;EXTENSION 5 (BIT #4)
870	000100	EXT40=	40	;EXTENSION 6 (BIT #5)
871	000200	SC1=	100	;SECTOR COUNT FIELD 0 (BIT #6)
872		SC2=	200	;SECTOR COUNT FIELD 1 (BIT #7)
873	001000	SC4=	400	;SECTOR COUNT FIELD 2 (BIT #8)
874	002000	SC10=	1000	;SECTOR COUNT FIELD 3 (BIT #9)
875	004000	SC20=	2000	;SECTOR COUNT FIELD 4 (BIT #10)
876	010000	TRK1=	4000	;TRACK FIELD 1 (BIT #11)
877	020000	TRK2=	10000	;TRACK FIELD 2 (BIT #12)
878	040000	TRK4=	20000	;TRACK FIELD 3 (BIT #13)
879	100000	TRK10=	40000	;TRACK FIELD 4 (BIT #14)
880		TRK20=	100000	;TRACK FIELD 5 (BIT #15)
881				
882				
883	000001	;RPO4 ERROR REGISTER #2 (RPER2) (#10)		
884	000002	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
885	000004	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
886	000010	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
887	000020	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
888	000040	MSE=	20	;MOTOR SEQUENCE ERROR (BIT #4)
889	000100	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
890	000200	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
891	000400	FEN=	200	;FAILSAFE ENABLED (BIT #7)
892	001000	WRU=	400	;WRITE READY UNSAFE (BIT #8)
893	002000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
894	004000	NHS=	2000	;NO HEAD SELECTION (BIT #10)
895	010000	IXE=	4000	;INDEX ERROR (BIT #11)
896	020000	VU30=	10000	;30VOLT UNSAFE (BIT #12)
897	100000	PLU=	20000	;PLO UNSAFE (BIT #13)
898		ACU=	100000	;AC UNSAFE (BIT #15)
899				
900				
901	000001	;RPO5/6 ERROR REGISTER #02 (RPER2) (#10)		
902	000002	WCU=	1	;WRITE CURRENT UNSAFE (BIT #0)
903	000004	CSF=	2	;CURRENT SINK FAILURE (BIT #1)
904	000010	WSU=	4	;WRITE SELECT UNSAFE (BIT #2)
905	000020	CSU=	10	;CURRENT SWITCH UNSAFE (BIT #3)
906	000040	RAW=	20	;READ AND WRITE (BIT #4)
907	000100	TDF=	40	;TRANSITIONS DETECTOR FAILURE (BIT #5)
908	000200	TUF=	100	;TRANSITIONS UNSAFE (BIT #6)
909	000400	ABS=	200	;ABNORMAL STOP (BIT #7)
910	001000	WRU=	400	;WRITE READY UNSAFE (BIT #8)
911	002000	MHS=	1000	;MULTIPLE HEAD SELECT (BIT #9)
		NHS=	2000	;NO HEAD SELECTION (BIT #10)



```

912          004000          IXE= 4000          ;INDEX ERROR (BIT #11)
913          020000          PLU= 20000         ;PLO UNSAFE (BIT #12)
914
915          ;OFFSET REGISTER (RPOF) (#11)
916
917          000001          OF25= 1          ;OFFSET 25 MICRO INCHES (BIT #0)
918          000002          OF50= 2          ;OFFSET 50 MICRO INCHES (BIT #1)
919          000004          OF100= 4         ;OFFSET 100 MICRO INCHES (BIT #2)
920          000010          OF200= 10        ;OFFSET 200 MICRO INCHES (BIT #3)
921          000020          OF400= 20        ;OFFSET 400 MICRO INCHES (BIT #4)
922          000040          OF800= 40        ;OFFSET 800 MICRO INCHES (BIT #5)
923          000200          OFREV= 200       ;OFFSET NEGATIVE (REVERSE) (BIT #5)
924          002000          HCI= 2000      ;HEADER COMPARE INHIBIT (BIT #10)
925          004000          ECI= 4000      ;ERROR CORRECTION CODE INHIBIT (BIT #11)
926          010000          FMT22= 10000  ;FORMAT BIT (BIT #12)
927
928          ;DESIRED CYLINDER ADDRESS (RPCA) (#12)
929          ;(EACH BIT IS CALLED BY BIT NUMBER)
930
931          ;CURRENT CYLINDER ADDRESS (RPCC) (#13)
932          ;(EACH BIT IS CALLED BY BIT NUMBER)
933
934          ;SERIAL NUMBER REGISTER (RPSN) (#14)
935          ;(EACH IS CALLED BY BIT NUMBER)
936
937          ;RPO4 ERROR REGISTER #03 (RPER3) (#15)
938
939          000001          PSU= 1          ;PACK SPEED UNSAFE (BIT #0)
940          000002          VUF= 2          ;VELOCITY UNSAFE (BIT #1)
941          000010          UWR= 10         ;ANY UNSAFE EXCEPT READ/WRITE (BIT #3)
942          000040          ACL= 40        ;AC LOW (BIT #5)
943          000100          DCL= 100       ;DC LOW (BIT #6)
944          040000          SKI= 40000     ;SEEK INCOMPLETE (BIT #14)
945          100000          OCYL= 100000  ;OFF CYLINDER (BIT #15)
946
947          ;RPO5/6 ERROR REGISTER #03 (RPER3) (#15)
948
949          000001          DCU= 1          ;DC UNSAFE (BIT #0)
950          000002          WAO= 2          ;WRITE AND OFFSET (BIT #1)
951          000040          ACL= 40        ;AC LOW (BIT #5)
952          000100          DCL= 100       ;DC LOW (BIT #6)
953          020000          OPE= 20000    ;OPERATOR PLUG ERROR (BIT #13)
954          040000          SKI= 40000    ;SEEK INCOMPLETE (BIT #14)
955          100000          OCYL= 100000  ;OFF CYLINDER ERROR (BIT #15)
956
957          ;ECC POSITION REGISTER (RPEC1) (#16)
958          ;(EACH BIT IS CALLED BY BIT NUMBER)
959
960          ;ECC PATTERN REGISTER (RPEC2) (#17)
961          ;(EACH BIT IS CALLED BY BIT NUMBER)
962
963          .SBTTL TRAP CATCHER
964
965          000000          .=0
966          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
967          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS

```

```

968 ;#LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
969
970 000174 000174
971 000176 000000
972 001
973
974
975 000
976
977
978 .SBTTL PROGRAM STARTING ADDRESS = 200
979 000200 000200
980 000204 000137 001460
981 000137 001450
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023

```

```

DISPREG: .WORD 0 ;:SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;:SOFTWARE SWITCH REGISTER
.SBTTL STARTING ADDRESS(ES)
JMP 2# ;:JUMP TO STARTING ADDRESS OF PROGRAM
.ENDC
.SBTTL PROGRAM STARTING ADDRESS = 200
JMP =200 START1 ;:START THE PROGRAM, USE THE DEFAULT RH11 ADDRESS
JMP START ;:START THE PROGRAM, CHANGE THE RH11 ADDRESS

.MACRO MORETAGS
CR = 15
LF = 12
DRVSEL: .WORD 0 ;:ADDRESS OF SELECTED DRIVE
DRIVE: .WORD 0 ;:DRIVE NUM STORAGE FOR DRIVER ERR CALLS
ATTN: .WORD 0 ;:ATTENTION REGISTER STORAGE FOR ERROR MESSAGES
$HEAD: .WORD 0 ;:HEAD ADDRESS STORED FOR ERROR MESSAGES
TOPLN: .WORD 0 ;:'TYPE THE HEADER' INDICATOR
OFFDIR: .WORD 0 ;:NEG OFFSET PASS IND
BITIND: .WORD 0 ;:SIGN CHANGE BIT STORED INDICATOR
SINCNG: .WORD 0 ;:SIGN CHANGE BIT STORAGE
PLUS: .WORD 0 ;:STORE THE POS SIGN CHANGE CODE
INREG: .WORD 0 ;:CONTAINS THE VALUE READ FROM THE REGISTER
TOLER: .WORD 0 ;:ALIGNMENT TOLERANCE FOR PRESENT CYLINDER
MAXCYL: .WORD 0 ;:CONTAINS THE MAXIMUM CYLINDER ADDRESS
OF DRIVE UNDER TEST
MAXPOS: .WORD 0 ;:CONTAINS MAXIMUM POSITIVE OFFSET
MAXNEG: .WORD 0 ;:CONTAINS MAXIMUM NEGATIVE OFFSET
MAXOFF: .WORD 0 ;:CONTAINS NUMBER OF OFFSET POSITIONS
OUTTOL: .WORD 0 ;:CONTAINS TOLERANCE FOR INNER AND OUTER CYLINDERS
MIDTOL: .WORD 0 ;:CONTAINS TOLERANCE FOR THE ALIGNMENT CYLINDER
INCYL: .WORD 0 ;:INNER CYLINDER ADDRESS
MIDCYL: .WORD 0 ;:ALIGNMENT CYLINDER ADDRESS
OUTCYL: .WORD 0 ;:OUTER CYLINDER ADDRESS
TABLE1: .WORD 0 ;:CONTAINS HEAD TABLE ADDRESS FOR 'INCYL' & 'OUTCYL'
TABLE2: .WORD 0 ;:CONTAINS ADDRESS OF OFFSET CODE TABLE
SEKCNT: .WORD 0 ;:CONTAINS SEEK COUNT
CHGADR: .WORD 0 ;:CHANGE RH11 BUS ADDRESS FLAG
PKV: .WORD 104,106 ;:KW11-P VECTOR ADDRESS
PKCS: .WORD 172540 ;:KW11-P CONTROL AND STATUS REG.
PKB: .WORD 172542 ;:KW11-P COUNT SET BUFFER
PKC: .WORD 172544 ;:KW11-P COUNTER
LKV: .WORD 100,102 ;:KW11-L VECTOR ADDRESS
LKS: .WORD 177546 ;:KW11-L STATUS REGISTER

STARS
.SBTTL RH11/RP04/5/6 ADDRESS & VECTOR LOCATIONS
STARS
$RPADR: .WORD 176700 ;:RH11/RP04/5/6 UNIBUS ADDRESS

```



CZRJCBO RPO4/5/6 HEAD ALNMT  
CZRJCB.P11 04-NOV-77 12:49

MACY11 30(1046) 04-NOV-77 17:31 PAGE 21  
PROGRAM STARTING ADDRESS = 200

SEQ 0021

1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039

```

SRPVEC: .WORD 254 ;RH11/RPO4/5/6 VECTOR ADDRESS

.MACRO .ENDM MORETAGS
$REG'A: .WORD 0 ;;CONTAINS (($REGAD)+'B)
.NLIST
SCM1=SCM1+1
SCM2=SCM2+2
.LIST
.ENDM $$SCMREG
.MACRO $$SCMTMP A
$TMP'A: .WORD 0 ;;USER DEFINED
.NLIST
SCM4=SCM4+1
.LIST
.ENDM $$SCMTMP

```

```

1040          .SBTTL COMMON TAGS
1041
1042          001      .IF B
1043          ;*****
1044          .IFF
1045          .NLIST
1046          .REPT
1047          .LIST
1048          ;*****
1049          .NLIST
1050          .ENDR
1051          .LIST
1052          000      .ENDC
1053          ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
1054          ;*USED IN THE PROGRAM.
1055
1056          001      .IF NB
1057          .=
1058          .IFF      .=
1059          .=1100
1060          .ENDC
1061          001100  $CMTAG:          ;;START OF COMMON TAGS
1062          001      .IF NB
1063          .WORD      0
1064          .IFF
1065          001100  000000  $PASS: .WORD      0          ;;CONTAINS PASS COUNT
1066          .ENDC
1067          001102  000      $TSTNM: .BYTE      0          ;;CONTAINS THE TEST NUMBER
1068          001103  000      $ERFLG: .BYTE      0          ;;CONTAINS ERROR FLAG
1069          001104  000000  $ICNT:  .WORD      0          ;;CONTAINS SUBTEST ITERATION COUNT
1070          001106  000000  $LPADR: .WORD      0          ;;CONTAINS SCOPE LOOP ADDRESS
1071          001110  000000  $LPERR: .WORD      0          ;;CONTAINS SCOPE RETURN FOR ERRORS
1072          001112  000000  $ERTTL: .WORD      0          ;;CONTAINS TOTAL ERRORS DETECTED
1073          001114  000      $ITEMB: .BYTE      0          ;;CONTAINS ITEM CONTROL BYTE
1074          001115  001      $ERMAX: .BYTE      1          ;;CONTAINS MAX. ERRORS PER TEST
1075          001116  000000  $ERRPC: .WORD      0          ;;CONTAINS PC OF LAST ERROR INSTRUCTION
1076          001120  000000  $GDADR: .WORD      0          ;;CONTAINS ADDRESS OF 'GOOD' DATA
1077          001122  000000  $BDADR: .WORD      0          ;;CONTAINS ADDRESS OF 'BAD' DATA
1078          001124  000000  $GDADR: .WORD      0          ;;CONTAINS 'GOOD' DATA
1079          001126  000000  $BDADR: .WORD      0          ;;CONTAINS 'BAD' DATA
1080          001130  000000  .WORD      0          ;;RESERVED--NOT TO BE USED
1081          001132  000000  .WORD      0
1082          001134  000      $AUTOB: .BYTE      0          ;;AUTOMATIC MODE INDICATOR
1083          001135  000      $INTAG: .BYTE      0          ;;INTERRUPT MODE INDICATOR
1084          001136  000000  .WORD      0
1085          001140  177570  $SWR:  .WORD      DSWR          ;;ADDRESS OF SWITCH REGISTER
1086          001142  177570  $DISPLAY: .WORD      DDISP          ;;ADDRESS OF DISPLAY REGISTER
1087          001144  177560  $TKS:  177560          ;;TTY KBD STATUS
1088          001146  177562  $TKB:  177562          ;;TTY KBD BUFFER
1089          001150  177564  $STPS: 177564          ;;TTY PRINTER STATUS REG. ADDRESS
1090          001152  177566  $STPB: 177566          ;;TTY PRINTER BUFFER REG. ADDRESS
1091          001154  000      $NULL: .BYTE      0          ;;CONTAINS NULL CHARACTER FOR FILLS
1092          001155  002      $FILLS: .BYTE      2          ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
1093          001156  012      $FILLC: .BYTE      12         ;;INSERT FILL CHARS. AFTER A "LINE FEED"
1094          001157  000      $STPFLG: .BYTE      0          ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
1095          001      .IF B

```



```

1096 .IFF
1097 .NLIST
1098 $CM3=
1099 .LIST
1100 .ENDC
1101 .IF NE $CM3
1102 $REGAD: .WORD 0 ;;CONTAINS THE ADDRESS FROM
1103 ;;WHICH ($REGO) WAS OBTAINED
1104 .NLIST
1105 $CM1=0
1106 $CM2=0
1107 .LIST
1108 .REPT $CM3
1109 $$CMREG \ $CM1, \ $CM2
1110 .ENDR
1111 .ENDC
1112 .IF NB
1113 .NLIST
1114 $CM4=0
1115 .LIST
1116 .REPT
1117 $$CMTMP \ $CM4
1118 .ENDR
1119 .ENDC
1120 .IF NE $$SWR&4000
1121 $TIMES: 0 ;;MAX. NUMBER OF ITERATIONS
1122 .ENDC
1123 .IF NE 1000&$$SWR
1124 $ESCAPE:0 ;;ESCAPE ON ERROR ADDRESS
1125 .ENDC
1126 .IF NE 2000&$$SWR
1127 $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
1128 .ENDC
1129 $QUES: .ASCII /?/ ;;QUESTION MARK
1130 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
1131 $LF: .ASCIZ <12> ;;LINE FEED
1132 .IF NE 18., .EVEN
1133 .IF B
1134 ;:*****
1135 .IFF
1136 .NLIST
1137 .REPT
1138 .LIST
1139 ;:*****
1140 .NLIST
1141 .ENDR
1142 .LIST
1143 .ENDC
1144 .IF NB
1145 .NLIST
1146 .MCALL .SAPTBL
1147 .LIST
1148 .SAPTBL <>
1149 .ENDC
1150 .IF NB <MORETAGS>
1151 CR = 15

```

000  
001

000  
001

000  
001

000  
001

000  
001

000  
077

001160  
001161

000012  
001

000  
001

000  
001

000015

```

1152          000012          LF          =          12
1153 001164 000000          DRVSEL: .WORD          0          ; ADDRESS OF SELECTED DRIVE
1154 001166 000000          DRIVE:  .WORD          0          ; DRIVE NUM STORAGE FOR DRIVER ERR CALLS
1155 001170 000000          ATTN:  .WORD          0          ; ATTENTION REGISTER STORAGE FOR ERROR MESSAGES
1156 001172 000000          SHEAD: .WORD          0          ; HEAD ADDRESS STORED FOR ERROR MESSAGES
1157 001174 000000          TOPLN: .WORD          0          ; 'TYPE THE HEADER' INDICATOR
1158 001176 000000          OFFDIR: .WORD          0          ; NEG OFFSET PASS IND
1159 001200 000000          BITIND: .WORD          0          ; SIGN CHANGE BIT STORED INDICATOR
1160 001202 000000          SINCNG: .WORD          0          ; SIGN CHANGE BIT STORAGE
1161 001204 000000          PLUS:  .WORD          0          ; STORE THE POS SIGN CHANGE CODE
1162 001206 000000          INREG:  .WORD          0          ; CONTAINS THE VALUE READ FROM THE REGISTER
1163 001210 000000          TOLER:  .WORD          0          ; ALIGNMENT TOLERANCE FOR PRESENT CYLINDER
1164 001212 000000          MAXCYL: .WORD          0          ; CONTAINS THE MAXIMUM CYLINDER ADDRESS
1165
1166 001214 000000          MAXPOS: .WORD          0          ; OF DRIVE UNDER TEST
1167 001216 000000          MAXNEG: .WORD          0          ; CONTAINS MAXIMUM POSITIVE OFFSET
1168 001220 000000          MAXOFF: .WORD          0          ; CONTAINS MAXIMUM NEGATIVE OFFSET
1169 001222 000000          OUTTOL: .WORD          0          ; CONTAINS NUMBER OF OFFSET POSITIONS
1170 001224 000000          MIDTOL: .WORD          0          ; CONTAINS TOLERANCE FOR INNER AND OUTER CYLINDERS
1171 001226 000000          INCYL:  .WORD          0          ; CONTAINS TOLERANCE FOR THE ALIGNMENT CYLINDER
1172 001230 000000          MIDCYL: .WORD          0          ; INNER CYLINDER ADDRESS
1173 001232 000000          OUTCYL: .WORD          0          ; ALIGNMENT CYLINDER ADDRESS
1174 001234 000000          TABLE1: .WORD          0          ; OUTER CYLINDER ADDRESS
1175 001236 000000          TABLE2: .WORD          0          ; CONTAINS HEAD TABLE ADDRESS FOR 'INCYL' & 'OUTCYL'
1176 001240 000000          SEKCNT:  .WORD          0          ; CONTAINS ADDRESS OF OFFSET CODE TABLE
1177 001242 000000          CHGADR:  .WORD          0          ; CONTAINS SEEK COUNT
1178 001244 000104 000106          PKV:    .WORD          104,106          ; CHANGE RH11 BUS ADDRESS FLAG
1179 001250 172540          PKCS:   .WORD          172540          ; KW11-P VECTOR ADDRESS
1180 001252 172542          PKB:    .WORD          172542          ; KW11-P CONTROL AND STATUS REG.
1181 001254 172544          PKC:    .WORD          172544          ; KW11-P COUNT SET BUFFER
1182 001256 000100 000102          LKV:    .WORD          100,102          ; KW11-P COUNTER
1183 001262 177546          LKS:    .WORD          177546          ; KW11-L VECTOR ADDRESS
1184
1185          002          .IF B
1186          ;*****
1187          .IFF
1188          .NLIST
1189          .REPT
1190          .LIST
1191          ;*****
1192          .NLIST
1193          .ENDR
1194          .LIST
1195          .ENDC          001
1196
1197          .SBTTL  RH11/RPO4/5/6 ADDRESS & VECTOR LOCATIONS
1198
1199          002          .IF B
1200          ;*****
1201          .IFF
1202          .NLIST
1203          .REPT
1204          .LIST
1205          ;*****
1206          .NLIST
1207          .ENDR

```



CZRJCBO, RPO4/5/6 HEAD ALNMT  
CZRJCB.P11 04-NOV-77 12:49

MACY11 30(1046) 04-NOV-77 17:31 PAGE 25  
RH11/RPO4/5/6 ADDRESS & VECTOR LOCATIONS

SEQ 0025

1208  
1209 001  
1210  
1211 001264 176700  
1212 001266 000254  
1213  
1214 000

.LIST  
.ENDC  
\$RPADR: .WORD 176700 ;RH11/RPO4/5/6 UNIBUS ADDRESS  
\$RPVEC: .WORD 254 ;RH11/RPO4/5/6 VECTOR ADDRESS  
.ENDC

1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270

.SBTTL ERROR POINTER TABLE

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

```

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

001270

\$ERRTB:

;ERROR 1

001270 020506  
001272 021237  
001274 021626  
001276 021740

EM1  
DH1  
DT1  
DF1

;RH11 INTERRUPT OCCURRED (RHAS=0)

;ERROR 2

001300 020547  
001302 021244  
001304 021632  
001306 021741

EM2  
DH2  
DT2  
DF2

;UNEXPECTED ATTENTION OCCURRED

;ERROR 3

001310 020605  
001312 021321  
001314 021650  
001316 021747

EM3  
DH3  
DT3  
DF3

;MASSBUS PARITY ERROR (MCPE=1)

;ERROR 4

001320 020643  
001322 021347  
001324 021660  
001326 021752

EM4  
DH4  
DT4  
DF4

;MASSBUS PARITY ERROR (PAR=1)

;ERROR 5

001330 020700  
001332 021244  
001334 021632  
001336 021741

EM5  
DH2  
DT2  
DF2

;ADDRESS PLUG CHANGE BIT SET

;ERROR 6

001340 020734  
001342 021406

EM6  
DH6

;RH11 DIDN'T RESPOND TO ADDRESSING



1271	001344	021672	DT6	
1272	001346	021740	DF1	
1273				
1274				; ERROR 7
1275				
1276	001350	000000	0	; UNUSED
1277	001352	000000	0	
1278	001354	000000	0	
1279	001356	000000	0	
1280				
1281				; ERROR 10
1282				
1283	001360	020776	EM10	; DRIVE OR DATA ERROR
1284	001362	021421	DH10	
1285	001364	021676	DT10	
1286	001366	021756	DF10	
1287				
1288				; ERROR 11
1289				
1290	001370	021022	EM11	; DRIVE UNSAFE ERROR
1291	001372	021421	DH10	
1292	001374	021676	DT10	
1293	001376	021756	DF10	
1294				
1295				; ERROR 12
1296				
1297	001400	021045	EM12	; HEAD OUT OF ALIGNMENT
1298	001402	021507	DH12	
1299	001404	021716	DT12	
1300	001406	021765	DF12	
1301				
1302				; ERROR 13
1303				
1304	001410	021073	EM13	; HEAD TOO FAR OUT OF ALIGNMENT
1305	001412	021600	DH13	
1306	001414	021726	DT13	
1307	001416	021770	DF13	
1308				
1309				; ERROR 14
1310				
1311	001420	021131	EM14	; SOFTWARE TIMEOUT
1312	001422	021421	DH10	
1313	001424	021676	DT10	
1314	001426	021756	DF10	
1315				
1316				; ERROR 15
1317				
1318	001430	021152	EM15	; UNCORRECTABLE MASSBUS PARITY ERROR
1319	001432	000000	0	
1320	001434	000000	0	
1321	001436	000000	0	
1322				
1323				; ERROR 16
1324				
1325	001440	021215	EM16	; DRIVE IS UNAVAILABLE
1326	001442	021617	DH16	

1327 001444 021734  
1328 001446 021740

DT16  
DF1

1330 001

```

      .IF B
      ;*****
      .IFF
      .NLIST
      .REPT
      .LIST
      ;*****
      .NLIST
      .ENDR
      .LIST
      .ENDC

```

1340 000

.SBTTL RP04/5/6 DRIVER COMMANDS

1343 001

```

      .IF B
      ;*****
      .IFF
      .NLIST
      .REPT
      .LIST
      ;*****
      .NLIST
      .ENDR
      .LIST
      .ENDC

```

1354 000

1356 000101  
1357 000103  
1358 000105  
1359 000107  
1360 000111  
1361 000113  
1362 000115  
1363 000117  
1364 000121  
1365 000123  
1366 000131  
1367 000141  
1368 000143  
1369 000145  
1370 000151  
1371 000153  
1372 000161  
1373 000163  
1374 000171  
1375 000173

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

1376 001

```

      .IF B
      ;*****
      .IFF
      .NLIST
      .REPT

```

1380  
1381  
1382



```
1383 .LIST  
1384 ;:*****  
1385 .NLIST  
1386 .ENDR  
1387 .LIST  
1388 000 .ENDC  
1389  
1390 .SBTTL PROGRAM START AND INITIALIZATION ROUTINES  
1391  
1392 001 .IF B  
1393 ;:*****  
1394 .IFF  
1395 .NLIST  
1396 .REPT  
1397 .LIST  
1398 ;:*****  
1399 .NLIST  
1400 .ENDR  
1401 .LIST  
1402 000 .ENDC  
1403  
1404 001450 012737 177777 001242 START: MOV #-1,CHGADR ;SET CHANGE RH11 ADDRESS FLAG  
1405 001456 000402 BR START2 ;FINISH  
1406 001460 005037 001242 START1: CLR CHGADR ;CLEAR RH11 CHANGE ADDRESS FLAG  
1407 001464 000005 START2: RESET ;CLEAR THE BUS  
1408 .SBTTL INITIALIZE THE COMMON TAGS  
1409 001 .IF DF $CMTAG  
1410 ;:CLEAR THE COMMON TAGS ($CMTAG) AREA  
1411 001466 012706 001100 MOV #$CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED  
1412 001472 005026 CLR (R6)+ ;:CLEAR MEMORY LOCATION  
1413 001474 022706 001140 CMP #SWR,R6 ;:DONE?  
1414 001500 001374 BNE #-6 ;:LOOP BACK IF NO  
1415 000 .ENDC  
1416 001 .IF NB  
1417 MOV #,SP ;:SETUP THE STACK POINTER  
1418 .IFF  
1419 001502 012706 001100 MOV #STACK,SP ;:SETUP THE STACK POINTER  
1420 000 .ENDC  
1421 .IFF NDF $SETUP,$SETUP=0  
1422 .IFF NE $SETUP&17 ;:INITIALIZE A FEW VECTORS  
1423 001 .IF NE $SETUP&1 ;:BIT00  
1424 MOV #SCOPE,@IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE  
1425 MOV #340,@IOTVEC+2 ;:LEVEL 7  
1426 .IFF NDF $CMTAG, CLRB $TSTNM ;:INITIALIZE THE TEST NUMBER  
1427 000 .ENDC  
1428 001 .IF NE $SETUP&2 ;:BIT01  
1429 001506 012737 004412 000030 MOV #ERROR,@EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE  
1430 001514 012737 000340 000032 MOV #340,@EMTVEC+2 ;:LEVEL 7  
1431 000 .ENDC  
1432 001 .IF NE $SETUP&4 ;:BIT02  
1433 001522 012737 007452 000034 MOV #TRAP,@TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS  
1434 001530 012737 000340 000036 MOV #340,@TRAPVEC+2 ;:LEVEL 7  
1435 000 .ENDC  
1436 001 .IF NE $SETUP&10 ;:BIT03  
1437 MOV #SPWRDN,@PWRVEC ;:POWER FAILURE VECTOR  
1438 MOV #340,@PWRVEC+2 ;:LEVEL 7
```

```

1439          000          .ENDC
1440          001          .IF NE $$SETUP&20          ;;BIT04
1441          .IIF NDF $CMTAG, CLR          $PASS          ;;CLEAR THE PASS COUNT
1442          MOV          SENDCT,SEOPCT          ;;SETUP END-OF-PROGRAM COUNTER
1443          000          .ENDC
1444          001          .IF NE $$SETUP&40          ;;BIT05
1445          001536 012737 176543 007132          MOV          #176543,$SHINUM          ;;PRIME THE RANDOM NUMBER GENERATOR
1446          001544 012737 123456 007134          MOV          #123456,$LONUM          ;;BOTH HIGH AND LOW WORDS
1447          000          .ENDC
1448          001          .IF NE $$SETUP&1          ;;BIT00
1449          002          .IF NE $$SWR&4000
1450          .IIF NDF $CMTAG, MOV          #1,$ICNT          ;;INITIALIZE THE ITERATION COUNTER
1451          CLR          $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
1452          001          .ENDC
1453          000          .ENDC
1454          001          .IF NE $$SETUP&2          ;;BIT01
1455          .IIF NDF $CMTAG, CLRB          $ERFLG          ;;CLEAR THE ERROR FLAG
1456          002          .IF NE 1000&$$SWR
1457          .IIF NDF $CMTAG, CLR          $ERTTL          ;;CLEAR THE ERROR COUNT
1458          CLR          $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1459          MOVB          #1,$ERMAX          ;;ALLOW ONE ERROR PER TEST
1460          001          .ENDC
1461          000          .ENDC
1462          001          .IF NE $$SETUP&20          ;;BIT04
1463          002          .IF NE 10000&$$SWR
1464          .MACRO $$SETUP ?N1,?N2,?N3
1465          ;;INITIALIZE THE "T-BIT" TRAP VECTOR. THEN LOAD LOCATION "$RTRN", IN
1466          ;;THE "END-OF-PASS" (SEOP) ROUTINE, WITH A "RTI" OR "RTT".
1467          MOV          #$RTRN,@TBITVEC          ;;SET "T" BIT VECTOR TO $RTRN
1468          MOV          #340,@TBITVEC+2          ;;LEVEL 7
1469          MOV          #RTI,$RTRN          ;;SET $RTRN TO A RTI
1470          MOV          #N2,@RESVEC          ;;TRY TO DO A RTT
1471          CLR          -(SP)          ;;DUMMY PS
1472          MOV          #N1,-(SP)          ;;AND PC
1473          RTT          ;;TRY THE RTT
1474          N1:          MOV          #RTT,$RTRN          ;;RTT IS LEGAL--SET $RTRN TO A RTT
1475          BR          N3
1476          N2:          ADD          #10,SP          ;;RTT ILLEGAL--CLEAN OFF THE STACK
1477          N3:          MOV          #RESVEC+2,@RESVEC          ;;RESTORE TRAP CATCHER
1478          CLR          $TBIT          ;;CLEAR "T" BIT SWITCH
1479          .ENDM          $$SETUP
1480          $$SETUP
1481          001          .ENDC
1482          000          .ENDC
1483          001          .IF NE $$SETUP&1
1484          .IIF NE $$SWR&40000, MOV          #,$LPADR          ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1485          .IIF NE 1000&$$SWR, MOV          #,$LPERR          ;;SETUP THE ERROR LOOP ADDRESS
1486          000          .ENDC
1487          ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1488          ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
1489          001552 013746 000004          MOV          @ERRVEC,-(SP)          ;;SAVE ERROR VECTOR
1490          001556 012737 001612 000004          MOV          #64,$ERRVEC          ;;SET UP ERROR VECTOR
1491          001564 012737 177570 001140          MOV          #DSWR,$SWR          ;;SETUP FOR A HARDWARE SWICH REGISTER
1492          001572 012737 177570 001142          MOV          #DDISP,$DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
1493          001600 022777 177777          CMP          #-1,$SWR          ;;TRY TO REFERENCE HARDWARE SWR
1494          001606 001012          BNE          66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED

```



```

1495                                     ;; AND THE HARDWARE SWR IS NOT = -1
1496 001610 000403                                     ;; BRANCH IF NO TIMEOUT
1497 001612 012716 001620 64$: MOV #65$, (SP) ;; SET UP FOR TRAP RETURN
1498 001616 000002                                     ;;
1499 001620 012737 000176 001140 65$: MOV #SWREG, SWR ;; POINT TO SOFTWARE SWR
1500 001626 012737 000174 001142 66$: MOV #DISPRÉG, DISPLAY ;;
1501 001634 012637 000004 66$: MOV (SP)+, @#ERRVEC ;; RESTORE ERROR VECTOR
1502
1503 001 .IF DF $MAIL
1504 .MACRO $$SETMAIL ?$ARG1
1505 CLR $PASS ;; CLEAR PASS COUNT
1506 BITB #APTSIZE, $ENVM ;; TEST USER SIZE UNDER APT
1507 BEQ $ARG1 ;; YES, USE NON-APT SWITCH
1508 MOV #$$SWREG, SWR ;; NO, USE APT SWITCH REGISTER
1509 $ARG1:
1510 .ENDM $$SETMAIL
1511 .ENDC
1512 000
1513 001640 005227 177777 INC #-1 ;; FIRST START ?
1514 001644 001006 BNE 1$ ;; BR IF NOT
1515 001646 104401 016546 TYPE ,TITLE ;; TYPE PROGRAM TITLE
1516 001652 104401 016644 TYPE 'DDU' ;; TYPE 'DDU' SETUP MESSAGE
1517 001656 104401 001161 TYPE, $CRLF ;; CR-LF
1518 001662 004737 005604 1$: JSR PC, $TKINT ;; TURN ON THE TTY KEYBOARD INTERRUPT
1519 001
1520 .IF NE $SETUP&100
1521 001666 005737 000042 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1522 001672 001006 TST @#42 ;; ARE WE RUNNING UNDER XXDP/ACT?
1523 002 BNE 67$ ;; BRANCH IF YES
1524 .IF DF $MAIL
1525 CMPB $ENV, #1 ;; ARE WE RUNNING UNDER APT?
1526 BEQ 67$ ;; BRANCH IF YES
1527 .ENDC
1528 001674 023727 001140 000176 CMP SWR, #SWREG ;; SOFTWARE SWITCH REG SELECTED?
1529 001702 001005 BNE 68$ ;; BRANCH IF NO
1530 001704 104406 GTSWR ;; GET SOFT-SWR SETTINGS
1531 001710 000403 BR 68$
1532 001716 112737 000001 001134 67$: MOV #1, $AUTOB ;; SET AUTO-MODE INDICATOR
1533 68$:
1534 .ENDC
1535 001716 005227 177777 INC #-1 ;; FIRST START ?
1536 001722 001010 BNE INIT ;; BR IF NOT
1537 001724 004737 022000 JSR PC, BUSADR ;; CHECK THE RH11 ADDRESS
1538 001730 013737 001264 007704 MOV $RPADR, RPADR ;; MOVE RH11 ADDRESS TO DRIVER
1539 001736 013737 001266 007706 MOV $RPVEC, RPVEC ;; VECTOR ADDRESS
1540
1541 ;GET THE DRIVE NUMBER FROM THE OPERATOR AND SETUP THE VARIABLE LOCATION
1542 ;FOR RPO4/5 OR RPO6'S
1543
1544 001744 012716 001100 INIT: MOV #STACK, (SP) ;; SETUP THE STACK ADDRESS
1545 001750 005037 177776 CLR PS ;; SET PROCESSOR PRIORITY TO ZERO
1546 001754 005037 001174 CLR TOPLN ;; CLEAR HEADER INDICATOR
1547 001760 004737 005604 JSR PC, $TKINT ;; INITIALIZE THE TTY KEYBOARD
1548 001764 104401 017545 TYPE ENTERD ;; ASK FOR DRIVE ASSIGNMENT
1549 001770 004737 004270 JSR PC, GETNUM ;; GET DRIVE NUMBER
1550 001774 000410 BR 1$ ;; 'CR' ENTERED

```



1551	001776	011637	001164			MOV	(SP) DRVSEL	: DRIVE NUMBER
1552	002002	112637	016120			MOV	(SP)+, DPB	: DRIVE NUMBER TO PAR BLOCK
1553	002006	023727	001164	000010		CMP	DRVSEL, #8.	: CHECK DRIVE NUMBER
1554	002014	103403				BLO	2\$	: BR IF DRIVE NUMBER OK
1555	002016	104401	001160		1\$:	TYPE	\$QUES	: TYPE QUESTION MARK
1556	002022	000750				BR	INIT	: TRY AGAIN
1557	002024	004737	004114		2\$:	JSR	PC, ST, CLK	: INITIALIZE THE CLOCK
1558	002030	004737	007722			JSR	PC, RPINIT	: INITIALIZE THE DRIVER
1559	002034	012737	177777	007646		MOV	#-1, SEEKFG	: SET SEEK ALLOW FLAG
1560	002042	012737	177777	007644		MOV	#-1, SAVEFG	: SAVE DRIVE REGISTERS
1561	002050	113700	016120			MOV	DPB, RO	: USE DRIVE NUMBER AS AN INDEX
1562	002054	105760	007556			TSTB	DRVSTA(RO)	: SEE IF DRIVE AVAILABLE
1563	002060	003004				BGT	3\$	: BRANCH IF IT IS
1564	002062	104401	017514			TYPE	NODRV	: 'DRIVE NOT AVAILABLE'
1565	002066	000137	001744			JMP	INIT	: TRY IT AGAIN
1566	002072	132760	000003	007566	3\$:	BITB	#BIT00!BIT01, DRV TYP(RO)	: RPO4/5 ?
1567	002100	001442				BEG	4\$	: BR IF NOT
1568	002102	012737	000536	001222		MOV	#350, OUTTOL	: SETUP 4 & 400 TOLERANCES
1569	002110	012737	000113	001224		MOV	#75, MIDTOL	: TOLERANCE AT CYLINDER 245
1570	002116	012737	000620	001226		MOV	#400, INCYL	: INNER ALIGNMENT CYLINDER
1571	002124	012737	000004	001232		MOV	#4, OUTCYL	: OUTER ALIGNMENT CYLINDER
1572	002132	012737	000365	001230		MOV	#245, MIDCYL	: ALIGNMENT CYLINDER
1573	002140	012737	016310	001234		MOV	#HEAD45, TABLE1	: HEAD ADDRESS TABLE
1574	002146	012737	016405	001236		MOV	#OFTBL4, TABLE2	: OFFSET TABLE ADDRESS
1575	002154	012737	000633	001212		MOV	#411, MAXCYL	: LAST CYLINDER + 1
1576	002162	012737	175520	001216		MOV	#-1200, MAXNEG	: MAXIMUM NEGATIVE OFFSET VALUE
1577	002170	012737	002260	001214		MOV	#1200, MAXPOS	: MAXIMUM POSITIVE OFFSET VALUE
1578	002176	012737	000141	001220		MOV	#97, MAXOFF	: NUMBER OF OFFSET POSITIONS
1579	002204	000441				BR	5\$	: CONTINUE WITH THE SETUP
1580	002206	012737	000257	001222	4\$:	MOV	#175, OUTTOL	: TOLERANCE AT CYLINDERS 8 & 800
1581	002214	012737	000113	001224		MOV	#75, MIDTOL	: TOLERANCE AT ALIGNMENT CYLINDER
1582	002222	012737	001440	001226		MOV	#800, INCYL	: INNER ALIGNMENT CYLINDER
1583	002230	012737	000010	001232		MOV	#8, OUTCYL	: OUTER ALIGNMENT CYLINDER
1584	002236	012737	000760	001230		MOV	#496, MIDCYL	: ALIGNMENT CYLINDER
1585	002244	012737	016260	001234		MOV	#HEAD6, TABLE1	: HEAD ADDRESS TABLE
1586	002252	012737	016324	001236		MOV	#OFTBL6, TABLE2	: OFFSET CODE TABLE
1587	002260	012737	001457	001212		MOV	#815, MAXCYL	: MAXIMUM CYLINDER ADDRESS + 1
1588	002266	012737	176650	001216		MOV	#-600, MAXNEG	: MAXIMUM NEGATIVE OFFSET VALUE
1589	002274	012737	001130	001214		MOV	#600, MAXPOS	: MAXIMUM POSITIVE OFFSET VALUE
1590	002302	012737	000061	001220		MOV	#49, MAXOFF	: NUMBER OF OFFSET POSITIONS
1591	002310	004537	003736		5\$:	JSR	RS, DRIVER	: RECALIBRATE
1592	002314	000107				.WORD	RECAL	: DRIVER CALL
1593	002316	032737	004000	016152		BIT	#BIT11, REG+RPDS1	: SEE IF WRITE LOCK SET
1594	002324	001003				BNE	6\$	: BR IF WRITE LOCKED
1595	002326	104401	017574			TYPE	WLOCK	: REPORT NOT WRITE LOCKED
1596	002332	000604				BR	INIT	: TRY IT AGAIN
1597	002334	104401	017751		6\$:	TYPE	,MODE	: SEE IF ALIGN, VERIFY, OR EXERCISE
1598	002340	104410				RDCHR		: GET THE ENTRY
1599	002342	012637	002442			MOV	(SP)+, 10\$	: SAVE THE ENTRY
1600	002346	122737	000126	002442		CMPB	#'V', 10\$	: WAS A 'V' ENTERED ?
1601	002354	001422				BEG	9\$	: BR IF IT WAS
1602	002356	022737	000101	002442		CMP	#'A', 10\$	: WAS AN 'A' ENTERED ?
1603	002364	001002				BNE	7\$	: BR IF NOT
1604	002366	000137	003530			JMP	DDUALN	: HEADS TO BE ALIGNED WITH DDU
1605	002372	022737	000105	002442	7\$:	CMP	#'E', 10\$	: SEE IF EXERCISE
1606	002400	001004				BNE	8\$	: BR IF 'E' NOT ENTERED



```

1607 002402 000137 003640          JMP      RANDOM      ;GO TO EXERCISE ROUTINE
1608 002406 104401 002442          TYPE    '10$        ;ECHO THE INVALID CHARACTER
1609 002412 104401 001160          8$:    TYPE    '8$        ;TYPE A QUESTION MARK
1610 002416 000137 001744          JMP      INIT        ;GO ALL THE WAY BACK TO DRIVE ENTRY
1611 002422 104401 020015          9$:    TYPE    '9$        ;TYPE 'VERIFY'
1612 002426 032777 000004          BIT     #SW2,DSWR    ;SWITCH 2 SET ?
1613 002434 001504          BEQ     CYLDER       ;BR IF NOT SET - CHECK ALL HEADS
1614 002436 000137 002444          JMP     ONEHD        ;CHECK ONLY THE SPECIFIED HEAD
1615 002442 000000          10$:   .WORD    0      ;OPERATOR ENTRY GOES HERE
1616
1617          001          .IF B
1618          ;*****
1619          .IFF
1620          .NLIST
1621          .REPT
1622          .LIST
1623          ;*****
1624          .NLIST
1625          .ENDR
1626          .LIST
1627          000          .ENDC
1628
1629          .SBTTL  SETUP TO CHECK ONLY THE SPECIFIED HEAD
1630
1631          001          .IF B
1632          ;*****
1633          .IFF
1634          .NLIST
1635          .REPT
1636          .LIST
1637          ;*****
1638          .NLIST
1639          .ENDR
1640          .LIST
1641          000          .ENDC
1642
1643 002444 104401 020355          ONEHD: TYPE    ENTCYL      ;ASK FOR CYLINDER ADDRESS
1644 002450 004737 004270          JSR     PC,GETNUM    ;GET THE CYLINDER ADDRESS
1645 002454 000427          BR     1$           ;'CR' ENTERED
1646 002456 012637 016132          MOV     (SP)+,DPB+CYL ;MOVE IT TO PARAMETER BLOCK
1647 002462 013737 001222 001210          MOV     OUTTOL,TOLER ;SET INITIAL ALIGNMENT TOLERANCE
1648 002470 023737 001232 016132          CMP     OUTCYL,DPB+CYL ;SEE IF OUTER CYLINDER
1649 002476 001432          BEQ     4$           ;BR IF IT IS
1650 002500 023737 001226 016132          CMP     INCYL,DPB+CYL ;SEE IF INNER CYLINDER
1651 002506 001426          BEQ     4$           ;BR IF IT IS
1652 002510 013737 001224 001210          MOV     MIDTOL,TOLER ;CHANGE TOLERANCE
1653 002516 023737 001230 016132          CMP     MIDCYL,DPB+CYL ;SEE IF ALIGNMENT CYLINDER
1654 002524 001414          BEQ     3$           ;BR IF YES
1655 002526 104401 020437          TYPE    ,BADCYL     ;REPORT INVALID CYLINDER
1656 002532 000744          BR     ONEHD        ;TRY IT AGAIN
1657 002534 023727 001212 000633 1$:    CMP     MAXCYL,#411. ;RPO4/5 ?
1658 002542 001403          BEQ     2$           ;BR IF YES
1659 002544 104401 020156          TYPE    ,CYL496     ;CYLINDER 496 DEFAULT MESSAGE
1660 002550 000402          BR     3$           ;CONTINUE
1661 002552 104401 020112 2$:    TYPE    ,CYL245     ;CYLINDER 245 DEFAULT MESSAGE
1662 002556 013737 001230 016132 3$:    MOV     MIDCYL,DPB+CYL ;CYLINDER ADDRESS

```

1663 002564 104401 020336  
 1664 002570 004737 004270  
 1665 002574 000411  
 1666 002576 012637 001172  
 1667 002602 022737 000022 001172  
 1668 002610 002007  
 1669 002612 104401 020410  
 1670 002616 000762  
 1671 002620 104401 020054  
 1672 002624 005037 001172  
 1673 002630 004537 003736  
 1674 002634 000105  
 1675 002636 004737 003032  
 1676 002642 000137 001744

4\$: TYPE ENTHD ;ASK FOR THE HEAD ADDRESS  
 JSR PC,GETNUM ;GET THE HEAD ADDRESS  
 BR 5\$ ;'CR' ENTERED  
 MOV (SP)+,\$HEAD ;SAVE THE HEAD ADDRESS  
 CMP #18.,\$HEAD ;ADDRESS VALID ?  
 BGE 6\$ ;BR IF YES  
 TYPE BADTRK ;INVALID HEAD ADDRESS MESSAGE  
 BR 4\$ ;TRY AGAIN  
 5\$: TYPE HDZERO ;HEAD DEFAULT MESSAGE  
 CLR \$HEAD ;CLEAR HEAD STORAGE  
 6\$: JSR R5,DRIVER ;SEEK TO THE SPECIFIED CYLINDER  
 .WORD SEEK ;SEEK CODE  
 JSR PC,CHECK ;CHECK THE HEAD  
 JMP INIT ;GET THE NEXT DRIVE/HEAD

1677  
 1678 001  
 1679  
 1680  
 1681  
 1682  
 1683  
 1684  
 1685  
 1686  
 1687  
 1688 000  
 1689  
 1690  
 1691  
 1692 001  
 1693  
 1694  
 1695  
 1696  
 1697  
 1698  
 1699  
 1700  
 1701  
 1702 000  
 1703

.IF B  
 ;\*\*\*\*\*  
 .IFF  
 .NLIST  
 .REPT  
 .LIST  
 ;\*\*\*\*\*  
 .NLIST  
 .ENDR  
 .LIST  
 .ENDC

.SBTTL MAIN ROUTINE - CHECK ALL HEADS AT ALL ALIGNMENT CYLNDERS

.IF B  
 ;\*\*\*\*\*  
 .IFF  
 .NLIST  
 .REPT  
 .LIST  
 ;\*\*\*\*\*  
 .NLIST  
 .ENDR  
 .LIST  
 .ENDC

1704 002646 013737 001230 016132  
 1705 002654 013737 001224 001210  
 1706 002662 004537 003736  
 1707 002666 000105  
 1708 002670 012704 016210  
 1709 002674 004737 003014  
 1710 002700 013737 001226 016132  
 1711 002706 013737 001222 001210  
 1712 002714 004537 003736  
 1713 002720 000105  
 1714 002722 013704 001234  
 1715 002726 004737 003014  
 1716 002732 013737 001232 016132  
 1717 002740 004537 003736  
 1718 002744 000105

CYLDER: MOV MIDCYL,DPB+CYL ;START AT THE ALIGNMENT CYLINDER  
 MOV MIDTOL,TOLER ;ALIGNMENT CHECK TOLERANCE  
 JSR R5,DRIVER ;SEEK TO THE ALIGNMENT CYLINDER  
 .WORD SEEK ;DRIVER SEEK CODE  
 MOV #HEAD1,R4 ;ADDRESS OF ALIGNMENT HEAD SELECTION TABLE  
 JSR PC,1\$ ;CHECK ALIGNMENT, ALL HEADS, AT THE ALIGNMENT CYLINDER  
 MOV INCYL,DPB+CYL ;CHANGE TO INNER CYLINDER  
 MOV OUTTOL,TOLER ;CHANGE TOLERANCE VALUE  
 JSR R5,DRIVER ;SEEK TO INNER CYLINDER  
 .WORD SEEK ;DRIVER SEEK CODE  
 MOV TABLE1,R4 ;CHANGE HEAD TABLE ADDRESS  
 JSR PC,1\$ ;CHECK ALIGNMENT AT INNER CYLINDER  
 MOV OUTCYL,DPB+CYL ;CHANGE CYLINDER TO OUTER CYLINDER  
 JSR R5,DRIVER ;SEEK TO OUTER CYLINDER  
 .WORD SEEK ;DRIVER SEEK CODE



1719	002746	004737	003014		JSR	PC,1\$	:CHECK ALIGNMENT AT OUTER CYLINDER	
1720	002752	013737	001230	016132	MOV	MIDCYL,DPB+CYL	:GO BACK TO THE ALIGNMENT CYLINDER	
1721	002760	013737	001224	001210	MOV	MIDTOL,TOLER	:CHANGE ALIGNMENT TOLERANCE	
1722	002766	004537	003736		JSR	R5,DRIVER	:SEEK	
1723	002772	000105			.WORD	SEEK	:DRIVER SEEK CODE	
1724	002774	012704	016210		MOV	#HEAD1,R4	:ADDRESS OF 0 - 18 HEAD ADDRESS TABLE	
1725	003000	004737	003014		JSR	PC,1\$	:RECHECK ALIGNMENT AT ALIGNMENT CYLINDER	
1726	003004	104401	020472		TYPE	ENDMSG	:TYPE 'DONE'	
1727	003010	000137	001744		JMP	INIT	:GET NEXT DRIVE OR HEAD	
1728	003014	012437	001172		1\$:	MOV	(R4)+,\$HEAD	:MOVE HEAD ADDRESS FROM TABLE
1729	003020	100403			BMI	2\$	:BR IF END OF TABLE	
1730	003022	004737	003032		JSR	PC,CHECK	:CHECK THE ALIGNMENT	
1731	003026	000772			BR	1\$	:GET NEXT HEAD VALUE	
1732	003030	000207			2\$:	RTS	PC	:RETURN
1733								
1734		001			.IF B			
1735					;;*****			
1736					.IFF			
1737					.NLIST			
1738					.REPT			
1739					.LIST			
1740					;;*****			
1741					.NLIST			
1742					.ENDR			
1743					.LIST			
1744		000			.ENDC			
1745								
1746					.SBTTL	ROUTINE TO FIND THE TRACK CENTER		
1747								
1748		001			.IF B			
1749					;;*****			
1750					.IFF			
1751					.NLIST			
1752					.REPT			
1753					.LIST			
1754					;;*****			
1755					.NLIST			
1756					.ENDR			
1757					.LIST			
1758		000			.ENDC			
1759								
1760	003032	005037	001176		CHECK:	CLR	OFFDIR	:CLEAR THE OFFSET DIRECTION INDICATOR
1761	003036	005037	001200			CLR	BITIND	:CLEAR THE SIGN BIT INDICATOR
1762	003042	013702	001214			MOV	MAXPOS,R2	:MAXIMUM POSITIVE OFFSET VALUE
1763	003046	013703	001220			MOV	MAXOFF,R3	:MAXIMUM NUMBER OF OFFSETS
1764	003052	013701	001236			MOV	TABLE2,R1	:OFFSET CODE TABLE ADDRESS
1765	003056	013746	001172			MOV	\$HEAD,-(SP)	:HEAD VALUE TO STACK
1766	003062	000316				SWAB	(SP)	:SWITCH FOR RPDA LOAD
1767	003064	004037	015012			JSR	RO,WRT.RP	:LOAD RPDA WITH HEAD ADDRESS
1768	003070	000006				RPDA		:REGISTER ADDRESS
1769	003072	001744				INIT		:UNCORRECTABLE PARITY ERROR RETURN
1770	003074	112137	016121		1\$:	MOVB	(R1)+,DPB+CODE	:OFFSET VALUE TO PARAMETER BLOCK
1771	003100	162702	000031			SUB	#25.,R2	:DECREMENT OFFSET VALUE
1772	003104	000404				BR	3\$	:BYPASS REVERSE DIRECTION SETUP
1773	003106	114137	016121		2\$:	MOVB	-(R1),DPB+CODE	:REVERSE OFFSET VALUE TO PAR BLOCK
1774	003112	062702	000031			ADD	#25.,R2	:INCREMENT OFFSET VALUE

```

1775 003116 004537 003736      3$: JSR      R5,DRIVER      ;OFFSET THE DRIVE
1776 003122 000115                .WORD    OFFSET          ;OFFSET OP CODE
1777 003124 005737 001200      TST      BITIND         ;INITIAL SIGN VALUE STORED ?
1778 003130 001011                BNE      4$             ;BR IF STORED
1779 003132 005137 001200      COM      BITIND         ;SET THE SIGN BIT INDICATOR
1780 003136 042737 077777 016172  BIC      #1CBIT15,REG+RPOF ;LEAVE THE SIGN BIT
1781 003144 013737 016172 001202  MOV      REG+RPOF,SINCNG ;STORE THE SIGN CHANGE BIT
1782 003152 000407                BR       5$
1783 003154 042737 077777 016172 4$: BIC      #1CBIT15,REG+RPOF ;KEEP ONLY THE SIGN VALUE
1784 003162 023737 016172 001202  CMP      REG+RPOF,SINCNG ;SIGN CHANGED ?
1785 003170 001044                BNE      8$             ;BRANCH IF IT DID
1786 003172 005303                DEC      R3             ;DECREMENT THE OFFSET ATTEMPT COUNTER
1787 003174 001036                BNE      7$             ;BRANCH IF OFFSETS NOT COMPLETED
1788 003176 032777 000001 175734  BIT      #SW0,2SWR      ;CHECK SWRO
1789 003204 001002                BNE      6$             ;BR IF SET
1790 003206 104013                ERROR    13             ;REPORT OUT OF LIMIT CONDITION
1791 003210 000534                BR       13$           ;SEE IF LOOP ON ERROR
1792 003212 005737 001174      6$: TST      TOPLN        ;TYPE THE HEADING LINE ?
1793 003216 001005                BNE      17$           ;BR IF NOT
1794 003220 104401 017657      TYPE     HEADER        ;TYPE THE HEADER
1795 003224 012737 177777 001174  MOV      4-1,TOPLN     ;SET THE BYPASS SWITCH
1796 003232
1797 003232 013746 001172      17$: MOV      $HEAD,-(SP)   ;;SAVE $HEAD FOR TYPEOUT
1798
1799 003236 104405      .IIF NB (<)>
1800 003240 013746 016132      TYPDS
1801      MOV      DPB+CYL,-(SP) ;;SAVE DPB+CYL FOR TYPEOUT
1802      .IIF NB (<)>
1803      TYPDS
1804      TYPE     ,TOLRG     ;;GO TYPE--DECIMAL ASCII WITH SIGN
1805      TYPE     ,$CRLF    ;HEAD OUT OF LIMIT
1806      BIT      #SW15,2SWR ;CR-LF
1807      BEQ      13$       ;HALT IF ERROR ?
1808      HALT
1809      BR       13$       ;BR IF NOT
1810      BEQ      1$         ;HEAD JUST TYPED OUT OF TOLERANCE
1811      BR       2$         ;CHECK FOR LOOP
1812      BR       2$         ;SEE WHICH DIRECTION
1813      TST      OFFDIR    ;BRANCH IF FORWARD
1814      BNE      9$         ;REVERSE DIRECTION
1815      MOV      R2,PLUS    ;FINISHED WITH REVERSE ?
1816      MOV      MAXNEG,R2 ;BRANCH IF YES
1817      MOV      MAXOFF,R3 ;SAVE THE SIGN CHANGE VALUE
1818      ADD      TABLE2,R1 ;MAXIMUM NEGATIVE OFFSET VALUE
1819      CLR      BITIND     ;NUMBER OF OFFSETS
1820      BR       2$         ;NUMBER OF OFFSETS
1821      BR       2$         ;BEGINNING OF REVERSE DIRECTION VALUES
1822      ADD      PLUS,R2    ;SET REVERSE DIRECTION INDICATOR
1823      ASR      R2         ;RESET SIGN BIT INDICATOR
1824      MOV      R2,PLUS    ;START REVERSE OFFSETS
1825      TST      R2         ;ADD THE TWO SIGN CHANGE VALUES
1826      BPL      10$        ;'DIVIDE' BY 2
1827      NEG      R2         ;SAVE THE RESULT
1828      BIT      #SW0,2SWR  ;SEE IF VALUE NEGATIVE
1829      BNE      11$        ;BRANCH IF POSITIVE
1830      CMP      TOLER,R2   ;RECOMPLEMENT THE VALUE
1831      ;SEE IF SWRO SET
1832      ;BRANCH IF IT IS
1833      ;SEE IF TRACK CENTERLINE WITHIN TOLERANCE

```



```

1831 003402 002043          BGE      14$          ;BRANCH IF VALUE OK
1832 003404 104012          ERROR    12           ;REPORT THE ERROR
1833 003406 000435          BR       13$          ;CHECK FOR LOOP ON ERROR
1834 003410 010246          11$:    MOV      R2,-(SP) ;SAVE TRACK CENTER VALUE
1835 003412 005737 001174    TST     TOPLN        ;TYPE HEADER ?
1836 003416 001005          BNE     12$          ;BR IF NOT
1837 003420 104401 017657    TYPE    HEADER        ;TYPE THE HEADER
1838 003424 012737 177777 001174    MOV     @-1,TOPLN    ;CLEAR THE INDICATOR
1839 003432
1840 003432 013746 001172    12$:    MOV     $HEAD,-(SP) ;SAVE $HEAD FOR TYPEOUT
1841
1842 003436 104405          .IIF NB (<)          ;GO TYPE--DECIMAL ASCII WITH SIGN
1843 003440 013746 016132    MOV     DPB+CYL,-(SP) ;SAVE DPB+CYL FOR TYPEOUT
1844
1845 003444 104405          .IIF NB (<)          ;GO TYPE--DECIMAL ASCII WITH SIGN
1846 003446 013746 001204    MOV     PLUS,-(SP)   ;SAVE PLUS FOR TYPEOUT
1847
1848 003452 104405          .IIF NB (<)          ;GO TYPE--DECIMAL ASCII WITH SIGN
1849 003454 104401 001161    TYPE    TYPE,        ;CR-LF
1850 003460 023726 001210    CMP     TOLER,(SP)+ ;SEE IF HEAD IN TOLERANCE
1851 003464 002012          BGE     14$          ;BR IF IN TOLERANCE
1852 003466 032777 100000 175444    BIT     @SW15,@SWR   ;HALT IF ERROR ?
1853 003474 001406          BEQ     14$          ;BR IF NOT
1854 003476 000000          HALT
1855 003500 000404          BR      14$          ;HEAD JUST TYPED OUT OF TOLERANCE
1856 003502 032777 001000 175430 13$:    BIT     @SW9,@SWR   ;CHECK FOR SWITCH 1
1857 003510 001004          BNE     15$          ;SEE IF SWITCH 9 SET
1858 003512 032777 000002 175420 14$:    BIT     @SW1,@SWR  ;BR IF SET
1859 003520 001402          BEQ     16$          ;SWITCH 1 SET ?
1860 003522 000137 003032 15$:    JMP     CHECK        ;NOT SET
1861 003526 000207 16$:    RTS     PC          ;DO THE HEAD AGAIN
1862
1863          001          ;RETURN
1864
1865          .IF B
1866          ;*****
1867          .IFF
1868          .NLIST
1869          .REPT
1870          .LIST
1871          ;*****
1872          .NLIST
1873          .ENDR
1874          .LIST
1875          000          .ENDC
1876
1877          001          .SBTTL ROUTINE TO SELECT HEAD FOR DDU CONTROLLED HEAD ALIGNMENT
1878          .IF B
1879          ;*****
1880          .IFF
1881          .NLIST
1882          .REPT
1883          .LIST
1884          ;*****
1885          .NLIST
1886          .ENDR
1887          .LIST

```

```

1887          000          .ENDC
1888
1889 003530 104401 020027 DDUALN: TYPE      ALIGN      :TYPE 'ALIGN'
1890 003534 022737 000633 001212  CMP      #411.,MAXCYL :RPO4/5 ?
1891 003542 001403          BEQ      1$          :BR IF YES
1892 003544 104401 020270          TYPE      ALN496. :'POSITIONED AT 496' MESSAGE
1893 003550 000402          BR      2$          :CONTINUE
1894 003552 104401 020222 1$:      TYPE      ALN245 :'POSITIONED AT 245'
1895 003556 013737 001230 016132 2$:      MOV      MIDCYL,DPB+CYL :LOAD CYLINDER ADDRESS
1896 003564 004537 003736          JSR      R5,DRIVER :SEEK TO ALIGNMENT CYLINDER
1897 003570 000105          .WORD   SEEK       :DRIVER CODE FOR SEEK
1898 003572 104401 020336 3$:      TYPE      ENTHD   :ASK FOR HEAD NUMBER
1899 003576 004737 004270          JSR      PC,GETNUM  :READ THE KEYBOARD
1900 003602 000403          BR      4$          :'CR' ENTERED
1901 003604 021627 000022          CMP      (SP),#18. :ENTRY CAN'T BE GREATER THAN 18
1902 003610 101403          BLOS   5$          :BR IF LESS OR EQUAL
1903 003612 104401 001160 4$:      TYPE      $QUES  :TYPE A QUESTION MARK
1904 003616 000755          BR      1$          :TRY AGAIN
1905 003620 011637 001172 5$:      MOV      (SP),$HEAD :HEAD NUMBER FOR TYPEOUT
1906 003624 000316          SWAB   (SP)       :PUT HEAD NUMBER INTO UPPER BYTE
1907 003626 004037 015012          JSR      RO,WRT.RP  :LOAD RPO4
1908 003632 000006          RPO4          :REGISTER ADDRESS
1909 003634 001744          INIT          :UNCORRECTABLE PARITY ERROR RETURN
1910 003636 000755          BR      3$          :WAIT FOR NEXT HEAD ENTRY

```

```

1911          001          .IF B
1912          :*****
1913          .IFF
1914          .NLIST
1915          .REPT
1916          .LIST
1917          :*****
1918          .NLIST
1919          .ENDR
1920          .LIST
1921          .ENDC
1922          000

```

```

1923          .SBTTL ROUTINE TO PERFORM 5,000 RANDOM SEEKS
1924
1925          001          .IF B
1926          :*****
1927          .IFF
1928          .NLIST
1929          .REPT
1930          .LIST
1931          :*****
1932          .NLIST
1933          .ENDR
1934          .LIST
1935          .ENDC
1936          000

```

```

1937
1938 003640 104401 020040 001240 RANDOM: TYPE      EXER      :TYPE 'EXERCISE'
1939 003644 012737 011610 1$:      MOV      #5000.,SEKCNT :NUMBER OF SEEKS
1940 003652 004737 007034          JSR      PC,$RAND  :CYCLE THE RANDOM NUMBER GENERATOR
1941 003656 013746 007132          MOV      $HNUM,-(SP) :PUT UPPER RANDOM NUMBER ON THE STACK
1942 003662 005046          CLR      -(SP)     :UPPER DIVIDEND

```



1943	003664	013746	001212	MOV	MAXCYL, -(SP)	; PUT DIVISOR ON STACK
1944	003670	004737	007136	JSR	PC, \$DIV	; GET THE REMAINDER (RANDOM CYLINDER ADDRESS)
1945	003674	021637	016132	CMP	(SP), DPB+CYL	; SEE IF SAME RANDOM CYLINDER AS LAST
1946	003700	001003		BNE	2\$	; BR IF DIFFERENT CYLINDER
1947	003702	062706	000004	ADD	#4, SP	; ADJUST THE STACK POINTER
1948	003706	000761		BR	1\$	; TRY AGAIN
1949	003710	012637	016132	2\$: MOV	(SP)+, DPB+CYL	; PUT CYL ADDR IN THE DPB
1950	003714	005726		TST	(SP)+	; CORRECT THE STACK POINTER
1951	003716	004537	003736	JSR	R5, DRIVER	; DO THE SEEK
1952	003722	000105		.WORD	SEEK	; SEEK CODE
1953	003724	005337	001240	DEC	SEKCNT	; DECREMENT THE SEEK COUNT
1954	003730	001350		BNE	1\$	; BR IF NOT DONE
1955	003732	000137	001744	JMP	INIT	; RETURN TO COMMAND ROUTINE

1956						
1957		001		.IF B		
1958				;;*****		
1959				.IFF		
1960				.NLIST		
1961				.REPT		
1962				.LIST		
1963				;;*****		
1964				.NLIST		
1965				.ENDR		
1966				.LIST		
1967		000		.ENDC		

.SBTTL COMMON ENTRY TO THE RPO4/5/6 DRIVER

1968						
1969						
1970						
1971		001		.IF B		
1972				;;*****		
1973				.IFF		
1974				.NLIST		
1975				.REPT		
1976				.LIST		
1977				;;*****		
1978				.NLIST		
1979				.ENDR		
1980				.LIST		
1981		000		.ENDC		

1982						
1983	003736	012537	016122	DRIVER: MOV	(R5)+, DPB+COMND	; COMMAND CODE
1984	003742	004037	010472	1\$: JSR	RD, RPO4	; DRIVER ENTRY
1985	003746	016120		DPB		; DATA PARAMETER BLOCK ADDRESS
1986	003750	000774		BR	1\$	; REQUEST NOT ACCEPTED
1987	003752	005737	016136	2\$: TST	DPB+STATUS	; SEE IF ORDER COMPLETE
1988	003756	001775		BEQ	2\$	; BR IF NOT COMPLETE
1989	003760	100401		BMI	4\$	; BRANCH IF ERROR
1990	003762	000205		3\$: RTS	R5	; RETURN
1991	003764	032737	000200 016136	4\$: BIT	#BIT7, DPB+STATUS	; DID THE ORDER TERMINATE ?
1992	003772	001020		BNE	5\$	; BRANCH IF IT DID
1993	003774	032737	060006 016136	BIT	#BIT14!BIT13!BIT02!BIT01, DPB+STATUS	; DRIVE OFFLINE OR
1994						; NON-EXISTENT ?
1995	004002	001024		BNE	6\$	; BR IF IT IS
1996	004004	032737	010000 016136	BIT	#BIT12, DPB+STATUS	; DRIVE UNSAFE ?
1997	004012	001022		BNE	7\$	; BR IF UNSAFE
1998	004014	032737	006000 016136	BIT	#BIT11!BIT10, DPB+STATUS	; UNCORRECTABLE PARITY ERROR ?



```

1999 004022 001020
2000 004024 032737 001400 016136 BNE 8$ ;BR IF YES
2001 004032 001016 BIT #BIT09:BIT08,DPB ;STATUS ;TIMEOUT ON THIS DRIVE ?
2002 004034 104010 BNE 9$ ;BR IF YES
2003 004036 032777 001000 175074 5$: ERROR 10 ;REPORT DRIVE OR DATA ERROR
2004 004044 001746 BIT #SW09,2SWR ;LOOP ON ERROR ?
2005 004046 162705 000006 BEQ 3$ ;BR IF NOT
2006 004052 000743 SUB #6,R5 ;CORRECT R5 FOR LOOP
2007 004054 104016 BR 3$ ;RETURN
2008 004056 000405 6$: ERROR 16 ;REPORT DRIVE UNAVAILABLE
2009 004060 104011 BR 10$ ;CHECK FOR LOOP
2010 004062 000403 7$: ERROR 11 ;REPORT PERSISTENT UNSAFE
2011 004064 104015 BR 10$ ;CHECK FOR LOOP
2012 004066 000401 8$: ERROR 15 ;UNCORRECTABLE PARITY ERROR
2013 004070 104014 BR 10$ ;CHECK FOR LOOP
2014 004072 032777 001000 175040 9$: ERROR 14 ;SOFTWARE TIMEOUT
2015 004100 001403 10$: BIT #SW09,2SWR;LOOP ON ERROR ?
2016 004102 162705 000006 BEQ 11$ ;BR IF NOT
2017 004106 000725 SUB #6,R5 ;CORRECT R5 FOR LOOP
2018 004110 000137 001744 BR 3$ ;RETURN
2019 JMP INIT ;RETURN TO USER
2020 001
2021 .IF B
2022 ;*****
2023 .IFF
2024 .NLIST
2025 .REPT
2026 .LIST
2027 ;*****
2028 .NLIST
2029 .ENDR
2030 .LIST
2031 .ENDC
2032 000
2033 .SBTTL SUBROUTINES
2034 001
2035 .IF B
2036 ;*****
2037 .IFF
2038 .NLIST
2039 .REPT
2040 .LIST
2041 ;*****
2042 .NLIST
2043 .ENDR
2044 .LIST
2045 .ENDC
2046 000
2046 ; THIS ROUTINE WILL DETERMINE IF THERE IS A CLOCK ON THE SYSTEM
2047 ; AND IF THERE IS IT WILL SETUP THE VECTOR AND START THE CLOCK.
2048 ; CALL
2049 ; JSR PC,2$ST.CLK
2050 ; RETURN
2051
2052 004114 010146 ST.CLK: MOV R1,-(SP) ;SAVE R1
2053 004116 012701 000006 MOV #ERRVEC+2,R1 ;SAVE AND SETUP TIMEOUT VECTOR
2054 004122 011146 MOV (R1),-(SP)

```



```

2055 004124 005011          CLR      (R1)          ;LEVEL 0
2056 004126 014146          MOV      -(R1),-(SP)
2057 004130 012711 004146    MOV      #15,(R1)      ;GO TO 1$ ON TIMEOUT
2058 004134 005777 175110    TST      @PKCS         ;IS THERE A KW11-P?
2059 004140 004737 004200    JSR      PC,ST.PCLK    ;START THE KW11-P
2060 004144 000411          BR       3$           ;GO TO EXIT
2061 004146 022626          1$:     CMP      (SP)+,(SP)+ ;CLEAN UP THE STACK
2062 004150 012711 004166    MOV      #2$,(R1)      ;IF TIMEOUT GO TO 2$
2063 004154 005777 175102    TST      @LK$         ;IS THERE A KW11-L?
2064 004160 004737 004232    JSR      PC,ST.LCLK    ;START THE KW11-L
2065 004164 000401          BR       3$           ;EXIT
2066 004166 022626          2$:     CMP      (SP)+,(SP)+ ;CLEAN UP THE STACK
2067 004170 012621 3$:     MOV      (SP)+,(R1)+ ;RESTORE THE TIMEOUT VECTOR
2068 004172 012621          MOV      (SP)+,(R1)+
2069 004174 012601          MOV      (SP)+,R1     ;RESTORE R1
2070 004176 000207          RTS      PC           ;RETURN
2071
2072 004200 012777 004256 175036 ST.PCLK: MOV    #SRVCLK,@PKV ;SETUP THE KW11-P VECTOR
2073 004206 012777 000300 175032    MOV      #300,@PKV+2
2074 004214 012777 000001 175030    MOV      #1,@PKB
2075 004222 012777 000115 175020    MOV      #115,@PKCS
2076
2077 004230 000207          RTS      PC           ;RETURN
2078
2079 004232 012777 004256 175016 ST.LCLK: MOV    #SRVCLK,@LKV ;SETUP THE KW11-L VECTOR
2080 004240 012777 000300 175012    MOV      #300,@LKV+2
2081 004246 012777 000100 175006    MOV      #100,@LKS
2082 004254 000207          RTS      PC           ;RETURN
2083
2084 004256 012746 000020          SRVCLK: MOV    #16,-(SP) ;TIME PER TICK IN MILLISECONDS
2085 004262 004737 014110          JSR      PC,@RPTMR     ;COUNT THE ELAPSED TIME
2086 004266 000002          RTI             ;RETURN AFTER INTERRUPT
2087
2088          ;ROUTINE TO GET DECIMAL KEYBOARD ENTRIES
2089          ;CALL
2090          ;
2091          ; JSR      PC,GETNUM
2092          ; RETURN1
2093          ; RETURN2
2094          ; 'CR' ENTERED
2095          ; NUMBER ENTERED, CONVERTED NUMBER ON THE STACK
2096
2094 004270 011646          GETNUM: MOV    (SP),-(SP) ;PROVIDE SPACE FOR FIRST CHAR
2095 004272 104411 1$:     RDLIN          ;READ AN ASCIZ LINE
2096 004274 012600          MOV      (SP)+,RO     ;ADDRESS OF 1ST CHAR
2097 004276 010037 004376    MOV      RO,6$        ;SAVE IN CASE OF BAD INPUT
2098 004302 105710          TSTB     (RO)         ;FIRST CHARACTER A 'CR' ?
2099 004304 001440          BEQ      7$          ;BR IF IT IS
2100 004306 005046          CLR      -(SP)       ;CLEAR DATA WORD
2101 004310 112001 2$:     MOV      (RO)+,R1     ;PICKUP THIS CHARACTER
2102 004312 001421          BEQ      4$          ;GET OUT IF ZERO
2103 004314 122701 000060    CMPB     #'0',R1     ;MAKE SURE THAT THIS CHARACTER
2104 004320 003016          BGT      4$          ;IS A DIGIT BETWEEN 0 & 9
2105 004322 122701 000071    CMPB     #'9',R1
2106 004326 002420          BLT      5$
2107 004330 006316          ASL      (SP)        ;*2
2108 004332 011646          MOV      (SP),-(SP)  ;SAVE FOR LATER
2109 004334 006316          ASL      (SP)        ;*4
2110 004336 006316          ASL      (SP)        ;*8.

```

```

2111 004340 062616          ADD      (SP)+,(SP)      ;*10.
2112 004342 102412          BVS      5$              ;OVERFLOW ISN'T ALLOWED
2113 004344 042701 000060    BIC      #60,R1         ;CLEAR THE UPPER BITS
2114 004350 060116          ADD      R1,(SP)        ;ADD THE NUMBER
2115 004352 102406          BVS      5$              ;OVERFLOW ISN'T ALLOWED
2116 004354 000755          BR       2$              ;LOOP
2117 004356 012666 000002    4$:     MOV      (SP)+,2(SP) ;SAVE THE RESULT
2118 004362 062716 000002    ADD      #2,(SP)        ;INCREMENT THE RETURN ADDRESS
2119 004366 000410          BR       8$              ;EXIT
2120 004370 005726          5$:     TST      (SP)+      ;CLEAN PARTIAL NUMBER FROM STACK
2121 004372 105010          CLRB    (R0)           ;SET A TERMINATOR
2122 004374 104401          TYPE    ;TYPE THE INPUT UP TO BAD CHAR
2123 004376 000000 001160    6$:     .WORD    0         ;POINTER GOES HERE
2124 004400 104401          TYPE    '$QUES         ;'?' 'CR' 'LF'
2125 004404 000732          BR       1$              ;TRY AGAIN
2126 004406 012616          7$:     MOV      (SP)+,(SP) ;RESTORE THE PC
2127 004410 000207          8$:     RTS       PC       ;RETURN
2128
2129          001
2130          .IF B
2131          ;*****
2132          .IFF
2133          .NLIST
2134          .REPT
2135          .LIST
2136          ;*****
2137          .NLIST
2138          .ENDR
2139          .LIST
2140          .ENDC
2141          000
2142          .SBTTL  MACRO ROUTINES
2143          .SBTTL  ERROR HANDLER ROUTINE
2144          001
2145          .IF B
2146          ;*****
2147          .IFF
2148          .NLIST
2149          .REPT
2150          .LIST
2151          ;*****
2152          .NLIST
2153          .ENDR
2154          .LIST
2155          .ENDC
2156          ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2157          ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2158          001
2159          .IF B $ERRTYP
2160          .IIF NE 20000&$$SWR,;*AND TYPE OUT THE PC OF THE ERROR INSTRUCTION
2161          .IFF
2162          ;*AND GO TO $ERRTYP ON ERROR
2163          .ENDC
2164          000
2165          .IIF NE 123000&$$SWR,;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
2166          .IIF NE 100000&$$SWR,;*SW15=1      HALT ON ERROR
2167          .IIF NE 20000&$$SWR,;*SW13=1      INHIBIT ERROR TYPEOUTS
2168          .IIF NE 2000&$$SWR,;*SW10=1      BELL ON ERROR

```



```

2167 .IIF NE 1000&$$SWR,;*SW09=1 LOOP ON ERROR
2168 ;*CALL
2169 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
2170
2171 004412 $ERROR:
2172 004412 104407 .IIF NE $SETUP&100, CKSWR ;;TEST FOR CHANGE IN SOFT-SWR
2173 001 .IF NB <<MOV R1,DRIVE>,<MOV R3,ATTN>
2174 004414 010137 001166 MOV R1,DRIVE
2175 004420 010337 001170 MOV R3,ATTN
2176 000 .ENDC
2177 004424 105237 001103 7$: INCB $ERFLG ;;SET THE ERROR FLAG
2178 004430 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
2179 004432 013777 001102 174502 MOV $STSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
2180 001 .IF EQ 2000&$$SWR ;SW10
2181 004440 005237 001112 INC $ERTTL ;;INC THE ERROR COUNT
2182
2183 .IFF
2184 BIT #BIT10,@SWR ;;BELL ON ERROR?
2185 BEQ 1$ ;;NO - SKIP
2186 TYPE $BELL ;;RING BELL
2187 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
2188 .ENDC
2189 004444 011637 001116 MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
2190 004450 162737 000002 001116 SUB #2, $ERRPC
2191 004456 117737 174434 001114 MOVB @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
2192 001 .IF NE 20000&$$SWR ;SW13
2193 004464 032777 020000 174446 BIT #BIT13,@SWR ;;SKIP TYPEOUT IF SET
2194 001 BNE 20$ ;;SKIP TYPEOUTS
2195 .IFTF
2196 002 .IF DF $4DCAT
2197 CMP (SP), #1002 ;;IF RETURN PC LESS THAN 1002
2198 BHI 12$ ;;ERROR IS ILLEGAL TRAP
2199 ;;PROCESS UNEXPECTED TRAP OR INTERRUPT
2200 MOV 4(SP), $ERRPC ;;GET PC AT TIME OF FALSE TRAP
2201 SUB #2, $ERRPC ;;ADJUST PC
2202 TYPE 10$ ;;TYPE HEADER
2203 TYPOCT $ERRPC ;;TYPE PC AT TIME OF ERROR
2204 TYPE 11$
2205 SUB #4, (SP) ;;GET FALSE TRAP VECTOR ADDR
2206 MOV (SP), $ERRPC
2207 TYPOCT $ERRPC
2208 TYPE $CRLF
2209 CMP (SP)+, (SP)+ ;;POP FALSE TRAP VECTOR PC&ADDR
2210 BR 20$
2211 10$: .ASCIZ '<200>'PC= '
2212 11$: .ASCIZ ' UNEXPECTED TRAP TO '
2213 .EVEN
2214 12$: .ENDC
2215 .IF NB $ERRTYP
2216 004474 004737 004520 JSR PC, $ERRTYP ;;GO TO USER ERROR ROUTINE
2217 001 .ENDC
2218 .IFT
2219 002 .IF B $ERRTYP
2220 TYPE $CRLF
2221 TYPOCT $ERRPC, <ERROR ADDRESS>
2222 001 .ENDC

```

```

2223 004500 104401 001161          TYPE      ,SCLF
2224                                .ENDC
2225 004504                          20$:
2226                                .IF DF $MAIL
2227                                CMPB      #APTENV,$ENV      ;;RUNNING IN APT MODE
2228                                BNE       2$              ;;NO SKIP APT ERROR REPORT
2229                                MOVB     $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
2230                                JSR      PC,$ATY4        ;;REPORT FATAL ERROR TO APT
2231                                21$:
2232                                .BYTE   0
2233                                .BYTE   0
2234                                BR       22$              ;;APT ERROR LOOP
2235                                .ENDC
2236 004504 005777 174430          2$:
2237 004510 100002                    .IF NE $SWR&100000 ;SW15      ;;HALT ON ERROR
2238 004512 000000                    TST      @SWR                ;;SKIP IF CONTINUE
2239 004514 104407                    BPL     3$                  ;;HALT ON ERROR!
2240                                .IIF NE $SETUP&100, CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
2241                                .IF NE $SWR&1000 ;SW09
2242                                3$:
2243                                BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
2244                                BEQ     4$                ;;BR IF NO
2245                                MOV     $LPERR,(SP)       ;;FUDGE RETURN FOR LOOPING
2246                                TST     $ESCAPE            ;;CHECK FOR AN ESCAPE ADDRESS
2247                                BEQ     5$                ;;BR IF NONE
2248                                MOV     $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
2249 004516                                5$:
2250                                .IFF
2251                                3$:
2252                                .ENDC
2253                                .IFF
2254                                .IF NE $SWR&1000 ;SW09
2255                                2$:
2256                                BIT      #BIT09,@SWR      ;;LOOP ON ERROR?
2257                                BEQ     3$                ;;BR IF NO
2258                                MOV     $LPERR,(SP)       ;;FUDGE RETURN FOR LOOPING
2259                                TST     $ESCAPE            ;;CHECK FOR AN ESCAPE ADDRESS
2260                                BEQ     4$                ;;BR IF NONE
2261                                MOV     $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
2262                                4$:
2263                                .IFF
2264                                2$:
2265                                .ENDC
2266                                .ENDC
2267                                .IF NE $SETUP&20 ;BIT04--(.SEOP)
2268                                CMP      #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
2269                                BNE     6$                ;;BRANCH IF NO
2270                                HALT                    ;;YES
2271 004516 000002                    6$:
2272                                .ENDC
2273                                .IF B (<)
2274                                RTI                          ;;RETURN
2275                                .IFF
2276                                .IRP LASTIN,<>
2277                                LASTIN
2278                                .ENDM
2279                                .ENDC
2280                                .IIF NDF $ERTTL,$ERTTL:
2281                                .IIF EQ .-$ERTTL,$ERTTL: .WORD 0 ;;ERROR COUNT

```



```

2279 .IIF NDF STSTNM,STSTNM:
2280 .IIF EQ .-STSTNM,STSTNM: .BYTE 0 ;;TEST NUMBER
2281 .IIF NDF SERFLG,SERFLG:
2282 .IIF EQ .-SERFLG,SERFLG: .BYTE 0 ;;ERROR FLAG
2283 .IIF NE 81,EVEN
2284 .IF NE $SWR&1000;SW09
2285 .IIF NDF $ESCAPE,$ESCAPE:
2286 .IIF EQ .-$ESCAPE,$ESCAPE: .WORD 0 ;;ESCAPE ON ERROR ADDRESS
2287 .ENDC
2288
2289 .IIF NDF $ERRPC,$ERRPC:
2290 .IIF EQ .-$ERRPC,$ERRPC: .WORD 0 ;;LAST ERROR INSTRUCTION EXECUTE
2291 .IF NE $SWR&2000;SW10
2292 .IIF NDF $BELL,$BELL:
2293 .IIF EQ .-$BELL,$BELL: .ASCIZ <207><377><377> ;;ASCII CODE FOR BELL
2294 .ENDC
2295
2296 .IIF NDF $ITEMB,$ITEMB:
2297 .IIF EQ .-$ITEMB,$ITEMB: .WORD 0 ;;ITEM BYTE
2298 .IIF NDF $QUES,$QUES:
2299 .IIF EQ .-$QUES,$QUES: .ASCII "?" ;;QUESTION MARK
2300 .IIF NDF $SCRLF,$SCRLF:
2301 .IIF EQ .-$SCRLF,$SCRLF: .ASCII <15> ;;CARRIAGE RETURN
2302 .IIF NDF $SLF,$SLF:
2303 .IIF EQ .-$SLF,$SLF: .ASCIZ <12> ;;LINEFEED
2304 .IIF NE 18,EVEN
2305 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
2306
2307 .IF B
2308 ;*****
2309 .IFF
2310 .NLIST
2311 .REPT
2312 .LIST
2313 ;*****
2314 .NLIST
2315 .ENDR
2316 .LIST
2317 .ENDC
2318 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
2319 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
2320 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
2321
2322 $ERRTYP:
2323 TYPE SCRLF ;; "CARRIAGE RETURN" & "LINE FEED"
2324 MOV RO,-(SP) ;; SAVE RO
2325 CLR RO ;; PICKUP THE ITEM INDEX
2326 BISB @#$ITEMB,RO
2327 BNE 1$ ;; IF ITEM NUMBER IS ZERO, JUST
2328 MOV $ERRPC,-(SP) ;; TYPE THE PC OF THE ERROR
2329 .IIF NB <ERROR ADDRESS>, ;; SAVE $ERRPC FOR TYPEOUT
2330 TYPOC ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
2331 .IF B 1
2332 BR 6$ ;; GET OUT
2333 .IFF
2334 BR 10$ ;; GET OUT
2335 .ENDC

```

```

004520
004520 104401 001161
004524 010046
004526 005000
004530 153700 001114
004534 001004
004536 013746 001116
004542 104402
001
004544 000445
000

```

2335	004546	005300		1\$:	DEC	RO	:: ADJUST THE INDEX SO THAT IT WILL
2336	004550	006300			ASL	RO	:: WORK FOR THE ERROR TABLE
2337	004552	006300			ASL	RO	
2338	004554	006300			ASL	RO	
2339	004556	062700	001270		ADD	#SERRTB,RO	:: FORM TABLE POINTER
2340	004562	012037	004572		MOV	(RO)+,2\$	:: PICKUP "ERROR MESSAGE" POINTER
2341	004566	001404			BEQ	3\$	:: SKIP TYPEOUT IF NO POINTER
2342	004570	104401			TYPE		:: TYPE THE "ERROR MESSAGE"
2343	004572	000000		2\$:	.WORD	0	:: "ERROR MESSAGE" POINTER GOES HERE
2344	004574	104401	001161		TYPE	, \$CRLF	:: "CARRIAGE RETURN" & "LINE FEED"
2345	004600	012037	004610	3\$:	MOV	(RO)+,4\$	:: PICKUP "DATA HEADER" POINTER
2346	004604	001404			BEQ	5\$	:: SKIP TYPEOUT IF 0
2347	004606	104401			TYPE		:: TYPE THE "DATA HEADER"
2348	004610	000000		4\$:	.WORD	0	:: "DATA HEADER" POINTER GOES HERE
2349	004612	104401	001161		TYPE	, \$CRLF	:: "CARRIAGE RETURN" & "LINE FEED"
2350		001			.IF B 1		
2351				5\$:	MOV	(RO),RO	:: PICKUP "DATA TABLE" POINTER
2352					BNE	7\$	:: GO TYPE THE DATA
2353				6\$:	MOV	(SP)+,RO	:: RESTORE RO
2354					TYPE	, \$CRLF	:: "CARRIAGE RETURN" & "LINE FEED"
2355					RTS	PC	:: RETURN
2356				7\$:	TYPOCT	2(RO)+	:: TYPE AN OCTAL NUMBER
2357					TST	(RO)	:: IS THERE ANOTHER NUMBER?
2358					BEQ	6\$	:: BR IF NO
2359					TYPE	8\$	:: TYPE TWO(2) SPACES
2360					BR	7\$	:: LOOP
2361				8\$:	.ASCIZ	/ /	:: TWO(2) SPACES
2362					.EVEN		
2363					.IFF		
2364	004616	010146		5\$:	MOV	R1, -(SP)	:: SAVE R1
2365	004620	012001			MOV	(RO)+,R1	:: PICKUP "DATA TABLE" POINTER
2366	004622	001415			BEQ	9\$	:: BR IF NO DATA TO BE TYPED
2367	004624	012000			MOV	(RO)+,RO	:: PICKUP "DATA FORMAT" POINTER
2368	004626	105720		6\$:	TSTB	(RO)+	:: "OCTAL" OR "DECIMAL"
2369	004630	001003			BNE	7\$	:: BR IF DECIMAL
2370	004632	013146			MOV	2(R1)+, -(SP)	:: SAVE 2(R1)+ FOR TYPEOUT
2371				.IIF NB	<>		
2372	004634	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
2373	004636	000402			BR	8\$	
2374	004640			7\$:			
2375	004640	013146			MOV	2(R1)+, -(SP)	:: SAVE 2(R1)+ FOR TYPEOUT
2376				.IIF NB	<>		
2377	004642	104405			TYPODS		:: GO TYPE--DECIMAL ASCII WITH SIGN
2378	004644	005711		8\$:	TST	(R1)	:: IS THERE ANOTHER NUMBER?
2379	004646	001403			BEQ	9\$	:: BR IF NO
2380	004650	104401	004670		TYPE	11\$	:: TYPE TWO(2) SPACES
2381	004654	000764			BR	6\$	:: LOOP
2382				9\$:	MOV	(SP)+,R1	:: RESTORE R1
2383	004656	012601		10\$:	MOV	(SP)+,RO	:: RESTORE RO
2384	004660	012600			TYPE	, \$CRLF	:: "CARRIAGE RETURN" & "LINE FEED"
2385	004662	104401	001161		RTS	PC	:: RETURN
2386	004666	000207		11\$:	.ASCIZ	/ /	:: TWO(2) SPACES
2387	004670	020040	000		.EVEN		
2388		004674			.ENDC		
2389		000			.SBTTL	TYPE ROUTINE	
2390							



```

2391      001      .IF B
2392      .:*****
2393      .IFF
2394      .NLIST
2395      .REPT
2396      .LIST
2397      .:*****
2398      .NLIST
2399      .ENDR
2400      .LIST
2401      .ENDC
2402      000
2403      *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
2404      *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
2405      *NOTE1:      $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
2406      *NOTE2:      $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
2407      *NOTE3:      $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2408      *
2409      *CALL:
2410      *1) USING A TRAP INSTRUCTION
2411      *      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2412      *OR
2413      *      TYPE
2414      *      MESADR
2415      *
2416
2417      004674      105737      001157      STYPE:      TSTB      $TFPLG      ;; IS THERE A TERMINAL?
2418      004700      100002      BPL      1$      ;; BR IF YES
2419      004702      000000      HALT      ;; HALT HERE IF NO TERMINAL
2420      004704      000407      BR      3$      ;; LEAVE
2421      004706      010046      1$:      MOV      RO,-(SP)      ;; SAVE RO
2422      004710      017600      000002      MOV      22(SP),RO      ;; GET ADDRESS OF ASCIZ STRING
2423      .IF DF $MAIL
2424      CMPB      #APTENV,$ENV      ;; RUNNING IN APT MODE
2425      BNE      62$      ;; NO GO CHECK FOR APT CONSOLE
2426      BITB      #APTSPool,$ENVM      ;; SPOOL MESSAGE TO APT
2427      BEQ      62$      ;; NO GO CHECK FOR CONSOLE
2428      MOV      RO,61$      ;; SETUP MESSAGE ADDRESS FOR APT
2429      JSR      PC,$ATY3      ;; SPOOL MESSAGE TO APT
2430      61$:      .WORD      0      ;; MESSAGE ADDRESS
2431      62$:      BITB      #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
2432      BNE      60$      ;; YES,SKIP TYPE OUT
2433      .ENDC
2434      000
2435      004714      112046      2$:      MOVB      (RO)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
2436      004716      001005      BNE      4$      ;; BR IF IT ISN'T THE TERMINATOR
2437      004720      005726      TST      (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
2438      004722      012600      60$:      MOV      (SP)+,RO      ;; RESTORE RO
2439      004730      000002      3$:      ADD      #2,(SP)      ;; ADJUST RETURN PC
2440      004732      122716      4$:      RTI      ;; RETURN
2441      004736      001430      CMPB      #HT,(SP)      ;; BRANCH IF <HT>
2442      004740      122716      8$:      BEQ      8$      ;; BRANCH IF NOT <CRLF>
2443      004744      001006      CMPB      #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
2444      004746      005726      BNE      5$      ;; POP <CR><LF> EQUIV
2445      004750      104401      TST      (SP)+      ;; TYPE A CR AND LF
2446      004752      001161      TYPE
      $CRLF

```

```

2447 004754 105037 005110          CLRB   $CHARCNT      ;; CLEAR CHARACTER COUNT
2448 004760 000755                   BR     2$             ;; GET NEXT CHARACTER
2449 004762 004737 005044          5$:   JSR   PC,$TYPEC  ;; GO TYPE THIS CHARACTER
2450 004766 123726 001156          6$:   CMPB  $FILLC,(SP)+  ;; IS IT TIME FOR FILLER CHARS.?
2451 004772 001350                   BNE   2$             ;; IF NO GO GET NEXT CHAR.
2452 004774 013746 001154          MOV   $NULL,-(SP)    ;; GET # OF FILLER CHARS. NEEDED
2453                                     AND   THE NULL CHAR.
2454 005000 105366 000001          7$:   DEC   1(SP)     ;; DOES A NULL NEED TO BE TYPED?
2455 005004 002770                   BLT   6$             ;; BR IF NO--GO POP THE NULL OFF OF STACK
2456 005006 004737 005044          JSR   PC,$TYPEC      ;; GO TYPE A NULL
2457 005012 105337 005110          DEC   $CHARCNT       ;; DO NOT COUNT AS A COUNT
2458 005016 000770                   BR    7$            ;; LOOP
2459
2460                                     ;HORIZONTAL TAB PROCESSOR
2461
2462 005020 112716 000040          8$:   MOV   #'(SP)    ;; REPLACE TAB WITH SPACE
2463 005024 004737 005044          9$:   JSR   PC,$TYPEC  ;; TYPE A SPACE
2464 005030 132737 000007 005110  BITB  #',$CHARCNT    ;; BRANCH IF NOT AT
2465 005036 001372                   BNE   9$             ;; TAB STOP
2466 005040 005726                   TST  (SP)+          ;; POP SPACE OFF STACK
2467 005042 000724                   BR    2$            ;; GET NEXT CHARACTER
2468 005044 105777 174100          $TYPEC: TSTB  @STPS    ;; WAIT UNTIL PRINTER IS READY
2469 005050 100375                   BPL  $TYPEC
2470 005052 116677 000002 174072  MOV   2(SP),@STPB    ;; LOAD CHAR TO BE TYPED INTO DATA REG.
2471 005060 122766 000015 000002  CMPB  #CR,2(SP)     ;; IS CHARACTER A CARRIAGE RETURN?
2472 005066 001003                   BNE  1$             ;; BRANCH IF NO
2473 005070 105037 005110          CLR   $CHARCNT      ;; YES--CLEAR CHARACTER COUNT
2474 005074 000406                   BR   $TYPEX
2475 005076 122766 000012 000002  1$:   CMPB  #LF,2(SP)  ;; IS CHARACTER A LINE FEED?
2476 005104 001402                   BEQ  $TYPEX         ;; BRANCH IF YES
2477 005106 105227                   INCB (PC)+          ;; COUNT THE CHARACTER
2478 005110 000000          $CHARCNT: .WORD 0  ;; CHARACTER COUNT STORAGE
2479 005112 000207          $TYPEX:  RTS      PC
2480
2481                                     .IIF NDF HT,HT= 11    ;; CODE FOR HORIZONTAL TAB
2482                                     .IIF NDF LF,LF= 12    ;; CODE FOR LINE FEED
2483                                     .IIF NDF CR,CR= 15    ;; CODE FOR CARRIAGE RETURN
2484                                     .IIF NDF CRLF,CRLF= 200  ;; CODE FOR CARRIAGE RETURN-LINE FEED
2485                                     .IIF NDF STPS,STPS: .WORD 177564  ;; TTY PRINTER STATUS REG. ADDRESS
2486                                     .IIF EQ -STPS,STPS: .WORD 177566  ;; TTY PRINTER BUFFER REG. ADDRESS
2487                                     .IIF NDF STPB,STPB: .WORD 177566  ;; TTY PRINTER BUFFER REG. ADDRESS
2488                                     .IIF EQ -STPB,STPB: .WORD 177566  ;; TTY PRINTER BUFFER REG. ADDRESS
2489                                     .IIF NDF $NULL,$NULL: .BYTE 0  ;; CONTAINS NULL CHARACTER FOR FILLS
2490                                     .IIF EQ -$NULL,$NULL: .BYTE 0  ;; CONTAINS NULL CHARACTER FOR FILLS
2491                                     .IIF NDF $FILLS,$FILLS: .BYTE 2  ;; CONTAINS # OF FILLER CHARACTER
2492                                     .IIF EQ -$FILLS,$FILLS: .BYTE 2  ;; CONTAINS # OF FILLER CHARACTER
2493                                     .IIF NDF $FILLC,$FILLC: .BYTE 12  ;; INSERT FILL CHARS. AFTER A "LI
2494                                     .IIF EQ -$FILLC,$FILLC: .BYTE 12  ;; INSERT FILL CHARS. AFTER A "LI
2495                                     .IIF NDF $STPFLG,$STPFLG: .BYTE 0  ;; "TERMINAL AVAILABLE" FLAG (BIT
2496                                     .IIF EQ -$STPFLG,$STPFLG: .BYTE 0  ;; "TERMINAL AVAILABLE" FLAG (BIT
2497                                     .IIF NDF $QUES,$QUES: .ASCII "?"  ;; QUESTION MARK
2498                                     .IIF EQ -$QUES,$QUES: .ASCII "?"  ;; QUESTION MARK
2499                                     .IIF NDF $CRLF,$CRLF: .ASCII <15>  ;; CARRIAGE RETURN
2500                                     .IIF EQ -$CRLF,$CRLF: .ASCII <15>  ;; CARRIAGE RETURN
2501                                     .IIF NDF $LF,$LF: .ASCIZ <12>  ;; LINEFEED
2502                                     .IIF EQ -$LF,$LF: .ASCIZ <12>  ;; LINEFEED

```



2503  
2504  
2505  
2506  
2507  
2508  
2509  
2510  
2511  
2512  
2513  
2514  
2515  
2516  
2517  
2518  
2519  
2520  
2521  
2522  
2523  
2524  
2525  
2526  
2527  
2528  
2529  
2530  
2531  
2532  
2533  
2534  
2535  
2536  
2537  
2538  
2539  
2540  
2541  
2542  
2543  
2544  
2545  
2546  
2547  
2548  
2549  
2550  
2551  
2552  
2553  
2554  
2555  
2556  
2557  
2558

001

000

```

.IIF NE 18. .EVEN
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

.IF B
;*****
.IFF
.NLIST
.REPT
.LIST
;*****
.NLIST
.ENDR
.LIST
.ENDC
*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
*OCTAL (ASCII) NUMBER AND TYPE IT.
*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
*CALL:
*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
*      TYPOS    ;;CALL FOR TYPEOUT
*      .BYTE    N                  ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
*      .BYTE    M                  ;;M=1 OR 0
*                                  ;;1=TYPE LEADING ZEROS
*                                  ;;0=SUPPRESS LEADING ZEROS
*$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
*$TYPOS OR $TYPOC
*CALL:
*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
*      TYPON    ;;CALL FOR TYPEOUT
*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
*CALL:
*      MOV      NUM,-(SP)          ;;NUMBER TO BE TYPED
*      TYPOC    ;;CALL FOR TYPEOUT
*$TYPOS: MOV      2(SP),-(SP)      ;;PICKUP THE MODE
        MOV      1(SP),SOFILL     ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+,SOMODE+1   ;;NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)         ;;ADJUST RETURN ADDRESS
        BR      $TYPON
*$TYPOC: MOV      #1,SOFILL       ;;SET THE ZERO FILL SWITCH
        MOV      #6,SOMODE+1     ;;SET FOR SIX(6) DIGITS
*$TYPON: MOV      #5,SOCNT       ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)        ;;SAVE R3
        MOV      R4,-(SP)        ;;SAVE R4
        MOV      R5,-(SP)        ;;SAVE R5
        MOV      SOMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,SOMODE       ;;SAVE IT FOR USE
        MOV      SOFILL,R4      ;;GET THE ZERO FILL SWITCH
        MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
        CLR      R3             ;;CLEAR THE OUTPUT WORD
        ROL     R5              ;;ROTATE MSB INTO "C"
        BR      3$             ;;GO DO MSB
1$:
        ROL     R5
        BR      3$

```

005114	017646	000000	
005120	116637	000001	005337
005126	112637	005341	
005132	062716	000002	
005136	000406		
005140	112737	000001	005337
005146	112737	000006	005341
005154	112737	000005	005336
005162	010346		
005164	010446		
005166	010546		
005170	113704	005341	
005174	005404		
005176	062704	000006	
005202	110437	005340	
005206	113704	005337	
005212	016605	000012	
005216	005003		
005220	006105		
005222	000404		

```

2559 005224 006105      2$:  ROL  R5          ;;FORM THIS DIGIT
2560 005226 006105      ROL  R5
2561 005230 006105      ROL  R5
2562 005232 010503      MOV  R5,R3
2563 005234 006103      3$:  ROL  R3          ;;GET LSB OF THIS DIGIT
2564 005236 105337 005340  DECB $OMODE        ;;TYPE THIS DIGIT?
2565 005242 100016      BPL  7$           ;;BR IF NO
2566 005244 042703 177770  BIC  #177770,R3  ;;GET RID OF JUNK
2567 005250 001002      BNE  4$           ;;TEST FOR 0
2568 005252 005704      TST  R4           ;;SUPPRESS THIS 0?
2569 005254 001403      BEQ  5$           ;;BR IF YES
2570 005256 005204      4$:  INC  R4          DON'T SUPPRESS ANYMORE 0'S
2571 005260 052703 000060  BIS  #'0,R3      MAKE THIS DIGIT ASCII
2572 005264 052703 000040  5$:  BIS  #' ,R3   MAKE ASCII IF NOT ALREADY
2573 005270 110337 005334  MOVB R3,R5       SAVE FOR TYPING
2574 005274 104401 005334  TYPE 8$          GO TYPE THIS DIGIT
2575 005300 105337 005336  7$:  DECB $OCNT    COUNT BY 1
2576 005304 003347      BGT  2$           BR IF MORE TO DO
2577 005306 002402      BLT  6$           BR IF DONE
2578 005310 005204      INC  R4           INSURE LAST DIGIT ISN'T A BLANK
2579 005312 000744      BR   2$          GO DO THE LAST DIGIT
2580 005314 012605      6$:  MOV  (SP)+,R5  RESTORE R5
2581 005316 012604      MOV  (SP)+,R4    RESTORE R4
2582 005320 012603      MOV  (SP)+,R3    RESTORE R3
2583 005322 016666 000002 000004  MOV  2(SP),4(SP) SET THE STACK FOR RETURNING
2584 005330 012616      MOV  (SP)+,(SP)
2585 005332 000002      RTI
2586 005334 000      8$:  .BYTE 0        RETURN
2587 005335 000      .BYTE 0        STORAGE FOR ASCII DIGIT
2588 005336 000      .BYTE 0        TERMINATOR FOR TYPE ROUTINE
2589 005337 000      $OCNT: .BYTE 0   OCTAL DIGIT COUNTER
2590 005340 000000  $OFILL: .BYTE 0  ZERO FILL SWITCH
2591      $OMODE: .WORD 0 NUMBER OF DIGITS TO TYPE
2592      .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2593      .IF B
2594      ;*****
2595      .IFF
2596      .NLIST
2597      .REPT
2598      .LIST
2599      ;*****
2600      .NLIST
2601      .ENDR
2602      .LIST
2603      .ENDC
2604      ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2605      ;SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2606      ;NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2607      ;BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2608      ;REPLACED WITH SPACES.
2609      ;CALL:
2610      ;*   MOV  NUM,-(SP)  ;;PUT THE BINARY NUMBER ON THE STACK
2611      ;*   TYPDS          ;;GO TO THE ROUTINE
2612
2613 005342      $TYPDS: MOV  R0,-(SP)  ;;PUSH R0 ON STACK
2614 005342 010046

```



MO4

CZRJCBO, RPO4/5/6 HEAD ALNMT  
CZRJCBO.P11 04-NOV-77 12:49

MACY11 30(1046) 04-NOV-77 17:31 PAGE 51  
CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0051

2615	005344	010146			MOV	R1,-(SP)	:: PUSH R1 ON STACK
2616	005346	010246			MOV	R2,-(SP)	:: PUSH R2 ON STACK
2617	005350	010346			MOV	R3,-(SP)	:: PUSH R3 ON STACK
2618	005352	010546			MOV	R5,-(SP)	:: PUSH R5 ON STACK
2619	005354	012746	020200		MOV	#20200,-(SP)	:: SET BLANK SWITCH AND SIGN
2620	005360	016605	000020		MOV	20(SP),R5	:: GET THE INPUT NUMBER
2621	005364	100004			BPL	1\$	:: BR IF INPUT IS POS.
2622	005366	005405			NEG	R5	:: MAKE THE BINARY NUMBER POS.
2623	005370	112766	000055	000001	MOVB	#'-,1(SP)	:: MAKE THE ASCII NUMBER NEG.
2624	005376	005000			CLR	R0	:: ZERO THE CONSTANTS INDEX
2625	005400	012703	005556		MOV	#\$DBLK,R3	:: SETUP THE OUTPUT POINTER
2626	005404	112723	000040		MOVB	#',(R3)+	:: SET THE FIRST CHARACTER TO A BLANK
2627	005410	005002			CLR	R2	:: CLEAR THE BCD NUMBER
2628	005412	016001	005546		MOV	\$DTBL(R0),R1	:: GET THE CONSTANT
2629	005416	160105			SUB	R1,R5	:: FORM THIS BCD DIGIT
2630	005420	002402			BLT	4\$	:: BR IF DONE
2631	005422	005202			INC	R2	:: INCREASE THE BCD DIGIT BY 1
2632	005424	000774			BR	3\$	
2633	005426	060105			ADD	R1,R5	:: ADD BACK THE CONSTANT
2634	005430	005702			TST	R2	:: CHECK IF BCD DIGIT=0
2635	005432	001002			BNE	5\$	:: FALL THROUGH IF 0
2636	005434	105716			TSTB	(SP)	:: STILL DOING LEADING 0'S?
2637	005436	100407			BMI	7\$	:: BR IF YES
2638	005440	106316			ASLB	(SP)	:: MSD?
2639	005442	103003			BCC	6\$	:: BR IF NO
2640	005444	116663	000001	177777	MOVB	1(SP),-1(R3)	:: YES--SET THE SIGN
2641	005452	052702	000060		BIS	#'0,R2	:: MAKE THE BCD DIGIT ASCII
2642	005456	052702	000040		BIS	#',R2	:: MAKE IT A SPACE IF NOT ALREADY A DIGIT
2643	005462	110223			MOVB	R2,(R3)+	:: PUT THIS CHARACTER IN THE OUTPUT BUFFER
2644	005464	005720			TST	(R0)+	:: JUST INCREMENTING
2645	005466	020027	000010		CMP	R0,#10	:: CHECK THE TABLE INDEX
2646	005472	002746			BLT	2\$	:: GO DO THE NEXT DIGIT
2647	005474	003002			BGT	8\$	:: GO TO EXIT
2648	005476	010502			MOV	R5,R2	:: GET THE LSD
2649	005500	000764			BR	6\$	:: GO CHANGE TO ASCII
2650	005502	105726			TSTB	(SP)+	:: WAS THE LSD THE FIRST NON-ZERO?
2651	005504	100003			BPL	9\$	:: BR IF NO
2652	005506	116663	177777	177776	MOVB	-1(SP),-2(R3)	:: YES--SET THE SIGN FOR TYPING
2653	005514	105013			CLRB	(R3)	:: SET THE TERMINATOR
2654	005516	012605			MOV	(SP)+,R5	:: POP STACK INTO R5
2655	005520	012603			MOV	(SP)+,R3	:: POP STACK INTO R3
2656	005522	012602			MOV	(SP)+,R2	:: POP STACK INTO R2
2657	005524	012601			MOV	(SP)+,R1	:: POP STACK INTO R1
2658	005526	012600			MOV	(SP)+,R0	:: POP STACK INTO R0
2659	005530	104401	005556		TYPE	\$DBLK	:: NOW TYPE THE NUMBER
2660	005534	016666	000002	000004	MOV	2(SP),4(SP)	:: ADJUST THE STACK
2661	005542	012616			MOV	(SP)+,(SP)	
2662	005544	000002			RTI		:: RETURN TO USER
2663	005546	023420			\$DTBL:	10000.	
2664	005550	001750				1000.	
2665	005552	000144				100.	
2666	005554	000012				10.	
2667	005556	000004			\$DBLK:	.BLKW 4	
2668					.SBTTL	TTY INPUT ROUTINE	
2669							
2670		001			.IF B		

```

2671 ;*****
2672 .IFF
2673 .NLIST
2674 .REPT
2675 .LIST
2676 ;*****
2677 .NLIST
2678 .ENDR
2679 .LIST
2680 .ENDC
2681 000
2682 .IIF NDF $TKS,$TKS:
2683 .IIF EQ .-$TKS,$TKS: .WORD 177560 ;; TTY KBD STATUS
2684 .IIF NDF $TKB,$TKB:
2685 .IIF EQ .-$TKB,$TKB: .WORD 177562 ;; TTY KBD BUFFER
2686 .ENABL LSB
2687 .IF NB 7
2688 005566 000000 $TKCNT: .WORD 0 ;; NUMBER OF ITEMS IN QUEUE
2689 005570 000000 $TKQIN: .WORD 0 ;; INPUT POINTER
2690 005572 000000 $TKQOUT: .WORD 0 ;; OUTPUT POINTER
2691 005574 000007 $TKQSRV: .BLKB 7 ;; TTY KEYBOARD QUEUE
2692 $TKQEND=
2693 .IIF NE (&1),.EVEN
2694
2695 ;*TK INITIALIZE ROUTINE
2696 ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
2697 ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
2698
2699 ;*CALL:
2700 * JSR PC,$TKINT
2701 * RETURN
2702 $TKINT: CLR $TKCNT ;; CLEAR COUNT OF ITEMS IN QUEUE
2703 005604 005037 005566 005570 MOV $TKQSRV,$TKQIN ;; MOVE THE STARTING ADDRESS OF THE
2704 005610 012737 005574 005572 MOV $TKQIN,$TKQOUT ;; QUEUE INTO THE INPUT & OUTPUT POINTERS.
2705 005616 013737 005570 005572 MOV $TKSRV,$TKVEC ;; INITIALIZE THE KEYBOARD VECTOR
2706 005624 012737 005654 000060 MOV #200,$TKVEC+2 ;; "BR" LEVEL 4
2707 005632 012737 000200 000062 TST $TKB ;; CLEAR DONE FLAG
2708 005640 005777 173302 000062 MOV #100,$TKS ;; ENABLE TTY KEYBOARD INTERRUPT
2709 005644 012777 000100 173272 MOV #100,$TKS ;; RETURN TO CALLER
2710 005652 000207
2711
2712 ;*TK SERVICE ROUTINE
2713 ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
2714 ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
2715 ;*IT IN THE QUEUE.
2716 002
2717 .IF NB START1
2718 ;*IF THE CHARACTER IS A "CONTROL-C" (↑C) $TKINT IS CALLED AND
2719 ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (START1)
2720 .ENDC
2721 005654 117746 173266 $TKSRV: MOVB $TKB,-(SP) ;; PICKUP THE CHARACTER
2722 005660 042716 177600 BIC #↑C177,(SP) ;; STRIP THE JUNK
2723 .IF NB START1
2724 005664 021627 000003 CMP (SP),#3 ;; IS IT A CONTROL C?
2725 005670 001007 BNE 1$ ;; BRANCH IF NO
2726 005672 104401 006773 TYPE $CNTLC ;; TYPE A CONTROL-C (↑C)
2727 005676 004737 005604 JSR PC,$TKINT ;; INIT THE KEYBOARD

```



```

2727 005702 005726          TST      (SP)+      ;; CLEAN UP STACK
2728 005704 000137 001460   JMP      START1    ;; CONTROL C RESTART
2729          001
2730          002
2731 005710 021627 000007   .ENDC
2732 005714 001004          .IF NE $SETUP&100
2733 005716 022737 000176 001140   IS:    CMP      (SP), #7      ;; IS IT A CONTROL G?
2734 005724 001500          BNE     2$         ;; BRANCH IF NO
2735          001          CMP      #SWREG, SWR    ;; IS SOFT-SWR SELECTED?
2736          001          BEQ     6$         ;; GO TO SWR CHANGE
2737 005726          .ENDC
2738          .IIF NE $SETUP&100, 2$:
2739          .IIF EQ $SETUP&100, 1$:
2740 005726 022737 000007 005566   CMP      #7, $STKCNT  ;; IS THE QUEUE FULL?
2741 005736 104401 006767   BNE     3$         ;; BRANCH IF NO
2742 005742 005726          TYPE     $BELL      ;; RING THE TTY BELL
2743 005744 000451          TST     (SP)+      ;; CLEAN CHARACTER OFF OF STACK
2744 005746 021627 000023   BR      5$         ;; EXIT
2745 005752 001021          3$:    CMP      (SP), #23    ;; IS IT A CONTROL-S?
2746 005754 005077 173164   BNE     32$        ;; BRANCH IF NO
2747 005760 005726          CLR     @STKS      ;; DISABLE TTY KEYBOARD INTERRUPTS
2748 005762 105777 173156   TST     (SP)+      ;; CLEAN CHAR OFF STACK
2749 005766 100375          31$:  TSTB    @STKS      ;; WAIT FOR A CHAR
2750 005770 117746 173152   BPL     31$        ;; LOOP UNTIL ITS THERE
2751 005774 042716 177600   MOVB   @STKB, -(SP)  ;; GET THE CHARACTER
2752 006000 022627 000021   BIC    #C177, (SP)  ;; MAKE IT 7-BIT ASCII
2753 006004 001366          CMP     (SP)+, #21  ;; IS IT A CONTROL-Q?
2754 006006 012777 000100 173130   BNE     31$        ;; BRANCH IF NO
2755 006014 000002          MOV     #100, @STKS ;; REENABLE TTY KEYBOARD INTERRUPTS
2756 006016 005237 005566   RTI
2757 006022 021627 000140   32$:  INC     $STKCNT    ;; COUNT THIS CHARACTER
2758 006026 002405          CMP     (SP), #140  ;; IS IT UPPER CASE?
2759 006030 021627 000175   BLT     4$         ;; BRANCH IF YES
2760 006034 003002          CMP     (SP), #175  ;; IS IT A SPECIAL CHAR?
2761 006036 042716 000040   BGT     4$         ;; BRANCH IF YES
2762 006042 112677 177522   BIC    #40, (SP)    ;; MAKE IT UPPER CASE
2763 006046 005237 005570          MOVB   (SP)+, @STKQIN ;; AND PUT IT IN QUEUE
2764 006052 023727 005570 005603   INC     $STKQIN     ;; UPDATE THE POINTER
2765 006060 001003          CMP     $STKQIN, @STKQEND ;; GO OFF THE END?
2766 006062 012737 005574 005570   BNE     5$         ;; BRANCH IF NO
2767 006070 000002          MOV     #STKQSR, $STKQIN ;; RESET THE POINTER
2768          000          RTI
2769          .ENDC
2770          001
2771          002
2772          .IF NE $SETUP&100
2773          .IF B
2774          *****
2775          .IIF
2776          .NLIST
2777          .REPT
2778          .LIST
2779          *****
2780          .NLIST
2781          .ENDR
2782          .LIST
2783          .ENDC
2784          .IF NB 7

```



```

2783 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2784 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2785 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP
2786 ;*CALL WHEN OPERATING IN TTY INTERRUPT MODE.
2787 006072 022737 000176 001140 $CKSWR: CMP #SWREG,SWR ;: IS THE SOFT-SWR SELECTED
2788 006100 001124 ;: BNE 15$ ;: EXIT IF NOT
2789 006102 105777 173036 ;: TSTB @STKS ;: IS A CHAR WAITING?
2790 006106 100121 ;: BPL 15$ ;: IF NOT, EXIT
2791 006110 117746 173032 ;: MOVB @STKB,-(SP) ;: YES
2792 006114 042716 177600 ;: BIC #1C17,(SP) ;: MAKE IT 7-BIT ASCII
2793 006120 021627 000007 ;: CMP (SP),#7 ;: IS IT A CONTROL-G?
2794 006124 001300 ;: BNE 2$ ;: IF NOT, PUT IT IN THE TTY QUEUE
2795 ;: AND EXIT
2796
2797 003 .IF B
2798 ;:*****
2799 .IFF
2800 .NLIST
2801 .REPT
2802 .LIST
2803 ;:*****
2804 .NLIST
2805 .ENDR
2806 .LIST
2807 002 .ENDC
2808 ;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE
2809 ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A
2810 ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.
2811 006126 123727 001134 000001 6$: CMPB $AUTOB,#1 ;: ARE WE RUNNING IN AUTO-MODE?
2812 006134 001674 ;: BEQ 2$ ;: BRANCH IF YES
2813 006136 005726 ;: TST (SP)+ ;: CLEAR CONTROL-G OFF STACK
2814 006140 004737 005604 ;: JSR PC,$TKINT ;: FLUSH THE TTY INPUT QUEUE
2815 006144 005077 172774 ;: CLR @STKS ;: DISABLE TTY KEYBOARD INTERRUPTS
2816 006150 112737 000001 001135 ;: MOVB #1,$INTAG ;: SET INTERRUPT MODE INDICATOR
2817
2818 .IFF
2819 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
2820 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
2821 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
2822 ;*WHEN OPERATING IN TTY FLAG MODE.
2823 $CKSWR: CMP #SWREG,SWR ;: IS THE SOFT-SWR SELECTED?
2824 ;: BNE 15$ ;: BRANCH IF NO
2825 ;: TSTB @STKS ;: CHAR THERE?
2826 ;: BPL 15$ ;: IF NO, DON'T WAIT AROUND
2827 ;: MOVB @STKB,-(SP) ;: SAVE THE CHAR
2828 ;: BIC #1C17,(SP) ;: STRIP-OFF THE ASCII
2829 ;: CMP #7,(SP)+ ;: IS IT A CONTROL G?
2830 ;: BNE 15$ ;: NO, RETURN TO USER
2831 ;: CMPB $AUTOB,#1 ;: ARE WE RUNNING IN AUTO-MODE?
2832 ;: BEQ 15$ ;: BRANCH IF YES
2833
2834 .IFTF
2835 006156 104401 007005 ;: SGTSWR: TYPE , $CNTLG ;: ECHO THE CONTROL-G (+G)
2836 006162 104401 007012 ;: MOV $MSWR ;: TYPE CURRENT CONTENTS
2837 006166 013746 000176 ;: MOV SWREG,-(SP) ;: SAVE SWREG FOR TYPEOUT
2838 ;: .IIF NB (>), ;:

```



2839	006172	104402			TYPOC		:: GO TYPE--OCTAL ASCII(ALL DIGITS)
2840	006174	104401	007023		TYPE	, \$MNEW	:: PROMPT FOR NEW SWR
2841	006200	005046		19\$:	CLR	-(SP)	:: CLEAR COUNTER
2842	006202	005046			CLR	-(SP)	:: THE NEW SWR
2843	006204	105777	172734	7\$:	TSTB	\$STKS	:: CHAR THERE?
2844	006210	100375			BPL	7\$	:: IF NOT TRY AGAIN
2845							
2846	006212	117746	172730		MOVB	\$STKB, -(SP)	:: PICK UP CHAR
2847	006216	042716	177600		BIC	#1C177, (SP)	:: MAKE IT 7-BIT ASCII
2848							
2849		003			.IF NB START1		
2850	006222	021627	000003		CMP	(SP), #3	:: IS IT A CONTROL-C?
2851	006226	001015			BNE	9\$	:: BRANCH IF NOT
2852	006230	104401	006773		TYPE	\$CNTLC	:: YES, ECHO CONTROL-C (↑C)
2853	006234	062706	000006		ADD	#6, SP	:: CLEAN UP STACK
2854	006240	123727	001135	000001	CMPB	\$INTAG, #1	:: REENABLE TTY KEYBOARD INTERRUPTS?
2855	006246	001003			BNE	8\$	:: BRANCH IF NO
2856	006250	012777	000100	172666	MOV	#100, \$STKS	:: ALLOW TTY KEYBOARD INTERRUPTS
2857	006256	000137	001460	8\$:	JMP	START1	:: CONTROL-C RESTART
2858		002			.ENDC		
2859							
2860							
2861	006262	021627	000025	9\$:	CMP	(SP), #25	:: IS IT A CONTROL-U?
2862	006266	001005			BNE	10\$	:: BRANCH IF NOT
2863	006270	104401	007000		TYPE	\$CNTLU	:: YES, ECHO CONTROL-U (↑U)
2864	006274	062706	000006	20\$:	ADD	#6, SP	:: IGNORE PREVIOUS INPUT
2865	006300	000737			BR	19\$	:: LET'S TRY IT AGAIN
2866							
2867							
2868	006302	021627	000015	10\$:	CMP	(SP), #15	:: IS IT A <CR>?
2869	006306	001022			BNE	16\$	:: BRANCH IF NO
2870	006310	005766	000004		TST	4(SP)	:: YES, IS IT THE FIRST CHAR?
2871	006314	001403			BEQ	11\$	:: BRANCH IF YES
2872	006316	016677	000002	172614	MOV	2(SP), \$SWR	:: SAVE NEW SWR
2873	006324	062706	000006	11\$:	ADD	#6, SP	:: CLEAR UP STACK
2874	006330	104401	001161	14\$:	TYPE	\$SCRLF	:: ECHO <CR> AND <LF>
2875	006334	123727	001135	000001	CMPB	\$INTAG, #1	:: RE-ENABLE TTY KBD INTERRUPTS?
2876	006342	001003			BNE	15\$	:: BRANCH IF NOT
2877	006344	012777	000100	172572	MOV	#100, \$STKS	:: RE-ENABLE TTY KBD INTERRUPTS
2878	006352	000002		15\$:	RTI		:: RETURN
2879	006354	004737	005044	16\$:	JSR	PC, \$TYPEC	:: ECHO CHAR
2880	006360	021627	000060		CMP	(SP), #60	:: CHAR < 0?
2881	006364	002420			BLT	18\$	:: BRANCH IF YES
2882	006366	021627	000067		CMP	(SP), #67	:: CHAR > ??
2883	006372	003015			BGT	18\$	:: BRANCH IF YES
2884	006374	042726	000060		BIC	#60, (SP)+	:: STRIP-OFF ASCII
2885	006400	005766	000002		TST	2(SP)	:: IS THIS THE FIRST CHAR
2886	006404	001403			BEQ	17\$	:: BRANCH IF YES
2887	006406	006316			ASL	(SP)	:: NO, SHIFT PRESENT
2888	006410	006316			ASL	(SP)	:: CHAR OVER TO MAKE
2889	006412	006316			ASL	(SP)	:: ROOM FOR NEW ONE.
2890	006414	005266	000002	17\$:	INC	2(SP)	:: KEEP COUNT OF CHAR
2891	006420	056616	177776		BIS	-2(SP), (SP)	:: SET IN NEW CHAR
2892	006424	000667			BR	7\$	:: GET THE NEXT ONE
2893	006426	104401	001160	18\$:	TYPE	\$QUES	:: TYPE ?<CR><LF>
2894	006432	000720			BR	20\$	:: SIMULATE CONTROL-U

```

2895          001          .ENDC
2896          000          .ENDC
2897          .DSABL  LSB
2898
2899
2900          001          .IF B
2901          ;*****
2902          .IFF
2903          .NLIST
2904          .REPT
2905          .LIST
2906          ;*****
2907          .NLIST
2908          .ENDR
2909          .LIST
2910          000          .ENDC
2911          001          .IF NB ?
2912          .IFTF
2913          ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2914          ;*CALL:
2915          .IFT
2916          ;*      RDCHR          ;;GET A CHARACTER FROM THE QUEUE
2917          .IFF
2918          ;*      RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
2919          .IFTF
2920          ;*      RETURN HERE      ;;CHARACTER IS ON THE STACK
2921          ;*                      ;;WITH PARITY BIT STRIPPED OFF
2922          ;
2923          ;
2924          .IFT
2925          006434 011646 $RDCHR: MOV      (SP), -(SP)      ;; PUSH DOWN THE PC AND
2926          006436 016666          MOV      4(SP), 2(SP)      ;; THE PS
2927          006444 005066 000004 000002 CLR      4(SP)          ;; GET READY FOR A CHARACTER
2928          .IF IDN <#0>, #0
2929          006450 005046          CLR      -(SP)          ;; PUT NEW PS ON STACK
2930          .IFF
2931          .ENDC          MOV      #0, -(SP)          ;; PUT NEW PS ON STACK
2932          001
2933          006452 012746 006460          MOV      #64$, -(SP)      ;; PUT NEW PC ON STACK
2934          006456 000002          RTI          ;; POP NEW PC AND PS
2935          006460          64$:
2936          006460 005737 005566          1$:  TST      $STKCNT      ;; WAIT ON A CHARACTER
2937          006464 001775          BEQ      1$
2938          006466 005337 005566          DEC      $STKCNT      ;; DECREMENT THE COUNTER
2939          006472 117766 177074 000004 MOVB    @STKQOUT, 4(SP)  ;; GET ONE CHARACTER
2940          006500 005237 005572          INC      $STKQOUT      ;; UPDATE THE POINTER
2941          006504 023727 005572 005603 CMP      $STKQOUT, #STKGEND ;; DID IT GO OFF OF THE END?
2942          006512 001003          BNE     2$          ;; BRANCH IF NO
2943          006514 012737 005574 005572 MOV      #STKQSR, $STKQOUT ;; RESET THE POINTER
2944          006522 000002          RTI          ;; RETURN
2945          .IFF
2946          $RDCHR: MOV      (SP), -(SP)      ;; PUSH DOWN THE PC
2947          MOV      4(SP), 2(SP)      ;; SAVE THE PS
2948          1$:  TSTB    @STKS          ;; WAIT FOR
2949          BPL     1$          ;; A CHARACTER
2950          MOVB   @STKB, 4(SP)      ;; READ THE TTY

```



```

2951          BIC      #1C<177>,4(SP)    ;; GET RID OF JUNK IF ANY
2952          CMP      4(SP),#23          ;; IS IT A CONTROL-S?
2953          BNE      3$                  ;; BRANCH IF NO
2954          2$:      TSTB   2$TKS        ;; WAIT FOR A CHARACTER
2955          BPL      2$                  ;; LOOP UNTIL ITS THERE
2956          MOVB   2$TKB,-(SP)          ;; GET CHARACTER
2957          BIC      #1C177,(SP)        ;; MAKE IT 7-BIT ASCII
2958          CMP      (SP)+,#21          ;; IS IT A CONTROL-Q?
2959          BNE      2$                  ;; IF NOT DISCARD IT
2960          BR      1$                  ;; YES, RESUME
2961          3$:      CMP      4(SP),#140   ;; IS IT UPPER CASE?
2962          BLT      4$                  ;; BRANCH IF YES
2963          CMP      4(SP),#175         ;; IS IT A SPECIAL CHAR?
2964          BGT      4$                  ;; BRANCH IF YES
2965          BIC      #40,4(SP)          ;; MAKE IT UPPER CASE
2966          RTI                          ;; GO BACK TO USER
2967          4$:      .ENDC
2968          .IF NB 7
2969          .IFF
2970          .NLIST
2971          $RDSZ=B.
2972          .LIST
2973          .ENDC
2974          .IF GT $RDSZ-1
2975          .IF B
2976          ;*****
2977          .IFF
2978          .NLIST
2979          .REPT
2980          .LIST
2981          ;*****
2982          .NLIST
2983          .ENDR
2984          .LIST
2985          .ENDC
2986          001
2987          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2988          ;*CALL:
2989          ;*      RDLIN
2990          ;*      RETURN HERE
2991          ;*      ;; INPUT A STRING FROM THE TTY
2992          ;*      ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2993          ;*      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2994          006524 010346 $RDLIN: MOV      R3,-(SP)    ;; SAVE R3
2995          .IF NB <X>
2996          .ENDC
2997          006526 005046 .IF NB <X>
2998          CLR      -(SP)                ;; CLEAR THE RUBOUT KEY
2999          006530 012703 006760 1$:      MOV      #$TTYIN,R3    ;; GET ADDRESS
3000          .IF B <7>
3001          2$:      CMP      #$TTYIN+8.,R3    ;; BUFFER FULL?
3002          .IFF
3003          2$:      CMP      #$TTYIN+7,R3    ;; BUFFER FULL?
3004          .ENDC
3005          BLOS     4$                    ;; BR IF YES
3006          RDCHR   (SP)+,(R3)            ;; GO READ ONE CHARACTER FROM THE TTY
3007          MOVB   (SP)+,(R3)            ;; GET CHARACTER
3008          .IF B 7
3009          .IF NB START1

```

```

3007                                CMPB    #3,(R3)                ;; IS IT A CONTROL-C?
3008                                BNE     10$                    ;; BRANCH IF NO
3009                                TYPE    $CNTLC                ;; TYPE A CONTROL-C (↑C)
3010                                .IIF NB <X>, ↑ST (SP)+          ;; CLEAN RUBOUT KEY OFF OF THE STACK
3011                                MOV     (SP)+,R3                ;; RESTORE R3
3012                                JMP     START1                 ;; GOTO CONTROL-C RESTART
3013                                .ENDC
3014                                .ENDC
3015 006546 122713 000177          10$: CMPB    #177,(R3)                ;; IS IT A RUBOUT
3016                                .IF NB <X>
3017                                BNE     5$                    ;; BR IF NO
3018 006552 001022                    TST     (SP)                ;; IS THIS THE FIRST RUBOUT?
3019 006554 005716                    BNE     6$                    ;; BR IF NO
3020 006556 112737 000134 006756  MOVB    #' \,9$                ;; TYPE A BACK SLASH
3021 006566 104401 006756          TYPE    9$
3022 006572 012716 177777          MOV     #-1,(SP)            ;; SET THE RUBOUT KEY
3023 006576 005303                    DEC     R3                    ;; BACKUP BY ONE
3024 006600 020327 006760          CMP     R3,#$TTYIN          ;; STACK EMPTY?
3025 006604 103434                    BLO     4$                    ;; BR IF YES
3026 006606 111337 006756          MOVB    (R3),9$            ;; SETUP TO TYPEOUT THE DELETED CHAR.
3027 006612 104401 006756          TYPE    9$                    ;; GO TYPE
3028 006616 000746                    BR     2$                    ;; GO READ ANOTHER CHAR.
3029 006620 005716                    TST     (SP)                ;; RUBOUT KEY SET?
3030 006622 001406                    BEQ     7$                    ;; BR IF NO
3031 006624 112737 000134 006756  MOVB    #' \,9$                ;; TYPE A BACK SLASH
3032 006632 104401 006756          TYPE    9$
3033 006636 005016                    CLR     (SP)                ;; CLEAR THE RUBOUT KEY
3034 006640 122713 000025          7$: CMPB    #25,(R3)            ;; IS CHARACTER A CTRL U?
3035 006644 001003                    BNE     8$                    ;; BR IF NO
3036 006646 104401 007000          TYPE    $CNTLU              ;; TYPE A CONTROL "U"
3037 006652 000726                    BR     1$                    ;; GO START OVER
3038 006654 122713 000022          8$: CMPB    #22,(R3)            ;; IS CHARACTER A "↑R"?
3039 006660 001011                    BNE     3$                    ;; BRANCH IF NO
3040 006662 105013                    CLRB   (R3)                ;; CLEAR THE CHARACTER
3041 006664 104401 001161          TYPE    $SCRLF              ;; TYPE A "CR" & "LF"
3042 006670 104401 006760          TYPE    $TTYIN              ;; TYPE THE INPUT STRING
3043 006674 000717                    BR     2$                    ;; GO PICKUP ANOTHER CHARACTER
3044                                .IFF
3045                                BNE     3$                    ;; SKIP IF NOT
3046                                .ENDC
3047 006676 104401 001160          4$: TYPE    $QUES              ;; TYPE A '??'
3048 006702 000712                    BR     1$                    ;; CLEAR THE BUFFER AND LOOP
3049 006704 111337 006756          3$: MOVB    (R3),9$            ;; ECHO THE CHARACTER
3050 006710 104401 006756          TYPE    9$
3051 006714 122723 000015          CMPB    #15,(R3)+          ;; CHECK FOR RETURN
3052 006720 001305                    BNE     2$                    ;; LOOP IF NOT RETURN
3053 006722 105063 177777          CLRB   -1(R3)              ;; CLEAR RETURN (THE 15)
3054 006726 104401 001162          TYPE    $LF                  ;; TYPE A LINE FEED
3055 006732 005726                    .IIF NB <X>, ↑ST (SP)+          ;; CLEAN RUBOUT KEY FROM THE STACK
3056 006734 012603                    MOV     (SP)+,R3            ;; RESTORE R3
3057 006736 011646                    MOV     (SP),-(SP)          ;; ADJUST THE STACK AND PUT ADDRESS OF THE
3058 006740 016666 000004 000002  MOV     4(SP),2(SP)          ;; FIRST ASCII CHARACTER ON IT
3059 006746 012766 006760 000004  MOV     #$TTYIN,4(SP)
3060 006754 000002                    RTI
3061 006756 000                    9$: .BYTE    0                ;; RETURN
3062 006757 000                    .BYTE    0                ;; STORAGE FOR ASCII CHAR. TO TYPE
                                     ;; TERMINATOR

```



```

3063          002          .IF B <7>
3064          $TTYIN: .BLKB  8.          ;;RESERVE 8 BYTES FOR TTY INPUT
3065          .IFF
3066 006760 000007 $TTYIN: .BLKB  7          ;;RESERVE 7 BYTES FOR TTY INPUT
3067          001          .ENDC
3068          .IIF NDF $QUES,$QUES:
3069          .IIF EQ .-$QUES,$QUES: .ASCII  "?"          ;;QUESTION MARK
3070          .IIF NDF $CRLF,$CRLF:
3071          .IIF EQ .-$CRLF,$CRLF: .ASCII  <15>          ;CARRIAGE RETURN
3072          .IIF NDF $LF,$LF:
3073          .IIF EQ .-$LF,$LF: .ASCIZ  <12>          ;;LINE FEED
3074          000          .ENDC
3075          001          .IF NB 7
3076          .IIF NDF $BELL,$BELL:
3077 006767 207 177777 000 .IIF EQ .-$BELL,$BELL: .ASCIZ  <207><377><377> ;;CODE FOR BELL
3078          000          .ENDC
3079          001          .IF NB START1
3080          .IIF NDF $CNTLC,$CNTLC:
3081 006773 136 006503 000012 .IIF EQ .-$CNTLC,$CNTLC: .ASCIZ  /?C/<15><12>          ;;CONTROL "C"
3082          000          .ENDC
3083 007000 052536 005015 000 $CNTLU: .ASCIZ  /?U/<15><12>          ;;CONTROL "U"
3084 007005 136 006507 000012 $CNTLG: .ASCIZ  /?G/<15><12>          ;;CONTROL "G"
3085 007012 005015 053523 020122 $MSWR: .ASCIZ  <15><12>/SWR = /
3086 007020 020075 000
3087 007023 040 047040 053505 $MNEW: .ASCIZ  / NEW = /
3088 007030 036440 000040
3089          001
3090          .IF NE $SETUP & 100
3091          .IIF NDF $AUTOB,$AUTOB:
3092          .IIF EQ .-$AUTOB,$AUTOB: .BYTE  0          ;;AUTO MODE FLAG
3093          .IIF NDF $INTAG,$INTAG:
3094          .IIF EQ .-$INTAG,$INTAG: .BYTE  0          ;;INTERRUPT MODE FLAG
3095          .ENDC
3096          .IIF NE 18..EVEN
3097          .SBTTL  RANDOM NUMBER GENERATOR ROUTINE
3098          001
3099          .IF B
3100          ;*****
3101          .IFF
3102          .NLIST
3103          .REPT
3104          .LIST
3105          ;*****
3106          .NLIST
3107          .ENDR
3108          .LIST
3109          .ENDC
3110          ;*THIS ROUTINE IS A DOUBLE PRECISION PSEUDO RANDOM NUMBER GENERATOR
3111          ;*WITH A RANGE OF 0 TO 2(+33)-1.
3112          ;*CALL:
3113          ;*      JSR      PC,$RAND          ;;CALL THE ROUTINE
3114          ;*      RETURN          ;;RETURN HERE THE RANDOM
3115          ;*                      ;;NUMBER WILL BE IN
3116          ;*                      ;;$HINUM,$LONUM
3117 007034
3118 007034 010046 $RAND: MOV      RO,-(SP)          ;;PUSH RO ON STACK

```

3119 007036 010146  
3120 007040 010246  
3121 007042 013700 007134  
3122 007046 013701 007132  
3123 007052 012702 177771  
3124 007056 006300  
3125 007060 006101  
3126 007062 005202  
3127 007064 001374  
3128 007066 063700 007134  
3129 007072 005501  
3130 007074 063701 007132  
3131 007100 062700 001057  
3132 007104 005501  
3133 007106 062701 047401  
3134 007112 010037 007134  
3135 007116 010137 007132  
3136 007122 012602  
3137 007124 012601  
3138 007126 012600  
3139 007130 000207  
3140 007132 176543  
3141 007134 123456  
3142  
3143  
3144 001  
3145  
3146  
3147  
3148  
3149  
3150  
3151  
3152  
3153  
3154 000  
3155  
3156  
3157  
3158  
3159  
3160  
3161  
3162  
3163  
3164  
3165  
3166  
3167  
3168  
3169  
3170  
3171  
3172  
3173  
3174

```

MOV R1,-(SP) ;; PUSH R1 ON STACK
MOV R2,-(SP) ;; PUSH R2 ON STACK
MOV $LONUM,R0 ;; SET R0 WITH LOW
MOV $HINUM,R1 ;; SET R1 WITH HIGH
MOV #-7,R2 ;; SET SHIFT COUNT
1$: ASL R0 ;; SHIFT R0 LEFT AND
ROL R1 ;; ROTATE CARRY INTO R1 AND
INC R2 ;; CHECK FOR DONE
BNE 1$ ;; CONTINUE SHIFT LOOP
ADD $LONUM,R0 ;; ADD NUMBER TO MAKE X 129
ADC R1 ;; PROPOGATE CARRY
ADD $HINUM,R1 ;; ADD NUMBER TO MAKE X 129
ADD #1057,R0 ;; ADD LOW CONSTANT
ADC R1 ;; PROPOGATE CARRY
ADD #47401,R1 ;; ADD HIGH CONSTANT
MOV R0,$LONUM ;; SAVE R0
MOV R1,$HINUM ;; SAVE R1
MOV (SP)+,R2 ;; POP STACK INTO R2
MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,R0 ;; POP STACK INTO R0
RTS PC ;; RETURN

```

\$HINUM: .WORD 176543  
\$LONUM: .WORD 123456  
.SBTL INTEGER DIVIDE ROUTINE

.IF B  
;\*\*\*\*\*

.IFF  
.NLIST  
.REPT  
.LIST  
;\*\*\*\*\*

.NLIST  
.ENDR  
.LIST  
.ENDC

\*THIS ROUTINE WILL DIVIDE A 32-BIT TWO'S COMPLEMENT INTEGER  
\*DIVIDEND BY A 16-BIT TWO'S COMPLEMENT INTEGER DIVISOR GIVING  
\*A 16-BIT TWO'S COMPLEMENT INTEGER QUOTIENT AND A 16-BIT REMAINDER.  
\*DIVISION WILL BE PERFORMED SO THAT THE REMAINDER IS OF THE  
\*SAME SIGN AS THE DIVIDEND.

\*CALL:  
MOV LOW DIVIDEND,-(SP) ;; THE HIGH DIVIDEND MUST BE < 1/2  
MOV HIGH DIVIDEND,-(SP); AS LARGE AS THE DIVISOR  
MOV DIVISOR,-(SP)  
JSR PC,\$DIV  
RETURN ;; QUOTIENT & REMAINDER ARE ON THE STACK  
"V"=0 IMPLIES NO ERROR  
"V"=1 IMPLIES ERROR OCCURRED  
"C"=0 DIVIDE OVERFLOW OCCURRED  
"C"=1 ATTEMPTED TO DIVIDE BY ZERO

STACK	NO ERROR	OVERFLOW	DIVIDE BY ZERO
----	----	----	----
TOP	REMAINDER	ALL ZEROS	ALL ONES



```

3175 ;* +2 QUOTIENT ALL ZEROS ALL ONES
3176
3177 007136 $DIV: TRAP ;;PUSH OLD PSW AND PC ON STACK
3178 007136 104400
3179 001
3180 .IF NB MOV (SP)+, ;;SAVE THE PSW IN
3181 .ENDC
3182 007140 042716 000017 BIC #17,(SP) ;;STRIP AWAY CONDITION CODES
3183 007144 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
3184 007146 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
3185 007150 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
3186 007152 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
3187 007154 005046 CLR -(SP) ;;SAVE A PLACE FOR SIGNS
3188 007156 012746 000021 MOV #17,-(SP) ;;SETUP THE ITERATION COUNTER
3189 007162 016601 000024 MOV 24(SP),R1 ;;PICKUP THE DIVIDEND
3190 007166 016600 000022 MOV 22(SP),RO
3191 007172 100005 BPL 1$ ;;CHECK THE SIGN
3192 007174 105366 000003 DECB 3(SP) ;;KEEP TRACK OF THE SIGN
3193 007200 005400 NEG RO ;;AND NEGATE THE ORIGINAL
3194 007202 005401 NEG R1 ;;NUMBER
3195 007204 005600 RO
3196 007206 016602 000020 1$: MOV 20(SP),R2 ;;PICKUP THE DIVISOR
3197 007212 002407 BLT 2$ ;;CHECK THE SIGN
3198 007214 003011 BGT 3$ ;;DIVISOR OF 0 IS A NO-NO
3199 007216 052766 000003 000014 BIS #3,14(SP) ;;SET "V" & "C"
3200 007224 012700 177777 MOV #-1,RO ;;SET REMAINDER TO ALL ONES
3201 007230 000424 BR 7$ ;;EXIT
3202 007232 005266 000002 2$: INC 2(SP) ;;KEEP TRACK OF DIVISORS SIGN
3203 007236 000401 BR 4$
3204 007240 005402 3$: NEG R2 ;;NEGATE THE ORIGINAL NUMBER
3205 007242 000241 4$: CLC ;;CLEAR "C"
3206 007244 000405 BR 6$ ;;START FORMING QUOTIENT
3207 007246 006100 5$: ROL RO ;;POSITION MSB'S
3208 007250 010003 MOV RO,R3 ;;COPY
3209 007252 060203 ADD R2,R3 ;;COMPARE DIVIDEND & DIVISOR
3210 007254 103001 BCC 6$ ;;BR IF DIVIDEND > DIVISOR
3211 007256 010300 MOV R3,RO ;;REMAINDER AFTER THIS LOOP
3212 007260 006101 6$: ROL R1 ;;QUOTIENT BIT ENTERS HERE
3213 007262 005316 DEC (SP) ;;DONE?
3214 007264 001370 BNE 5$ ;;BR IF NO
3215 007266 005701 TST R1 ;;OVERFLOW?
3216 007270 100005 BPL 8$ ;;BR IF NO
3217 007272 052766 000002 000014 BIS #2,14(SP) ;;SET "V" IN RETURN STATUS WORD
3218 007300 005000 CLR RO ;;SET REMAINDER TO ALL ZEROS
3219 007302 010001 7$: MOV RO,R1 ;;COPY REMAINDER INTO QUOTIENT
3220 007304 005726 8$: TST (SP)+ ;;CLEAR COUNTER FROM STACK
3221 007306 005716 TST (SP) ;;REMAINDER SIGN CORRECTION NEEDED?
3222 007310 002004 BGE 9$ ;;BR IF NO
3223 007312 005400 NEG RO ;;NEGATE REMAINDER
3224 007314 105066 000001 CLRB 1(SP) ;;CLEAR SIGN
3225 007320 005316 DEC (SP) ;;BUT DON'T FORGET QUOTIENT
3226 007322 005726 9$: TST (SP)+ ;;QUOTIENT SIGN CORRECTION NEEDED?
3227 007324 001401 BEQ 10$ ;;BR IF NO
3228 007326 005401 NEG R1 ;;NEGATE QUOTIENT
3229 007330 010166 000020 10$: MOV R1,20(SP) ;;RETURN QUOTIENT AND
3230 007334 010066 000016 MOV RO,16(SP) ;;REMAINDER TO USER

```

```

3231 007340 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
3232 007342 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
3233 007344 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
3234 007346 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
3235 007350 012666 000002  MOV      (SP)+,2(SP)    ;; SETUP TO RETURN CONDITION CODES
3236 007354 000002      RTI                    ;; RETURN
3237                                     .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
3238
3239                                     .IF B
3240                                     ;*****
3241                                     .IFF
3242                                     .NLIST
3243                                     .REPT
3244                                     .LIST
3245                                     ;*****
3246                                     .NLIST
3247                                     .ENDR
3248                                     .LIST
3249                                     .ENDC
3250                                     ;*SAVE R0-R5
3251                                     ;*CALL:
3252                                     ;*   SAVREG
3253                                     ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3254                                     ;*
3255                                     ;*TOP---(+16)
3256                                     ;* +2---(+18)
3257                                     ;* +4---R5
3258                                     ;* +6---R4
3259                                     ;* +8---R3
3260                                     ;*+10---R2
3261                                     ;*+12---R1
3262                                     ;*+14---R0
3263
3264 007356      $SAVREG:
3265 007356 010046      MOV      R0,-(SP)      ;; PUSH R0 ON STACK
3266 007360 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
3267 007362 010246      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
3268 007364 010346      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
3269 007366 010446      MOV      R4,-(SP)      ;; PUSH R4 ON STACK
3270 007370 010546      MOV      R5,-(SP)      ;; PUSH R5 ON STACK
3271 007372 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PS OF MAIN FLOW
3272 007376 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PC OF MAIN FLOW
3273 007402 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PS OF CALL
3274 007406 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PC OF CALL
3275 007412 000002      RTI
3276
3277                                     ;*RESTORE R0-R5
3278                                     ;*CALL:
3279                                     ;*   RESREG
3280 007414      $RESREG:
3281 007414 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PC OF CALL
3282 007420 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PS OF CALL
3283 007424 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PC OF MAIN FLOW
3284 007430 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PS OF MAIN FLOW
3285 007434 012605      MOV      (SP)+,R5      ;; POP STACK INTO R5
3286 007436 012604      MOV      (SP)+,R4      ;; POP STACK INTO R4

```



```

3287 007440 012603      MOV      (SP)+,R3      ;; POP STACK INTO R3
3288 007442 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
3289 007444 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
3290 007446 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
3291 007450 000002      RTI
3292      .SBTTL TRAP DECODER
3293
3294      001      .IF B
3295      ;*****
3296      .IFF
3297      .NLIST
3298      .REPT
3299      .LIST
3300      ;*****
3301      .NLIST
3302      .ENDR
3303      .LIST
3304      000      .ENDC
3305      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
3306      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3307      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3308      ;*GO TO THAT ROUTINE.
3309
3310      001      .IF B
3311 007452 010046 $TRAP: MOV      RO,-(SP)      ;;SAVE RO
3312      .IFF
3313 $TRAP: MOV      2(SP),-(SP)    ;; ASSUME THE STATUS OF
3314      BIC      #20,(SP)        ;; THE CALLER--DO NOT ALLOW
3315      MOV      #1$,-(SP)      ;; T-BIT TRAPS
3316      RTI
3317      1$:      MOV      RO,-(SP)    ;;SET THE NEW STATUS
3318      .ENDC      ;;SAVE RO
3319 007454 016600 000002      MOV      2(SP),RO      ;; GET TRAP ADDRESS
3320 007460 005740      TST      -(RO)         ;; BACKUP BY 2
3321 007462 111000      MOVVB   (RO),RO        ;; GET RIGHT BYTE OF TRAP
3322      001
3323      .IF NB
3324      BPL      $TRAP1        ;; NON-USER TRAP BELOW 200
3325      BIC      #1C177,RO     ;; STRIP AWAY THE JUNK
3326      JMP      (PC)          ;; USER TRAP ABOVE 177, GO TO
3327      .WORD
3328 $TRAP1:
3329      .ENDC
3330      .IF NB
3331      CMP      #1$TERM,RO    ;; CHECK FOR OUT OF BOUNDS
3332      BGT      .+6          ;; BR IF OK
3333      HALT
3334      BR      .-2          ;; OUT OF BOUNDS
3335      .ENDC      ;; HANGUP
3336 007464 006300      ASL      RO              ;; POSITION FOR INDEXING
3337 007466 016000 007506      MOV      $TRPAD(RO),RO   ;; INDEX TO TABLE
3338 007472 000200      RTS      RO              ;; GO TO ROUTINE
3339
3340      ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
3341
3342 007474 011646 $TRAP2: MOV      (SP),-(SP)    ;; MOVE THE PC DOWN

```

3343 007476 016666 000004 000002  
 3344 007504 000002  
 3345  
 3346  
 3347  
 3348  
 3349  
 3350  
 3351  
 3352  
 3353  
 3354  
 3355  
 3356  
 3357  
 3358  
 3359  
 3360  
 3361  
 3362  
 3363  
 3364  
 3365  
 3366  
 3367  
 3368  
 3369  
 3370  
 3371 001  
 3372 002  
 3373  
 3374  
 3375  
 3376  
 3377  
 3378  
 3379  
 3380 007506 007474  
 3381 001  
 3382 007510 004674  
 3383 000  
 3384 001  
 3385 002  
 3386  
 3387  
 3388  
 3389  
 3390  
 3391  
 3392  
 3393  
 3394  
 3395 007512 001  
 3396 005140  
 3397 002  
 3398

```

MOV 4(SP),2(SP) ;;MOVE THE PSW DOWN
RTI ;;RESTORE THE PSW

.MACRO SETTRAP A,B,MSG
$$SET A,B,\(TRAP+STRP),\STRP,<MSG>
.NLIST
STRP=STRP+1
.LIST
.ENDM SETTRAP
.MACRO $$SET A,B,C,D,COMNT
.IF EQ STRP-1
.SBTTL TRAP TABLE

; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
; ROUTINE
; -----
STRPAD: .WORD STRAP2
.ENDC
.IIF NDF GNS,.NLIST
A= C
.IIF NDF GNS,.LIST
B ;;CALL=A TRAP+D(C) COMNT
.ENDM $$SET
.MACRO TRMTRP
$TERM=-STRPAD
.ENDM TRMTRP
.IF DF $TYPE
.IF EQ STRP-1
.SBTTL TRAP TABLE

; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
; ROUTINE
; -----
STRPAD: .WORD STRAP2
.ENDC
$TYPE ;;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
.IF DF $TYPOC
.IF EQ STRP-1
.SBTTL TRAP TABLE

; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
; *BY THE "TRAP" INSTRUCTION.
; ROUTINE
; -----
STRPAD: .WORD STRAP2
.ENDC
$TYPOC ;;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
.IF EQ STRP-1
.SBTTL TRAP TABLE

```



3399  
3400  
3401  
3402  
3403  
3404  
3405  
3406  
3407  
3408  
3409  
3410  
3411  
3412  
3413  
3414  
3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428  
3429  
3430  
3431  
3432  
3433  
3434  
3435  
3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
3450  
3451  
3452  
3453  
3454

007514 005114 001 002  
007516 005154 001 000 001 002  
007520 005342 001 000 001  
007522 006162 001 000  
001 002

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

ROUTINE

\$TRPAD: .WORD STRAP2  
.ENDC  
STYPOS ;;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)  
.IF EQ STRP-1  
.SBTTL TRAP TABLE

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

ROUTINE

\$TRPAD: .WORD STRAP2  
.ENDC  
STYPON ;;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)  
.ENDC  
.IF DF STYPDS  
.IF EQ STRP-1  
.SBTTL TRAP TABLE

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

ROUTINE

\$TRPAD: .WORD STRAP2  
.ENDC  
STYPDS ;;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)  
.ENDC  
.IF DF STYPBN  
SETTRAP TYPBN,STYPBN,↑/TYPE BINARY (ASCII) NUMBER/  
.ENDC  
.IF DF \$GTSWR  
.IF EQ STRP-1  
.SBTTL TRAP TABLE

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

ROUTINE

\$TRPAD: .WORD STRAP2  
.ENDC  
\$GTSWR ;;CALL=GTSWR TRAP+6(104406) GET SOFT-SWR SETTING  
.ENDC  
.IF DF \$CKSWR  
.IF EQ STRP-1  
.SBTTL TRAP TABLE

;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED

3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482  
3483  
3484  
3485  
3486  
3487  
3488  
3489  
3490  
3491  
3492  
3493  
3494  
3495  
3496  
3497  
3498  
3499  
3500  
3501  
3502  
3503  
3504  
3505  
3506  
3507  
3508  
3509  
3510

007524 006072 001  
000  
001  
002

007526 006434 001  
000  
001  
002

007530 006524 001  
000  
001  
000  
001  
000  
001  
002

007532 007356 001  
002

;\*BY THE "TRAP" INSTRUCTION.

; ROUTINE  
-----

\$TRPAD: .WORD STRAP2  
.ENDC

SCKSWR ;;CALL=CKSWR TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR

.ENDC  
.IF DF \$RDCHR  
.IF EQ \$TRP-1  
.SBTTL TRAP TABLE

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

; ROUTINE  
-----

\$TRPAD: .WORD STRAP2  
.ENDC

\$RDCHR ;;CALL=RDCHR TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE

.ENDC  
.IF DF \$RDLIN  
.IF EQ \$TRP-1  
.SBTTL TRAP TABLE

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

; ROUTINE  
-----

\$TRPAD: .WORD STRAP2  
.ENDC

\$RDLIN ;;CALL=RDLIN TRAP+11(104411) TTY TYPEIN STRING ROUTINE

.ENDC  
.IF DF \$RDOCT  
SETTRAP RDOCT,\$RDOCT,↑/READ AN OCTAL NUMBER FROM TTY/  
.ENDC  
.IF DF \$RDDEC  
SETTRAP RDDEC,\$RDDEC,↑/READ A DECIMAL NUMBER FROM TTY/  
.ENDC  
.IF DF \$SAVREG  
.IF EQ \$TRP-1  
.SBTTL TRAP TABLE

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

; ROUTINE  
-----

\$TRPAD: .WORD STRAP2  
.ENDC

\$SAVREG ;;CALL=SAVREG TRAP+12(104412) SAVE R0-R5 ROUTINE

.IF EQ \$TRP-1  
.SBTTL TRAP TABLE

;\*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED



3511  
3512  
3513  
3514  
3515  
3516  
3517  
3518  
3519  
3520  
3521  
3522  
3523  
3524  
3525  
3526  
3527  
3528  
3529  
3530  
3531  
3532  
3533  
3534  
3535  
3536  
3537  
3538  
3539  
3540  
3541  
3542  
3543  
3544  
3545  
3546  
3547  
3548  
3549  
3550  
3551  
3552  
3553  
3554  
3555  
3556

007534 007414  
001  
000  
001

000

001

000

007536 000000 000000 000000  
007544 000000

007546 000  
007547 000

```

;#BY THE "TRAP" INSTRUCTION.
;
; ROUTINE
;-----
$TRPAD: .WORD STRAP2
.ENDC
$RESREG ;;CALL=RESREG TRAP+13(104413) RESTORE R0-R5 ROUTINE
.ENDC
; IF DF $R2A
SETTRAP R2AZ,$R2AZ
SETTRAP R2AZ,$R2AZ
SETTRAP R2AZQ,$R2AZQ
.ENDC
;MACRO ERRCAL N,M
; IF DF $ESCAPE
JSR RO,ES.SAV ;SAVE THE ADDRESS IN 'ESCAPE'
.ENDC
ERROR N ;'M
.ENDM ERRCAL

.SBTTL SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)
;COPYRIGHT (C) 1976
;DIGITAL EQUIPMENT CORP.
;MAYNARD, MA 01754
;AUTHOR(S): JIM LACEY/CHUCK HESS

; IF B
;*****
; IF
.NLIST
.REPT
.LIST
;*****
.NLIST
.ENDR
.LIST
.ENDC

;STORAGE FOR RPDS1, RPER1, RPER2, AND RPER3 ON AN ERROR "2"
;RPERRS = RPDS1
;RPERRS+2 = RPER1
;RPERRS+4 = RPER2
;RPERRS+6 = RPER3
RPERRS: .WORD 0,0,0,0

;TABLE OF DRIVE ACTIVE INDICATORS (DRVACT=8 BYTES)
;DRVACT=0 IF DRIVE IS IDLE
;DRVACT>0 IF DRIVE IS ACTIVE WITH A COMMAND
;DRVACT<0 IF DRIVE IS ACTIVE WITH AN ERROR RECOVERY OPERATION
DRVACT: .BYTE 0 ;DRIVE 0
.BYTE 0 ;DRIVE 1

```

3567 007550 000  
 3568 007551 000  
 3569 007552 000  
 3570 007553 000  
 3571 007554 000  
 3572 007555 000

.BYTE 0 ;DRIVE 2  
 .BYTE 0 ;DRIVE 3  
 .BYTE 0 ;DRIVE 4  
 .BYTE 0 ;DRIVE 5  
 .BYTE 0 ;DRIVE 6  
 .BYTE 0 ;DRIVE 7

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

3578 007556 000  
 3579 007557 000  
 3580 007560 000  
 3581 007561 000  
 3582 007562 000  
 3583 007563 000  
 3584 007564 000  
 3585 007565 000

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

;TABLE OF DRIVE TYPES (DRV TYP=8 BYTES)  
 ;DRV TYP=0 IF DRIVE IS NONEXISTENT (DRVSTA=0, ALSO)  
 ;DRV TYP=1 IF DRIVE IS RP04  
 ;DRV TYP=2 IF DRIVE IS RP05  
 ;DRV TYP=4 IF DRIVE IS RP06  
 ;DRV TYP=-1 IF NOT RP04/5/6

3595 007566 000  
 3596 007567 000  
 3597 007570 000  
 3598 007571 000  
 3599 007572 000  
 3600 007573 000  
 3601 007574 000  
 3602 007575 000

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

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

3608 007576 000  
 3609 007577 000  
 3610 007600 000  
 3611 007601 000  
 3612 007602 000  
 3613 007603 000  
 3614 007604 000  
 3615 007605 000

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

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

3621 007606 000  
 3622 007607 000

DPRQS: .BYTE 0 ;DRIVE 0  
 .BYTE 0 ;DRIVE 1



```

3623 007610 000 .BYTE 0 ;DRIVE 2
3624 007611 000 .BYTE 0 ;DRIVE 3
3625 007612 000 .BYTE 0 ;DRIVE 4
3626 007613 000 .BYTE 0 ;DRIVE 5
3627 007614 000 .BYTE 0 ;DRIVE 6
3628 007615 000 .BYTE 0 ;DRIVE 7
3629
3630 ;TRANSFER WAIT FLAG (TRNSWT=1 WORD)
3631 ;THIS IS A ONE WORD QUEUE. IT WILL CONTAIN THE ADDRESS OF
3632 ;"DPB" OF THE I/O OPERATION.
3633
3634 007616 000000 TRNSWT: .WORD 0
3635
3636 ;SEARCH WAIT KEYS (SRCHWT=1 WORD)
3637 ;THIS IS A ONE WORD QUEUE THAT WILL CONTAIN A KEY FOR EACH OF
3638 ;THE DRIVES THAT ARE PERFORMING A SEARCH COMMAND FOR THE I/O
3639 ;REQUEST THAT IS AT THE TOP OF THEIR REQUEST QUEUE.
3640 ;EACH DRIVE IS ASSIGNED ONE BIT, STARTING AT BIT00 FOR DRIVE 0.
3641
3642 007620 000000 SRCHWT: .WORD 0
3643
3644 ;RPO4/5/6 DRIVER ACTIVE FLAG (ACTDRV=1 BYTE)
3645 ;ACTDRV=0 IF DRIVER IS INACTIVE
3646 ;ACTDRV>0 IF DRIVER IS ACTIVE
3647
3648 007622 000 ACTDRV: .BYTE 0
3649
3650 ;SOFTWARE TIMER ROUTINE ACTIVE FLAG (ACTSTR=1 BYTE)
3651 ;ACTSTR=0 IF SOFTWARE TIMER ROUTINE IS INACTIVE
3652 ;ACTSTR>0 IF SOFTWARE TIMER ROUTINE IS ACTIVE
3653
3654 007623 000 ACTSTR: .BYTE 0
3655
3656 ;UNLOAD FLAG (ULDFLG=8 BYTES)
3657 ;ULDFLG=0 IF NO UNLOAD COMMAND
3658 ;ULDFLG>0 IF UNLOAD COMMAND IN PROGRESS
3659 ;ULDFLG<0 IF UNLOAD COMMAND IN WAIT QUEUE
3660
3661 007624 000 ULDFLG: .BYTE 0 ;DRIVE 0
3662 007625 000 .BYTE 0 ;DRIVE 1
3663 007626 000 .BYTE 0 ;DRIVE 2
3664 007627 000 .BYTE 0 ;DRIVE 3
3665 007630 000 .BYTE 0 ;DRIVE 4
3666 007631 000 .BYTE 0 ;DRIVE 5
3667 007632 000 .BYTE 0 ;DRIVE 6
3668 007633 000 .BYTE 0 ;DRIVE 7
3669
3670 ;LOOK AHEAD COUNT (LACNT=8 BYTES)
3671 ;LACNT WILL INDICATE THE NUMBER OF LOOK AHEADS PERFORMED
3672
3673 007634 000 LACNT: .BYTE 0 ;DRIVE 0
3674 007635 000 .BYTE 0 ;DRIVE 1
3675 007636 000 .BYTE 0 ;DRIVE 2
3676 007637 000 .BYTE 0 ;DRIVE 3
3677 007640 000 .BYTE 0 ;DRIVE 4
3678 007641 000 .BYTE 0 ;DRIVE 5

```

F06

CZRJCBO, RPO4/5/6 HEAD ALNMT  
CZRJCB.P11 04-NOV-77 12:49

MACY11 30(1046) 04-NOV-77 17:31 PAGE 70  
SINGLE/DUAL PORT RH11/RPO4/5/6 DRIVER (REV 1.0)

SEG 0070

```

3679 007642 000 .BYTE 0 ;DRIVE 6
3680 007643 000 .BYTE 0 ;DRIVE 7
3681
3682 ;SAVE REGISTERS FLAG (SAVEFG =1 WORD)
3683 ;SAVEFG <0 IF SAVE THE RH11/RPO4/5/6 REGISTERS WHEN THE
3684 ;OPERATION IS COMPLETED AS PER (DPB+14).
3685 ;SAVEFG=0 IF SAVE THE RH11/RPO4/5/6 REGISTERS, AS PER
3686 ;(DPB+14), AFTER AN ERROR.
3687
3688 007644 000000 SAVEFG: .WORD 0
3689
3690 ;SEEK FLAG (SEEKFG=1 WORD)
3691 ;SEEKFG=0 IF WHEN THE DISK ADDRESS ISN'T IN THE WINDOW
3692 ;FOR A DATA TRANSFER START A SEARCH COMMAND
3693 ;SEEKFG<0 IF DATA TRANSFER WILL DO IMPLIED SEEKS,
3694 ;DISREGARD THE WINDOW
3695
3696 007646 000000 SEEKFG: .WORD 0
3697
3698 ;TIMEOUT TABLE (TIMER=8 WORDS)
3699 ;THIS TABLE CONTAINS THE TIME ALLOWED FOR AN OPERATION
3700
3701 007650 177777 TIMER: .WORD -1 ;DRIVE 0
3702 007652 177777 .WORD -1 ;DRIVE 1
3703 007654 177777 .WORD -1 ;DRIVE 2
3704 007656 177777 .WORD -1 ;DRIVE 3
3705 007660 177777 .WORD -1 ;DRIVE 4
3706 007662 177777 .WORD -1 ;DRIVE 5
3707 007664 177777 .WORD -1 ;DRIVE 6
3708 007666 177777 .WORD -1 ;DRIVE 7
3709
3710 ;DATA TRANSFER UNDERWAY INDICATOR (DTUW=1 WORD)
3711 ;DTUW<0 IF NO DATA TRANSFER UNDERWAY
3712 ;DTUW=+N (WHERE N=0 TO 7) IMPLIES DATA TRANSFER UNDERWAY ON DRIVE N
3713
3714 007670 177777 DTUW: .WORD -1
3715
3716 ;ATTENTION BITS TABLE (ATABIT=8 BYTES)
3717 ;THIS TABLE CONTAINS THE CORRESPONDING BIT TO EACH DRIVES
3718 ;ATTENTION BIT
3719
3720 007672 001 ATABIT: .BYTE 1 ;DRIVE 0
3721 007673 002 .BYTE 2 ;DRIVE 1
3722 007674 004 .BYTE 4 ;DRIVE 2
3723 007675 010 .BYTE 10 ;DRIVE 3
3724 007676 020 .BYTE 20 ;DRIVE 4
3725 007677 040 .BYTE 40 ;DRIVE 5
3726 007700 100 .BYTE 100 ;DRIVE 6
3727 007701 200 .BYTE 200 ;DRIVE 7
3728
3729 ;RPO4/5/6 TO RH11 "MASSBUS CONTROL BUS PARITY ERRORS" (MCPE) ALLOWED BEFORE
3730 ;CALLING IT FATAL (MCPEMX=1 WORD)
3731
3732 007702 000003 MCPEMX: .WORD 3
3733
3734 ;STORAGE FOR RPADR (THE FIRST ADDRESS (776700) OF THE RH11/RPO4/5/6),

```



3735  
3736  
3737 007704 176700  
3738 007706 000254 000240  
3739  
3740  
3741  
3742 007712 000004  
3743  
3744  
3745 007714 001000  
3746  
3747  
3748 007716 000200  
3749  
3750  
3751 007720 000005  
3752  
3753  
3754  
3755 000000  
3756 000002  
3757 000004  
3758 000006  
3759 000010  
3760 000012  
3761 000014  
3762 000016  
3763 000020  
3764 000022  
3765 000024  
3766 000026  
3767 000030  
3768 000032  
3769 000034  
3770 000036  
3771 000040  
3772 000042  
3773 000044  
3774 000046  
3775  
3776  
3777  
3778  
3779  
3780  
3781  
3782  
3783  
3784  
3785  
3786  
3787  
3788  
3789 007722 104412  
3790 007724 013746 177776

;RPVEC (THE VECTOR ADDRESS (254)), AND RPVEC+2 (THE BR LEVEL (5)).

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

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

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

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

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

MXWNDW: .WORD 5

;DEFINITIONS OF THE RH11/RPO4/5/6 ADDRESS INDEXES

RPCS1=0 ;CONTROL AND STATUS REGISTER #1 (DRIVE REG. 00)  
 RPWC=2 ;WORD COUNT REGISTER (NOT A DRIVE REG)  
 RPBA=4 ;UNIBUS ADDRESS REGISTER (NOT A DRIVE REG)  
 RPDA=6 ;DESIRED SECTOR/TRACK ADDRESS REGISTER (DRIVE REG. 05)  
 RPCS2=10 ;CONTROL AND STATUS REGISTER #2 (NOT A DRIVE REG)  
 RPDS1=12 ;DRIVE STATUS REGISTER (DRIVE REG 01)  
 RPER1=14 ;ERROR REGISTER #1 (DRIVE REG. 02)  
 RPAS=16 ;ATTENTION SUMMARY PSEUDO REGISTER (DRIVE REG. 04)  
 RPLA=20 ;LOOK AHEAD REGISTER (DRIVE REG. 07)  
 RPDB=22 ;DATA BUFFER REGISTER (NOT A DRIVE REG.)  
 RPMR=24 ;MAINTAINABILITY REGISTER (DRIVE REG. 03)  
 RPDT=26 ;DRIVE TYPE REGISTER (DRIVE REG. 06)  
 RPSN=30 ;SERIAL NUMBER REGISTER (DRIVE REG. 10)  
 RPOF=32 ;OFFSET REGISTER (DRIVE REG. 11)  
 RPCA=34 ;DESIRED CYLINDER ADDRESS REGISTER (DRIVE REG. 12)  
 RPCC=36 ;CURRENT CYLINDER ADDRESS REGISTER (DRIVE REG. 13)  
 RPER2=40 ;ERROR REGISTER #2 (DRIVE REG. 14)  
 RPER3=42 ;ERROR REGISTER #3 (DRIVE REG. 15)  
 RPEC1=44 ;ECC POSITION REGISTER (DRIVE REG. 16)  
 RPEC2=46 ;ECC PATTERN REGISTER (DRIVE REG. 17)

;RH11/RPO4/5/6 DRIVER INITIALIZATION CODE  
 ;THIS ROUTINE WILL DETERMINE WHICH RPO4/5/6 DRIVES ARE  
 ;AVAILABLE FOR TESTING AND SET THE DRVSTA INDICATOR  
 ;TO THE PROPER STATE FOR EACH DRIVE.  
 ;NOTE: THIS ROUTINE CALLS DRVINT

;CALL

JSR PC,RPINIT  
RETURN

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

RPINIT: SAVREG ;SAVE R0 - R5  
 MOV 2#PS,-(SP) ;SAVE THE PRESENT PROCESSOR STATUS

```

3791 007730 012737 000240 177776      MOV      #(<5*32.>),@#PS      ;CHANGE THE PRIORITY TO 5
3792 007736 004737 015662              JSR      PC CLRQUE          ;CLEAR ALL REQUEST QUEUES
3793 007742 012701 007536              MOV      #RPERRS,R1        ;FIRST ADDRESS TO BE CLEARED
3794 007746 012702 007646              MOV      #SEEKFG,R2        ;LAST ADDRESS TO BE CLEARED
3795 007752 005021              1$:     CLR      (R1)+          ;CLEAR
3796 007754 020102              CMP      R1,R2             ;ARE WE DONE?
3797 007756 101775              BLOS    1$                ;BRANCH IF NO
3798 007760 012702 007670              MOV      #DTUW,R2         ;LAST ADDRESS
3799 007764 012721 177777              2$:     MOV      #-1,(R1)+   ;INITIALIZE
3800 007770 020102              CMP      R1,R2            ;DONE?
3801 007772 101774              BLOS    2$                ;LOOP IF NO
3802 007774 005037 007556              CLR      DRVSTA           ;SET ALL DRIVES TO OFFLINE
3803 010000 005037 007560              CLR      DRVSTA+2
3804 010004 005037 007562              CLR      DRVSTA+4
3805 010010 005037 007564              CLR      DRVSTA+6
3806 010014 013703 007706              MOV      RPVEC,R3         ;SETUP THE RH11/RPO4/5/6 VECTOR
3807 010020 012723 012566              MOV      #ISR,(R3)+
3808 010024 013713 007710              MOV      RPVEC+2,(R3)
3809 010030 013704 007704              MOV      RPADR,R4        ;FIRST ADDRESS OF RH11/RPO4
3810 010034 012764 000040 000010      MOV      #BIT05,RPCS2(R4) ;MASSBUS INIT
3811 010042 005001              CLR      R1               ;START WITH DRIVE 0
3812 010044 004037 010134              3$:     JSR      R0,DRVINT     ;INIT THE DRIVE
3813 010050 000401              BR      4$                ;'DVA' NOT SET OR PARITY ERROR
3814 010052 000402              BR      5$                ;NORMAL RETURN
3815 010054 105061 007556              4$:     CLRB    DRVSTA(R1)   ;SET DRIVE STATUS TO OFFLINE
3816 010060 005201              5$:     INC      R1          ;GO TO NEXT DRIVE
3817 010062 042701 177770              BIC     #C7,R1           ;MASK OUT UNUSED BITS
3818 010066 001366              BNE     3$                ;BR IF MORE DRIVES TO GO
3819 010070 012701 000007              MOV     #7,R1            ;START WITH DRIVE 7
3820 010074 005037 177776              CLR     @#PS             ;CLEAR THE PROCESSOR STATUS
3821 010100 105761 007576              6$:     TSTB    DPINT(R1)   ;WAITING FOR DRIVE TO SWITCH PORTS ?
3822 010104 001405              BEQ     8$                ;BR NOT WAITING
3823 010106 004737 015316              JSR     PC SET.IE        ;SET INTERRUPT
3824 010112 105761 007576              7$:     TSTB    DPINT(R1)   ;DRIVE SWITCHED PORTS ?
3825 010116 001375              BNE     7$                ;BR IF NOT
3826 010120 005301              8$:     DEC     R1          ;GO TO THE NEXT DRIVE
3827 010122 100366              BPL     6$                ;CHECK NEXT DRIVE
3828 010124 012637 177776              MOV     (SP)+,@#PS       ;RESTORE THE PROCESSOR STATUS
3829 010130 104413              RESREG ;RESTORE R0 - R5
3830 010132 000207              RTS     PC                ;BYE-BYE
3831
3832 ;DRIVE INITIALIZATION ROUTINE
3833 ;THIS ROUTINE DETERMINES IF A DRIVE EXIST AND IF IT IS
3834 ;AN RPO4/5/6. IF IT IS, A "READ-IN PRESET" IS ISSUED AND FMT22
3835 ;IS SET TO A "1". THEN MOL, DPR, DRY, AND VV ARE CHECKED TO
3836 ;INSURE THEY ARE ALL ON A "1". AND DEPENDING ON THEIR STATE,
3837 ;DRVSTA IS SET TO THE PROPER CONDITION.
3838
3839 ;CALL
3840 :     MOV     #DRVNUM,R1   ;DRIVE NUMBER TO R1
3841 :     MOV     RPADR,R4     ;UNIBUS ADDRESS OF RH11/RPO4/5/6 (RPCS1)
3842 :     JSR     R0,DRVINT    ;CALLED BY A JSR
3843 :     RETURN1 ;ERROR OCCURRED (PARITY)
3844 :     RETURN2 ;NORMAL RETURN
3845
3846 010134 010546      DRVINT: MOV     R5,-(SP)   ;SAVE R5

```



3847	010136	105061	007556		CLRB	DRVSTA(R1)	; START DRIVE STATUS AS OFFLINE
3848	010142	105061	007566		CLRB	DRVSTYP(R1)	; CLEAR THE DRIVE TYPE INDICATOR
3849	010146	105061	007624		CLRB	ULDFLG(R1)	; CLEAR THE UNLOAD FLAG
3850	010152	010164	000010		MOV	R1,RPCS2(R4)	; SELECT A DRIVE
3851	010156	112764	000111	000000	MOV	#11,RPCS1(R4)	; DO A DRIVE CLEAR COMMAND (& SEIZE DRIVE)
3852	010164	032764	010000	000010	BIT	#BIT12,RPCS2(R4)	; NONEXISTENT DRIVE?
3853	010172	001403			1\$		; NO---BRANCH
3854	010174	004737	015316		JSR	PC,SET.IE	; GO SET "IE" WITHOUT A "TRE"
3855	010200	000520			BR	6\$	; LEAVE THIS ROUTINE
3856	010202	105061	007556		CLRB	DRVSTA(R1)	; SET DRIVE STATUS TO OFFLINE
3857	010206	032764	004000	000000	BIT	#BIT11,RPCS1(R4)	; SEE IF DRIVE AVAILABLE
3858	010214	001514			BEQ	7\$	; BR IF DRIVE NOT AVAILABLE
3859	010216	004037	014636		JSR	RO,RD.RP	; READ THE DRIVE TYPE REG.
3860	010222	000026			RPDT		
3861	010224	010466			8\$		; ERROR RETURN ADDRESS
3862	010226	012605			MOV	(SP)+,R5	; PUT DRIVE TYPE IN R5
3863	010230	112761	000001	007566	MOV	#1,DRVSTYP(R1)	; SET RPO4 INDICATOR
3864	010236	022705	020020		CMP	#20020,R5	; IS IT A SINGLE PORT RPO4?
3865	010242	001431			BEQ	2\$	; BRANCH IF YES
3866	010244	022705	024020		CMP	#24020,R5	; IS IT A DUAL PORT RPO4?
3867	010250	001426			BEQ	2\$	; BR IF YES
3868	010252	112761	000002	007566	MOV	#2,DRVSTYP(R1)	; SET RPO5 INDICATOR
3869	010260	022705	020021		CMP	#20021,R5	; SINGLE PORT RPO5 ?
3870	010264	001420			BEQ	2\$	; BR IF YES
3871	010266	022705	024021		CMP	#24021,R5	; DUAL PORT RPO5 ?
3872	010272	001415			BEQ	2\$	; BR IF YES
3873	010274	112761	000004	007566	MOV	#4,DRVSTYP(R1)	; SET RPO6 INDICATOR
3874	010302	022705	020022		CMP	#20022,R5	; SINGLE PORT RPO6 ?
3875	010306	001407			BEQ	2\$	; BR IF YES
3876	010310	022705	024022		CMP	#24022,R5	; DUAL PORT RPO6 ?
3877	010314	001404			BEQ	2\$	; BR IF YES
3878	010316	112761	177777	007566	MOV	#-1,DRVSTYP(R1)	; SET INDICATOR TO 'OTHER'
3879	010324	000446			BR	6\$	; EXIT
3880	010326	012746	000121		MOV	#121,-(SP)	; DO A "READ-IN PRESET"
3881	010332	004037	015012		JSR	RO,WRT.RP	
3882	010336	000000			RPCS1		
3883	010340	010466			8\$		
3884	010342	012746	010000		MOV	#BIT12,-(SP)	; SET FMT22=1
3885	010346	004037	015012		JSR	RO,WRT.RP	
3886	010352	000032			RPOF		
3887	010354	010466			6\$		
3888	010356	004037	014636		JSR	RO,RD.RP	; READ RPDS1
3889	010362	000012			RPDS1		
3890	010364	010466			8\$		
3891	010366	012605			MOV	(SP)+,R5	; AND SAVE IT IN R5
3892	010370	100015			BPL	4\$	; BRANCH IF ATA=0
3893	010372	116164	007672	000016	MOV	ATABIT(R1),RPAS(R4)	; CLEAR ATTENTION BIT
3894	010400	004037	014636		JSR	RO,RD.RP	; FIND OUT WHY ATA=1
3895	010404	000014			RPER1		
3896	010406	010466			8\$		
3897	010410	006126			ROL	(SP)+	; IS IT UNSAFE?
3898	010412	100004			BPL	4\$	; BR IF NOT
3899	010414	112761	177777	007556	MOV	#-1,DRVSTA(R1)	; SET UNSAFE INDICATOR
3900	010422	000407			BR	6\$	; EXIT
3901	010424	005105			COM	R5	; CHECK MOL, DPR, DRY, AND VV
3902	010426	042705	167077		BIC	#1<BIT12!BIT08!BIT07!BIT06>,R5	

```

3903 010432 001003          BNE      6$          ;BRANCH IF MOL, DPR, DRY, OR VV IS CLEAR
3904 010434 112761 000001 007556  MOVB    #1,DRVSTA(R1) ;SET DRIVE STATUS TO ONLINE
3905 010442 005720          TST     (R0)+        ;STEP OVER THE ERROR RETURN
3906 010444 000410          BR      8$          ;EXIT
3907 010446 006301          ASL     R1           ;CHANGE INDEX TO ADDRESS WORDS
3908 010450 012761 003720 007650  MOV     #2000.,TIMER(R1) ;START 2 SEC TIMER
3909 010456 006201          ASR     R1           ;RESTORE R1
3910 010460 105161 007576  COMB    DPINT(R1)    ;SET PORT INITIALIZE INIDICATOR
3911 010464 005720          TST     (R0)+
3912 010466 012605          MOV     (SP)+,R5    ;RESTORE R5
3913 010470 000200          RTS      R0         ;EXIT
3914
3915          ;REQUEST PRE-PROCESSOR-HANDLES SUBSYSTEM REQUEST
3916          ;CALL
3917          ;
3918          ;
3919          ;
3920          ;
3921          ;
3922          ;
3923          ;
3924          ;
3925          ;
3926          ;
3927          ;
3928          ;
3929          ;
3930          ;
3931          ;
3932          ;
3933          ;
3934          ;
3935          ;
3936          ;
3937          ;
3938          ;
3939          ;
3940          ;
3941          ;
3942          ;
3943          ;
3944          ;
3945          ;
3946          ;
3947          ;
3948          ;
3949          ;
3950          ;
3951          ;
3952          ;
3953          ;
3954          ;
3955          ;
3956          ;
3957          ;
3958          ;

```

:CALL THE RPO4/5/6 DRIVER  
 ADDRESS OF POINTER OF DRIVES PARAMETER BLOCK  
 RETURN HERE IF QUEUE IS FULL  
 RETURN HERE IF REQUEST IS IN QUEUE OR THERE  
 IS AN ERROR CONDITION

```

RPO4:  MOV     @#PS,-(SP)    ;SAVE THE CALLING STATUS
        MOV     RPVEC+2,@#PS ;DON'T ALLOW ANY RPO4/5/6 INTERRUPTS
        MOVB    #1,ACTDRV  ;SET "ACTIVE DRIVER" FLAG
        SAVREG          ;SAVE R0 - R5
        MOV     (R0),R2    ;PICKUP THE DRIVE PARAMETER BLOCK POINTER
        CLR     16(R2)    ;CLEAR THE STATUS/ERROR INDICATOR
        MOVB    (R2),R1    ;PICKUP THE DRIVE NUMBER
        MOV     RPADR,R4   ;UNIBUS ADDRESS OF RPCS1
        TSTB   DRVSTA(R1) ;CHECK DRIVES STATUS
        BGT     1$        ;BRANCH IF ONLINE
        TSTB   ULDFLG(R1) ;UNLOAD COMMAND IN QUEUE?
        BNE    3$        ;BRANCH IF YES
        TSTB   DPINT(R1)  ;TRYING TO INIT THE DRIVE
        BR     5$        ;BR IF YES
        JSR    R0,DRVINT  ;GO INIT. THE DRIVE
        BR     4$        ;ERROR RETURN
        TSTB   DRVSTA(R1) ;IS DRIVE STATUS ONLINE?
        BLE    6$        ;BR IF NOT
        TSTB   DPRQS(R1)  ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
        BNE    5$        ;BR IF YES
        MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
        JSR    R0,DRVQUE  ;PUT THIS REQUEST IN QUEUE
        BR     9$        ;QUEUE IS FULL
        CMPB   #103,2(R2) ;IS THIS REQ. FOR AN UNLOAD?
        BNE    2$        ;BR IF NO
        MOVB   #-1,ULDFLG(R1) ;SET THE "UNLOAD IN QUEUE" FLAG
        TSTB   DRVACT(R1) ;IS THIS DRIVE ACTIVE?
        BNE    8$        ;BR IF YES
        JSR    PC,OPT     ;CALL THE OPTIMIZER
        BR     8$
        MOV     #BIT15!BIT13,16(R2) ;SET THE "UNLOAD IN QUEUE" ERROR FLAG
        BR     8$
        JSR    PC,C17    ;GO HANDLE THE PARITY ERROR
        BR     8$

```



```

3959 010656 004037 015760      5$: JSR    RD,DRVQUE      ;PUT REQUEST IN QUEUE
3960 010662 000431              BR      9$              ;QUEUE IS FULL
3961 010664 032714 000100      BIT     #BIT06,(R4)     ;IS 'IE' SET ALREADY ?
3962 010670 001023              BNE     8$              ;BR IF IT IS
3963 010672 004737 015316      JSR     PC,SET.IE      ;SET INTERRUPT
3964 010676 000420              BR      8$              ;RETURN, REQUEST IN QUEUE
3965 010700 105761 007556      6$: TSTB   DRVSTA(R1)    ;SEE IF DRIVE OFFLINE OR UNSAFE
3966 010704 002412              BLT     7$              ;BR IF UNSAFE
3967 010706 012762 140000 000016  MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE ERROR INDICATOR
3968 010714 105761 007556      TSTB   DRVSTYP(R1)    ;SEE IF OFFLINE OR NONEXISTENT
3969 010720 001007              BNE     8$              ;BR IF OFFLINE
3970 010722 012762 100002 000016  MOV     #BIT15:BIT01,16(R2) ;REPORT DRIVE NONEXISTENT
3971 010730 000403              BR      8$              ;GO TO EXIT
3972 010732 012762 110000 000016  7$: MOV     #BIT15:BIT12,16(R2) ;DRIVE IS UNSAFE
3973 010740 104413              8$: RESREG                ;RESTORE R0 - R5
3974 010742 005720              TST     (R0)+           ;SETUP FOR NORMAL RETURN
3975 010744 000401              BR      10$            ;FINISH UP, THEN EXIT
3976 010746 104413              9$: RESREG                ;RESTORE R0 - R5
3977 010750 005720              10$: TST     (R0)+           ;CORRECT THE RETURN ADDRESS
3978 010752 105037 007622      CLRB   ACTDRV         ;CLEAR "ACTIVE DRIVER" FLAG
3979 010756 012637 177776      MOV     (SP)+,2#PS     ;RETURN "PS" TO USER LEVEL
3980 010762 000200              RTS      RD             ;RETURN TO CALLER
3981
3982 ;OPTIMIZER-CALLED FOR A PARTICULAR DRIVE
3983
3984 ;CALL
3985
3986 ;
3987 ;
3988 010764 104412              MOV     #DRVNUM,R1     ;DRIVE NUMBER TO R1
3989 010766 013746 177776              JSR     PC,OPT         ;SETUP A COMMAND
3990 010772 146137 007672 007620  OPT:  SAVREG                ;SAVE R0 - R5
3991 011000 004737 016034              MOV     2#PS, -(SP)    ;SAVE PROC. STATUS
3992 011004 005702              BICB   ATABIT(R1),SRCHWT ;CLEAR "SEARCH WAIT" KEY
3993 011006 001505              JSR     PC,GETREQ     ;GET "DPB" POINTER OF REQUEST
3994 011010 032764 004000 000000  TST     R2              ;IS THERE A REQUEST IN QUEUE?
3995 011016 001407              BEQ     7$              ;NO--BRANCH TO EXIT
3996 011020 032764 000100 000012  BIT     #BIT11,RPCS1(R4) ;IS DVA SET?
3997 011026 001003              BEQ     10$            ;BRANCH IF NOT
3998 011030 004037 010134              BIT     #BIT6,RPDS1(R4) ;IS VV SET?
3999 011034 000470              BNE     10$            ;BR IF IT IS
4000 011036 105761 007556      9$: JSR     RD,DRVINT    ;SEE IF DRIVE STILL ONLINE ?
4001 011042 003014              BR      6$              ;PARITY OR 'DVA' NOT SET
4002 011044 004737 016056      10$: TSTB   DRVSTA(R1)    ;IS DRIVE ONLINE?
4003 011050 012762 140000 000016  BGT     1$              ;YES--BRANCH
4004 011056 105761 007556      JSR     PC,POPQUE     ;NO--REMOVE REQUEST FROM QUEUE
4005 011062 100064              MOV     #BIT15:BIT14,16(R2) ;SET OFFLINE STATUS/ERROR INDICATOR
4006 011064 012762 110000 000016  TSTB   DRVSTA(R1)    ;IS DRIVE UNSAFE ?
4007 011072 000460              BPL     8$              ;BR TO EXIT IF NOT
4008 011074 012746 000111              MOV     #BIT15:BIT12,16(R2) ;SET UNSAFE STATUS/ERROR INDICATOR
4009 011100 004037 015012      1$: BR      8$              ;BRANCH TO EXIT
4010 011104 000000              MOV     #111, -(SP)   ;LOAD COMMAND ONTO THE STACK
4011 011106 011216              JSR     RD,WRT.RP     ;LOAD THE REGISTER
4012 011110 032714 004000      RPCS1                ;REGISTER INCREMENT
4013 011114 001427              BIT     #BIT11,(R4)   ;ERROR RETURN ADDRESS
4014 011116 122762 000150 000002  BEQ     5$              ;DRIVE AVAILABLE ?
                                ;BR IF NOT
                                CMPB   #150,2(R2)      ;IS THE REQUEST FOR I/O?

```

```

4015 011124 002403          BLT      2$          ;YES--BRANCH
4016 011126 004737 011460    JSR      PC,C14     ;CALL THE COMMAND INITIATOR
4017 011132 000440          BR       8$          ;BRANCH TO EXIT
4018 011134 005737 007670    2$:     TST      DTUW   ;DATA TRANSFER UNDERWAY?
4019 011140 002012          BGE     4$          ;YES--GO START A SEARCH
4020 011142 005737 007646    TST     SEEKFG     ;DO IMPLIED SEEKS?
4021 011146 100404          BMI     3$          ;YES---BRANCH
4022 011150 004037 012430    JSR     RD,LA      ;NO--DO LOOK AHEAD
4023 011154 000427          BR      8$          ;RETURN HERE ON A PARITY ERROR
4024 011156 000403          BR      4$          ;GO START A SEARCH
4025 011160 004737 011244    3$:     JSR     PC,C11 ;START A DATA TRANSFER
4026 011164 000423          BR      8$
4027 011166 004737 011352    4$:     JSR     PC,C13 ;START A SEARCH
4028 011172 000420          BR      8$          ;GO TO THE EXIT
4029 011174 112761 177777 007606 5$:     MOVB    #-1,DPRQS(R1) ;SET PORT REQUEST INDICATOR
4030 011202 010103          MOV     R1,R3      ;SET UP TO ADDRESS WORDS
4031 011204 006303          ASL    R3          ;CONVERT TO WORD INDEX
4032 011206 012763 023420 007650 MOV     #10000.,TIMER(R3) ;START 10 SEC TIMER
4033 011214 000402          BR      7$          ;EXIT
4034 011216 004737 012074    6$:     JSR     PC,C17 ;PROCESS THE PARITY ERROR
4035 011222 032714 000100    7$:     BIT     #BIT06,(R4) ;SEE IF 'IE' ALREADY SET
4036 011226 001002          BNE    8$          ;BR IF SET
4037 011230 004737 015316    JSR     PC,SET.IE  ;SET "IE" WITHOUT A "TRE"
4038 011234 012637 177776    8$:     MOV     (SP)+,2#PS ;RESTORE PROC. STATUS
4039 011240 104413          RESREG ;RESTORE R0 - R5
4040 011242 000207          RTS     PC
4041
4042 ;COMMAND INITIATOR
4043 ;CALL
4044 ;
4045 ;   MOV     #DRVNUM,R1 ;DRIVE NUMBER
4046 ;   MOV     #DPB,R2    ;ADDRESS OF DPB
4047 ;   JSR     PC,C1?    ;C1?= C11,C13, OR C14
4048 ;   ;WHERE:
4049 ;   C11=DATA TRANSFER
4050 ;   C12=SEARCH REQUESTED BY DATA XFER
4051 ;   C14=NOT DATA TRANSFER
4052 ;
4053 C11: JSR     PC,POPQUE ;REMOVE REQUEST FROM "DRIVES WAIT" QUEUE
4054     MOV     R2,TRNSWT ;PUT REQ. IN TRANSFER WAIT QUEUE
4055     MOV     R2,R3    ;DPB ADDRESS TO R3
4056     MOV     RPADR,R4 ;RPCS1 ADDRESS
4057     MOV     R1,RPCS2(R4) ;SELECT DRIVE
4058     ADD     #4,R3    ;DESIRED WORD COUNT
4059     ADD     #2,R4    ;RPWC ADDRESS
4060     MOV     (R3)+,(R4)+ ;LOAD WORD COUNT
4061     MOV     (R3)+,(R4)+ ;LOAD BUFFER ADDRESS
4062     MOV     (R3)+,-(SP) ;LOAD SECTOR AND TRACK
4063     JSR     RD,WRT.RP ;CALL THE LOAD(WRITE) ROUTINE
4064     RPD    RPDA     ;INDEX OF REGISTER TO LOAD
4065     CI7    CI7      ;ERROR RETURN ADDRESS
4066     MOV     (R3)+,-(SP) ;LOAD CYLINDER ADDRESS
4067     JSR     RD,WRT.RP
4068     RPCA   RPCA     ;
4069     CI7    CI7      ;
4070     MOV     2(R2),-(SP) ;LOAD "COMMAND+GO", "A17&A16", AND "PSEL"

```



4071	011332	004037	015012		JSR	RO,WRT.RP	
4072	011336	000000			RPCS1		
4073	011340	012074			CI7		
4074	011342	010137	007670		MOV	R1,DTUW	;SET "DATA TRANSFER UNDERWAY"
4075	011346	000137	012036		JMP	CI5	
4076	011352	013704	007704	CI3:	MOV	RPADR,R4	;RPCS1 ADDRESS
4077	011356	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
4078	011362	016246	000012		MOV	12(R2),-(SP)	;DESIRED CYLINDER ADDRESS
4079	011366	004037	015012		JSR	RO,WRT.RP	
4080	011372	000034			RPCA		
4081	011374	012074			CI7		
4082	011376	116203	000010		MOV	10(R2),R3	;PICKUP SECTOR ADDRESS
4083	011402	163703	007720		SUB	MXWINDW,R3	;BACKUP BY MAX. SEARCH FOR I/O WINDOW
4084	011406	002002			IS		
4085	011410	062703	000026		ADD	#22,R3	
4086	011414	010346		IS:	MOV	R3,-(SP)	;COMBINE THE ADJUSTED SECTOR WITH
4087	011416	116266	000011	000001	MOV	11(R2),1(SP)	;THE DESIRED TRACK
4088	011424	004037	015012		JSR	RO,WRT.RP	;LOAD DESIRED TRACK & SECTOR
4089	011430	000006			RPDA		
4090	011432	012074			CI7		
4091	011434	012746	000131		MOV	#131,-(SP)	;START A SEARCH
4092	011440	004037	015012		JSR	RO,WRT.RP	
4093	011444	000000			RPCS1		
4094	011446	012074			CI7		
4095	011450	156137	007672	007620	BIS	ATABIT(R1),SRCHWT	;SET "SEARCH WAIT" KEY
4096	011456	000567			BR	CI5	
4097	011460	013704	007704	CI4:	MOV	RPADR,R4	;RPCS1 ADDRESS
4098	011464	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE
4099	011470	116203	000002		MOV	2(R2),R3	;PICKUP THE REQUESTED COMMAND
4100	011474	122703	000131		CMP	#131,R3	;IS IT A SEARCH COMMAND?
4101	011500	001007			BNE	IS	;BRANCH IF NO
4102	011502	016246	000010		MOV	10(R2),-(SP)	;LOAD DESIRED TRACK & SECTOR
4103	011506	004037	015012		JSR	RO,WRT.RP	
4104	011512	000006			RPDA		
4105	011514	012074			CI7		
4106	011516	000403			BR	2\$	;GO LOAD CYLINDER
4107	011520	122703	000105	IS:	CMP	#105,R3	;IS IT A SEEK COMMAND
4108	011524	001007			BNE	3\$	;BRANCH IF NO
4109	011526	016246	000012	2\$:	MOV	12(R2),-(SP)	;LOAD DESIRED CYLINDER
4110	011532	004037	015012		JSR	RO,WRT.RP	
4111	011536	000034			RPCA		
4112	011540	012074			CI7		
4113	011542	000546			BR	CI6	
4114	011544	122703	000115	3\$:	CMP	#115,R3	;IS IT AN "OFFSET" COMMAND?
4115	011550	001013			BNE	4\$	;BR IF NO
4116	011552	004037	014636		JSR	RO,RO.RP	;MERGE THE OFFSET VALUE INTO RPOF
4117	011556	000032			RPOF		;BUT DON'T CHANGE THE UPPER
4118	011560	012074			CI7		
4119	011562	116216	000001		MOV	1(R2),(SP)	;BYTE WHEN LOADING THE
4120	011566	004037	015012		JSR	RO,WRT.RP	;REGISTER (RPOF)
4121	011572	000032			RPOF		
4122	011574	012074			CI7		
4123	011576	000530			BR	CI6	;GO START THE COMMAND
4124	011600	122703	000107	4\$:	CMP	#107,R3	;IS IT A "RECALIBRATE" COMMAND?
4125	011604	001525			BEQ	CI6	;BRANCH IF YES
4126	011606	122703	000117		CMP	#117,R3	;IS IT A RETURN TO CENTER?



4127	011612	001522			BEQ	CI6		; BRANCH IF YES
4128	011614	122703	000103		CMPB	#103,R3		; IS IT AN "UNLOAD" COMMAND?
4129	011620	001016			BNE	5\$		; BRANCH IF NO
4130	011622	112761	000001	007546	MOVB	#1,DRVACT(R1)		; SET THE DRIVE ACTIVE INDICATOR
4131	011630	105061	007556		CLRB	DRVSTA(R1)		; PUT DRIVE STATUS TO OFFLINE
4132	011634	112761	000001	007624	MOVB	#1,ULDFLG(R1)		; SET "UNLOAD IN PROGRESS" FLAG
4133	011642	010346			MOV	R3,-(SP)		; START THE "UNLOAD" COMMAND
4134	011644	004037	015012		JSR	RO,WRT.RP		
4135	011650	000000			RPCS1			
4136	011652	012074			CI7			
4137	011654	000207			RTS	PC		; RETURN TO USER
4138	011656	122703	000143		5\$: CMPB	#143,R3		; IS IT A "SET FORMAT" COMMAND?
4139	011662	001014			BNE	6\$		; BRANCH IF NO
4140	011664	004037	014636		JSR	RO,RD.RP		; READ THE OFFSET REGISTER
4141	011670	000032			RPOF			
4142	011672	012074			CI7			
4143	011674	116266	000001	000001	MOVB	1(R2),1(SP)		; COMBINE "FMT22" "ECI" AND "HCI"
4144	011702	004037	015012		JSR	RO,WRT.RP		; LOAD "FMT22", "ECI", AND/OR "HCI".
4145	011706	000032			RPOF			
4146	011710	012074			CI7			
4147	011712	000436			BR	12\$		
4148	011714	122703	000141		6\$: CMPB	#141,R3		; IS IT A "GET REGISTER" COMMAND?
4149	011720	001023			BNE	10\$		; BRANCH IF NO
4150	011722	016203	000006		7\$: MOV	6(R2),R3		; POINTS TO 1ST ADDRESS OF WHERE
4151								; TO PUT THE REGISTER(S)
4152	011726	116237	000010	011744	MOVB	10(R2),9\$		; INIT. THE INDEX FOR THE FIRST REG.
4153	011734	116205	000011		MOVB	11(R2),R5		; INDEX OF LAST REG. TO MOVE
4154	011740	004037	014636		8\$: JSR	RO,RD.RP		; READ RPO4/5/6 REGISTER
4155	011744	000000			9\$: RPCS1			; INDEX OF REG. TO READ
4156	011746	012074			CI7			
4157	011750	012623			MOV	(SP)+,(R3)+		; GET THE CONTENTS OF RH11/RPO4/5/6 REG.
4158	011752	023705	011744		CMP	9\$ R5		; LAST REG. BEEN READ?
4159	011756	001414			BEQ	12\$		; GET OUT IF YES
4160	011760	062737	000002	011744	ADD	#2,9\$		; INCREASE THE INDEX BY 2
4161	011766	000764			BR	8\$		; LOOP--MORE TO READ
4162	011770	122703	000145		10\$: CMPB	#145,R3		; IS IT A "SELECT DRIVE" COMMAND?
4163	011774	001405			BEQ	12\$		; BRANCH IF YES
4164	011776	010346			11\$: MOV	R3,-(SP)		; LOAD THE COMMAND
4165	012000	004037	015012		JSR	RO,WRT.RP		
4166	012004	000000			RPCS1			
4167	012006	012074			CI7			
4168	012010	004737	016056		12\$: JSR	PC,POPQUE		; REMOVE REG. FROM QUEUE
4169	012014	052762	000200	000016	BIS	#BIT07,16(R2)		; SET THE "DONE" BIT
4170	012022	005737	007644		TST	SAVEFG		; SAVE THE RH11/RPO4/5/6 REGISTERS?
4171	012026	100002			BPL	13\$		; BRANCH IF NO
4172	012030	004737	015200		JSR	PC,SVRH11		; YES--GO SAVE THE REGISTERS
4173	012034	000207			13\$: RTS	PC		; RETURN TO USER
4174	012036	006301			CI5: ASL	R1		
4175	012040	012761	001750	007650	MOV	#1000.,TIMER(R1)		; SET A ONE SECOND TIMER
4176	012046	006201			ASR	R1		
4177	012050	112761	000001	007546	MOVB	#1,DRVACT(R1)		; SET THE DRIVE ACTIVE
4178	012056	000207			RTS	PC		; RETURN TO THE USER
4179	012060	010346			CI6: MOV	R3,-(SP)		; LOAD THE COMMAND
4180	012062	004037	015012		JSR	RO,WRT.RP		
4181	012066	000000			RPCS1			
4182	012070	012074			CI7			



4183	012072	000761				BR	C15	
4184	012074	032764	010000	000010	CI7:	BIT	#BIT12,RPCS2(R4)	;DRIVE NON-EXISTENT ?
4185	012102	001034				BNE	C18	;BR IF YES
4186	012104	005702			1\$:	TST	R2	;ANYTHING IN QUEUE ?
4187	012106	001405				BEQ	CI7B	;BR IF NOT
4188	012110	012762	104000	000016		MOV	#BIT15!BIT11,16(R2)	;SET "PARITY" ERROR INDICATOR
4189	012116	004737	015200			JSR	PC,SVRH11	;GO SAVE THE RH11/RPO4/5/6 REGISTERS
4190	012122	012746	000111		CI7B:	MOV	#11,-(SP)	;DO A "DRIVE CLEAR"
4191	012126	004037	015012			JSR	RO,WRT.RP	
4192	012132	000000				RPCS1		
4193	012134	012174				C18		
4194	012136	004737	015740			JSR	PC,EMPTYQ	;EMPTY THE QUEUE
4195	012142	105061	007624			CLRB	ULDFLG(R1)	;CLEAR THE UNLOAD IN QUEUE FLAG
4196	012146	105061	007546			CLRB	DRVACT(R1)	;DRIVE IS IDLE
4197	012152	020137	007670			CMP	R1,DTUW	;IF THIS DRIVE HAD AN I/O REQUEST
4198	012156	001005				BNE	1\$	;IN PROGRESS CLEAR ALL OF THE FLAGS
4199	012160	005037	007616			CLR	TRNSWT	
4200	012164	012737	177777	007670		MOV	#-1,DTUW	
4201	012172	000207			1\$:	RTS	PC	
4202	012174	104412			CI8:	SAVREG		;SAVE R0 - R5
4203	012176	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	;IS 'NED' SET ?
4204	012204	001002				BNE	1\$	;BR IF YES
4205	012206	005001				CLR	R1	
4206	012210	005003				CLR	R3	
4207	012212	105761	007546		1\$:	TSTB	DRVACT(R1)	;DRIVE ACTIVE?
4208	012216	001443				BEQ	5\$	;BRANCH IF NO
4209	012220	013702	007616			MOV	TRNSWT,R2	;GET THE "TRANSFER WAIT" QUEUE
4210	012224	020137	007670			CMP	R1,DTUW	;DID THIS DRIVE HAVE AN I/O IN PROGRESS?
4211	012230	001402				BEQ	2\$	;BRANCH IF YES
4212	012232	004737	016034			JSR	PC,GETREQ	;GET THE OPB POINTER
4213	012236	005702			2\$:	TST	R2	;QUEUE ENTRY FOR DRIVE ?
4214	012240	001415				BEQ	4\$	;BR IF NOT
4215	012242	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	; 'NED' SET ?
4216	012250	001404				BEQ	3\$	;BR IF NOT
4217	012252	012762	100002	000016		MOV	#BIT15!BIT01,16(R2)	;SET 'DRIVE NON-EXISTENT' INDICATOR
4218	012260	000405				BR	4\$	;CONTINUE
4219	012262	012762	102000	000016	3\$:	MOV	#BIT15!BIT10,16(R2)	;SET "NON-CLEARABLE PARITY" ERROR INDICATOR
4220	012270	004737	015200			JSR	PC,SVRH11	;SAVE RH11/RPO4/5/6 REGISTERS
4221	012274	012763	177777	007650	4\$:	MOV	#-1,TIMER(R3)	;STOP THE TIMER
4222	012302	105061	007546			CLRB	DRVACT(R1)	;SET "DRIVE ACTIVE" TO IDLE
4223	012306	020137	007670			CMP	R1,DTUW	;IS THIS DRIVE SETUP FOR A TRANSFER
4224	012312	001005				BNE	5\$	;BR IF NOT
4225	012314	012737	177777	007670		MOV	#-1,DTUW	;RESET THE INDICATOR
4226	012322	005037	007616			CLR	TRNSWT	;CLEAR THE TRANSFER QUEUE
4227	012326	105061	007624		5\$:	CLRB	ULDFLG(R1)	;CLEAR UNLOAD FLAG
4228	012332	032764	010000	000010		BIT	#BIT12,RPCS2(R4)	; 'NED' SET ?
4229	012340	001021				BNE	6\$	;BR IF YES
4230	012342	005201				INC	R1	;MOVE TO THE NEXT DRIVE
4231	012344	062703	000002			ADD	#2,R3	
4232	012350	042701	177770			BIC	#1C7,R1	
4233	012354	001316				BNE	1\$	;BRANCH IF MORE DRIVES
4234	012356	012737	177777	007670		MOV	#-1,DTUW	;NO DATA TRANSFERS UNDERWAY
4235	012364	005037	007616			CLR	TRNSWT	;CLEAR THE 'TRANSFER WAIT' QUEUE
4236	012370	004737	015662			JSR	PC,CLRQUE	;CLEAR ALL OF THE REQUEST QUEUES
4237	012374	012764	000040	000010		MOV	#BIT05,RPCS2(R4)	;DO A MASSBUS INIT.
4238	012402	000406				BR	7\$	;CONTINUE



4239	012404	004737	015740	6\$:	JSR	PC,EMPTY0	;CLEAR THE DRIVE'S QUEUE	
4240	012410	105061	007556		CLRB	DRVSTA(R1)	;SET DRIVE TO OFFLINE	
4241	012414	105061	007566		CLRB	DRVTYP(R1)	;CLEAR THE DRIVE TYPE INDICATOR	
4242	012420	004737	015316	7\$:	JSR	PC,SET.IE	;SET "IE" WITHOUT "TRE"	
4243	012424	104413			RESREG	RO	;RESTORE RO - R5	
4244	012426	000207			RTS	PC	;RETURN	
4245								
4246								
4247								
4248								
4249								
4250								
4251								
4252								
4253								
4254								
4255								
4256	012430	013704	007704	LA:	MOV	RPADR,R4	;GET RPCS1'S ADDRESS	
4257	012434	010164	000010		MOV	R1,RPCS2(R4)	;SELECT DRIVE	
4258	012440	004037	014636		JSR	RO,RO.RP	;READ CURRENT CYLINDER	
4259	012444	000036			RPCC			
4260	012446	012560			4\$		;ERROR RETURN ADDRESS	
4261	012450	022662	000012		CMP	(SP)+,12(R2)	;IS CURRENT CYLINDER=DESIRED	
4262							;CYLINDER?	
4263	012454	001037			BNE	3\$	;EXIT IF NO	
4264	012456	105261	007634		INCB	LACNT(R1)	;INCREMENT THE LOOK AHEAD COUNT	
4265	012462	126137	007634	007712	CMPB	LACNT(R1),MXLACT	;EXCEED MAX?	
4266	012470	003026			BGT	2\$	;BRANCH IF YES	
4267	012472	116203	000010		MOVSB	10(R2),R3	;GET DESIRED SECTOR ADDRESS AND	
4268	012476	000303			SWAB	R3	;MULT. BY 64--ALIGN WITH	
4269	012500	006203			ASR	R3	;LOOK AHEAD REGISTER	
4270	012502	006203			ASR	R3		
4271	012504	012737	000340	177776	MOV	#340,#PS	;PRIORITY LEVEL "7"	
4272	012512	004037	014636		JSR	RO,RO.RP	;READ LOOK AHEAD REGISTER	
4273	012516	000020			RPLA			
4274	012520	012560			4\$			
4275	012522	162603			SUB	(SP)+,R3	;CALCULATE THE DELTA	
4276	012524	002002			BGE	1\$		
4277	012526	062703	002600		ADD	#(22.*64.),R3	;MAKE THE DELTA POSITIVE	
4278	012532	023703	007714	1\$:	CMP	MXDLTA,R3	;CHECK THE DELTA TO SEE	
4279	012536	002406			BLT	3\$	;IF IT IS WITHIN THE	
4280	012540	023703	007716		CMP	MNDLTA,R3	;WINDOW---IF YES, ZERO	
4281	012544	002003			BGE	3\$	;THE LOOK AHEAD COUNT	
4282	012546	105061	007634	2\$:	CLRB	LACNT(R1)	;AND TAKE THE I/O EXIT	
4283	012552	005720			TST	(RO)+		
4284	012554	005720		3\$:	TST	(RO)+	;ADJUST THE RETURN ADDRESS	
4285	012556	000402			BR	5\$	;EXIT	
4286	012560	004737	012074	4\$:	JSR	PC,CI7	;PROCESS THE ERROR	
4287	012564	000200		5\$:	RTS	RO	;RETURN	
4288								
4289								
4290								
4291	012566	112737	000001	007622	ISR:	MOVB	#1,ACTDRV	;SET "ACTIVE DRIVER" FLAG
4292	012574	104412			SAVREG		;SAVE RO - R5	
4293	012576	013704	007704		MOV	RPADR,R4	;ADDRESS OF RHSCS1	
4294	012602	013701	007670		MOV	DTUW,R1	;GET "DATA TRANSFER UNDERWAY" INDICATOR	

;LOOK AHEAD ROUTINE

;CALL

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

MOV	RPADR,R4	;GET RPCS1'S ADDRESS
MOV	R1,RPCS2(R4)	;SELECT DRIVE
JSR	RO,RO.RP	;READ CURRENT CYLINDER
RPCC		
4\$		;ERROR RETURN ADDRESS
CMP	(SP)+,12(R2)	;IS CURRENT CYLINDER=DESIRED
		;CYLINDER?
BNE	3\$	;EXIT IF NO
INCB	LACNT(R1)	;INCREMENT THE LOOK AHEAD COUNT
CMPB	LACNT(R1),MXLACT	;EXCEED MAX?
BGT	2\$	;BRANCH IF YES
MOVSB	10(R2),R3	;GET DESIRED SECTOR ADDRESS AND
SWAB	R3	;MULT. BY 64--ALIGN WITH
ASR	R3	;LOOK AHEAD REGISTER
ASR	R3	
MOV	#340,#PS	;PRIORITY LEVEL "7"
JSR	RO,RO.RP	;READ LOOK AHEAD REGISTER
RPLA		
4\$		
SUB	(SP)+,R3	;CALCULATE THE DELTA
BGE	1\$	
ADD	#(22.*64.),R3	;MAKE THE DELTA POSITIVE
CMP	MXDLTA,R3	;CHECK THE DELTA TO SEE
BLT	3\$	;IF IT IS WITHIN THE
CMP	MNDLTA,R3	;WINDOW---IF YES, ZERO
BGE	3\$	;THE LOOK AHEAD COUNT
CLRB	LACNT(R1)	;AND TAKE THE I/O EXIT
TST	(RO)+	
TST	(RO)+	;ADJUST THE RETURN ADDRESS
BR	5\$	;EXIT
JSR	PC,CI7	;PROCESS THE ERROR
RTS	RO	;RETURN

;INTERRUPT SERVICE ROUTINE

MOVB	#1,ACTDRV	;SET "ACTIVE DRIVER" FLAG
SAVREG		;SAVE RO - R5
MOV	RPADR,R4	;ADDRESS OF RHSCS1
MOV	DTUW,R1	;GET "DATA TRANSFER UNDERWAY" INDICATOR



```

4295 012606 002403          BLT      1$          ;BRANCH IF NO DATA TRANSFER UNDERWAY
4296 012610 004737 012632    JSR      PC,TD      ;CALL TRANSFER DONE
4297 012614 000402          BR       2$          ;EXIT
4298 012616 004737 012772    1$:     JSR      PC,SC ;CALL SPECIAL CONDITIONS
4299 012622 104413          2$:     RESREG      ;RESTORE R0 - R5
4300 012624 105037 007622    CLRB    ACTDRV     ;CLEAR "ACTIVE DRIVER" FLAG
4301 012630 000002          RTI              ;RETURN
4302
4303          ;TRANSFER DONE ROUTINE
4304
4305 012632 105061 007546    TD:     CLRB    DRVACT(R1) ;SET DRIVE ACTIVE INDICATOR TO IDLE
4306 012636 012737 177777 007670    MOV     #-1,DTUW   ;NO DATA TRANSFERS UNDERWAY
4307 012644 006301          ASL     R1
4308 012646 012761 177777 007650    MOV     #-1,TIMER(R1) ;CANCEL TIMEOUT
4309 012654 006201          ASR     R1
4310 012656 013702 007616    MOV     TRNSWT,R2   ;GET "DPB" ADDRESS FROM THE
4311 012662 005037 007616    CLR     TRNSWT      ;TRANSFER WAIT QUEUE--CLEAR QUEUE
4312 012666 052762 000200 000016    BIS     #BIT07,16(R2) ;SET DONE
4313 012674 010164 000010    MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
4314 012700 004037 014636    JSR     R0,RD.RP    ;TRANSFER ERROR(TRE=1)?
4315 012704 000000          RPCS1
4316 012706 012074          CI7
4317 012710 006126          ROL     (SP)+
4318 012712 100413          BMI     3$          ;BR IF YES
4319 012714 005737 007644    TST     SAVEFG      ;SAVE THE RH11/RPO4/5/6 REGISTERS?
4320 012720 100002          BPL     1$          ;BRANCH IF NO
4321 012722 004737 015200    JSR     PC,SVRH11   ;YES--SAVE THE REGISTERS
4322          001
4323          .IF NB
4324 1$:     JSR     PC,WC   ;SEE IF WRITE CHECK TO BE PUT IN QUEUE
4325          JSR     PC,GETREQ ;GET DPB POINTER
4326          TST     R2       ;ENTRY FOR DRIVE ?
4327          BEQ     2$       ;BR IF NOT
4328          JSR     PC,OPT   ;CALL OPTIMIZER
4329 012726 004737 010764    .IFF
4330          1$:     JSR     PC,OPT   ;CALL OPTIMIZER
4331          .ENDC
4332 012732 000417          BR     SC
4333 012734 012714 000113    2$:     MOV     #113,(R4)   ;CHECK OTHER DRIVES
4334 012740 000414          BR     SC               ;RELEASE THE DRIVE
4335 012742 052762 100100 000016 3$:     BIS     #BIT15:BIT06,16(R2) ;CHECK FOR OTHER DRIVES
4336 012750 004737 015740    JSR     PC,EMPTYQ     ;SET DATA ERROR FLAG
4337 012754 004737 015200    JSR     PC,EMPTYQ     ;EMPTY THE "DRIVE'S WAIT" QUEUE
4338 012760 012714 040111    JSR     PC,SVRH11    ;SAVE THE RH11/RPO4/5/6 REGISTERS
4339 012764 012714 000113    MOV     #40111,(R4)  ;ISSUE A "DRIVE CLEAR"
4340 012770 000400          MOV     #113,(R4)   ;ISSUE A RELEASE TO THE DRIVE
4341          BR     SC               ;CHECK FOR OTHER DRIVES
4342          .IF NB
4343          ;FORCED WRITE CHECK ROUTINE
4344 WC:     TST     AUTOCK   ;AUTOMATIC WRITE CHECKS ?
4345          BEQ     2$       ;BR IF NOT
4346          CMPB   #2,$CODE(R2) ;LAST OPERATION WRITE DATA ?
4347          BEQ     1$       ;BR IF IT WAS
4348          CMPB   #3,$CODE(R2) ;LAST OPERATION WRITE HEADER & DATA ?
4349          BNE     2$       ;BR IF NOT
4350          1$:     JSR     R0,DRVQUE ;PUT THE OPERATION IN THE QUEUE

```

```

4351 BR 2$ ;QUEUE IS FULL
4352 CLR 16(R2) ;CLEAR 'DONE' BIT IN DPB
4353 MOV $RPCS1(R2), $PREV0(R2) ;SAVE WRITE OPERATION CODE
4354 MOV $CYL(R2), $PREVA+2(R2) ;SAVE CYLINDER
4355 MOV $SEC(R2), $PREVA(R2) ;SAVE SECTOR AND TRACK ADDRESSES
4356 BICB #2, $CODE(R2) ;CHANGE WRITE TO CHECK
4357 BICB #20, $COMND(R2) ;CHANGE DRIVER CODE TO WRITE CHECK
4358 BISB #10, $COMND(R2) ;FINISH CHANGING CODE TO WRITE CHECK
4359 2$: RTS PC ;EXIT
4360
4361 000 .ENDC
4362 ;SPECIAL CONDITION ROUTINE
4363
4364 012772 116403 000016 SC: MOVB RPAS(R4), R3 ;READ "RPAS"
4365 012776 001012 BNE 2$ ;BRANCH IF ANY 'ATA' BITS SET
4366 013000 004037 014636 JSR RO, RD.RP ;READ CONTROL AND STATUS REGISTER
4367 013004 000000 RPCS1
4368 013006 012174 CIB
4369 013010 106126 ROLB (SP)+ ;IS "IE"=1?
4370 013012 100403 BMI 1$ ;YES, NO DRIVES TO CHECK
4371 001 .IF DF $ESCAPE
4372 JSR RO, ES.SAV ;SAVE THE ADDRESS IN '$ESCAPE'
4373 000 .ENDC
4374 013014 104001 ERROR 1 ;REPORT AN ILLEGAL INTERRUPT
4375 013016 004737 015316 JSR PC, SET.IE ;SET INTERRUPT ENABLE
4376 013022 000207 1$: RTS PC ;RETURN
4377 013024 005046 2$: CLR -(SP) ;PROCESS ALL DRIVES THAT HAVE
4378 013026 110316 MOVB R3, (SP) ;AN "ATA"=1
4379 013030 012703 000001 MOV #1, R3
4380 013034 005001 CLR R1
4381 013036 030316 SC3: BIT R3, (SP) ;ATA=1?
4382 013040 001005 BNE SC5 ;YES--BRANCH
4383 013042 005201 SC4: INC R1 ;MOVE TO THE NEXT DRIVE
4384 013044 106303 ASLB R3
4385 013046 001373 BNE SC3 ;BRANCH IF MORE TO CHECK?
4386 013050 005726 TST (SP)+ ;CLEAN OFF THE STACK
4387 013052 000207 RTS PC ;RETURN TO USER
4388 013054 105761 007576 SC5: TSTB DPINT(R1) ;INITIALIZING THE DRIVE ?
4389 013060 001402 BEQ 1$ ;BR IF NOT
4390 013062 000137 013750 JMP SC13 ;PROCESS THE DRIVE
4391 013066 105761 007606 1$: TSTB DPRQS(R1) ;PORT REQUEST OUTSTANDING ?
4392 013072 001402 BEQ 2$ ;BR IF NOT
4393 013074 000137 013750 JMP SC13 ;START THE OUTSTANDING COMMAND
4394 013100 105761 007556 2$: TSTB DRVSTA(R1) ;CHECK THE DRIVE STATUS
4395 013104 003025 BGT 5$ ;BRANCH IF ONLINE
4396 013106 105761 007624 TSTB ULDFLG(R1) ;UNLOAD IN PROGRESS?
4397 013112 003422 BLE 5$ ;BRANCH IF NOT
4398 013114 004737 016034 JSR PC, GETREQ ;GET DPB POINTER
4399 013120 004737 015200 JSR PC, SVRH11 ;SAVE THE RH11/RPO4/5/6 REGISTERS
4400 013124 004737 013700 JSR PC, SC12 ;SAVE RPD51, RPER1, RPER2, AND RPER3
4401 ;ALSO DO A DRIVE INIT (DRVINT)
4402 013130 105761 007556 TSTB DRVSTA(R1) ;DID DRIVE COME ONLINE?
4403 013134 003416 BLE 6$ ;NO---BRANCH
4404 013136 032737 040000 007536 BIT #BIT14, RPERRS ;WAS THERE AN ERROR?
4405 013144 001002 BNE 3$ ;BR IF ERROR
4406 013146 000137 013610 JMP SC11 ;NO ERROR

```



4407	013152	013705	007540	3\$:	MOV	RPERRS+2,R5	:YES -- PICKUP RPER1 AND
4408	013156	000476			BR	SC6A	:GO PROCESS THE ERROR
4409	013160	105761	007546	5\$:	TSTB	DRVACT(R1)	:DRIVE ACTIVE WITH COMMAND OR ERROR RECOVERY ?
4410	013164	001027			BNE	SC6	:BR IF EITHER
4411	013166	004737	013700		JSR	PC,SC12	:SAVE RPDS1, RPER1, RPER2, AND RPER3
4412							:ALSO DO A DRVINT
4413	013172	105761	007576	6\$:	TSTB	DPINT(R1)	:TRYING TO INIT THE DRIVE ?
4414	013176	001321			BNE	SC4	:BR IF YES, CHECK ON MORE DRIVES
4415	013200	105761	007556		TSTB	DRVSTA(R1)	:CHECK ON DRIVE'S STATUS
4416	013204	100412			BMI	7\$	:BR IF UNSAFE
4417	013206	032737	020000 007544		BIT	#BIT13,RPERRS+6	:ADDRESS PLUG CHANGED ?
4418	013214	001011			BNE	8\$	:BR IF YES
4419	013216	012746	000113		MOV	#113, -(SP)	:RELEASE COMMAND
4420	013222	004037	015012		JSR	RO,WRT.RP	:WRITE THE COMMAND INTO RPCS1
4421	013226	000000			RPCS1		:REGISTER INDEX
4422	013230	013560			SCB		:PARITY EXIT ADDRESS
4423	013232	011605		7\$:	MOV	(SP),R5	:PICKUP (RPAS) BEFORE THE ERROR CALL
4424		001		.IF DF	\$ESCAPE		
4425					JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
4426		000		.ENDC			
4427	013234	104002			ERROR	2	:REPORT THE UNEXPECTED ATTENTION
4428	013236	000701			BR	SC4	:GO CHECK FOR MORE ATA'S
4429	013240			8\$:			
4430		001		.IF DF	\$ESCAPE		
4431					JSR	RO,ES.SAV	:SAVE THE ADDRESS IN '\$ESCAPE'
4432		000		.ENDC			
4433	013240	104005			ERROR	5	:REPORT THE ADDRESS PLUG CHANGE
4434	013242	000677			BR	SC4	:CHECK FOR MORE DRIVES
4435	013244	006301		SC6:	ASL	R1	:SETUP TO ADDRESS WORDS
4436	013246	012761	177777 007650		MOV	#-1,TIMER(R1)	:STOP THE TIMER
4437	013254	006201			ASR	R1	:RESTORE THE DRIVE ADDRESS
4438	013256	004737	016034		JSR	PC,GETREQ	:GET THE DPB POINTER FROM THE QUEUE
4439	013262	010164	000010		MOV	R1,RPCS2(R4)	:SELECT DRIVE
4440	013266	004037	014636		JSR	RO,RD.RP	:READ THE RPO4'S STATUS REG.
4441	013272	000012			RPDS1		
4442	013274	013560			SCB		
4443	013276	011605			MOV	(SP),R5	:AND PUT IT IN R5
4444	013300	006126			ROL	(SP)+	:WAS THERE AN ERROR?
4445	013302	100407			BMI	1\$	:BR IF ERROR
4446	013304	105761	007546		TSTB	DRVACT(R1)	:CHECK DRIVE'S STATE
4447	013310	003137			SC11		:BR IF DRIVE ACTIVE WITH ORDER
4448	013312	052762	100210 000016		BIS	#BIT15!BIT07!BIT03,16(R2)	:INFORM USER OF ERROR RECOVER COMPLETION
4449	013320	000470			BR	SC7	
4450	013322	004037	014636	1\$:	JSR	RO,RD.RP	:READ ERROR REGISTER #1
4451	013326	000014			RPER1		
4452	013330	013560			SCB		
4453	013332	012605			MOV	(SP)+,R5	:AND SAVE IT IN R5
4454	013334	004737	015200		JSR	PC,SVRH11	:SAVE RH11/RPO4/5/6 REGISTERS
4455	013340	012746	000111		MOV	#111, -(SP)	:ISSUE A DRIVE CLEAR
4456	013344	004037	015012		JSR	RO,WRT.RP	
4457	013350	000000			RPCS1		
4458	013352	013560			SCB		
4459	013354	006105		SC6A:	ROL	R5	:WAS "UNSAFE" CONDITION =1?
4460	013356	100406			BMI	1\$	:BRANCH IF YES
4461	013360	005702			TST	R2	:ANYTHING IN QUEUE ?
4462	013362	001447			BEQ	SC7	:BR IF NOT

4463	013364	052762	100240	000016		BIS	#BIT15!BIT07!BIT05,16(R2)	; INFORM USER OF ERROR
4464	013372	000443				BR	SC7	
4465	013374	004037	014636		1\$:	JSR	RO, RD.RP	; READ DRIVE STATUS REG. #1
4466	013400	000012				RPDS1		
4467	013402	013560				SC8		
4468	013404	011605				MOV	(SP), R5	; SAVE RPDS1 IN R5
4469	013406	006126				ROL	(SP)+	; "ERR"=1?
4470	013410	100011				BPL	2\$	; BR IF NO--UNSAFE CLEARED
4471	013412	112761	177777	007556		MOVB	#-1, DRVSTA(R1)	; DRIVE IS UNSAFE
4472	013420	004737	015200			JSR	PC, SVRH11	; SAVE RH11/RPO4/5/6 REGISTERS
4473	013424	052762	110000	000016		BIS	#BIT15!BIT12,16(R2)	; INFORM USER OF UNSAFE ERROR
4474	013432	000423				BR	SC7	
4475	013434	032705	010000		2\$:	BIT	#BIT12,R5	; "MOL" = 1 ?
4476	013440	001015				BNE	3\$	; BR IF YES
4477	013442	112761	177777	007546		MOVB	#-1, DRVACT(R1)	; ACTIVE ERROR RECOVER
4478	013450	112761	000001	007556		MOVB	#1, DRVSTA(R1)	; ONLINE
4479	013456	006301				ASL	R1	
4480	013460	012761	072460	007650		MOV	#30000., TIMER(R1)	; START 30 SECOND TIMER
4481	013466	006201				ASR	R1	
4482	013470	000137	013042			JMP	SC4	
4483	013474	052762	100220	000016	3\$:	BIS	#BIT15!BIT07!BIT04,16(R2)	; INFORM USER OF ERROR
4484	013502	105061	007546		SC7:	CLRB	DRVACT(R1)	; DRIVE IS IDLE
4485	013506	004737	015740			JSR	PC, EMPTYQ	; DUMP THE QUEUE
4486	013512	105761	007624			TSTB	ULDFLG(R1)	; UNLOAD IN PROGRESS OR QUEUE?
4487	013516	003002				BGT	1\$	; BR IF NOT
4488	013520	105061	007624			CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
4489	013524	116164	007672	000016	1\$:	MOVB	ATABIT(R1), RPAS(R4)	; CLEAR ATTENTION BIT
4490	013532	105761	007556			TSTB	DRVSTA(R1)	; IS THE DRIVE UNSAFE ?
4491	013536	100406				BMI	2\$	; BR IF IT IS
4492	013540	012746	000113			MOV	#113, -(SP)	; RELEASE COMMAND
4493	013544	004037	015012			JSR	RO, WRT.RP	; WRITE THE COMMAND INTO RPCS1
4494	013550	000000				RPCS1		
4495	013552	013560				SC8		
4496	013554	000137	013042		2\$:	JMP	SC4	; CHECK FOR MORE DRIVES
4497	013556	105761	007546		SC8:	TSTB	DRVACT(R1)	; IS DRIVE IDLE?
4498	013564	001405				BEQ	1\$	; YES--BRANCH
4499	013566	004737	016034			JSR	PC, GETREQ	; GET DPB POINTER
4500	013572	004737	012074			JSR	PC, CI7	; PROCESS THE PARITY ERROR
4501	013576	000402				BR	2\$	; CONTINUE
4502	013600	004737	012122		1\$:	JSR	PC, CI7B	; PROCESS THE UNCORRECTABLE PARITY ERROR
4503	013604	000137	013042		2\$:	JMP	SC4	; CHECK MORE DRIVES
4504	013610	105761	007624		SC11:	TSTB	ULDFLG(R1)	; "UNLOAD IN PROGRESS"?
4505	013614	003402				BLE	1\$	; BRANCH IF NO
4506	013616	105061	007624			CLRB	ULDFLG(R1)	; CLEAR UNLOAD FLAG
4507	013622	105061	007546		1\$:	CLRB	DRVACT(R1)	; SET DRIVE IDLE
4508	013626	136137	007672	007620		BITB	ATABIT(R1), SRCHWT	; DOING A SEARCH OPERATION FOR
4509								; AN I/O COMMAND?
4510	013634	001012				BNE	2\$	; BRANCH IF YES
4511	013636	004737	016056			JSR	PC, POPQUE	; REMOVE REQUEST FROM QUEUE
4512	013642	052762	000200	000016		BIS	#BIT07,16(R2)	; SET "DONE" BIT
4513	013650	005737	007644			TST	SAVEFG	; SAVE THE REGISTERS?
4514	013654	100002				BPL	2\$	; BRANCH IF NO
4515	013656	004737	015200			JSR	PC, SVRH11	; YES--SAVE ALL OF THE RH11/RPO4/5/6 REG'S
4516	013662	116164	007672	000016	2\$:	MOVB	ATABIT(R1), RPAS(R4)	; CLEAR ATTENTION BIT
4517	013670	004737	010764			JSR	PC, OPT	; START A REQUEST
4518	013674	000137	013042			JMP	SC4	; CHECK FOR MORE DRIVES



```

4519 013700 010164 000010          SC12:  MOV    R1,RPCS2(R4) ;SELECT DRIVE
4520 013704 016437 000012 007536  MOV    RPOS1(R4),RPERRS ;SAVE THE FOUR REGISTERS THAT
4521 013712 016437 000014 007540  MOV    RPER1(R4),RPERRS+2 ;WILL TELL US SOMETHING
4522 013720 016437 000040 007542  MOV    RPER2(R4),RPERRS+4
4523 013726 016437 000042 007544  MOV    RPER3(R4),RPERRS+6
4524 013734 004037 010134          JSR    RO,DRVINT ;INIT. THE STATE OF THE DRIVE
4525 013740 000401          BR     1$ ;TAKE ERROR EXIT
4526 013742 000207          PC     ;RETURN
4527 013744 005726          1$:   TST    (SP)+ ;POP PC OFF OF THE STACK
4528 013746 000704          BR     SC8 ;PROCESS THE PARITY ERROR
4529 013750 006301          SC13: ASL    R1 ;SETUP TO ADDRESS WORDS
4530 013752 012761 177777 007650  MOV    #-1,TIMER(R1) ;STOP THE TIMER
4531 013760 006201          ASR    R1
4532 013762 010164 000010 000016  MOV    R1,RPCS2(R4) ;SELECT THE DRIVE
4533 013766 116164 007672          MOVB  ATABIT(R1),RPAS(R4) ;CLEAR THE ATTENTION BIT
4534 013774 032714 004000          BIT   #BIT11,(R4) ;DRIVE AVAILABLE ?
4535 014000 001006          BNE   1$ ;BR IF AVAILABLE
4536 014002 006301          ASL    R1
4537
4538 014004 012761 023420 007650  MOV    #10000.,TIMER(R1) ;START 10 SEC TIMER AGAIN
4539
4540 014012 006201          ASR    R1
4541 014014 000433          BR     3$ ;EXIT
4542 014016 105761 007576          1$:   TSTB  DPINT(R1) ;INITIALIZING THE DRIVE ?
4543 014022 001424          BEQ   2$ ;BR IF NOT
4544 014024 105061 007576          CLRB  DPINT(R1) ;CLEAR THE INIT INDICATOR
4545 014030 004037 010134          JSR   RO,DRVINT ;GO INIT THE DRIVE
4546 014034 000240          NOP   ;DUMMY PARITY ERROR RETURN
4547 014036 105761 007556          TSTB  DRVSTA(R1) ;DRIVE ONLINE ?
4548 014042 003014          BGT   2$ ;BR IF YES -- START ORDER
4549 014044 005702          TST   R2 ;QUEUE ENTRY FOR THE DRIVE
4550 014046 001416          BEQ   3$ ;BR IF NOT
4551 014050 004737 016034          JSR   PC,GETREQ ;GET DPB ADDRESS
4552 014054 052762 140000 000016  BIS   #BIT15:BIT14,16(R2) ;INFORM USER THAT DRIVE OFFLINE
4553 014062 004737 015200          JSR   PC,SVRH11 ;SAVE THE REGISTERS
4554 014066 004737 015740          JSR   PC,EMPTYQ ;EMPTY THE REQUEST QUEUE
4555 014072 000404          BR     3$
4556 014074 105061 007606          2$:   CLRB  DPRQS(R1) ;CLEAR THE PORT REQUEST INDICATOR
4557 014100 004737 010764          JSR   PC,OPT ;START THE PENDING REQUEST
4558 014104 000137 013042          3$:   JMP   SC4 ;PROCESS OTHER DRIVES
4559
4560          ;RPO4/5/6 TIMER ROUTINE
4561          ;CALL
4562          ;
4563          ;   MOV    #TIME -(SP) ;ELAPSED TIME IN MILLISECONDS ON THE STACK
4564          ;   JSR    PC,RPTMR ;CALL RPO4/5/6 TIME ROUTINE
4565
4565 014110 005737 007622          RPTMR: TST   ACTDRV ;CHECK "ACTDRV & ACTSTR"
4566 014114 001030          BNE   4$ ;IF NON ZERO EXIT
4567 014116 112737 000001 007623  MOVB  #1,ACTSTR ;SET "ACTSTR"
4568 014124 104412          SAVREG ;SAVE R0 - R5
4569 014126 005001          CLR   R1 ;START WITH DRIVE 0
4570 014130 005003          CLR   R3
4571 014132 005763 007650          1$:   TST   TIMER(R3) ;IS THE TIMER RUNNING?
4572 014136 002407          BLT   2$ ;BRANCH IF NO
4573 014140 166663 000002 007650  SUB   2(SP),TIMER(R3) ;COUNT THE INTERVAL
4574 014146 003003          BGT   2$ ;BR IF NO SOFTWARE TIMEOUT

```

```

4575 014150 004737 014202      JSR    PC,STO      ;CALL SOFTWARE TIMEOUT ROUTINE
4576 014154 000405              BR     3$          ;GO TO THE EXIT
4577 014156 005201      2$:  INC     R1      ;MOVE TO NEXT DRIVE
4578 014160 005723              TST    (R3)+
4579 014162 022701 000010      CMP    #8.,R1     ;OUT OF DRIVES?
4580 014166 003361              BGT    1$          ;BRANCH IF NO
4581 014170 104413      3$:  RESREG          ;RESTORE R0 - R5
4582 014172 105037 007623      CLRB  ACTSTR      ;ZERO ACTIVE SOFTWARE TIMEOUT ROUTINE FLAG
4583 014176 012616      4$:  MOV    (SP)+,(SP) ;ADJUST THE STACK
4584 014200 000207      RTS    PC          ;RETURN

;SOFTWARE TIMEOUT ROUTINE
;NOTE: THIS ROUTINE MUST BE ENTERED AT PRIORITY 6
;OR GREATER
;CALL:  STO      ;DRIVE NUMBER
;        MOV     #DRVNUM,R1 ;DRIVE NUMBER
;        JSR    PC,STO      ;CALL
;        RETURN

4596 014202 010146      STO:  MOV     R1,-(SP) ;SAVE R1
4597 014204 010346      MOV     R3,-(SP)    ;SAVE R3
4598 014206 013704 007704      MOV     RPADR,R4   ;GET ADDRESS OF "RPCS1"
4599 014212 010164 000010      MOV     R1,RPCS2(R4) ;SELECT THE DRIVE
4600 014216 004037 014636      JSR    R0,RD.RP    ;READ "DRIVE STATUS REG"
4601 014222 000012      RPOS1
4602 014224 014524      STOS
4603 014226 105726      TSTB  (SP)+        ;IS "DRY"=1?
4604 014230 100477      BMI   ST02         ;BR IF YES
4605 014232 105761 007576      ST01: TSTB  DPINT(R1) ;TRYING TO INTIALIZE THE DRIVE ?
4606 014236 001074      BNE   ST02         ;BR IF YES
4607 014240 105761 007606      TSTB  DPRQS(R1)   ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
4608 014244 001071      BNE   ST02         ;BR IF YES
4609 014246 013702 007616      MOV     TRNSWT,R2  ;PICKUP TRANSFER WAIT QUEUE
4610 014252 020137 007670      CMP    R1,DTUW    ;TRANSFER UNDERWAY ON THIS DRIVE?
4611 014256 001402      BEQ   1$          ;BRANCH IF YES
4612 014260 004737 016034      JSR    PC,GETREQ  ;GET DPB ADDRESS
4613 014264 052762 101000 000016 1$:  BIS    #BIT15:BIT09,16(R2) ;SET THE ERROR FLAGS
4614 014272 004737 015200      JSR    PC,SVRH11  ;SAVE RH11/RPO4/5/6 REGISTERS
4615 014276 012764 000040 000010      MOV     #BIT05,RPCS2(R4) ;"INIT" THE MASS BUS
4616 014304 105061 007546      CLRB  DRVACT(R1)  ;DRIVE IS IDLE
4617 014310 105061 007624      CLRB  ULDFLG(R1) ;CLEAR THE UNLOAD FLAG
4618 014314 005001      CLR   R1          ;START WITH DRIVE 0
4619 014316 005003      CLR   R3
4620 014320 004037 010134      2$:  JSR    R0,DRVINT  ;INIT. THIS DRIVE
4621 014324 000477      BR     ST05        ;PARITY ERROR RETURN
4622 014326 105761 007546      TSTB  DRVACT(R1)  ;DRIVE IDLE BEFORE THE INIT.?
4623 014332 001414      BEQ   4$          ;YES--BRANCH
4624 014334 013702 007616      MOV     TRNSWT,R2  ;GET TRANSFER WAIT QUEUE
4625 014340 023701 007670      CMP    DTUW,R1     ;WAS THERE I/O ON THIS DRIVE?
4626 014344 001402      BEQ   3$          ;YES--BRANCH
4627 014346 004737 016034      JSR    PC,GETREQ  ;GET THE DPB POINTER FROM QUEUE
4628 014352 052762 100400 000016 3$:  BIS    #BIT15:BIT08,16(R2) ;INFORM USER OF INIT.
4629 014360 105061 007546      CLRB  DRVACT(R1)  ;SET DRIVE ACTIVE TO IDLE
4630 014364 105061 007624      4$:  CLRB  ULDFLG(R1) ;NO UNLOAD

```



```

4631 014370 012763 177777 007650      MOV      #-1,TIMER(R3)      ;STOP THE TIMER
4632 014376 005723                    TST      (R3)+            ;UPDATE THE INDEX
4633 014400 005201                    INC      R1                ;INCREMENT THE DRIVE NUMBER
4634 014402 022701 000010              CMP      #B.,R1           ;LAST DRIVE BEEN CHECKED?
4635 014406 003344                    BGT     2$                ;NO--LOOP
4636 014410 012737 177777 007670      MOV      #-1,DTUW         ;NO DATA TRANSFERS UNDERWAY
4637 014416 005037 007616              CLR     TRNSWT           ;CLEAR TRANSFER WAIT QUEUE
4638 014422 004737 015662              JSR     PC,CLRQUE        ;CLEAR ALL REQUEST QUEUES
4639 014426 000500                    BR      ST09              ;EXIT
4640 014430 116405 000016              MOV     RPAS(R4),R5      ;READ ATTENTION REG
4641 014434 136105 007672              BIT     ATABIT(R1),R5   ;IS ATTENTION FOR THIS DRIVE UP ?
4642 014440 001017                    BNE     ST03              ;YES--BRANCH
4643 014442 105761 007576              TST     DPINT(R1)        ;TRYING TO INITIALIZE THE DRIVE ?
4644 014446 001031                    BNE     ST06              ;BR IF YES - DRIVE NOT ONLINE
4645 014450 105761 007606              TST     DPRQS(R1)        ;OUTSTANDING PORT REQUEST FOR THE DRIVE ?
4646 014454 001045                    BNE     ST07              ;BR IF YES - NO RESPONSE TO REQUEST
4647 014456 020137 007670              CMP     R1,DTUW          ;DATA TRANSFER UNDERWAY FOR THIS DRIVE
4648 014462 001263                    BNE     ST01              ;BR IF NO
4649 014464 004037 014636              JSR     RD,RD.RP         ;YES--CHECK "RDY"
4650 014470 000000                    RPCS1
4651 014472 014524                    ST05
4652 014474 105726                    TST     (SP)+
4653 014476 100255                    BPL     ST01              ;BR IF "RDY"=0
4654 014500 105761 007576              ST03:  TST     DPINT(R1)   ;INITIALIZING THE DRIVE ?
4655 014504 001003                    BNE     1$                ;BR IF INIT PENDING
4656 014506 105761 007606              TST     DPRQS(R1)        ;PORT REQUEST PENDING ?
4657 014512 001446                    BEQ     ST09              ;BR IF NOT
4658 014514 012763 177777 007650  1$:  MOV      #-1,TIMER(R3)    ;STOP THE TIMER
4659 014522 000442                    BR      ST09              ;EXIT
4660 014524 004737 012174              ST05:  JSR     PC,CIB       ;GO HANDLE THE PARITY ERROR
4661 014530 000437                    BR      ST09
4662 014532 105051 007576              ST06:  CLRB   DPINT(R1)    ;CLEAR THE INITIALIZE INDICATOR
4663 014536 105061 007556              CLRB   DRVSTA(R1)        ;SET UNIT OFFLINE
4664 014542 012763 177777 007650      MOV      #-1,TIMER(R3)   ;STOP THE TIMER
4665 014550 004737 016034              JSR     PC,GETREQ        ;GET THE DPB ADDRESS
4666 014554 005702                    TST     R2                ;REQUEST IN QUEUE ?
4667 014556 001424                    BEQ     ST09              ;BR IF NOT
4668 014560 052762 140000 000016      BIS     #BIT15:BIT14,16(R2) ;INFORM THE USER DRIVE NOT AVAILABLE
4669 014566 000414                    BR      ST08              ;FINISH
4670 014570 012763 177777 007650  ST07:  MOV      #-1,TIMER(R3)    ;STOP THE TIMER
4671 014576 105061 007606              CLRB   DPRQS(R1)        ;CLEAR PORT REQUEST INDICATOR
4672 014602 004737 016034              JSR     PC,GETREQ        ;GET DPB ADDRESS
4673 014606 005702                    TST     R2                ;QUEUE ENTRY FOR DRIVE ?
4674 014610 001407                    BEQ     ST09              ;BR IF NONE
4675 014612 012762 100004 000016      MOV     #BIT15:BIT2,16(R2) ;INFORM USER OF PORT REQUEST ERROR
4676 014620 004737 015740              ST08:  JSR     PC,EMPTYQ   ;CLEAR THE QUEUE FOR THE DRIVE
4677 014624 004737 015200              JSR     PC,SVRH11        ;SAVE THE REGISTERS
4678 014630 012603                    ST09:  MOV     (SP)+,R3      ;RESTORE R3
4679 014632 012601                    MOV     (SP)+,R1        ;RESTORE R1
4680 014634 000207                    RTS      PC              ;RETURN
4681
4682 ;ROUTINE TO READ A RH11/RPO4/5/6 REGISTER
4683 ;
4684 ;CALL
4685 ;
4686 ;      JSR     RD,RD.RP      ;GO READ A REGISTER
;      INDEX  ;REG. INDEX FROM BASE

```

```

4687      ; ERRADR      ;ERROR ADDRESS--PROCESS ERROR STARTING
4688      ; RETURN      ;AT THIS ADDRESS
4689      ;              ;CONTENTS OF REG. IS ON THE STACK
4690
4691 014636 013737 007702 015000 RD.RP:  MOV    MCPEMX, RD.RP2  ;MAX. RETRYS ALLOWED
4692 014644 011646          015000      MOV    (SP), -(SP)  ;SAVE RD FOR RETURN
4693 014646 013737 007704 014662      MOV    RPADR, RD.ADR ;FORM THE DESIRED ADDRESS
4694 014654 062037 014662          014662      ADD    (RD)+, RD.ADR ;USING THE BASE AND THE INDEX
4695 014660 013727          014662      RD.RP1: MOV    @ (PC)+, (PC)+ ;READ THE DESIRED REGISTER OF THE RPO4
4696 014662 000000          014662      RD.ADR: .WORD  0      ;ADDRESS IS FORMED HERE
4697 014664 000000          014662      RD.WRD: .WORD  0      ;REG. CONTENTS PUT HERE
4698 014666 013766 014664 000002      MOV    RD.WRD, 2(SP) ;RETURN IT TO THE USER
4699 014674 013746 007704          014662      MOV    RPADR, -(SP)  ;PUT THE ADDRESS ON THE STACK
4700 014700 062716 000010          014662      ADD    #RPCS2, (SP)  ;FORM THE ADDRESS OF RPCS2
4701 014704 032736 010000          014662      BIT    #BIT12, @ (SP)+ ;CHECK THE 'NED' BIT
4702 014710 001035          014662      BNE    RD.RP3        ;BR IF DRIVE NON-EXISTENT
4703 014712 017746 172766          014662      MOV    @RPADR, -(SP) ;READ RPCS1
4704 014716 032716 020000          014662      BIT    #BIT13, (SP)  ;DID MCPE SET?
4705 014722 001002          014662      BNE    IS           ;BRANCH IF YES
4706 014724 022620          014662      CMP    (SP)+, (RD)+ ;ADJUST FOR RETURN
4707 014726 000430          014662      BR     RD.RP4       ;EXIT
4708 014730
4709      001
4710      .IF DF $ESCAPE
4711      000
4712 014730 104003          000      .ENDC
4713 014732 005737 007670          000      ERROR  3
4714 014736 100405          007670      TST    DTUW
4715 014740 032716 040000          007670      BMI    2$
4716 014744 001402          040000      BIT    #BIT14, (SP) ;NO--"TRE"=1?
4717 014746 005726          040000      BEQ    2$
4718 014750 000415          040000      TST    (SP)+
4719 014752 052716 040000          040000      BR     RD.RP3
4720 014756 000316          040000      2$:   BIS    #BIT14, (SP) ;YES--CLEAN OFF THE STACK AND
4721 014760 013737 007704 014774      SWAB   (SP)          ;TAKE THE FATAL ERROR EXIT
4722 014766 005237 014774          014774      MOV    RPADR, 3$    ;CLEAR "MCPE" BY SENDING A "1" TO "TRE"
4723 014772 112637          014774      INC    3$           ;POSITION BEFORE WRITING
4724 014774 000000          014774      MOV    (SP)+, @ (PC)+ ;FORM ADDRESS OF HIGH BYTE
4725 014776 005327          014774      3$:   .WORD  0
4726 015000 000003          015000      RD.RP2: DEC    (PC)+ ;WRITE THE HIGH BYTE OF RPCS1
4727 015002 002326          015000      RD.RP3: .WORD  3
4728 015004 011000          015000      BGE    RD.RP1       ;ADDRESS STORAGE
4729 015006 012616          015000      RD.RP3: MOV    (RD), RD ;EXCEEDED MAX. RETRYS
4730 015010 000200          015010      RD.RP4: MOV    (SP)+, (SP) ;BRANCH IF NO
4731      ;ROUTINE TO WRITE A REGISTER
4732      ;CALL
4733      ;
4734      ;
4735      ;
4736      ;
4737      ;
4738      ;
4739      ;
4740      ;
4741 015012 013737 007702 015162 WRT.RP: MOV    MCPEMX, WRT.R2 ;DATA TO BE LOADED ON THE STACK
4742 015020 016637 000002 015100      MOV    2(SP), WRT.WD ;CALL THE ROUTINE TO LOAD(WRITE) THE REG.

```



```

4743 015026 012616          MOV      (SP)+,(SP)          ;ADJUST THE STACK
4744 015030 012037 015102    MOV      (RO)+,WRT.AD        ;GET INDEX OF REGISTER TO BE WRITTEN
4745 015034 001015          BNE      1$                  ;BRANCH IF NOT RPCS1
4746 015036 122737 000150 015100  CMPB     #150,WRT.WD         ;IS THE COMMAND FOR DATA TRANSFERS?
4747 015044 002411          BLT      1$                  ;YES---DON'T GET THE OLD A16 & A17, & PSEL
4748 015046 004037 014636    JSR      RO,RO.RP            ;NO---COMBINE A16&A17, & PSEL WITH
4749 015052 000000          RPCS1                          ;THE COMMAND BEFORE SENDING IT TO
4750 015054 015170          WRT.R3                          ;THE RH11/RPO4
4751 015056 000316          SWAB     (SP)
4752 015060 042716 177770    BIC      #1C7,(SP)
4753 015064 112637 015101    MOVB     (SP)+,WRT.WD+1
4754 015070 063737 007704 015102  1$:     ADD      RPADR,WRT.AD        ;FORM THE ADDRESS OF THE DISK REG.
4755 015076 012737          WRT.R1: MOV      (PC)+,@(PC)+   ;LOAD THE DESIRED REG.
4756 015100 000000          WRT.WD: .WORD     0          ;WORD TO WRITE GOES HERE
4757 015102 000000          WRT.AD: .WORD     0          ;ADDRESS IS FORMED HERE
4758 015104 013746 007704    MOV      RPADR,-(SP)         ;PUT THE ADDRESS ON THE STACK
4759 015110 062716 000010    ADD      #RPCS2,(SP)        ;FORM THE ADDRESS OF RPCS2
4760 015114 032736 010000    BIT      #BIT12,@(SP)+     ;CHECK THE 'NED' BIT
4761 015120 001023          BNE      WRT.R3              ;BR IF DRIVE NON-EXISTENT
4762 015122 004037 014636    JSR      RO,RO.RP            ;CHECK FOR PARITY ERROR ON WRITE
4763 015126 000014          RPER1
4764 015130 015170          WRT.R3
4765 015132 032726 000010    BIT      #BIT03,(SP)+
4766 015136 001416          BEQ      WRT.R4              ;BRANCH IF "PAR=0"
4767 015140 016037 177776 015152  MOV      -2(RO),1$          ;PICKUP THE INDEX
4768 015146 004037 014636    JSR      RO,RO.RP            ;READ THE REG.
4769 015152 000000          1$:     .WORD     0          ;REG. INDEX
4770 015154 015170          WRT.R3                          ;RETURN TO THIS ADDRESS ON ERROR
4771          001
4772          .IF DF $ESCAPE
4773          .ENDC
4774 015156 104004          ERROR    4                  ;SAVE THE ADDRESS IN '$ESCAPE'
4775 015160 005327          WRT.R2: DEC      (PC)+       ;REPORT THE PARITY ON WRITE ERROR
4776 015162 000003          BGE      WRT.R1              ;DECREMENT THE ERROR COUNT
4777 015164 002344          TST      (SP)+              ;RETRY COUNTER
4778 015166 005726          WRT.R3: MOV      (RO) RO     ;TRY AGAIN IF NOT FINISHED
4779 015170 011000          BR       WRT.R5              ;CLEAN OFF THE STACK
4780 015172 000401          WRT.R4: TST      (RO)+       ;TAKE THE "PARITY ON WRITE" ERROR EXIT
4781 015174 005720          WRT.R5: RTS      RO          ;EXIT
4782 015176 000200          ;ROUTINE TO SAVE THE RH11/RPO4/5/6 REGISTERS AS PER DPB+14
4783
4784          ;CALL
4785
4786          MOV      #DPBNUM,R2   ;DPB POINTER TO R2
4787          JSR      PC,SVRH11    ;SAVE THE DRIVES REG'S
4788
4789          SVRH11: SAVREG
4790          TST      R2           ;SAVE R0 - R5
4791          BEQ      4$           ;QUEUE ENTRY FOR THE DRIVE ?
4792          MOV      RPADR,R4    ;BR IF NONE
4793          MOV      (R2),RPCS2(R4)
4794          MOVB     14(R2),R3    ;SELECT DRIVE
4795          BEQ      6$           ;GET THE ERROR TABLE POINTER
4796          CLR      3$           ;EXIT IF NO ADDRESS
4797          CMP      3$,#RPDB    ;COUNTER & POINTER
4798          ;REACHED THE BUFFER REGISTER ?

```



```

4799 015236 001006          BNE      2$          ;BR IF NOT
4800 015240 032764 000200 000010  BIT      #BIT07,RPCS2(R4) ;'OR' SET ?
4801 015246 001002          BNE      2$          ;BR IF SET
4802 015250 005023          CLR      (R3)+      ;STORE RPDB AS ZEROES
4803 015252 000405          BR       4$          ;CONTINUE
4804 015254 004037 014636  2$: JSR      RO,RO.RP  ;READ THE SELECTED REGISTER
4805 015260 000000  3$: .WORD  0          ;REGISTER INDEX
4806 015262 015306          SS       ;ERROR RETURN ADDRESS
4807 015264 012623          MOV      (SP)+,(R3)+ ;STORE THE REGISTER CONTENTS
4808 015266 023727 015260 000046  4$: CMP      3$,#RPEC2 ;REACHED THE END ?
4809 015274 001406          BEQ     6$          ;BR IF YES
4810 015276 062737 000002 015260  ADD      #2,3$      ;INCREMENT THE REGISTER INDEX
4811 015304 000751          BR       1$          ;CONTINUE READING THE REGISTERS
4812 015306 004737 012074  5$: JSR      PC,C17   ;PROCESS THE UNCORRECTABLE PARITY ERROR
4813 015312 104413  6$: RESREG ;RESTORE RO - R5
4814 015314 000207          RTS      PC          ;RETURN
4815
4816 ;ROUTINE TO SET THE INTERRUPT WITHOUT GETTING A "TRE"
4817 ;CALL
4818 ;
4819 ;   MOV      #DRVNUM,R1      ;DRIVE NUMBER TO R1
4820 ;   JSR      PC,SET.IE      ;SET "IE"
4821 ;
4822 ;
4823 ;SET.IE: MOV      R4,-(SP)    ;SAVE R4
4824 ;         MOV      RPADR,R4  ;PICKUP ADDRESS OF RPCS1
4825 ;         MOV      R1,RPCS2(R4) ;SELECT DRIVE
4826 ;         MOV      (R4),-(SP) ;READ RPCS1
4827 ;         BIS      #BIT14,(SP) ;SET THE "TRE" BIT OF THE WORD READ
4828 ;         SWAB   (SP)       ;ADJUST FOR DATO
4829 ;         MOVB   #BIT06,(R4) ;SET "IE"
4830 ;         BIT    #BIT12,RPCS2(R4) ;IS "NED"=1?
4831 ;         BNE    1$         ;YES--CLEAR "TRE"
4832 ;         TST   (SP)+      ;CLEAN OFF THE STACK
4833 ;         BR     2$         ;
4834 ;         MOVB   (SP)+,1(R4) ;CLEAR "TRE"
4835 ;         MOV    (SP)+,R4   ;RESTORE R4
4836 ;         RTS    PC        ;RETURN TO CALLER
4837 ;
4838 ;QUEUE COUNT
4839 ;QCNT: .BYTE  0          ;DRIVE 0
4840 ;       .BYTE  0          ;DRIVE 1
4841 ;       .BYTE  0          ;DRIVE 2
4842 ;       .BYTE  0          ;DRIVE 3
4843 ;       .BYTE  0          ;DRIVE 4
4844 ;       .BYTE  0          ;DRIVE 5
4845 ;       .BYTE  0          ;DRIVE 6
4846 ;       .BYTE  0          ;DRIVE 7
4847 ;
4848 ;QUEUE INPUT POINTERS
4849 ;QINPT: .WORD  QDRV0      ;DRIVE 0
4850 ;        .WORD  QDRV1      ;DRIVE 1
4851 ;        .WORD  QDRV2      ;DRIVE 2
4852 ;        .WORD  QDRV3      ;DRIVE 3
4853 ;        .WORD  QDRV4      ;DRIVE 4
4854 ;        .WORD  QDRV5      ;DRIVE 5

```



```

4855 015414 015622          .WORD  QDRV6          ;DRIVE 6
4856 015416 015642          .WORD  QDRV7          ;DRIVE 7
4857
4858          ;QUEUE OUTPUT POINTERS
4859
4860 015420 015462  QOUTPT: .WORD  QDRV0          ;DRIVE 0
4861 015422 015502          .WORD  QDRV1          ;DRIVE 1
4862 015424 015522          .WORD  QDRV2          ;DRIVE 2
4863 015426 015542          .WORD  QDRV3          ;DRIVE 3
4864 015430 015562          .WORD  QDRV4          ;DRIVE 4
4865 015432 015602          .WORD  QDRV5          ;DRIVE 5
4866 015434 015622          .WORD  QDRV6          ;DRIVE 6
4867 015436 015642          .WORD  QDRV7          ;DRIVE 7
4868
4869 015440 015462  QSTART: .WORD  QDRV0          ;DRIVE 0 START ADDRESS
4870 015442 015502  QSTOP:  .WORD  QDRV1          ;DRIVE 0 STOP ADDRESS & DRIVE 1 START ADDRESS
4871 015444 015522          .WORD  QDRV2          ;STOP DRIVE 1--START DRIVE 2
4872 015446 015542          .WORD  QDRV3          ;STOP DRIVE 2--START DRIVE 3
4873 015450 015562          .WORD  QDRV4          ;STOP DRIVE 3--START DRIVE 4
4874 015452 015602          .WORD  QDRV5          ;STOP DRIVE 4--START DRIVE 5
4875 015454 015622          .WORD  QDRV6          ;STOP DRIVE 5--START DRIVE 6
4876 015456 015642          .WORD  QDRV7          ;STOP DRIVE 6--START DRIVE 7
4877 015460 015662          .WORD  QTERM
4878
4879          ;DRIVE REQUEST QUEUES
4880
4881 015462 000010  QDRV0:  .BLKW  10
4882 015502 000010  QDRV1:  .BLKW  10
4883 015522 000010  QDRV2:  .BLKW  10
4884 015542 000010  QDRV3:  .BLKW  10
4885 015562 000010  QDRV4:  .BLKW  10
4886 015602 000010  QDRV5:  .BLKW  10
4887 015622 000010  QDRV6:  .BLKW  10
4888 015642 000010  QDRV7:  .BLKW  10
4889          QTERM=.
4890
4891          ;ROUTINE TO CLEAR ALL OF THE REQUEST QUEUES
4892
4893          ;CALL
4894          ;
4895          JSR      PC,CLRQUE
4896 015662 104412  CLRQUE: SAVREG          ;SAVE R0 - R5
4897 015664 012702 015370  MOV      #QCNT,R2          ;ZERO THE QUEUE COUNTS
4898 015670 005022          CLR      (R2)+          ;DRIVES 0 & 1
4899 015672 005022          CLR      (R2)+          ;DRIVES 2 & 3
4900 015674 005022          CLR      (R2)+          ;DRIVES 4 & 5
4901 015676 005022          CLR      (R2)+          ;DRIVES 6 & 7
4902 015700 012703 000010  MOV      #8,R3          ;MOVE THE STARTING
4903 015704 012701 015440  MOV      #QSTART,R1          ;ADDRESS OF THE QUEUE INTO
4904 015710 012122          1$:  MOV      (R1)+,(R2)+          ;THE QUEUE INPUT POINTER
4905 015712 005303          DEC      R3
4906 015714 001375          BNE     1$
4907 015716 012703 000010  MOV      #8,R3          ;MOVE THE STARTING ADDRESS
4908 015722 012701 015440  MOV      #QSTART,R1          ;OF THE QUEUE INTO THE
4909 015726 012122          2$:  MOV      (R1)+,(R2)+          ;QUEUE OUTPUT POINTER
4910 015730 005303          DEC      R3

```

```

4911 015732 001375          BNE      2$
4912 015734 104413          RESREG
4913 015736 000207          RTS      PC          ;RESTORE R0 - R5
4914
4915          ;EMPTY THE QUEUE SPECIFIED BY R1
4916          ;CALL
4917          ;
4918          ;MOV      DRVNUM,R1          ;DRIVE NUMBER TO R1
4919          ;JSR      PC,EMPTYQ
4920
4921 015740 105061 015370      EMPTYQ: CLR  B   QCNT(R1)          ;CLEAR NUMBER OF ITEMS IN QUEUE
4922 015744 006301          ASL      R1
4923 015746 016161 015400 015420      MOV      QINPT(R1),QOUTPT(R1) ;SET OUTPUT QUEUE POINTER=INPUT POINTER
4924 015754 006201          ASR      R1
4925 015756 000207          RTS      PC
4926
4927          ;ROUTINE TO PUT A REQUEST IN QUEUE
4928          ;CALL
4929          ;
4930          ;MOV      #DRVNUM,R1          ;DRIVE NUMBER
4931          ;MOV      #DPB,R2          ;ADDRESS OF PARAMETER BLOCK
4932          ;JSR      R0,DRVQUE          ;GO PUT REQUEST IN QUEUE
4933          ;RETURN1          ;RETURN HERE IF QUEUE IS FULL
4934          ;RETURN2          ;RETURN HERE IF REQUEST IS IN QUEUE
4935
4936 015760 122761 000010 015370      DRVQUE: CMP  B   #10,QCNT(R1)          ;IS QUEUE FULL?
4937 015766 001421          BEQ      2$          ;BR IF YES-TAKE RETURN1
4938 015770 105261 015370          INCB    QCNT(R1)          ;INCREMENT QUEUE COUNT
4939 015774 006301          ASL      R1
4940 015776 010271 015400          MOV      R2,QINPT(R1)          ;PUT THIS REQUEST IN QUEUE
4941 016002 062761 000002 015400          ADD     #2,QINPT(R1)          ;UPDATE THE QUEUE POINTER
4942 016010 026161 015400 015442          CMP     QINPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER
4943 016016 001003          BNE     1$          ;BRANCH IF NO
4944 016020 016161 015440 015400          MOV     QSTART(R1),QINPT(R1) ;YES--RESET POINTER
4945 016026 006201          1$:    ASR      R1
4946 016030 005720          TST     (R0)+          ;TAKE RETURN 2
4947 016032 000200          2$:    RTS      R0          ;RETURN TO USER
4948
4949          ;ROUTINE TO GET THE "DPB" ADDRESS OF NEXT REQUEST IN QUEUE
4950          ;CALL
4951          ;
4952          ;MOV      #DRVNUM,R1          ;DRIVE NUMBER TO R1
4953          ;JSR      PC,GETREQ          ;GO GET THE REQUEST
4954          ;RETURN          ;R2="DPB" ADDRESS OF THE REQUEST
4955          ;R2=0 IF NO REQUEST IN QUEUE
4956
4957 016034 005002          GETREQ: CLR  R2
4958 016036 105761 015370          TSTB   QCNT(R1)          ;IS THERE ANY REQUEST IN QUEUE?
4959 016042 001404          BEQ     2$          ;NO---BRANCH
4960 016044 006301          1$:    ASL      R1
4961 016046 017102 015420          MOV     QOUTPT(R1),R2          ;PICKUP "DPB" POINTER FOR THIS DRIVE
4962 016052 006201          ASR     R1
4963 016054 000207          2$:    RTS      PC          ;RETURN TO USER
4964
4965          ;ROUTINE TO "POP" THE REQUEST FROM QUEUE
4966          ;

```



```

4967      ;CALL
4968      ;
4969      ;
4970      ;
4971      ;
4972      016056 105361 015370      POPQUE: DECB      QCNT(R1)      ;DECREMENT QUEUE COUNT
4973      016062 006301      ASL      R1
4974      016064 017102 015420      MOV      @QOUTPT(R1),R2 ;GET THE "DPB" POINTER
4975      016070 062761 000002 015420      ADD      #2,QOUTPT(R1) ;UPDATE THE QUEUE POINTER
4976      016076 026161 015420 015442      CMP      QOUTPT(R1),QSTOP(R1) ;TIME TO RESET THE POINTER?
4977      016104 001003      BNE      IS ;NO--BRANCH TO EXIT
4978      016106 016161 015440 015420      MOV      QSTART(R1),QOUTPT(R1) ;YES--RESET THE POINTER
4979      016114 006201      IS:      ASR      R1
4980      016116 000207      RTS      PC ;RETURN TO USER
4981
4982      001
4983      ; IF DF $ESCAPE
4984      ; ROUTINE TO SAVE THE CONTENTS OF 'ESCAPE' WHEN THE DRIVER
4985      ; REPORTS AN ERROR DIRECTLY.
4986      ;
4987      ;CALL
4988      ;
4989      ;
4990      ;
4991      ;
4992      ;
4993      ;
4994      ;
4995      ;
4996      ;
4997      000
4998      ;
4999      ;
5000      001
5001      ; *****
5002      ; .IFF
5003      ; .NLIST
5004      ; .REPT
5005      ; .LIST
5006      ; *****
5007      ; .NLIST
5008      ; .ENDR
5009      ; .LIST
5010      000
5011      ; .ENDC
5012      ; .SBTTL DATA PARAMETER BLOCK
5013      ;
5014      001
5015      ; *****
5016      ; .IFF
5017      ; .NLIST
5018      ; .REPT
5019      ; .LIST
5020      ; *****
5021      ; .NLIST
5022      ; .ENDR

```

```

5023
5024      000
5025
5026 016120 000
5027 016121 000
5028 016122 000
5029 016123 000
5030 016124 000000
5031 016126 001206
5032
5033 016130 000
5034
5035 016131 000
5036
5037 016132 000000
5038 016134 016140
5039
5040
5041
5042
5043
5044 016136 000000
5045
5046
5047
5048
5049
5050 016140 000000
5051 016142 000000
5052 016144 000000
5053 016146 000000
5054 016150 000000
5055 016152 000000
5056 016154 000000
5057 016156 000000
5058 016160 000000
5059 016162 000000
5060 016164 000000
5061 016166 000000
5062 016170 000000
5063 016172 000000

```

```

.LIST
.ENDC
DPB:  .BYTE 0
      .BYTE 0
      .BYTE 0
      .BYTE 0
      .WORD 0
      .WORD INREG
      .BYTE 0
      .BYTE 0
      .WORD 0
      .WORD REG
      .WORD 0
REG:  .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0
      .WORD 0

```

```

;DRIVE NUMBER
;OFFSET VALUE OR FMT22, ECI, AND HCI
;COMMAND
;PSEL AND A17 AND A16
;WORD COUNT (MUST BE NEG)
;BUFFER ADDRESS OR
;REGISTER TABLE POINTER
;SECTOR ADDRESS OR
;FIRST REGISTER INDEX
;TRACK ADDRESS OR
;LAST REGISTER INDEX
;CYLINDER ADDRESS
;ERROR TABLE POINTER
;POINTS TO THE FIRST OF TWENTY
;LOCATIONS WHERE THE DRIVER IS
;TO STORE THE RH11/RPO4 REGISTERS
;ON AN ERROR. IF ZERO, REGISTERS
;ARE NOT SAVED.
;STATUS/ERROR INDICATOR
;BIT15 = 1 => ERROR OCCURRED
;BIT07 = 1 => DONE
;BIT 14 - BIT10 AND BIT06 - BIT03
;INDICATE TYPE OF ERROR
;STORE RPO4 REGISTERS HERE
;RPWC
;RPBA
;RPDA
;RPCS2
;RPDS1
;RPER1
;RPAS
;RPLA
;RPDB
;RPMR
;RPDT
;RPSN
;RPOF

```



CZRJCBO RPO4/5/6 HEAD ALNMT  
CZRJCB.P11 04-NOV-77 12:49

MACY11 30(1046) 04-NOV-77 17:31 PAGE 95  
DATA PARAMETER BLOCK

SEQ 0095

5064 016174 000000  
5065 016176 000000

.WORD 0  
.WORD 0

:RPCA  
:RPCC

```

5066 016200 000000 .WORD 0 ;RPER2
5067 016202 000000 .WORD 0 ;RPER3
5068 016204 000000 .WORD 0 ;RPEC1
5069 016206 000000 .WORD 0 ;RPEC2
5070
5071 000001 CODE=1 ;DPB INDEX EQUATES
5072 000002 COMND=2
5073 000006 BUF=6
5074 000010 SEC=10
5075 000011 TRK=11
5076 000012 CYL=12
5077 000016 STATUS=16
5078
5079 001 .IF B
5080 ;*****
5081 .IFF
5082 .NLIST
5083 .REPT
5084 .LIST
5085 ;*****
5086 .NLIST
5087 .ENDR
5088 .LIST
5089 .ENDC
5090 000
5091 .SBTTL HEAD CODE TABLE
5092
5093 001 .IF B
5094 ;*****
5095 .IFF
5096 .NLIST
5097 .REPT
5098 .LIST
5099 ;*****
5100 .NLIST
5101 .ENDR
5102 .LIST
5103 .ENDC
5104 000
5105 016210 000000 HEAD1: .WORD 0 ;HEAD ADDRESSES FOR CYLINDERS 245 & 496
5106 016212 000001 .WORD 1
5107 016214 000002 .WORD 2
5108 016216 000003 .WORD 3
5109 016220 000004 .WORD 4
5110 016222 000005 .WORD 5
5111 016224 000006 .WORD 6
5112 016226 000007 .WORD 7
5113 016230 000010 .WORD 10
5114 016232 000011 .WORD 11
5115 016234 000012 .WORD 12
5116 016236 000013 .WORD 13
5117 016240 000014 .WORD 14
5118 016242 000015 .WORD 15
5119 016244 000016 .WORD 16
5120 016246 000017 .WORD 17
5121 016250 000020 .WORD 20

```



```

S122 016252 000021 .WORD 21
S123 016254 000022 .WORD 22
S124 016256 100000 .WORD 100000 ;TABLE TERMINATOR
S125
S126 016260 000000 HEAD6: .WORD 0 ;HEAD ADDRESSES FOR CYLS 800 & 008
S127 016262 000001 .WORD 1
S128 016264 000012 .WORD 10.
S129 016266 000021 .WORD 17.
S130 016270 000022 .WORD 18.
S131 016272 100000 .WORD 100000
S132 016274 000000 .WORD 0
S133 016276 000001 .WORD 1
S134 016300 000012 .WORD 10.
S135 016302 000021 .WORD 17.
S136 016304 000022 .WORD 18.
S137 016306 100000 .WORD 100000 ;TERMINATOR
S138
S139 016310 000000 HEAD45: .WORD 0 ;HEAD ADDRESSES FOR CYLINDERS 400 & 4
S140 016312 000022 .WORD 18.
S141 016314 100000 .WORD 100000
S142 016316 000000 .WORD 0
S143 016320 000022 .WORD 18.
S144 016322 100000 .WORD 100000 ;TERMINATOR
S145
S146 001 .IF B
S147 ;*****
S148 .IFF
S149 .NLIST
S150 .REPT
S151 .LIST
S152 ;*****
S153 .NLIST
S154 .ENDR
S155 .LIST
S156 .ENDC
S157 000
S158 .SBTTL OFFSET CODE TABLES
S159 001
S160 .IF B
S161 ;*****
S162 .IFF
S163 .NLIST
S164 .REPT
S165 .LIST
S166 ;*****
S167 .NLIST
S168 .ENDR
S169 .LIST
S170 .ENDC
S171 000
S172 016324 OFTBL6: ;OFFSET TABLE FOR CYLINDER 496
S173 016324 030 .TE 30
S174 016325 027 .BYTE 27
S175 016326 026 .BYTE 26
S176 016327 025 .BYTE 25
S177 016330 024 .BYTE 24

```

5178	016331	023	.BYTE	23
5179	016332	022	.BYTE	22
5180	016333	021	.BYTE	21
5181	016334	020	.BYTE	20
5182	016335	017	.BYTE	17
5183	016336	016	.BYTE	16
5184	016337	015	.BYTE	15
5185	016340	014	.BYTE	14
5186	016341	013	.BYTE	13
5187	016342	012	.BYTE	12
5188	016343	011	.BYTE	11
5189	016344	010	.BYTE	10
5190	016345	007	.BYTE	7
5191	016346	006	.BYTE	6
5192	016347	005	.BYTE	5
5193	016350	004	.BYTE	4
5194	016351	003	.BYTE	3
5195	016352	002	.BYTE	2
5196	016353	001	.BYTE	1
5197	016354	000	.BYTE	0
5198	016355	201	.BYTE	201
5199	016356	202	.BYTE	202
5200	016357	203	.BYTE	203
5201	016360	204	.BYTE	204
5202	016361	205	.BYTE	205
5203	016362	206	.BYTE	206
5204	016363	207	.BYTE	207
5205	016364	210	.BYTE	210
5206	016365	211	.BYTE	211
5207	016366	212	.BYTE	212
5208	016367	213	.BYTE	213
5209	016370	214	.BYTE	214
5210	016371	215	.BYTE	215
5211	016372	216	.BYTE	216
5212	016373	217	.BYTE	217
5213	016374	220	.BYTE	220
5214	016375	221	.BYTE	221
5215	016376	222	.BYTE	222
5216	016377	223	.BYTE	223
5217	016400	224	.BYTE	224
5218	016401	225	.BYTE	225
5219	016402	226	.BYTE	226
5220	016403	227	.BYTE	227
5221	016404	230	.BYTE	230
5222				
5223	016405			
5224	016405	060	.BYTE	60
5225	016406	057	.BYTE	57
5226	016407	056	.BYTE	56
5227	016410	055	.BYTE	55
5228	016411	054	.BYTE	54
5229	016412	053	.BYTE	53
5230	016413	052	.BYTE	52
5231	016414	051	.BYTE	51
5232	016415	050	.BYTE	50
5233	016416	047	.BYTE	47

OFTBL4:

;OFFSET TABLE FOR CYLINDER 245



5234	016417	046	.BYTE	46
5235	016420	045	.BYTE	45
5236	016421	044	.BYTE	44
5237	016422	043	.BYTE	43
5238	016423	042	.BYTE	42
5239	016424	041	.BYTE	41
5240	016425	040	.BYTE	40
5241	016426	037	.BYTE	37
5242	016427	036	.BYTE	36
5243	016430	035	.BYTE	35
5244	016431	034	.BYTE	34
5245	016432	033	.BYTE	33
5246	016433	032	.BYTE	32
5247	016434	031	.BYTE	31
5248	016435	030	.BYTE	30
5249	016436	027	.BYTE	27
5250	016437	026	.BYTE	26
5251	016440	025	.BYTE	25
5252	016441	024	.BYTE	24
5253	016442	023	.BYTE	23
5254	016443	022	.BYTE	22
5255	016444	021	.BYTE	21
5256	016445	020	.BYTE	20
5257	016446	017	.BYTE	17
5258	016447	016	.BYTE	16
5259	016450	015	.BYTE	15
5260	016451	014	.BYTE	14
5261	016452	013	.BYTE	13
5262	016453	012	.BYTE	12
5263	016454	011	.BYTE	11
5264	016455	010	.BYTE	10
5265	016456	007	.BYTE	7
5266	016457	006	.BYTE	6
5267	016460	005	.BYTE	5
5268	016461	004	.BYTE	4
5269	016462	003	.BYTE	3
5270	016463	002	.BYTE	2
5271	016464	001	.BYTE	1
5272	016465	000	.BYTE	0
5273	016466	201	.BYTE	201
5274	016467	202	.BYTE	202
5275	016470	203	.BYTE	203
5276	016471	204	.BYTE	204
5277	016472	205	.BYTE	205
5278	016473	206	.BYTE	206
5279	016474	207	.BYTE	207
5280	016475	210	.BYTE	210
5281	016476	211	.BYTE	211
5282	016477	212	.BYTE	212
5283	016500	213	.BYTE	213
5284	016501	214	.BYTE	214
5285	016502	215	.BYTE	215
5286	016503	216	.BYTE	216
5287	016504	217	.BYTE	217
5288	016505	220	.BYTE	220
5289	016506	221	.BYTE	221

5290	016507	222	.BYTE	222
5291	016510	223	.BYTE	223
5292	016511	224	.BYTE	224
5293	016512	225	.BYTE	225
5294	016513	226	.BYTE	226
5295	016514	227	.BYTE	227
5296	016515	230	.BYTE	230
5297	016516	231	.BYTE	231
5298	016517	232	.BYTE	232
5299	016520	233	.BYTE	233
5300	016521	234	.BYTE	234
5301	016522	235	.BYTE	235
5302	016523	236	.BYTE	236
5303	016524	237	.BYTE	237
5304	016525	240	.BYTE	240
5305	016526	241	.BYTE	241
5306	016527	242	.BYTE	242
5307	016530	243	.BYTE	243
5308	016531	244	.BYTE	244
5309	016532	245	.BYTE	245
5310	016533	246	.BYTE	246
5311	016534	247	.BYTE	247
5312	016535	250	.BYTE	250
5313	016536	251	.BYTE	251
5314	016537	252	.BYTE	252
5315	016540	253	.BYTE	253
5316	016541	254	.BYTE	254
5317	016542	255	.BYTE	255
5318	016543	256	.BYTE	256
5319	016544	257	.BYTE	257
5320	016545	260	.BYTE	260

001

```

.IF B
:*****
.IFF
.NLIST
.REPT
.LIST
:*****
.NLIST
.ENDR
.LIST
.ENDC

```

000

.SBTTL MESSAGES

001

```

.IF B
:*****
.IFF
.NLIST
.REPT
.LIST
:*****
.NLIST
.ENDR

```

5290  
5291  
5292  
5293  
5294  
5295  
5296  
5297  
5298  
5299  
5300  
5301  
5302  
5303  
5304  
5305  
5306  
5307  
5308  
5309  
5310  
5311  
5312  
5313  
5314  
5315  
5316  
5317  
5318  
5319  
5320  
5321  
5322  
5323  
5324  
5325  
5326  
5327  
5328  
5329  
5330  
5331  
5332  
5333  
5334  
5335  
5336  
5337  
5338  
5339  
5340  
5341  
5342  
5343  
5344  
5345  
5346  
5347  
5348  
5349  
5350  
5351  
5352  
5353  
5354  
5355  
5356  
5357  
5358  
5359  
5360  
5361  
5362  
5363  
5364  
5365  
5366  
5367  
5368  
5369  
5370  
5371  
5372  
5373  
5374  
5375  
5376  
5377  
5378  
5379  
5380  
5381  
5382  
5383  
5384  
5385  
5386  
5387  
5388  
5389  
5390  
5391  
5392  
5393  
5394  
5395  
5396  
5397  
5398  
5399  
5400  
5401  
5402  
5403  
5404  
5405  
5406  
5407  
5408  
5409  
5410  
5411  
5412  
5413  
5414  
5415  
5416  
5417  
5418  
5419  
5420  
5421  
5422  
5423  
5424  
5425  
5426  
5427  
5428  
5429  
5430  
5431  
5432  
5433  
5434  
5435  
5436  
5437  
5438  
5439  
5440  
5441  
5442  
5443  
5444  
5445  
5446  
5447  
5448  
5449  
5450  
5451  
5452  
5453  
5454  
5455  
5456  
5457  
5458  
5459  
5460  
5461  
5462  
5463  
5464  
5465  
5466  
5467  
5468  
5469  
5470  
5471  
5472  
5473  
5474  
5475  
5476  
5477  
5478  
5479  
5480  
5481  
5482  
5483  
5484  
5485  
5486  
5487  
5488  
5489  
5490  
5491  
5492  
5493  
5494  
5495  
5496  
5497  
5498  
5499  
5500  
5501  
5502  
5503  
5504  
5505  
5506  
5507  
5508  
5509  
5510  
5511  
5512  
5513  
5514  
5515  
5516  
5517  
5518  
5519  
5520  
5521  
5522  
5523  
5524  
5525  
5526  
5527  
5528  
5529  
5530  
5531  
5532  
5533  
5534  
5535  
5536  
5537  
5538  
5539  
5540  
5541  
5542  
5543  
5544  
5545  
5546  
5547  
5548  
5549  
5550  
5551  
5552  
5553  
5554  
5555  
5556  
5557  
5558  
5559  
5560  
5561  
5562  
5563  
5564  
5565  
5566  
5567  
5568  
5569  
5570  
5571  
5572  
5573  
5574  
5575  
5576  
5577  
5578  
5579  
5580  
5581  
5582  
5583  
5584  
5585  
5586  
5587  
5588  
5589  
5590  
5591  
5592  
5593  
5594  
5595  
5596  
5597  
5598  
5599  
5600  
5601  
5602  
5603  
5604  
5605  
5606  
5607  
5608  
5609  
5610  
5611  
5612  
5613  
5614  
5615  
5616  
5617  
5618  
5619  
5620  
5621  
5622  
5623  
5624  
5625  
5626  
5627  
5628  
5629  
5630  
5631  
5632  
5633  
5634  
5635  
5636  
5637  
5638  
5639  
5640  
5641  
5642  
5643  
5644  
5645  
5646  
5647  
5648  
5649  
5650  
5651  
5652  
5653  
5654  
5655  
5656  
5657  
5658  
5659  
5660  
5661  
5662  
5663  
5664  
5665  
5666  
5667  
5668  
5669  
5670  
5671  
5672  
5673  
5674  
5675  
5676  
5677  
5678  
5679  
5680  
5681  
5682  
5683  
5684  
5685  
5686  
5687  
5688  
5689  
5690  
5691  
5692  
5693  
5694  
5695  
5696  
5697  
5698  
5699  
5700  
5701  
5702  
5703  
5704  
5705  
5706  
5707  
5708  
5709  
5710  
5711  
5712  
5713  
5714  
5715  
5716  
5717  
5718  
5719  
5720  
5721  
5722  
5723  
5724  
5725  
5726  
5727  
5728  
5729  
5730  
5731  
5732  
5733  
5734  
5735  
5736  
5737  
5738  
5739  
5740  
5741  
5742  
5743  
5744  
5745  
5746  
5747  
5748  
5749  
5750  
5751  
5752  
5753  
5754  
5755  
5756  
5757  
5758  
5759  
5760  
5761  
5762  
5763  
5764  
5765  
5766  
5767  
5768  
5769  
5770  
5771  
5772  
5773  
5774  
5775  
5776  
5777  
5778  
5779  
5780  
5781  
5782  
5783  
5784  
5785  
5786  
5787  
5788  
5789  
5790  
5791  
5792  
5793  
5794  
5795  
5796  
5797  
5798  
5799  
5800  
5801  
5802  
5803  
5804  
5805  
5806  
5807  
5808  
5809  
5810  
5811  
5812  
5813  
5814  
5815  
5816  
5817  
5818  
5819  
5820  
5821  
5822  
5823  
5824  
5825  
5826  
5827  
5828  
5829  
5830  
5831  
5832  
5833  
5834  
5835  
5836  
5837  
5838  
5839  
5840  
5841  
5842  
5843  
5844  
5845  
5846  
5847  
5848  
5849  
5850  
5851  
5852  
5853  
5854  
5855  
5856  
5857  
5858  
5859  
5860  
5861  
5862  
5863  
5864  
5865  
5866  
5867  
5868  
5869  
5870  
5871  
5872  
5873  
5874  
5875  
5876  
5877  
5878  
5879  
5880  
5881  
5882  
5883  
5884  
5885  
5886  
5887  
5888  
5889  
5890  
5891  
5892  
5893  
5894  
5895  
5896  
5897  
5898  
5899  
5900  
5901  
5902  
5903  
5904  
5905  
5906  
5907  
5908  
5909  
5910  
5911  
5912  
5913  
5914  
5915  
5916  
5917  
5918  
5919  
5920  
5921  
5922  
5923  
5924  
5925  
5926  
5927  
5928  
5929  
5930  
5931  
5932  
5933  
5934  
5935  
5936  
5937  
5938  
5939  
5940  
5941  
5942  
5943  
5944  
5945  
5946  
5947  
5948  
5949  
5950  
5951  
5952  
5953  
5954  
5955  
5956  
5957  
5958  
5959  
5960  
5961  
5962  
5963  
5964  
5965  
5966  
5967  
5968  
5969  
5970  
5971  
5972  
5973  
5974  
5975  
5976  
5977  
5978  
5979  
5980  
5981  
5982  
5983  
5984  
5985  
5986  
5987  
5988  
5989  
5990  
5991  
5992  
5993  
5994  
5995  
5996  
5997  
5998  
5999  
6000



5346				
5347		000		
5348				
5349	016546	005015	055132	041455
5350	016554	051132	041512	041055
5351	016562	005015		
5352	016564	050122	032060	032457
5353	016572	033057	044040	040505
5354	016600	020104	046101	043511
5355	016606	046516	047105	020124
5356	016614	042526	044522	044506
5357	016622	040503	044524	047117
5358	016630	050040	047522	051107
5359	016636	046501	005015	000012
5360				
5361	016644	047504	047040	052117
5362	016652	051440	046105	041505
5363	016660	020124	051104	053111
5364	016666	020105	047125	044524
5365	016674	020114	042510	042101
5366	016702	040440	044514	047107
5367	016710	042515	052116	052040
5368	016716	051505	020124	047502
5369	016724	020130	040510	020123
5370	016732	042502	047105	041440
5371	016740	047117	042516	052103
5372	016746	042105	005015	
5373	016752	047040	052117	035105
5374	016760	020040	044527	044124
5375	016766	052040	042510	040440
5376	016774	044514	047107	042515
5377	017002	052116	050040	041501
5378	017010	020113	047515	047125
5379	017016	042524	026104	041440
5380	017024	047117	052123	047101
5381	017032	020124	047111	042504
5382	017040	020130	051105	047522
5383	017046	051522	046440	054501
5384	017054	047440	041503	051125
5385	017062	006456	012	
5386	017065	040	044124	051505
5387	017072	020105	047111	042504
5388	017100	020130	051105	047522
5389	017106	051522	040440	042522
5390	017114	041440	052501	042523
5391	017122	020104	054502	052040
5392	017130	042510	040440	044514
5393	017136	047107	042515	052116
5394	017144	050040	041501	020113
5395	017152	047101	020104	040515
5396	017160	006531	012	
5397	017163	040	047516	020124
5398	017170	041517	052503	020122
5399	017176	044527	044124	051440
5400	017204	046517	020105	046101
5401	017212	043511	046516	047105

.LIST  
.ENDC

TITLE: .ASCII <CR><LF>/ZZ-CZRJC-B/<CR><LF>

.ASCIZ RPO4/5/6 HEAD ALIGNMENT VERIFICATION PROGRAM<CR><LF><LF>

DDU: .ASCII /DO NOT SELECT DRIVE UNTIL HEAD ALIGNMENT TEST BOX HAS BEEN CONNECTED/<C

.ASCII / NOTE: WITH THE ALIGNMENT PACK MOUNTED, CONSTANT INDEX ERRORS MAY OCCU

.ASCII / THESE INDEX ERRORS ARE CAUSED BY THE ALIGNMENT PACK AND MAY/<CR><LF>

.ASCII / NOT OCCUR WITH SOME ALIGNMENT PACKS. TO INHIBIT THE ALIGNMENT PACK/<C

5402	017220	020124	040520	045503
5403	017226	027123	020040	047524
5404	017234	044440	044116	041111
5405	017242	052111	052040	042510
5406	017250	040440	044514	047107
5407	017256	042515	052116	050040
5408	017264	041501	006513	012
5409	017271	040	042522	040514
5410	017276	042524	020104	047111
5411	017304	042504	020130	051105
5412	017312	047522	026122	043440
5413	017320	047522	047125	020104
5414	017326	044124	020105	047506
5415	017334	046114	053517	047111
5416	017342	020107	044520	051516
5417	017350	006472	012	
5418	017353	040	020040	020040
5419	017360	051040	030120	035064
5420	017366	050040	047111	040440
5421	017374	040461	032460	034060
5422	017402	024040	052506	041516
5423	017410	044524	047117	023440
5424	017416	052053	047120	042504
5425	017424	042530	023522	006451
5426	017432	012		
5427	017433	040	020040	020040
5428	017440	051040	030120	027465
5429	017446	035066	020040	044520
5430	017454	020116	030104	051064
5431	017462	034461	024040	052506
5432	017470	041516	044524	047117
5433	017476	023440	044455	042116
5434	017504	054105	024447	005015
5435	017512	000012		
5436				
5437	017514	005015	051104	053111
5438	017522	020105	047516	020124
5439	017530	053101	044501	040514
5440	017536	046102	006505	005012
5441	017544	000		
5442				
5443	017545	015	042412	052116
5444	017552	051105	042040	044522
5445	017560	042526	047040	046525
5446	017566	042502	035122	000040
5447				
5448	017574	005015	051104	053111
5449	017602	020105	047516	020124
5450	017610	051127	052111	020105
5451	017616	051120	052117	041505
5452	017624	042524	006504	005012
5453	017632	000		
5454				
5455	017633	040	044040	040505
5456	017640	020104	052517	020124
5457	017646	043117	051040	047101

.ASCII / RELATED INDEX ERROR, GROUND THE FOLLOWING PINS://<CR><LF>

.ASCII / RP04: PIN A1A0508 (FUNCTION '+TPINDEXER')//<CR><LF>

.ASCIZ @ RP05/6: PIN D04R19 (FUNCTION '-INDEX')@<CR><LF><LF>

NODRV: .ASCIZ <CR><LF>/DRIVE NOT AVAILABLE//<CR><LF><LF>

ENTERD: .ASCIZ <CR><LF>/ENTER DRIVE NUMBER: /

WLOCK: .ASCIZ <CR><LF>/DRIVE NOT WRITE PROTECTED//<CR><LF><LF>

TOLRG: .ASCIZ / HEAD OUT OF RANGE/



5458	017654	042507	000		
5459					
5460	017657	015	020012	020040	HEADER: .ASCII <CR><LF>/
5461	017664	020040	020040	020040	TRK CENTER/<CR><LF>
5462	017672	020040	020040	020040	
5463	017700	045522	041440	047105	
5464	017706	042524	006522	012	
5465	017713	040	020040	042510	.ASCIZ / HEAD CYL (IN U INCHES)/<CR><LF><LF>
5466	017720	042101	020040	054503	
5467	017726	020114	044450	020116	
5468	017734	020125	047111	044103	
5469	017742	051505	006451	005012	
5470	017750	000			
5471					
5472	017751	101	046050	043511	MODE: .ASCIZ /A(LIGN), V(ERIFY), OR E(XERCISE) ? /
5473	017756	024516	020054	024126	
5474	017764	051105	043111	024531	
5475	017772	020054	051117	042440	
5476	020000	054050	051105	044503	
5477	020006	042523	020051	020077	
5478	020014	000			
5479					
5480	020015	126	051105	043111	VERIFY: .ASCIZ /VERIFY/<CR><LF><LF>
5481	020022	006531	005012	000	
5482					
5483	020027	101	044514	047107	ALIGN: .ASCIZ /ALIGN/<CR><LF><LF>
5484	020034	005015	000012		
5485					
5486	020040	054105	051105	044503	EXER: .ASCIZ /EXERCISE/<CR><LF><LF>
5487	020046	042523	005015	000012	
5488					
5489	020054	042510	042101	030040	HDZERO: .ASCIZ /HEAD 0 SELECTED BY DEFAULT/<CR><LF><LF>
5490	020062	051440	046105	041505	
5491	020070	042524	020104	054502	
5492	020076	042040	043105	052501	
5493	020104	052114	005015	000012	
5494					
5495	020112	054503	044514	042116	CYL245: .ASCIZ /CYLINDER 245 SELECTED BY DEFAULT/<CR><LF><LF>
5496	020120	051105	031040	032464	
5497	020126	051440	046105	041505	
5498	020134	042524	020104	054502	
5499	020142	042040	043105	052501	
5500	020150	052114	005015	000012	
5501					
5502	020156	054503	044514	042116	CYL496: .ASCIZ /CYLINDER 496 SELECTED BY DEFAULT/<CR><LF><LF>
5503	020164	051105	032040	033071	
5504	020172	051440	046105	041505	
5505	020200	042524	020104	054502	
5506	020206	042040	043105	052501	
5507	020214	052114	005015	000012	
5508					
5509	020222	044504	045523	044440	ALN245: .ASCIZ /DISK IS POSITIONED AT CYLINDER 245/<CR><LF><LF>
5510	020230	020123	047520	044523	
5511	020236	044524	047117	042105	
5512	020244	040440	020124	054503	
5513	020252	044514	042116	051105	

5514	020260	031040	032464	005015	
5515	020266	000012			
5516					
5517	020270	044504	045523	044440	ALN496: .ASCIZ /DISK IS POSITIONED AT CYLINDER 496/<CR><LF><LF>
5518	020276	020123	047520	044523	
5519	020304	044524	047117	042105	
5520	020312	040440	020124	054503	
5521	020320	044514	042116	051105	
5522	020326	032040	033071	005015	
5523	020334	000012			
5524					
5525	020336	005015	047105	042524	ENTHD: .ASCIZ <CR><LF>/ENTER HEAD: /
5526	020344	020122	042510	042101	
5527	020352	020072	000		
5528					
5529	020355	015	042412	052116	ENTCYL: .ASCIZ <CR><LF>/ENTER CYLINDER ADDRESS: /
5530	020362	051105	041440	046131	
5531	020370	047111	042504	020122	
5532	020376	042101	051104	051505	
5533	020404	035123	000040		
5534					
5535	020410	047111	040526	044514	BADTRK: .ASCIZ /INVALID HEAD ADDRESS/<CR><LF>
5536	020416	020104	042510	042101	
5537	020424	040440	042104	042522	
5538	020432	051523	005015	000	
5539					
5540	020437	111	053116	046101	BADCYL: .ASCIZ /INVALID CYLINDER ADDRESS/<CR><LF>
5541	020444	042111	041440	046131	
5542	020452	047111	042504	020122	
5543	020460	042101	051104	051505	
5544	020466	006523	000012		
5545					
5546	020472	005015	042012	047117	ENDMSG: .ASCIZ <CR><LF><LF>/DONE/<CR><LF><LF>
5547	020500	006505	005012	000	
5548					
5549					
5550		020506			.EVEN
5551					
5552	020506	044122	030461	044440	EM1: .ASCIZ /RH11 INTERRUPT OCCURRED (RPAS=0)/
5553	020514	052116	051105	052522	
5554	020522	052120	047440	041503	
5555	020530	051125	042522	020104	
5556	020536	051050	040520	036523	
5557	020544	024460	000		
5558					
5559	020547	125	042516	050130	EM2: .ASCIZ /UNEXPECTED ATTENTION OCCURRED/
5560	020554	041505	042524	020104	
5561	020562	052101	042524	052116	
5562	020570	047511	020116	041517	
5563	020576	052503	051122	042105	
5564	020604	000			
5565					
5566	020605	115	051501	041123	EM3: .ASCIZ /MASSBUS PARITY ERROR (MCPE=1)/
5567	020612	051525	050040	051101	
5568	020620	052111	020131	051105	
5569	020626	047522	020122	046450	



5570	020634	050103	036505	024461		
5571	020642	000				
5572						
5573	020643	115	051501	041123	EM4:	.ASCIZ /MASSBUS PARITY ERROR (PAR=1)/
5574	020650	051525	050040	051101		
5575	020656	052111	020131	051105		
5576	020664	047522	020122	050050		
5577	020672	051101	030475	000051		
5578						
5579	020700	042101	051104	051505	EM5:	.ASCIZ /ADDRESS PLUG CHANGE BIT SET/
5580	020706	020123	046120	043525		
5581	020714	04144C	040510	043516		
5582	020722	020105	044502	020124		
5583	020730	042523	000124			
5584						
5585	020734	044122	030461	042040	EM6:	.ASCIZ /RH11 DIDN'T RESPOND TO ADDRESSING/
5586	020742	042111	023516	020124		
5587	020750	042522	050123	047117		
5588	020756	020104	047524	040440		
5589	020764	042104	042522	051523		
5590	020772	047111	000107			
5591						
5592	020776	051104	053111	020105	EM10:	.ASCIZ /DRIVE OR DATA ERROR/
5593	021004	051117	042040	052101		
5594	021012	020101	051105	047522		
5595	021020	000122				
5596						
5597	021022	051104	053111	020105	EM11:	.ASCIZ /DRIVE UNSAFE ERROR/
5598	021030	047125	040523	042506		
5599	021036	042440	051122	051117		
5600	021044	000				
5601						
5602	021045	110	040505	020104	EM12:	.ASCIZ /HEAD OUT OF ALIGNMENT/
5603	021052	052517	020124	043117		
5604	021060	040440	044514	047107		
5605	021066	042515	052116	000		
5606						
5607	021073	110	040505	020104	EM13:	.ASCIZ /HEAD TOO FAR OUT OF ALIGNMENT/
5608	021100	047524	020117	040506		
5609	021106	020122	052517	020124		
5610	021114	043117	040440	044514		
5611	021122	047107	042515	052116		
5612	021130	000				
5613						
5614	021131	123	043117	053524	EM14:	.ASCIZ /SOFTWARE TIMEOUT/
5615	021136	051101	020105	044524		
5616	021144	042515	052517	000124		
5617						
5618	021152	047125	047503	051122	EM15:	.ASCIZ /UNCORRECTABLE MASSBUS PARITY ERROR/
5619	021160	041505	040524	046102		
5620	021166	020105	040515	051523		
5621	021174	052502	020123	040520		
5622	021202	044522	054524	042440		
5623	021210	051122	051117	000		
5624						
5625	021215	104	044522	042526	EM16:	.ASCIZ /DRIVE UNAVAILABLE/





```

5682
5683 021617 104 044522 042526 DH16: .ASCIZ /DRIVE/
5684 021624 000
5685
5686 021626 .EVEN
5687
5688 021626 001170 000000 DT1: .WORD ATTN,0
5689
5690 021632 001166 007536 007540 DT2: .WORD DRIVE,RPERRS,RPERRS+2,RPERRS+4,RPERRS+6,ATTN,0
5691 021640 007542 007544 001170
5692 021646 000000
5693
5694 021650 001166 014662 014664 DT3: .WORD DRIVE,RD.ADR,RD.WRD,0
5695 021656 000000
5696
5697 021660 001166 015102 015100 DT4: .WORD DRIVE,WRT.AD,WRT.WD,RD.WRD,0
5698 021666 014664 000000
5699
5700 021672 001264 000000 DT6: .WORD SRPADR,0
5701
5702 021676 001164 016140 016150 DT10: .WORD DRVSEL,REG,REG+RPCS2,REG+RPDS1,REG+RPER1,REG+RPER2,REG+RPER3,0
5703 021704 016152 016154 016200
5704 021712 016202 000000
5705
5706 021716 001172 016132 001204 DT12: .WORD $HEAD,DPB+CYL,PLUS,0
5707 021724 000000
5708
5709 021726 001172 016132 000000 DT13: .WORD $HEAD,DPB+CYL,0
5710
5711 021734 001164 000000 DT16: .WORD DRVSEL,0
5712
5713
5714
5715 021740 000 DF1: .BYTE 0
5716
5717 021741 000 000 000 DF2: .BYTE 0,0,0,0,0,0
5718 021744 000 000 000
5719
5720 021747 000 000 000 DF3: .BYTE 0,0,0
5721
5722 021752 000 000 000 DF4: .BYTE 0,0,0,0
5723 021755 000
5724
5725 021756 000 000 000 DF10: .BYTE 0,0,0,0,0,0,0
5726 021761 000 000 000
5727 021764 000
5728
5729 021765 001 001 001 DF12: .BYTE 1,1,1
5730
5731 021770 001 001 DF13: .BYTE 1,1
5732
5733
5734
5735
5736
5737

```

```

.SBTTL BUSADR - GET BUS ADDRESS AND VECTOR ADDRESS FOR RH11
;THIS ROUTINE IS USED TO INSURE THE BUS ADDRESS
;OF THE RH11 IS SETUP FOR THE PROPER ADDRESS.
;IT WILL ALSO READ THE ADDRESS FROM THE TTY IF

```

```

5738 ;REQUIRED.
5739 ;NOTE: THIS ROUTINE DESTROYS R0-R4
5740 ;CALL
5741 ;
5742 ; JSR PC,BUSADR
5743 ; RETURN
5744 ;
5745 ;
5746 ;
5747 021772 177700 HIAD: .WORD 177700
5748 021774 000776 HIVEC: .WORD 776
5749 021776 000000 LIMIT: .WORD
5750 022000 005737 001242 BUSADR: TST CHGADR ;INPUT FROM TTY REQUESTED?
5751 022004 001460 BEQ 7$ ;NO--BRANCH
5752 022006 005037 001242 CLR CHGADR ;YES--CLEAR THE REQUEST FLAG
5753 022012 012700 001264 1$: MOV #SRPADR,R0 ;FIRST ADDRESS
5754 022016 104401 022220 TYPE MRPCS1 ;"RPCS1="
5755 022022 012046 MOV (R0)+,-(SP) ;PRESENT RPCS1 ADDRESS
5756 022024 104402 TYPOC TYPE IT ;TYPE IT
5757 022026 104401 022242 TYPE ,LINSF ;2 SPACES
5758 022032 104411 RDLIN ;GET THE ENTRY
5759 022034 012601 MOV (SP)+,R1 ;ADDRESS OF ASCII TEXT
5760 022036 013737 021772 021776 MOV HIAD,LIMIT ;SET UP ADDRESS-MAX
5761 001 .IF NB
5762 MOV ,R2 ;UPPER LIMIT OF INPUT
5763 .ENDC
5764 022044 004537 022246 JSR R5,CK.NUM ;CHECK THE NUMBER
5765 022050 022070 3$ ;CARRIAGE RETURN ONLY ENTERED
5766 022052 022146 7$ ;PERIOD ONLY ENTERED
5767 022054 022012 1$ ;ILLEGAL INPUT
5768 022056 022064 2$ ;TERMINATED WITH A CARRIAGE RETURN
5769 022060 022012 1$ ;TERMINATED WITH A "."
5770 022062 022142 4$ ;TERMINATED WITH A ".'"
5771 022064 010260 177776 2$: MOV R2,-2(R0) ;SAVE NEW RPCS1
5772 022070 104401 022231 3$: TYPE MRHVEC ;"RHVEC="
5773 022074 012700 001266 MOV #SRPVEC,R0 ;GET THE DEFAULT VECTOR ADDRESS
5774 022100 012046 MOV (R0)+,-(SP) ;PRESENT RH11 VECTOR ADDRESS ON THE STACK
5775 022102 104402 TYPOC TYPE IT ;TYPE IT
5776 022104 104401 022242 TYPE ,LINSF ;2 SPACES
5777 022110 104411 RDLIN ;READ THE ENTRY
5778 022112 012601 MOV (SP)+,R1 ;ASCII TEXT ADDRESS
5779 022114 013737 021774 021776 MOV HIVEC,LIMIT ;SET UP THE VECTOR-MAX
5780 001 .IF NB
5781 MOV ,R2 ;UPPER LIMIT OF INPUT
5782 .ENDC
5783 022122 004537 022246 JSR R5,CK.NUM ;CHECK THE NUMBER
5784 022126 022146 7$ ;CARRIAGE RETURN ONLY ENTERED
5785 022130 022146 7$ ;PERIOD ONLY ENTERED
5786 022132 022070 3$ ;ILLEGAL INPUT
5787 022134 022142 4$ ;TERMINATED WITH A CARRIAGE RETURN
5788 022136 022070 3$ ;TERMINATED WITH A "."
5789 022140 022142 4$ ;TERMINATED WITH A ".'"
5790 022142 010260 177776 4$: MOV R2,-2(R0) ;SAVE INPUT
5791 022146 013701 000004 7$: MOV ERRVEC,R1 ;SAVE THE ERROR VECTOR
5792 022152 012737 022206 000004 MOV #8$,ERRVEC ;SETUP FOR TRAP
5793 022160 005777 157100 TST JSRPADR ;CHECK FOR RH11

```



```

5794 022164 010137 000004      MOV      R1,ERRVEC      ;RESTORE ERROR VECTOR
5795 022170 012700 001264      MOV      #SRPADR,R0     ;FIRST ADDRESS OF NEW PARAMETERS
5796 022174 012701 007704      MOV      #RPADR,R1     ;FIRST ADDRESS OF WHERE TO PUT THEM
5797 022200 012021          MOV      (R0)+,(R1)+   ;BUS ADDRESS
5798 022202 012021          MOV      (R0)+,(R1)+   ;VECTOR ADDRESS
5799 022204 000207          RTS      PC            ;RETURN
5800 022206 010137 000004      8$:     MOV      R1,ERRVEC      ;RESTORE ERROR VECTOR
5801 022212 022626          CMP      (SP)+,(SP)+   ;CLEAN OFF THE STACK
5802 022214 104006          ERROR   6             ;REPORT THE ERROR
5803 022216 000675          BR      1$            ;ASK FOR BUS ADDRESS
5804
5805 022220 050122 051503 020061 MRPCS1: .ASCIZ  @RPCS1 = @
5806 022226 020075          000
5807 022231          122 053110 041505 MRHVEC: .ASCIZ  @RHVEC = @
5808 022236 036440 000040
5809 022242 020040          000
5810
5811          022246
5812
5813          .SBTTL CK.NUM - CHECK NUMBER (OCTAL)
5814          ;THIS ROUTINE CHECKS AN ASCIZ STRING FOR LEGAL CHARACTERS
5815          ;AND FORMS AN OCTAL NUMBER IN R2
5816          ;CALL:
5817          ;
5818          ;
5819          ;
5820          ;
5821          ;
5822          ;
5823          ;
5824          ;
5825          ;
5826          ;
5827 022246 010446      CK.NUM: MOV      R4,-(SP)     ;SAVE R4
5828 022250 010346      MOV      R3,-(SP)     ;SAVE R3
5829 022252 010246      MOV      R2,-(SP)     ;SAVE R2
5830 022254 005004      CLR      R4           ;RETURN POINTER
5831 022256 005003      CLR      R3           ;START NUMBER AT ZERO
5832 022260 005002      CLR      R2           ;STORE RESULT
5833 022262 004537 022464      JSR      R5,CK.CHR    ;CHECK ONE CHARACTER
5834 022266 022366      6$          ;ILLEGAL CHARACTER
5835 022270 022372      8$          ;CARRIAGE RETURN
5836 022272 022366      6$          ;
5837 022274 022370      7$          ;
5838 022276 022302      1$          ;DIGIT 0-7
5839 022300 022366      6$          ;DIGIT 8-9
5840 022302 062705 000004      1$:     ADD      #4,R5       ;INCREMENT RETURN PAST "CR" AND "PERIOD" ONLY RETURNS
5841 022306 006303      2$:     ASL      R3         ;FOR THE OCTAL NUMBER IN R3
5842 022310 103426          BCS      6$          ;DON'T LET IT GET TO BIG
5843 022312 006303      ASL      R3
5844 022314 103424          BCS      6$
5845 022316 006303      ASL      R3
5846 022320 103422          BCS      6$
5847 022322 060203      ADD      R2,R3
5848 022324 004537 022464      JSR      R5,CK.CHR    ;CHECK ONE CHARACTER
5849 022330 022372      8$          ;ILLEGAL CHARACTER

```

5850	022332	022354		5\$		:CARRIAGE RETURN
5851	022334	022352		4\$		:"
5852	022336	022344		3\$		:"
5853	022340	022306		2\$		:DIGIT 0-7
5854	022342	022372		8\$		:DIGIT 8-9
5855	022344	105711	3\$:	TSTB	(R1)	:DOES A "CR" FOLLOW THE "PERIOD"
5856	022346	001011		BNE	8\$	:BR IF NOT
5857	022350	005724		TST	(R4)+	:INCREMENT THE RETURN
5858	022352	005724	4\$:	TST	(R4)+	:INCREMENT THE RETURN INDEX
5859	022354	005724	5\$:	TST	(R4)+	:INCREMENT THE RETURN INDEX
5860	022356	023703	021776	CMP	LIMIT,R3	:INPUT VALUE TOO LARGE ?
5861	022362	101003		BHI	8\$	:BR IF IT IS
5862	022364	000401		BR	7\$	:BR IF NOT
5863	022366	005725	6\$:	TST	(R5)+	:INCREMENT THE RETURN ADDRESS
5864	022370	005725	7\$:	TST	(R5)+	:INCREMENT THE RETURN ADDRESS
5865	022372	060405	8\$:	ADD	R4,R5	:SETUP FOR PROPER RETURN
5866	022374	010302		MOV	R3,R2	:LOAD ENTERED VALUE
5867	022376	005726		TST	(SP)+	:CLEAN OFF THE STACK
5868	022400	012603		MOV	(SP)+,R3	:RESTORE R3
5869	022402	012604		MOV	(SP)+,R4	:RESTORE R4
5870	022404	011505		MOV	(R5),R5	:GET RETURN ADDRESS
5871	022406	000205		RTS	R5	:RETURN

```

5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905

```

```

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

```

```

CK.OCT: CMPB    (R1),#'0      : LESS THAN ZERO?
        BLO     1$           : YES -- BRANCH
        CMPB    (R1),#'7      : GREATER THAN SEVEN?
        BHI     1$           : YES -- BRANCH
        MOVB    (R1),R2       : GET THE CHARACTER
        BIC     #1C7,R2       : STRIP AWAY THE ASCII
        TST     (R5)+         : ADJUST FOR RETURN
1$:     RTS      R5           : RETURN

```

```

: THIS ROUTINE IS USED TO CHECK AN ASCII CHARACTER
: AND DETERMINE IF IT IS A DIGIT BETWEEN 0 AND 9.
: CALL
:       MOV      #ADR,R1      : ADDRESS OF ASCII CHARACTER
:       JSR      R5,CK.DEC    : CHECK THE CHARACTER
:       RETURN1  : NOT BETWEEN 0 AND 9
:       RETURN2  : BETWEEN 0 AND 9
:                   : R2 = DIGIT

```

```

CK.DEC: CMPB    (R1),#'0      : LESS THAN ZERO?
        BLO     1$           : YES -- BRANCH
        CMPB    (R1),#'9      : GREATER THAN NINE?
        BHI     1$           : YES -- BRANCH

```



5906	022452	111102	
5907	022454	042702	000060
5908	022460	005725	
5909	022462	000205	
5910			
5911			
5912			
5913			
5914			
5915			
5916			
5917			
5918			
5919			
5920			
5921			
5922			
5923			
5924	022464	105711	
5925	022466	001417	
5926	022470	121127	000054
5927	022474	001413	
5928	022476	121127	000056
5929	022502	001407	
5930	022504	004537	022436
5931	022510	000410	
5932	022512	004537	022410
5933	022516	005725	
5934	022520	005725	
5935	022522	005725	
5936	022524	005725	
5937	022526	005725	
5938	022530	005201	
5939	022532	011505	
5940	022534	000205	
5941			
5942			
5943			
5944			
5945			
5946			
5947			
5948			
5949			
5950			
5951			
5952			
5953			
5954			
5955			
5956			
5957			
5958			
5959			
5960			
5961			
5962			
5963			
5964			
5965			
5966			
5967			
5968			
5969			
5970			
5971			
5972			
5973			
5974			
5975			
5976			
5977			
5978			
5979			
5980			
5981			
5982			
5983			
5984			
5985			
5986			
5987			
5988			
5989			
5990			
5991			
5992			
5993			
5994			
5995			
5996			
5997			
5998			
5999			
6000			
6001			
6002			
6003			
6004			
6005			
6006			
6007			
6008			
6009			
6010			
6011			
6012			
6013			
6014			
6015			
6016			
6017			
6018			
6019			
6020			
6021			
6022			
6023			
6024			
6025			
6026			
6027			
6028			
6029			
6030			
6031			
6032			
6033			
6034			
6035			
6036			
6037			
6038			
6039			
6040			
6041			
6042			
6043			
6044			
6045			
6046			
6047			
6048			
6049			
6050			
6051			
6052			
6053			
6054			
6055			
6056			
6057			
6058			
6059			
6060			
6061			
6062			
6063			
6064			
6065			
6066			
6067			
6068			
6069			
6070			
6071			
6072			
6073			
6074			
6075			
6076			
6077			
6078			
6079			
6080			
6081			
6082			
6083			
6084			
6085			
6086			
6087			
6088			
6089			
6090			
6091			
6092			
6093			
6094			
6095			
6096			
6097			
6098			
6099			
6100			

```

MOV      (R1),R2          ;GET THE CHARACTER
BIC      #'0,R2          ;STRIP AWAY THE ASCII
TST      (R5)+           ;ADJUST FOR RETURN
1$:      RTS              ;RETURN

;THIS ROUTINE WILL CHECK AN ASCII CHARACTER TO
;DETERMINE WHAT IT IS.
CALL

;      MOV      #ADR,R1          ;ADDRESS OF ASCII CHARACTER
;      JSR      R5,CK.CHR        ;CHECK CHARACTER
;      RETURN   ADR1            ;UNKNOWN CHARACTER
;      RETURN   ADR2            ;CARRIAGE RETURN * (R1)=ADR+1
;      RETURN   ADR3            ;COMMA * (R1)=ADR+1
;      RETURN   ADR4            ;PERIOD * (R1)=ADR+1
;      RETURN   ADR5            ;DIGIT BETWEEN 0 AND 7.
;      RETURN   ADR6            ;DIGIT BETWEEN 8 AND 9.
;                                  ;R2 = DIGIT * (R1)=ADR+1

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

        .END

```

ABS	=	000200	BPTVEC=	000014	DH6	=	021406	EM13	=	021073	INREG	=	001206	
ACK	=	000123	BUF	=	000006	DIGB	=	000004	EM14	=	021131	IOTVEC=	000020	
ACL	=	000040	BUSADR	=	022000	DISPLA	=	001142	EM15	=	021152	IR	=	000100
ACTDRV	=	007622	CHECK	=	003032	DISPRE	=	000174	EM16	=	021215	ISR	=	012566
ACTSTR	=	007623	CHGADR	=	001242	DLT	=	100000	EM2	=	020547	IXE	=	004000
ACU	=	100000	CI1	=	011244	DL64	=	000020	EM3	=	020605	LA	=	012430
ALIGN	=	020027	CI3	=	011352	DMD	=	000001	EM4	=	020643	LACNT	=	007634
ALN245	=	020222	CI4	=	011460	DPB	=	016120	EM5	=	020700	LF	=	000012
ALN496	=	020270	CI5	=	012036	DPINT	=	007576	EM6	=	020734	LIMIT	=	021776
AOE	=	001000	CI6	=	012060	DPR	=	000400	ENDMSG	=	020472	LINSP	=	022242
ATA	=	100000	CI7	=	012074	DPRQS	=	007606	ENTCYL	=	020355	LKS	=	001262
ATABIT	=	007672	CI7B	=	012122	DRIVE	=	001166	ENTERD	=	017545	LKV	=	001256
ATTN	=	001170	CI8	=	012174	DRIVER	=	003736	ENTHD	=	020336	LST	=	002000
ATO	=	000001	CKSWR	=	104407	DRQ	=	004000	ERR	=	040000	MAXCYL	=	001212
AT1	=	000002	CK.CHR	=	022464	DRVACT	=	007546	ERRVEC=	=	000004	MAXNEG	=	001216
AT2	=	000004	CK.DEC	=	022436	DRVCLR=	=	000111	EXER	=	020040	MAXOFF	=	001220
AT3	=	000010	CK.NUM	=	022246	DRVINT	=	010134	EXT1	=	000001	MAXPOS	=	001214
AT4	=	000020	CK.OCT	=	022410	DRVQUE	=	015760	EXT10	=	000010	MCLK	=	000002
AT5	=	000040	CLR	=	000040	DRVSEL	=	001164	EXT2	=	000002	MCPE	=	020000
AT6	=	000100	CLRQUE	=	015662	DRVSTA	=	007556	EXT20	=	000020	MCPEMX	=	007702
AT7	=	000200	CODE	=	000001	DRVSTYP	=	007566	EXT4	=	000004	MHS	=	001000
A16	=	000400	COMND	=	000002	DRY	=	000200	EXT40	=	000040	MIDCYL	=	001230
A17	=	001000	CR	=	000015	DSWR	=	177570	FEN	=	000200	MIDTOL	=	001224
BADCYL	=	020437	CRLF	=	000200	DTE	=	010000	FER	=	000020	MINX	=	000004
BADTRK	=	020410	CSF	=	000002	DTSY	=	000200	FMT22	=	010000	MNDLTA	=	007716
BAI	=	000010	CSU	=	000010	DTUW	=	007670	F1	=	000002	MODE	=	017751
BITIND	=	001200	CYL	=	000012	DT00	=	000001	F2	=	000004	MOH	=	020000
BIT0	=	000001	CYLDER	=	002646	DT01	=	000002	F3	=	000010	MOL	=	010000
BIT00	=	000001	CYL245	=	020112	DT02	=	000004	F4	=	000020	MPE	=	000400
BIT01	=	000002	CYL496	=	020156	DT03	=	000010	F5	=	000040	MRD	=	000020
BIT02	=	000004	DCK	=	100000	DT04	=	000020	GETNUM	=	004270	MRHVEC	=	022231
BIT03	=	000010	DCL	=	000100	DT05	=	000040	GETREG=	=	000141	MRPCS1	=	022220
BIT04	=	000020	DCU	=	000001	DT06	=	000100	GETREQ	=	016034	MSE	=	000020
BIT05	=	000040	DDISP	=	177570	DT07	=	000200	GO	=	000001	MSTCK	=	000010
BIT06	=	000100	DDU	=	016644	DT08	=	000400	GRV	=	000010	MWR	=	000040
BIT07	=	000200	DDUALN	=	003530	DT1	=	021626	GTSWR	=	104406	MXDLTA	=	007714
BIT08	=	000400	DE1	=	000040	DT10	=	021676	HCE	=	000200	MXF	=	001000
BIT09	=	001000	DF20	=	000002	DT12	=	021716	HCI	=	002000	MXLACT	=	007712
BIT1	=	000002	DF1	=	021740	DT13	=	021726	HCRC	=	000400	MXWINDW	=	007720
BIT10	=	002000	DF10	=	021756	DT16	=	021734	HDZERO	=	020054	N	=	000261
BIT11	=	004000	DF12	=	021765	DT2	=	021632	HEADER	=	017657	NBA	=	100000
BIT12	=	010000	DF13	=	021770	DT3	=	021650	HEAD1	=	016210	NED	=	010000
BIT13	=	020000	DF2	=	021741	DT4	=	021660	HEAD45	=	016310	NEM	=	004000
BIT14	=	040000	DF3	=	021747	DT6	=	021672	HEAD6	=	016260	NHS	=	002000
BIT15	=	100000	DF4	=	021752	DVA	=	004000	HIAD	=	021772	NODRV	=	017514
BIT2	=	000004	DH1	=	021237	ECH	=	000100	HIVEC	=	021774	OCYL	=	100000
BIT3	=	000010	DH10	=	021421	ECI	=	004000	HT	=	000011	OFFDIR	=	001176
BIT4	=	000020	DH12	=	021507	EMPTYQ	=	015740	IAE	=	002000	OFFSET=	=	000115
BIT5	=	000040	DH13	=	021600	EMTVEC=	=	000030	IE	=	000100	OFREV	=	000200
BIT6	=	000100	DH16	=	021617	EM1	=	020506	ILF	=	000001	OFTBL4	=	016405
BIT7	=	000200	DH2	=	021244	EM10	=	020776	ILR	=	000002	OFTBL6	=	016324
BIT8	=	000400	DH3	=	021321	EM11	=	021022	INCYL	=	001226	OF100	=	000004
BIT9	=	001000	DH4	=	021347	EM12	=	021045	INIT	=	001744	OF200	=	000010



OF25 = 000001  
 OF400 = 000020  
 OF50 = 000002  
 OF800 = 000040  
 ONEHD = 002444  
 OPE = 020000  
 OPI = 020000  
 OPT = 010764  
 OR = 000200  
 OUTCYL = 001232  
 OUTTOL = 001222  
 PAR = 000010  
 PAT = 000020  
 PGE = 002000  
 PGM = 001000  
 PIP = 020000  
 PIRQ = 177772  
 PIRQVE = 000240  
 PKB = 001252  
 PKC = 001254  
 PKCS = 001250  
 PKV = 001244  
 FLU = 020000  
 PLUS = 001204  
 POPQUE = 016056  
 PRESET = 000121  
 PRO = 000000  
 PR1 = 000040  
 PR2 = 000100  
 PR3 = 000140  
 PR4 = 000200  
 PR5 = 000240  
 PR6 = 000300  
 PR7 = 000340  
 PS = 177776  
 PSEL = 002000  
 PSU = 000001  
 PSM = 177776  
 PWRVEC = 000024  
 QCNT = 015370  
 QDRV0 = 015462  
 QDRV1 = 015502  
 QDRV2 = 015522  
 QDRV3 = 015542  
 QDRV4 = 015562  
 QDRV5 = 015602  
 QDRV6 = 015622  
 QDRV7 = 015642  
 QINPT = 015400  
 QOUTPT = 015420  
 QSTART = 015440  
 QSTOP = 015442  
 QTERM = 015662

RANDOM = 003640  
 RAW = 000020  
 RDCHR = 104410  
 RDATA = 000171  
 RDHD = 000173  
 RDLIN = 104411  
 RDY = 000200  
 RD.ADR = 014662  
 RD.RP = 014636  
 RD.RP1 = 014660  
 RD.RP2 = 015000  
 RD.RP3 = 015004  
 RD.RP4 = 015010  
 RD.WRD = 014664  
 RECAL = 000107  
 REG = 016140  
 REL = 000113  
 RESREG = 104413  
 RESVEC = 000010  
 RMR = 000004  
 RNOP = 000101  
 RPADR = 007704  
 RPAS = 000016  
 RPBA = 000004  
 RPCA = 000034  
 RPCC = 000036  
 RPCS1 = 000000  
 RPCS2 = 000010  
 RPDA = 000006  
 RPDB = 000022  
 RPDS1 = 000012  
 RPDY = 000026  
 RPEC1 = 000044  
 RPEC2 = 000046  
 RPERRS = 007536  
 RPER1 = 000014  
 RPER2 = 000040  
 RPER3 = 000042  
 RPINIT = 007722  
 RPLA = 000020  
 RPMR = 000024  
 RPOF = 000032  
 RPSN = 000030  
 RPTMR = 014110  
 RPVEC = 007706  
 RPWC = 000002  
 RPD4 = 010472  
 RTC = 000117  
 R6 = %000006  
 R7 = %000007  
 SAVEFG = 007644  
 SAVREG = 104412  
 SC = 012772

SC1 = 000100  
 SC10 = 001000  
 SC11 = 013610  
 SC12 = 013700  
 SC13 = 013750  
 SC2 = 000200  
 SC20 = 002000  
 SC3 = 013036  
 SC4 = 013042  
 SC5 = 013054  
 SC6 = 013244  
 SC6A = 013354  
 SC7 = 013502  
 SC8 = 013560  
 SEARCH = 000131  
 SEC = 000010  
 SEEK = 000105  
 SEEKFG = 007646  
 SEKCNT = 001240  
 SELDRV = 000145  
 SETFMT = 000143  
 SET.IE = 015316  
 SINCNG = 001202  
 SKI = 040000  
 SRCHWT = 007620  
 SRVCLK = 004256  
 STACK = 001100  
 START = 001450  
 START1 = 001460  
 START2 = 001464  
 STATUS = 000016  
 STKLMT = 177774  
 ST0 = 014202  
 ST01 = 014232  
 ST02 = 014430  
 ST03 = 014500  
 ST05 = 014524  
 ST06 = 014532  
 ST07 = 014570  
 ST08 = 014620  
 ST09 = 014630  
 ST.CLK = 004114  
 ST.LCL = 004232  
 ST.PCL = 004200  
 SVRH11 = 015200  
 SWR = 001140  
 SWREG = 000176  
 SWO = 000001  
 SW00 = 000001  
 SW01 = 000002  
 SW02 = 000004  
 SW03 = 000010  
 SW04 = 000020

SW05 = 000040  
 SW06 = 000100  
 SW07 = 000200  
 SW08 = 000400  
 SW09 = 001000  
 SW1 = 000002  
 SW10 = 002000  
 SW11 = 004000  
 SW12 = 010000  
 SW13 = 020000  
 SW14 = 040000  
 SW15 = 100000  
 SW2 = 000004  
 SW3 = 000010  
 SW4 = 000020  
 SW5 = 000040  
 SW6 = 000100  
 SW7 = 000200  
 SW8 = 000400  
 SW9 = 001000  
 TABLE1 = 001234  
 TABLE2 = 001236  
 TAP = 040000  
 TBITVE = 000014  
 TD = 012632  
 TDF = 000040  
 TIMER = 007650  
 TITLE = 016546  
 TKVEC = 000060  
 TOLER = 001210  
 TOLRG = 017633  
 TOPLN = 001174  
 TPVEC = 000064  
 TRAPVE = 000034  
 TRE = 040000  
 TRK = 000011  
 TRK1 = 004000  
 TRK10 = 040000  
 TRK2 = 010000  
 TRK20 = 100000  
 TRK4 = 020000  
 TRNSWT = 007616  
 TRTVEC = 000014  
 TUF = 000100  
 TYPDS = 104405  
 TYPE = 104401  
 TYPOC = 104402  
 TYPON = 104404  
 TYPOS = 104403  
 ULDFLG = 007624  
 UNLOAD = 000103  
 UNS = 040000  
 UPE = 020000

US1 = 000001  
 US2 = 000002  
 US4 = 000004  
 UWR = 000010  
 VERIFY = 020015  
 VUF = 000002  
 VU30 = 010000  
 VV = 000100  
 WAO = 000002  
 WCE = 040000  
 WCF = 000040  
 WCHKD = 000151  
 WCHKHD = 000153  
 WCU = 000001  
 WLE = 004000  
 WLOCK = 017574  
 WRL = 004000  
 WRTDAT = 000161  
 WRTHD = 000163  
 WRT.AD = 015102  
 WRT.RP = 015012  
 WRT.R1 = 015076  
 WRT.R2 = 015162  
 WRT.R3 = 015170  
 WRT.R4 = 015174  
 WRT.R5 = 015176  
 WRT.WD = 015100  
 WRU = 000400  
 WSU = 000004  
 \$AUTOB = 001134  
 \$BDADR = 001122  
 \$BDDAT = 001126  
 \$BELL = 006767  
 \$CHARC = 005110  
 \$CKSWR = 006072  
 \$CMTAG = 001100  
 \$CM3 = 000000  
 \$CNTLC = 006773  
 \$CNTLG = 007005  
 \$CNTLU = 007000  
 \$CRLF = 001161  
 \$DBLK = 005556  
 \$DIV = 007136  
 \$DTBL = 005546  
 \$ERFLG = 001103  
 \$ERMAX = 001115  
 \$ERROR = 004412  
 \$ERRPC = 001116  
 \$ERRTB = 001270  
 \$ERRTY = 004520  
 \$ERTTL = 001112  
 \$FILLC = 001156  
 \$FILLS = 001155

\$GDADR 001120	\$LPERR 001110	\$RESRE 007414	\$TKQOU 005572	\$STSTM 001102
\$GDDAT 001124	\$MNEW 007023	\$RPADR 001264	\$TKQSR 005574	\$TTYIN 006760
\$GTSWR 006162	\$MSWR 007012	\$RPVEC 001266	\$TKS 001144	\$TYPDS 005342
\$HD = 000001	\$NULL 001154	\$SAVRE 007356	\$TKSRV 005654	\$TYPE 004674
\$HEAD 001172	\$OCNT 005336	\$SETUP= 000146	\$TN = 000001	\$TYPEC 005044
\$HINUM 007132	\$OMODE 005340	\$STUP = 177777	\$TPB = 001152	\$TYPEX 005112
\$ICNT 001104	\$PASS 001100	\$SWR = 120000	\$TPFLG 001157	\$TYPC 005140
\$INTAG 001135	\$QUES 001160	\$TKB 001146	\$TPS 001150	\$TYPON 005154
\$ITEMB 001114	\$RAND 007034	\$TKCNT 005566	\$TRAP 007452	\$TYPOS 005114
\$LF 001162	\$RDCHR 006434	\$TKINT 005604	\$TRAP2 007474	\$OFILL 005337
\$LONUM 007134	\$RDLIN 006524	\$TKQEN= 005603	\$TRP = 000014	. = 022536
\$LPADR 001106	\$RDSZ = 000007	\$TKQIN 005570	\$TRPAD 007506	

. ABS. 022536 000

ERRORS DETECTED: 0

RM03:CZRJCB,CZRJCB.SEQ/SOL/LI:ME/NL:MC=RPO456.011,CZRJCB.P11  
RUN-TIME: 21 15 .6 SECONDS  
RUN-TIME RATIO: 92/37=2.4  
CORE USED: 34K (67 PAGES)



L09