

RP04
RP05, RP06

RP04/5/6 DSKLS 2
CZRJHE0

AH-9215E-MC
FICHE 1 OF 1

APR 1982
COPYRIGHT © 76-82
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of data. Each cell in the grid contains a small, dense table or set of data points. The text is extremely small and difficult to read, but the overall structure is a regular grid of information. The data appears to be organized in a way that allows for comparison across rows and columns, possibly representing different parameters or measurements over time or across different categories.

.REM a

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

IDENTIFICATION

PRODUCT CODE: AC-9213E-MC
 PRODUCT NAME: CZRJHE0 RP04/5/6 DISKLESS TEST, PT 2
 PRODUCT DATE: NOVEMBER 1981
 MAINTAINER: CX DIAGNOSTIC GROUP
 AUTHOR: PETE BLACKSTONE
 REVISED BY: MIKE LEAVITT

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

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

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

COPYRIGHT (C) 1976,1979,1982 DIGITAL EQUIPMENT CORPORATION

CONTENTS

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

- 1. ABSTRACT
- 2. REQUIREMENTS
 - 2.1 EQUIPMENT
 - 2.2 STORAGE
 - 2.3 PRELIMINARY PROGRAMS
- 3. LOADING PROCEDURE
 - 3.1 METHOD
- 4. STARTING PROCEDURE
 - 4.1 CONTROL SWITCH SETTINGS
 - 4.2 STARTING ADDRESS OR ADDRESSES
 - 4.3 PROGRAM AND/OR OPERATOR ACTION
- 5. OPERATING PROCEDURE
 - 5.1 OPERATIONAL SWITCH SETTINGS
 - 5.2 SUB-ROUTINE ABSTRACTS
- 6. ERRORS
 - 6.1 'FATAL' ERRORS
- 7. RESTRICTIONS
- 8. MISCELLANEOUS
 - 8.1 EXECUTION TIME
 - 8.2 STACK POINTER
 - 8.3 OPERATOR SELECTABLE SCOPE LOOPS
 - 8.4 PROGRAM REVISION HISTORY
- 9. PROGRAM DESCRIPTION

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

1.0 ABSTRACT

THIS DIAGNOSTIC TESTS THE RH11 AND DCL OF AN RP04/5/6 SUBSYSTEM. IT DOES NOT USE THE DISK SURFACE OR ANY SIGNALS FROM THE MDLI. IT REQUIRES THAT THE DCL CABLE BE PLUGGED INTO THE MDLI OR BE APPROPRIATELY TERMINATED. IF THE DISK IS POWERED UP, IT IS REQUIRED TO GET THE DISK TO THE 'HEADS UNLOADED' POSITION. AFTER A SUCCESSFUL RUN (WITH NO ERRORS) OF THIS DIAGNOSTIC IT CAN BE ASSERTED THAT, "THAT PART OF THE DCL THAT HANDLES DATA OR DATA ASSOCIATED LOGIC IS WORKING PROPERLY". THIS IMPLIES THAT, THAT PART OF THE LOGIC WHICH HANDLES MECHANICAL COMMANDS OR ITS ASSOCIATED LOGIC IS NOT TESTED IN THIS DIAGNOSTIC. ALL DATA COMMANDS USE THE MAINTENANCE REGISTER IN THE WRAPAROUND MODE.

THE DIAGNOSTIC DOES NOT DO ANY TESTING OF THE RH70 CONTROLLER WHEN IT IS USED TO TEST RP04/5/6 DISK DRIVES CONNECTED TO THAT TYPE OF CONTROLLER. IT IS ASSUMED THAT THE RH70 SPECIFIC CONTROLLER DIAGNOSTICS HAVE BEEN SUCCESSFULLY RUN TO COMPLETION BEFORE THIS PROGRAM IS RUN.

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11 COMPUTER WITH CONSOLE TELETYPE, AND AN RP04/5/6 DISK SYSTEM. THE RP04/5/6 DISK SYSTEM WILL CONSIST OF AN RH11/RH70 CONTROLLER, AND DISK CONTROL LOGIC (DCL). THE CABLE FROM THE DCL CAN BE CONNECTED TO THE MDLI, BUT IF NOT THAT CABLE MUST BE PROPERLY TERMINATED.

2.2 STORAGE

THIS PROGRAM REQUIRES 16K WORDS OF MEMORY.

2.3 PRELIMINARY PROGRAMS

THIS PROGRAM ASSUMES THAT MAINDEC-11-DZRJG- (LATEST REV) HAS BEEN RUN WITHOUT ERRORS

3.0 LOADING PROCEDURE

USE STANDARD PROCEDURE FOR LOADING .ABS TAPES

4.0 STARTING PROCEDURE

SWITCH 12 MUST BE SET WHEN THIS PROGRAM IS TO BE RUN USING AN RH70 CONTROLLER. IT CAN BE SET AT THE FRONT PANEL, OR IN THE SOFTWARE SWITCH REGISTER IF THE OPERATOR SO DESIRES. SEE PARAGRAPH 5.1 FOR A DESCRIPTION OF SOFTWARE SWITCH REGISTER OPERATION.

4.1 CONTROL SWITCH SETTINGS

SEE SECTION 5.1

4.2 STARTING ADDRESS

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

START AT ADDRESS 200---FOR NORMAL RUN
START AT ADDRESS 204---TO SELECT NON-DEFAULT PARAMETERS
START AT ADDRESS 210---FOR UNIT SELECTION

200 START
ALL SWITCHES MUST BE DOWN FOR WORST CASE RUN. WITH THIS STARTING ADDRESS ALL THE RP04/5/6'S ON THE SYSTEM WILL BE TESTED ONE AT A TIME BEFORE 'END PASS' IS PRINTED OUT. TESTING WILL START WITH THE LOWEST UNIT NUMBER DRIVE THAT IS POWERED UP (THAT IS THE LOWEST UNIT NUMBER RHAS REGISTER THAT RESPONDS) THEN GO ON TO THE NEXT HIGHER UNIT NUMBER THAT IS POWERED UP.

204 RESTART
SAME AS 200 START WITH FOLLOWING EXCEPTIONS: PROGRAM WILL QUERY OPERATOR FOR THE CORRECT CSR AND VECTOR ADDRESS OF THE RHXX CONTROLLER. WHEN THIS ACTION HAS BEEN COMPLETED, THE PROGRAM WILL AUTOMATICALLY RESTART FROM ADDRESS 200, WITH THE SAME CONVENTIONS AS DESCRIBED FOR A 200 START.

210 START
ALL SWITCHES MUST BE DOWN FOR WORST CASE RUN. WITH THIS STARTING ADDRESS THE CONSOLE TELETYPE WILL ASK FOR THE UNIT NUMBER TO BE TESTED. THEN ONLY THAT UNIT WILL BE TESTED FOR EACH PASS OF THE PROGRAM.

4.3 PROGRAM AND/OR OPERATOR ACTION

1. LOAD THE PROGRAM INTO MEMORY.
2. SET STARTING ADDRESS ON THE SWITCH REGISTER
3. PRESS 'LOAD ADDRESS'.
4. SET 'OPERATIONAL SWITCH SETTINGS' (SEE SECTION 5.1) WORST CASE IS ALL SWITCHES DOWN.
5. PRESS 'START'.
6. FOR THE FIRST PASS EACH TEST WILL BE EXECUTED ONCE ON THE DRIVES PRESENT OR DRIVE SELECTED BEFORE 'END PASS' IS PRINTED. THE FIRST PASS WILL REQUIRE OPERATOR INTERVENTION IF THE PROGRAM IS NOT RUN UNDER AN 'ACT-11' MONITOR. THE SECOND AND SUBSEQUENT PASSES WILL EXECUTE EACH TEST FOUR TIMES ON EACH DRIVES PRESENT OR DRIVE SELECTED BEFORE 'END PASS' IS PRINTED. THE SECOND AND SUBSEQUENT PASSED DO NOT NEED ANY OPERATOR INTERVENTION.

5.0 OPERATING PROCEDURE

5.1 OPERATIONAL SWITCH SETTINGS

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 AN 'SOFTWARE' SWITCH REGISTER. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE

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

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 RP04/5/6 INTERRUPT. THE 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

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

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

SWITCH DEFINITIONS ARE GIVEN IN SECTION 9 'OPERATIONAL SWITCH SETTINGS' HOWEVER THE DETAIL DESCRIPTIONS ARE GIVEN HERE.

SWITCH 15 - HALT ON ERROR
WHEN THIS SWITCH IS SET, IF THE PROGRAM FINDS AN ERROR THEN THE APPROPRIATE INFORMATION WILL BE PRINTED OUT AND THEN THE PROGRAM WILL HALT. AFTER THIS HALT, PRESSING 'CONTINUE' WILL CONTINUE WITH THE PROGRAM TILL THE NEXT ERROR IS FOUND WHEN THE SAME THING WILL HAPPEN.

SWITCH 14 - LOOP ON TEST
WHEN THIS SWITCH IS SET THE PROGRAM WILL BEGIN TO LOOP ON THE CURRENT TEST BEING EXECUTED. FOR EXAMPLE IF THIS SWITCH IS SET WHEN THE PROGRAM IS IN TEST 10 THEN THE PROGRAM WILL KEEP EXECUTING ALL OF TEST 10 REPEATEDLY. ONE WAY TO BE SURE THAT THE PROGRAM IS IN THE EXPECTED TEST IS TO SET THIS SWITCH DURING AN ERROR PRINTOUT OR DURING A PROGRAM HALT.

SWITCH 13 - INHIBIT ERROR TYPEOUTS
WHEN THIS SWITCH IS SET FURTHER ERROR PRINTOUTS WILL CEASE, HOWEVER OPERATOR INSTRUCTIONS SUCH AS 'STOP DRIVE X' WILL CONTINUE. AT THE END OF PASS 'TOTAL NUMBER OF ERRORS ON THIS PASS ON DRIVE X' WILL BE TRUE, THAT IS, ALTHOUGH PRINTOUTS WERE INHIBITED IF THAT PASS FOUND 6 ERRORS, IT WILL SAY SO.

SWITCH 12 - RH70 CONTROLLER SELECT
THIS SWITCH MUST BE SET AT THE START OF THE PROGRAM WHEN THE DISK DRIVES TO BE TESTED ARE CONNECTED TO AN RH70 CONTROLLER. IT MUST NOT BE SET WHEN DISK DRIVES TO BE TESTED ARE CONNECTED TO AN RH11 CONTROLLER.

SWITCH 11 - INHIBIT ITERATIONS

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

WHEN THIS SWITCH IS SET THE PROGRAM ON SECOND PASS WILL NOT REPEAT EACH TEST FOUR TIMES BUT WILL DO EACH TEST ONCE ONLY.

SWITCH 10 - BELL ON ERROR
WHEN THIS SWITCH IS SET, IF THE PROGRAM FINDS AN ERROR THE "BELL" OR "ALARM" WILL BE SOUNDED. THIS SWITCH IS USEFUL WHEN SWITCH 11 IS SET YET INFORMATION IS NEEDED WHEN ANY ERROR IS DETECTED. TAKE THE EXAMPLE OF A PROGRAM LOOPING ON A TEST WITH SWITCH 11 SET TO HELP SCOPING. THEN IF THIS SWITCH IS SET AND THE BELL OR ALARM SOUNDS IT MEANS THAT THE ERROR IS PRESENT BUT IF THE BELL OR ALARM STOPS IT MEANS THAT THE ERROR IS NOT PRESENT.

SWITCH 9 - LOOP ON ERROR
WHEN THIS SWITCH IS SET, IF THE PROGRAM FINDS AN ERROR THEN GENERALLY THE PROGRAM WILL LOOP BACK TO THE LAST EXECUTED "SCOPE" STATEMENT. IF ON THE SECOND TIME THROUGH AN ERROR IS FOUND IT WILL AGAIN LOOP BACK TO THAT "SCOPE" STATEMENT. THIS LOOPING WILL CONTINUE AS LONG AS THE ERROR IS PRESENT AND THIS SWITCH IS SET. HOWEVER IF THE ERROR IS NOT PRESENT AT ANY TIME THEN IT WILL CONTINUE NORMALLY WITH THE PROGRAM. EACH TIME THE ERROR IS ENCOUNTERED PRINTOUT WILL TAKE PLACE UNLESS SWITCH 11 IS ALSO SET. DURING BEGUG, USING A SCOPE, IT IS RECOMMENDED THAT SWITCH 11 IS ALSO SET.

NOTE: ALSO SEE SECTION 8.3

SWITCH 8 - LOOP ON TEST IN SWR <7:0>
THIS IS A SPECIAL SWITCH. WHEN SET SWITCHES 0 THRU 7 HAVE ONE MEANING AND WHEN RESET SWITCHES 0 THRU 7 HAVE ANOTHER MEANING. THIS MEANS THAT ANY SETTING OF SWITCH 0 THRU 7 MUST BE DONE WITH SWITCH 8 IN THE APPROPRIATE POSITION. WHEN THIS SWITCH IS SET THEN SWITCHES 0 THRU 7 GIVE THE TEST NUMBER TO BE LOOPED ON. FOR EXAMPLE WITH SWITCH 8 SET AND SWITCH 3 SET THE PROGRAM WILL LOOP ON TEST 10. HOWEVER THIS SETTING MUST BE DONE AT THE BEGINNING OF THE PROGRAM THEN ALL THE TESTS FROM 1 TO 10 WILL BE EXECUTED AND THEN TEST 10 WILL BE REPEATED OVER AND OVER AGAIN. WHEN THIS SWITCH IS NOT SET THEN SWITCHES 0 THRU 7 HAVE THE MEANING ITS NAME INDICATES.
FOR EXAMPLE SWITCH 7 IS "STOP FURTHER COMPARES: THAT IS IF SWITCH 8 IS NOT SET AND SWITCH 7 IS SET THEN WHEN A DATA ERROR IS DETECTED NO FURTHER COMPARES WILL BE DONE. FOR EXAMPLE IN A 256 WORD BUFFER IF ALL THE WORDS ARE IN ERROR THEN AFTER SEEING THE PRINTOUT FOR THE FIRST FEW WORDS SETTING SWITCH 7 ONLY WILL STOP FURTHER PRINTOUTS OF THIS ERROR AND GO ON WITH THE TEST RATHER THAN PRINT ALL THE 256 WORDS. HOWEVER IF THIS WAS DONE WITH SWITCH 11 THEN THE NEXT ERROR THAT THE PROGRAM DETECTS IN A SUBSEQUENT TEST WILL ALSO BE LOST. BUT WITH SWITCH 7, ONLY THIS GROUP OF DATA ERRORS ARE NOT PRINTED OUT. ANOTHER EXAMPLE OF SWITCH 8 BEING LOW IS WITH SWITCH 6, WHICH

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

IS 'ECC TEST-COMPARE END RESULT ONLY'. THAT IS IF SWITCH 8 IS NOT SET AND SWITCH 6 IS SET THEN ON ECC TESTS (TEST 120 THRU TEST 134) INSTEAD OF COMPARING CONTENTS OF THE POSITION REGISTER AND PATTERN REGISTER AFTER EVERY CLOCK, COMPARES WILL ONLY BE DONE AT THE END OF ALL THE CLOCKS.

NOTE: ALSO SEE SECTION 8.3

SWITCH 7 - STOP FURTHER COMPARES IF SW08 IS LOW. IF SWITCH 8 IS SET AND THIS SWITCH IS ALSO SET THEN THIS SWITCH GIVES THE TEST NUMBER TO BE LOOPED ON AS INDICATED IN THE DESCRIPTION OF SWITCH 8. IF SWITCH 8 IS NOT SET AND THIS SWITCH IS SET THEN THE PROGRAM WILL DO AS THE NAME INDICATES. FOR EXAMPLE IN A 256 WORD BUFFER IF ALL THE WORDS ARE IN ERROR THEN AFTER SEEING THE ERROR PRINTOUTS FOR THE FIRST FEW WORDS THEN SETTING SWITCH 7 WITH SWITCH 8 NOT SET WILL STOP THE PRINTOUT OF ALL 256 WORDS BUT WILL NOT STOP THE PRINTOUT OF ANOTHER ERROR IN ANY SUBSEQUENT TEST. IT IS EXPECTED THAT SWITCH 7 AFTER BEING SET FOR A WHILE TO STOP PRINTING ALL THE 256 WORDS WILL BE RESET AGAIN TO ENABLE THE PRINTING OF OTHER DATA ERRORS.

SWITCH 6 - ECC TEST-COMPARE END RESULTS ONLY IF SW08 IS LOW IF SWITCH 8 IS SET AND THIS SWITCH IS ALSO SET THEN THIS SWITCH GIVES THE TEST NUMBER TO BE LOOPED ON AS INDICATED IN THE DESCRIPTION OF SWITCH 8. IF SWITCH 8 IS NOT SET AND THIS SWITCH IS SET THEN ON ECC TESTS (TEST 120 THRU TEST 134) INSTEAD OF COMPARING CONTENTS OF THE POSITION AND PATTERN REGISTERS AFTER EVERY CLOCK, COMPARES WILL BE DONE ONLY AT THE END OF ALL THE CLOCKS.

5.2 SUB-ROUTINE ABSTRACTS

SEE SECTION 9 "SUBROUTINES"

6.0 ERRORS

ERROR PRINTOUTS CONTAIN THE ERROR ADDRESS AND OTHER PERTINENT INFORMATION CONCERNING THE PARTICULAR FAILURE. THIS INFORMATION MAY BE THE CONTENTS OF RELEVANT RP04 REGISTERS OR GOOD/RECEIVED DATA. IF THE ERROR OCCURRED IN A SUBROUTINE, THE ADDRESS OF THE SUBROUTINE CALL IS ALSO GIVEN. REFER TO THE PROGRAM LISTING AT THE STATED ADDRESS TO DETERMINE THE CAUSE OF THE ERROR.

6.1 'FATAL' ERRORS

IN THE EVENT THAT THE DISK DRIVE BECOMES UNAVAILABLE TO THE CONTROLLER, POWERS DOWN, OR CERTAIN CRITICAL STATUS BITS CANNOT BE CLEARED PRIOR TO THE START OF A TEST SEQUENCE - THIS INFORMATION WILL BE COMMUNICATED TO THE OPERATOR. IN ADDITION, THE TTY BELL WILL RING AND THE PROGRAM WILL HALT. IT IS SUGGESTED THAT IF THIS HAPPENS THE OPERATOR LOAD

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

ADDRESS 200 (210) AND RESTART THE PROGRAM AS A FIRST ATTEMPT TO SOLVE THE PROBLEM. IF THE FAILURE CONTINUES TO OCCUR, THERE ARE TWO OPTIONS FOR THE OPERATOR:

1. LOOK IN THE TEST LISTING FOR THE 'HALT' INSTRUCTION AND REPLACE IT PLUS THE TWO WORDS ('TYPE ,CPHALT') ABOVE WITH 'NOP'S. WITH TTY ERROR PRINTOUTS INHIBITED, A SCOPE LOOP CAN BE INITIATED FOR THE TEST IN QUESTION.
2. GO BACK AND RERUN DZRPS AS IT IS QUITE POSSIBLE THAT A HARD FAILURE HAS OCCURRED IN ONE OF THE HARDWARE REGISTERS.

IT IS ALSO POSSIBLE TO CONTINUE FROM THE HALT POINT, BUT THIS IS NOT RECOMMENDED AS ALL FOLLOWING TESTS WILL EXHIBIT THE SAME SYMPTOMS AND GIVE MISLEADING ERROR PRINTOUTS.

7.0 RESTRICTIONS

IF THERE IS A DRIVE CONNECTED THEN THE OPERATOR MUST HAVE THE DRIVE PORT SWITCH LOCKED EITHER ON PORT A OR PORT B BUT NEVER LEAVE IT IN THE PROGRAMMABLE STATE. IF THERE IS NO DRIVE CONNECTED THEN THE CABLE NORMALLY GOING FROM THE DCL TO THE MDLI MUST BE PROPERLY TERMINATED.

SWITCH 12 MUST BE SET WHEN RUNNING ON AN RH70 CONTROLLER AND IT MUST NOT BE SET WHEN RUNNING ON AN RH11 CONTROLLER. BECAUSE OF THIS FACT, THE PROGRAM CANNOT BE RUN IN CHAIN MODE WHEN USING THE SOFTWARE SWITCH REGISTER AS THE ROUTINE WHICH ASKS FOR THE SWITCH REGISTER SETTINGS IS NOT OPERABLE WHEN IN CHAIN MODE.

8.0 MISCELLANEOUS

8.1 EXECUTION TIME

THE FIRST PASS OF THE PROGRAM WILL TAKE 1.75 MINUTES PER DRIVE. SUBSEQUENT PASSES WILL TAKE 7 MINUTE.

8.2 STACK POINTER

THE STACK IS INITIALLY SET TO 1000

8.3 OPERATOR SELECTABLE SCOPE LOOPS

HERE IS A DETAILED EXPLANATION OF HOW THE LOOP ON ERROR WORKS. FOR INSTRUCTIONS REGARDING USAGE OF THESE LOOPS, HIT CONTROL C ANY TIME WHILE THE PROGRAM IS RUNNING. ON HITTING AN ERROR IF THE LOOP ON ERROR SWITCH IS SET, THE PROGRAM GOES BACK - USUALLY BACK TO THE BEGINNING OF THE TEST.

WHEN THIS OPERATOR SELECTABLE SCOPE LOOP IS USED THEN THE POINT THE PROGRAM GOES BACK TO CAN BE CHANGED.

THE RESTRICTIONS TO THE POINT WHERE THE PROGRAM CAN GO ARE: -

1. IT MUST BE WITHIN THE TEST UNDER CONSIDERATION
2. LOOP ON ERROR SWITCH MUST BE SET

343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362

3. THE ERROR MUST OCCUR WITHIN THE TEST UNDER CONSIDERATION
IF THE ERROR DOES NOT OCCUR WITHIN THE TEST UNDER CONSIDERATION
THE PROGRAM WILL REVERT TO NORMAL OPERATION. HOWEVER, IF LOOP ON
TEST SWITCH IS SET AND THIS OPERATOR SELECTABLE SCOPE LOOP IS USED
THEN THE PROGRAM WILL LOOP BACK TO THE SELECTED POINT WHEN IT
COMES TO THE END OF THE TEST UNDER CONSIDERATION.

AFTER LOOPING FOR SOME TIME IF THE LOOP SWITCH IS PUT DOWN THEN
NORMAL OPERATION WILL CONTINUE.

8.4 PROGRAM REVISION HISTORY

9.0 PROGRAM DESCRIPTION

THE FOLLOWING SECTIONS DESCRIBE EACH TEST AND SUBROUTINES
IN DETAIL AND CAN ALSO BE USED AS AN INDEX TO THE LISTING.
THE LEFT MOST COLUMN IS THE LINE NUMBER WITHIN THE LISTING
WHERE THAT ITEM WILL BE FOUND.

a

1
540
541

;*LAST REVISION 02-NOV-81

.TITLE CZRJHEO RP04/5/6 DSKLS PT2
;*COPYRIGHT (C) 1976,1978,1981
;*DIGITAL EQUIPMENT CORPORATION
;*COLORADO SPGS., CO. 80919
;*;*PROGRAM BY PETE BLACKSTONE
;*;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11-DZQAC-C5), 18-MAR-81
;*

542

.SBTTL OPERATIONAL SWITCH SETTINGS
;*

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	RH70 CONTROLLER SELECT
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<7:0>
7	STOP FURTHER COMPARES IF SW08 IS LOW
6	ECC TEST-COMPARE END RESULTS ONLY IF SW08 IS LOW

543
544
545
546

.SBTTL BASIC DEFINITIONS

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

001100
104000
000004STACK = 1100
ERROR = EMT ::BASIC DEFINITION OF ERROR CALL
SCOPE = IOT ::BASIC DEFINITION OF SCOPE CALL

;*MISCELLANEOUS DEFINITIONS

000011
000012
000015
000200
177776
177776
177774
177772
177570
177570HT = 11 ::CODE FOR HORIZONTAL TAB
LF = 12 ::CODE FOR LINE FEED
CR = 15 ::CODE FOR CARRIAGE RETURN
CRLF = 200 ::CODE FOR CARRIAGE RETURN-LINE FEED
PS = 177776 ::PROCESSOR STATUS WORD
PSW=PS
STKLMT = 177774 ::STACK LIMIT REGISTER
PIRQ = 177772 ::PROGRAM INTERRUPT REQUEST REGISTER
DSWR = 177570 ::HARDWARE SWITCH REGISTER
DDISP = 177570 ::HARDWARE DISPLAY REGISTER

;*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007R0 = %0 ::GENERAL REGISTER
R1 = %1 ::GENERAL REGISTER
R2 = %2 ::GENERAL REGISTER
R3 = %3 ::GENERAL REGISTER
R4 = %4 ::GENERAL REGISTER
R5 = %5 ::GENERAL REGISTER
R6 = %6 ::GENERAL REGISTER
R7 = %7 ::GENERAL REGISTER
SP = %6 ::STACK POINTER
PC = %7 ::PROGRAM COUNTER

```
000000      ;*PRIORITY LEVEL DEFINITIONS
000040      PR0      = 0          ;;PRIORITY LEVEL 0
000100      PR1      = 40         ;;PRIORITY LEVEL 1
000140      PR2      = 100        ;;PRIORITY LEVEL 2
000200      PR3      = 140        ;;PRIORITY LEVEL 3
000240      PR4      = 200        ;;PRIORITY LEVEL 4
000300      PR5      = 240        ;;PRIORITY LEVEL 5
000340      PR6      = 300        ;;PRIORITY LEVEL 6
000340      PR7      = 340        ;;PRIORITY LEVEL 7
```

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

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



```
000400 BIT8=BIT08
000200 BIT7=BIT07
000100 BIT6=BIT06
000040 BIT5=BIT05
000020 BIT4=BIT04
000010 BIT3=BIT03
000004 BIT2=BIT02
000002 BIT1=BIT01
000001 BIT0=BIT00
```

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

.SBTTL RH11/RH70 REGISTERS

```
;*WORD COUNT REGISTER (RHWC)
;*EACH BIT IS CALLED BY BIT NUMBER
```

```
;*BUS ADDRESS REGISTER (RHBA)
;*EACH BIT IS CALLED BY BIT NUMBER
```

;*CONTROL AND STATUS REGISTER 2 (RHCS2)

```
547
548
549
550
551
552
553
554
555
556
557
558
559
560 000001 US1= 1 ;:UNIT SELECT (BIT #0)
561 000002 US2= 2 ;:UNIT SELECT (BIT #1)
562 000004 US4= 4 ;:UNIT SELECT (BIT #2)
563 000010 BAI= 10 ;:BUS ADDRESS INCREMENT INHIBIT (BIT #3)
564 000020 PAT= 20 ;:INVERT PARITY ON MASS BUS TO EVEN (BIT #4)
565 000040 CLR= 40 ;:CLEAR (BIT #5)
566 000100 IR= 100 ;:INPUT READY (BIT #6)
567 000200 OR= 200 ;:OUTPUT READY (BIT #7)
568 000400 MPE= 400 ;:MASS BUS PARITY ERROR (BIT #8)
569 001000 MXF= 1000 ;:MISSED TRANSFER ERROR (BIT #9)
570 002000 PGE= 2000 ;:PROGRAM ERROR (BIT #10)
571 004000 NEM= 4000 ;:NON EXISTANT MEMORY (BIT #11)
572 010000 NED= 10000 ;:NON EXISTANT DRIVE (BIT #12)
573 020000 UPE= 20000 ;:UNIBUS PARITY ERROR (BIT #13)
574 040000 WCE= 40000 ;:WRITE CHECK ERROR (BIT #14)
575 100000 DLT= 100000 ;:DATA LATE (BIT #15)
576
577
```

```
;*DATA BUFFER REGISTER (RHDB)
;*EACH BIT IS CALLED BY BIT NUMBER
```

547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580


```

581
582      ;*RP04 REGISTERS
583
584      ;*CONTROL AND STATUS 1 REGISTER. (#00)
585
586      000001      GO=      1      ;GO (BIT #0)
587      000100      IE=      100     ;INTERRUPT ENABLE (BIT #6)
588      000200      RDY=     200     ;READY (BIT #7)
589      000400      A16=     400     ;HIGH ORDER UNIBUS BITS (BIT #8)
590      001000      A17=    1000    ;HIGH ORDER UNIBUS BITS (BIT #9)
591      002000      PSEL=    2000    ;PORT SELECT (BIT #10)
592      004000      DVA=    4000    ;DEVICE AVAILABLE (BIT #11)
593      020000      MCPE=   20000   ;MASSBUSS PARITY ERROR (BIT #13)
594      040000      TRE=    40000   ;TRANSFER ERROR (BIT #14)
595      100000      SC=    100000   ;SPECIAL CONDITION (BIT #15)
596
597      ;*STATUS REGISTER (RHDS1) (#01)
598
599      000001      DF5=      1      ;DRIVE FORWARD 5''/SEC. (BIT #0)
600      000002      DFF20=   2      ;DRIVE FORWARD 20''/SEC. (BIT #1)
601      000004      DIGB=    4      ;DRIVE TO INNER GAVRD BAND (BIT #2)
602      000010      GRV=    10      ;GO REVERSE (BIT #3)
603      000020      DL64=   20      ;DIFFERENCE LESS THAN 64 (BIT #4)
604      000040      DE1=    40      ;DIFFERENCE EQUALS 1 (BIT #5)
605      000100      VV=    100      ;VOLUME VALID (BIT #6)
606      000200      DRY=    200      ;DRIVE READY (BIT #7)
607      000400      DPR=    400      ;DRIVE PRESENT (BIT #8)
608      001000      PROG=   1000     ;PROGRAMABLE (BIT #9)
609      002000      LST=   2000     ;LAST SECTOR TRANSFERRED (BIT #10)
610      004000      WRL=   4000     ;WRITE LOCK (BIT #11)
611      010000      MOL=  10000    ;MEDIUM ON-LINE (BIT #12)
612      020000      PIP=  20000    ;POSITIONING OPERATION IN PROGRESS (BIT #13)
613      040000      ERR=  40000    ;COMPOSIT ERROR. (BIT #14)
614      100000      ATA= 100000    ;ATTENTION ACTIVE (BIT #15)
615
616      ;*ERROR REGISTER #01 (RHER1) (#02)
617
618      000001      ILF=      1      ;ILLEGAL FUNCTION (BIT #0)
619      000002      ILR=      2      ;ILLEGAL REGISTER (BIT #1)
620      000004      RMR=      4      ;REGISTER MODIFICATION REFUSED (BIT #2)
621      000010      PAR=     10      ;PARITY ERROR (BIT #3)
622      000020      FER=     20      ;FORMAT ERROR (BIT #4)
623      000040      WCF=     40      ;WRITE CLOCK FAIL (BIT #5)
624      000100      ECH=    100      ;ECC HARD ERROR (BIT #6)
625      000200      HCE=    200      ;HEADER COMPARE ERROR (BIT #7)
626      000400      HCRC=   400      ;HEADER CRC ERROR (BIT #8)
627      001000      AOE=   1000     ;ADDRESS OVERFLOW ERROR (BIT #9)
628      002000      IAE=   2000     ;INVALID ADDRESS ERROR (BIT #10)
629      004000      WLE=   4000     ;WRITE LOCK ERROR (BIT #11)
630      010000      DTE=  10000    ;DRIVE TIMING ERROR (BIT #12)
631      020000      OPI=  20000    ;OPERATION INCOMPLETE (BIT #13)
632      040000      UNS=  40000    ;DRIVE UNSAFE (BIT #14)
633      100000      DCK= 100000    ;DATA CHECK ERROR (BIT 15)
634
635      ;*MAINTAINABILITY REGISTER (RHMR) (#03)
636
637      000001      DMD=      1      ;DIAGINOSTIC MODE (BIT #0)
  
```


638	000002	MCLK= 2	:MAINTAINABILITY CLOCK (BIT #1)
639	000004	MINX= 4	:MAINTAINABILITY INDEX (BIT #2)
640	000010	MSTCK= 10	:MAINTAINABILITY SECTOR CLOCK (BIT #3)
641	000020	MRD= 20	:MAINTAINABILITY READ (BIT #4)
642	000040	MWR= 40	:MAINTAINABILITY WRITE (BIT #5)
643	000200	DENVL= 200	:DATA ENVELOPE (BIT #7)
644	000400	ZER= 400	:ZERO DETECT (BIT #8)
645	001000	DTSY= 1000	:MAINTAINABILITY SYNC DETECTED (BIT #9)
646			
647		:*ATTENTION SUMMARY PSEUDO-REGISTER (RHAS) (#04)	
648			
649	000001	AT0= 1	:DEVICE 0 (BIT #0)
650	000002	AT1= 2	:DEVICE 1 (BIT #1)
651	000004	AT2= 4	:DEVICE 2 (BIT #2)
652	000010	AT3= 10	:DEVICE 3 (BIT #3)
653	000020	AT4= 20	:DEVICE 4 (BIT #4)
654	000040	AT5= 40	:DEVICE 5 (BIT #5)
655	000100	AT6= 100	:DEVICE 6 (BIT #6)
656	000200	AT7= 200	:DEVICE 7 (BIT #7)
657			
658			
659		:*DESIRED SECTOR/TRACK ADDRESS REGISTER (RHDST) (#1)	
660		:*EACH BIT IS CALLED BY BIT NUMBER	
661			
662			
663		:*DRIVE TYPE REGISTER (RHDT) (#06)	
664		:*EACH BIT IS CALLED BY BIT NUMBER	
665			
666			
667		:*LOOK-AHEAD REGISTER (RHLA) (#07)	
668			
669	000001	EXT1= 1	:EXTENSION 1 (BIT #0)
670	000002	EXT2= 2	:EXTENSION 2 (BIT #1)
671	000004	EXT4= 4	:EXTENSION 3 (BIT #2)
672	000010	EXT10= 10	:EXTENSION 4 (BIT #3)
673	000020	EXT20= 20	:EXTENSION 5 (BIT #4)
674	000040	EXT40= 40	:EXTENSION 6 (BIT #5)
675	000100	SC1= 100	:SECTOR COUNT FIELD 0 (BIT #6)
676	000200	SC2= 200	:SECTOR COUNT FIELD 1 (BIT #7)
677	000400	SC4= 400	:SECTOR COUNT FIELD 2 (BIT #8)
678	001000	SC10= 1000	:SECTOR COUNT FIELD 3 (BIT #9)
679	002000	SC20= 2000	:SECTOR COUNT FIELD 4 (BIT #10)
680	004000	TRK1= 4000	:TRACK FIELD 1 (BIT #11)
681	010000	TRK2= 10000	:TRACK FIELD 2 (BIT #12)
682	020000	TRK4= 20000	:TRACK FIELD 3 (BIT #13)
683	040000	TRK10= 40000	:TRACK FIELD 4 (BIT #14)
684	100000	TRK20= 100000	:TRACK FIELD 5 (BIT #15)
685			
686		:*RP04 ERROR REGISTER #2 (RHER2) (#10)	
687			
688	000001	WCU= 1	:WRITE CURRENT UNSAFE (BIT #0)
689	000002	CSF= 2	:CURRENT SINK FAILURE (BIT #1)
690	000004	WSU= 4	:WRITE SELECT UNSAFE (BIT #2)
691	000010	CSU= 10	:CURRENT SWITCH UNSAFE (BIT #3)
692	000020	MSE= 20	:MOTOR SEQUENCE ERROR (BIT #4)
693	000040	TDF= 40	:TRANSITIONS DETECTOR FAILURE (BIT #5)
694	000100	TUF= 100	:TRANSITIONS UNSAFE (BIT #6)

695	000200	FEN=	200	:FAILSAFE ENABLED (BIT #7)
696	000400	WRU=	400	:WRITE READY UNSAFE (BIT #8)
697	001000	MHS=	1000	:MULTIPLE HEAD SELECT (BIT #9)
698	002000	NHS=	2000	:NO HEAD SELECTION (BIT #10)
699	004000	IXE=	4000	:INDEX ERROR (BIT #11)
700	010000	VU30=	10000	:30VOLT UNSAFE (BIT #12)
701	020000	PLU=	20000	:PLO UNSAFE (BIT #13)
702	100000	ACU=	100000	:ACUNSAFE (BIT #15)
703				
704		;*RP05/6 ERROR REGISTER #2 (RHER2) (#10)		
705				
706	000001	WCU=	1	:WRITE CURENT UNSAFE
707	000002	CSF=	2	:CURRENT SINK FAILURE
708	000004	WSU=	4	:CURRENT SELECT UNSAFE
709	000010	CSU=	10	:CURRENT SWITCH UNSAFE
710	000020	RAW=	20	:READ AND WRITE
711	000040	TDF=	40	:TRANSITIONS DETECTOR FAILURE
712	000100	TUF=	100	:TRANSITIONS UNSAFE
713	000200	ABS=	200	:ABNORMAL STOP
714	000400	WRU=	400	:WRITE READY UNSAFE
715	001000	MHS=	1000	:MULTIPLE HEAD SELECT
716	002000	NHS=	2000	:NO HEAD SELECTION
717	004000	IXE=	4000	:INDEX ERROR
718	020000	PLU=	20000	:PLO UNSAFE
719				
720		;*OFFSET REGISTER (RHOF) (#11)		
721				
722	000001	OF25=	1	:OFFSET 25 MICRO INCHES (BIT #0)
723	000002	OF50=	2	:OFFSET 50 MICRO INCHES (BIT #1)
724	000004	OF100=	4	:OFFSET 100 MICRO INCHES (BIT #2)
725	000010	OF200=	10	:OFFSET 200 MICRO INCHES (BIT #3)
726	000020	OF400=	20	:OFFSET 400 MICRO INCHES (BIT #4)
727	000040	OF800=	40	:OFFSET 800 MICRO INCHES (BIT #5)
728				
729	000200	OFREV=	200	:OFFSET NEGATIVE (REVERSE) (BIT #7)
730	002000	HCI=	2000	:HEADER COMPARE INHIBIT (BIT #10)
731	004000	ECI=	4000	:ERROR CORRECTION CODE INHIBIT (BIT #11)
732	010000	FMT22=	10000	:FORMAT BIT (BIT #12)
733				
734				
735		;*DESIRED CYLINDER ADDRESS (RHCA) (#12)		
736		;*EACH BIT IS CALLED BY BIT NUMBER.		
737				
738				
739		;*CURRENT CYLINDER ADDRESS (RHCC) (#13)		
740		;*EACH BIT IS CALLED BY BIT NUMBER		
741				
742				
743		;*SERIAL NUMBER REGISTER (RHSN) (#14)		
744		;*EACH IS CALLED BY BIT NUMBER		
745				
746				
747		;*ERROR REGISTER #03 (RHER3) (#15)		
748				
749	000001	PSU=	1	:PACK SPEED UNSAFE (BIT #0)
750	000002	VUF=	2	:VELOCITY UNSAFE (BIT #1)
751	000010	UWR=	10	:ANY UNSAFE EXCEPT READ/WRITE (BIT #3)

752	000020	PRE=	20	:DISK PACK ROTATION ERROR (BIT #4)
753	000040	ACL=	40	:AC LOW (BIT #5)
754	000100	OPDCL=	100	:DC LOW (BIT #6)
755	040000	SKI=	40000	:SEEK INCOMPLETE (BIT #14)
756	100000	OCYL=	100000	:OFF CYLINDER (BIT #15)

757
758
759 ;*ECC POSITION REGISTER (RHEC1) (#16)
760 ;*EACH BIT IS CALLED BY BIT NUMBER

761
762
763 ;*ECC PATTERN REGISTER (RHEC2) (#17)
764 ;*EACH BIT IS CALLED BY BIT NUMBER

765 .SBTTL MEMORY MANAGEMENT DEFINITIONS

766 ;*KT11 VECTOR ADDRESS

000250 MMVEC = 250

 ;*KT11 STATUS REGISTER ADDRESSES

177572	SR0 = 177572
177574	SR1 = 177574
177576	SR2 = 177576
172516	SR3 = 172516

 ;*KERNEL 'I' PAGE DESCRIPTOR REGISTERS

172300	KIPDR0 = 172300
172302	KIPDR1 = 172302
172304	KIPDR2 = 172304
172306	KIPDR3 = 172306
172310	KIPDR4 = 172310
172312	KIPDR5 = 172312
172314	KIPDR6 = 172314
172316	KIPDR7 = 172316

 ;*KERNEL 'I' PAGE ADDRESS REGISTERS

172340	KIPAR0 = 172340
172342	KIPAR1 = 172342
172344	KIPAR2 = 172344
172346	KIPAR3 = 172346
172350	KIPAR4 = 172350
172352	KIPAR5 = 172352
172354	KIPAR6 = 172354
172356	KIPAR7 = 172356

767

1

.SBTTL TRAP CATCHER

000000

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

000174 000174
000174 000000
000176 000000

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

.SBTTL STARTING ADDRESS(ES)

2 000200 000137 004620
3 000204 000137 035506
4 000210 000137 004626
5

JMP @#BEGIN ;:JUMP TO STARTING ADDRESS OF PROGRAM
JMP @#BASECH ;:MODIFY ADDRESSES
JMP @#BEGIN2 ;:SELECT DRIVE START

.SBTTL ACT11 HOOKS

::*****
:HOOKS REQUIRED BY ACT11

000046 000214
000052 000046
000052 030746
000214 000052
000214 020000
000214 000214

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

6

0

.SBTTL COMMON TAGS

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

001100	001100			SCMTAG: .=1100	:: START OF COMMON TAGS
001100	000000			\$PASS: .WORD 0	:: CONTAINS PASS COUNT
001102	000			\$TSTNM: .BYTE 0	:: CONTAINS THE TEST NUMBER
001103	000			\$ERFLG: .BYTE 0	:: CONTAINS ERROR FLAG
001104	000000			\$ICNT: .WORD 0	:: CONTAINS SUBTEST ITERATION COUNT
001106	000000			\$LPADR: .WORD 0	:: CONTAINS SCOPE LOOP ADDRESS
001110	000000			\$LPERR: .WORD 0	:: CONTAINS SCOPE RETURN FOR ERRORS
001112	000000			\$ERTTL: .WORD 0	:: CONTAINS TOTAL ERRORS DETECTED
001114	000			\$ITEMB: .BYTE 0	:: CONTAINS ITEM CONTROL BYTE
001115	001			\$ERMAX: .BYTE 1	:: CONTAINS MAX. ERRORS PER TEST
001116	000000			\$ERRPC: .WORD 0	:: CONTAINS PC OF LAST ERROR INSTRUCTION
001120	000000			\$GDADR: .WORD 0	:: CONTAINS ADDRESS OF 'GOOD' DATA
001122	000000			\$BDADR: .WORD 0	:: CONTAINS ADDRESS OF 'BAD' DATA
001124	000000			\$GDDAT: .WORD 0	:: CONTAINS 'GOOD' DATA
001126	000000			\$BDDAT: .WORD 0	:: CONTAINS 'BAD' DATA
001130	000000			.WORD 0	:: RESERVED--NOT TO BE USED
001132	000000			.WORD 0	
001134	000			\$AUTOB: .BYTE 0	:: AUTOMATIC MODE INDICATOR
001135	000			\$INTAG: .BYTE 0	:: INTERRUPT MODE INDICATOR
001136	000000			.WORD 0	
001140	177570			\$SWR: .WORD DSWR	:: ADDRESS OF SWITCH REGISTER
001142	177570			\$DISPLAY: .WORD DDISP	:: ADDRESS OF DISPLAY REGISTER
001144	177560			\$TKS: 177560	:: TTY KBD STATUS
001146	177562			\$TKB: 177562	:: TTY KBD BUFFER
001150	177564			\$TPS: 177564	:: TTY PRINTER STATUS REG. ADDRESS
001152	177566			\$TPB: 177566	:: TTY PRINTER BUFFER REG. ADDRESS
001154	000			\$NULL: .BYTE 0	:: CONTAINS NULL CHARACTER FOR FILLS
001155	002			\$FILLS: .BYTE 2	:: CONTAINS # OF FILLER CHARACTERS REQUIRED
001156	012			\$FILLC: .BYTE 12	:: INSERT FILL CHARS. AFTER A 'LINE FEED'
001157	000			\$TPFLG: .BYTE 0	:: 'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
001160	000000			\$REGAD: .WORD 0	:: CONTAINS THE ADDRESS FROM WHICH (\$REG0) WAS OBTAINED
001162	000000			\$REG0: .WORD 0	:: CONTAINS ((\$REGAD)+0)
001164	000000			\$REG1: .WORD 0	:: CONTAINS ((\$REGAD)+2)
001166	000000			\$REG2: .WORD 0	:: CONTAINS ((\$REGAD)+4)
001170	000000			\$REG3: .WORD 0	:: CONTAINS ((\$REGAD)+6)
001172	000000			\$REG4: .WORD 0	:: CONTAINS ((\$REGAD)+10)
001174	000000			\$REG5: .WORD 0	:: CONTAINS ((\$REGAD)+12)
001176	000000			\$TMP0: .WORD 0	:: USER DEFINED
001200	000000			\$TMP1: .WORD 0	:: USER DEFINED
001202	000000			\$TMP2: .WORD 0	:: USER DEFINED
001204	000000			\$TMP3: .WORD 0	:: USER DEFINED
001206	000000			\$TMP4: .WORD 0	:: USER DEFINED
001210	000000			\$TMP5: .WORD 0	:: USER DEFINED
001212	000000			\$TIMES: 0	:: MAX. NUMBER OF ITERATIONS
001214	000000			\$ESCAPE: 0	:: ESCAPE ON ERROR ADDRESS
001216	207	377	377	\$BELL: .ASCIZ <207><377><377>	:: CODE FOR BELL
001222	077			\$QUES: .ASCII /?/	:: QUESTION MARK
001223	015			\$CRLF: .ASCII <15>	:: CARRIAGE RETURN
001224	012	000		\$LF: .ASCIZ <12>	:: LINE FEED

;;*****

.SBTTL USER DEFINED TAGS

001226	000254	RPVEC:	254	:RP04/5/6 VECTOR ADDRESS
001230	176722	RHDB:	176722	:DATA BUFFER
001232	176702	RHWC:	176702	:WORD COUNT
001234	176704	RHBA:	176704	:BUS ADDRESS
001236	176710	RHCS2:	176710	:CONTROL AND STATUS
001240	176700	RHCS1:	176700	:CONTROL AND STATUS 1 SEE NOTE ABOVE
001242	176714	RHER1:	176714	:ERROR #1 SEE NOTE ABOVE
001244	176706	RHDST:	176706	:DESIRED SECTOR / TRACK ADDRESS
001246	176740	RHER2:	176740	:ERROR #2
001250	176732	RHOF:	176732	:OFFSET
001252	176734	RHCA:	176734	:DESIRED CYLINDER ADDRESS
001254	176742	RHER3:	176742	:ERROR #3
001256	176716	RHAS:	176716	:ATTENTION SUMMARY SEE NOTE ABOVE
001260	176724	RHMR:	176724	:MAINTAINABILITY
001262	176712	RHDS1:	176712	:DRIVE STATUS
001264	176726	RHDT:	176726	:DRIVE TYPE
001266	176730	RHSN:	176730	:SERIAL NUMBER SEE NOTE ABOVE
001270	176744	RHEC1:	176744	:ECC POSITION
001272	176746	RHEC2:	176746	:ECC PATTERN
001274	176720	RHLA:	176720	:LOOK AHEAD
001276	176736	RHCC:	176736	:CURRENT CYLINDER ADDRESS

:ADDITIONAL REGISTERS LOCATED IN THE RH70 CONTROLLER

001300	176752	RHCS3:	176752	:CONTROL AND STATUS REG #3
001302	176750	RHBAE:	176750	:BUS ADDRESS EXTENSION REGISTER

:THE FOLLOWING LOCATIONS ARE RESERVED FOR REGISTER SAVES
:ANY TIME THERE IS AN ERROR ALL THESE WILL BE FILLED
:ONLY SOME MAY BE PRINTED BUT ALL WILL BE FILLED TRUE
:FOR THE TIME JUST AFTER THE 'ERROR' ERROR COMMAND

001304	000000	DB:	0	:DATA BUFFER
001306	000000	WC:	0	:WORD COUNT
001310	000000	BA:	0	:BUS ADDRESS
001312	000000	CS2:	0	:CONTROL AND STATUS 2
001314	000000	CS1:	0	:CONTROL AND STATUS 1
001316	000000	ER1:	0	:ERROR #1
001320	000000	DST:	0	:DESIRED SECTOR/TRACK ADDRESS
001322	000000	ER2:	0	:ERROR #2
001324	000000	OF:	0	:OFFSET
001326	000000	CA:	0	:DESIRED CYLINDER ADDRESS
001330	000000	ER3:	0	:ERROR #3
001332	000000	AS:	0	:ATTENTION SUMMARY
001334	000000	MR:	0	:MAINTAINABILITY
001336	000000	DS1:	0	:DRIVE STATUS
001340	000000	DT:	0	:DRIVE TYPE
001342	000000	SN:	0	:SERIAL NUMBER
001344	000000	EC1:	0	:ECC POSITION
001346	000000	EC2:	0	:ECC PATTERN
001350	000000	LA:	0	:LOOK-AHEAD
001352	000000	CC:	0	:CURRENT CYLINDER ADDRESS

;FLAGS AND INTERNAL PROGRAM CONTROL WORDS

001354		UNITS: .BLKW 8.	:THIS IS FILLED WITH -1
001374	000000	UNIT: .WORD 0	:UNIT UNDER TEST
001376	000000	NOUNIT: .WORD 0	:NUMBER OF UNITS PRESENT
001400	000000	NUNIT: .WORD 0	:USED TO KEEP TRACK OF UNIT UNDER TEST
001402	000000	SELECT: .WORD 0	:USED TO DETERMIN IF THERE ARE MORE
001404	000000	UNITSL: .WORD 0	:THAN ONE UNIT
001406	000000	ERFLG\$: 0	:ALL ONES INDICATE UNIT TO BE SELECTED
001410	000000	SAVDT: 0	:UNIT NO. SELECTED
001412	000000	SAVSN: 0	:ERROR FLAG
001414	000000	PCJSR: 0	:SAVE DRIVE TYPE REGISTER
001416	000000	ATTENT: 0	:FOR COMPARISON IN DRIVE CLEAR TEST
001420	000000	TOTALAT: 0	:AND RH INIT TEST
001422	000000	TMPILL: 0	:SAVE SERIAL NUMBER REGISTER
001424	000000	TSECC: 0	:FOR COMPARISON IN DRIVE CLEAR TEST
001426	000000	TESDTE: 0	:AND RH INIT TEST
001430	000000	TAGDTE: 0	:SAVE PC OF JSR WHICH GAVE THE ERROR
			:ATTENTION BIT FOR PRESENT UNIT
			:TOTAL ATTENTION BITS
			:TEMPORARY ILLEGAL FUNCTION
			:FLAG TO SAY IF ECC TEST OR NOT
			:WHEN =177777 IT IS AN ECC TEST
			:WHEN =0IT IS NOT AN ECC TEST
			:FLAG TO SAY IF DRIVE TIMING ERROR OR NOT
			:WHEN = 177777 IT IS A DTE TEST
			:WHEN = 0 IT IS NOT A DTE TEST
			:TEMPORARY TAG USED IN DRIVE TIMING
			:ERROR TEST

;FUNCTION EQUATES

;TABLE OF COMMAND FUNCTIONS FOR RHCS1
 ;THEN "GO" BIT HAS TO BE SET

001432		FUTABL:	
001432	000000	NOPEA: 0	:NO OPERATION
001434	000002	UNLOAD: 2	:UNLOAD (STAND BY)
001436	000006	RECALI: 6	:RECALIBRATE
001440	000010	DCLEAR: 10	:DRIVE CLEAR
001442	000012	RELEAS: 12	:RELEASE (DUAL-PORT OPERATION)
001444	000030	SERCH: 30	:SEARCH COMMAND
001446	000050	WRCHK: 50	:WRITE CHECK DATA
001450	000052	WRCHDT: 52	:WRITE CHECK HEADER AND DATA
001452	000060	WRIDAT: 60	:WRITE DATA
001454	000062	WRIFOR: 62	:WRITE HEADER AND DATA (FORMAT)
001456	000070	READAT: 70	:READ DATA
001460	000072	REFOR: 72	:READ HEADER AND DATA
001462	000004	SEECOM: 4	:SEEK COMMAND
001464	000014	OFSETC: 14	:OFFSET COMMAND
001466	000016	RETCL: 16	:RETURN TO CENTERLINE
001470	000022	PKACK: 22	:PACK ACKNOWLEDGE
001472	000020	READIN: 20	:READ IN
001474	000000	ILLEGL: .WORD	:COMPUTED ILLEGAL FUNCTION

;DATA BUFFER FOR READ WRITE


```
001500      WRFROM: .BLKW 274.      :WRITE FROM THIS BUFFER
002544      REINTO: .BLKW 274.      :READ INTO THIS BUFFER

003610 000000 TSTNM: 0          :TEST NUMBER
003612 000000 FIRST: 0         :IF ZERO WILL TYPE HEADER
                                :IF ONES WILL NOT TYPE HEADER
003614 000000 RH70: 0          :FLAG = 1 FOR RH70 CONTROLLER
003616 000000 SILOSZ: .WORD 0       :FLAG = 0 FOR RH11
                                :RH SILO SIZE

                                ;TABLE FOR ATTENTION BITS ATTENTION TABLE
003620      001      002      004 ATABLE: .BYTE 1,2,4,10,20,40,100,200
```

.SBTTL ERROR POINTER TABLE

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

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

1	003630		\$ERRTB:		
2			:ERROR 1		
3					
4	003630	047474	EM1		:WRONG DATA IN READING OR WRITING HARDWARE REGISTER
5	003632	053655	DH1		:PC
6					:REG. ADDR.
7					:GOOD DATA
8					:RECEIVED DATA
9	003634	060144	DT1		:\$ERRPC,\$STSTNM,REGADR,\$GDDAT,\$BDDAT
10	003636	060676	DF1		:0,0,0,0,0
11					
12			:ERROR 2		
13					
14	003640	047557	EM2		:ERROR ON DATA COMMAND
15					
16	003642	056650	DH33		:PC
17					:PC OF JSR
18					:TEST NO
19					:WORD NO.
20					:GOOD DATA
21					:CONTENTS OF RHCS1
22					:CONTENTS OF RHDS1
23					:CONTENTS OF RHER1
24	003644	060530	DT33		:\$ERRPC,PCJSR,\$STSTNM,ERWORD,\$GDDAT,CS1,DS1,ER1
25	003646	061046	DF33		:0,0,0,1,0,0,0,0
26					
27			:ERROR 3		
28					
29	003650	047557	EM2		:ERROR ON DATA COMMAND
30					
31	003652	056434	DH32		:PC
32					:PC OF JSR
33					:TEST NO
34					:WORD NO.
35					:GOOD DATA
36					:BAD DATA
37					:CONTENTS OF RHCS1
38					:CONTENTS OF RHDS1
39					:CONTENTS OF RHER1
40					
41	003654	060504	DT32		:\$ERRPC,PCJSR,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
42	003656	061035	DF32		:0,0,0,1,0,0,0,0,0

43				
44			:ERROR 4	
45				
46	003660	047557	EM2	:ERROR ON DATA COMMAND
47				
48	003662	056237	DH31	:PC
49				:TEST NO
50				:WORD NO.
51				:GOOD DATA
52				:BAD DATA
53				:CONTENTS OF RHCS1
54				:CONTENTS OF RHDS1
55				:CONTENTS OF RHER1
56				
57	003664	060462	DT31	:\$ERRPC,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
58	003666	061025	DF31	:0,0,1,0,0,0,0,0,
59				
60			:ERROR 5	
61				
62	003670	000000	0	:
63	003672	000000	0	:
64	003674	060462	DT31	:\$ERRPC,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
65	003676	061025	DF31	:0,0,1,0,0,0,0,0,
66				
67			:ERROR 6	
68				
69	003700	047606	EM6	:ERROR ON WRITE HEADER AND DATA
70				
71	003702	056434	DH32	:PC
72				:PC OF JSR
73				:TEST NO
74				:WORD NO.
75				:GOOD DATA
76				:BAD DATA
77				:CONTENTS OF RHCS1
78				:CONTENTS OF RHDS1
79				:CONTENTS OF RHER1
80				
81	003704	060504	DT32	:\$ERRPC,PCJSR,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
82	003706	061035	DF32	:0,0,0,1,0,0,0,0,0,
83				
84			:ERROR 7	
85				
86	003710	047606	EM6	:ERROR ON WRITE HEADER AND DATA
87	003712	053772	DH2	:PC
88				:TEST NO
89				:WORD NO.
90				:GOOD DATA
91				:BAD DATA
92	003714	060172	DT3	:\$ERRPC,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT
93	003716	060707	DF3	:0,0,1,0,0,
94				
95			:ERROR 10	
96				
97	003720	000000	0	:
98	003722	000000	0	:
99	003724	060172	DT3	:\$ERRPC,\$STSTNM,ERWORD,\$GDDAT,\$BDDAT

100	003726	060707	DF3	:0,0,1,0,0,
101				
102			:ERROR 11	
103				
104	003730	047645	EM11	:CONTROLLER OR DRIVE STATUS
105	003732	054103	DH11	:PC
106				:TEST NO
107				:FAILING REG. ADDR
108				:CONTENTS OF RHCS1
109				:CONTENTS OF RHCS2
110				:CONTENTS OF RHDS1
111				:CONTENTS OF RHER1
112	003734	060206	DT11	:\$ERRPC,\$TSTNM,\$BDADR,CS1,CS2,DS1,ER1
113	003736	060714	DF11	:0,0,0,0,0,0
114				
115			:ERROR 12	
116				
117	003740	047645	EM11	:WRONG DATA FROM SILO
118				
119	003742	053655	DH1	:PC
120				:REG. ADDR
121				:GOOD DATA
122				:RECEIVED DATA
123	003744	060144	DT1	:\$ERRPC,REGADR,\$GDDAT,\$BDDAT
124	003746	060676	DF1	:0,0,0,0
125				
126			:ERROR 13	
127				
128	003750	000000	0	
129	003752	000000	0	
130	003754	060144	DT1	:\$ERRPC,TSTNM,REGADR,\$GDDAT,\$BDDAT
131	003756	060676	DF1	:0,0,0,0,0
132				
133			:ERROR 14	
134				
135	003760	047700	EM14	:REGISTER FAILED
136	003762	054257	DH14	:PC
137				:FAILING REG. ADDR
138				:CONTENTS OF FAILING REG.
139				:CONTENTS OF RHCS1
140				:CONTENTS OF RHCS2
141				:CONTENTS OF RHDS1
142				:CONTENTS OF RHER1
143	003764	060226	DT14	:\$ERRPC,\$BDADR,\$BDDAT,CS1,CS2,DS1,ER1
144	003766	060723	DF14	:0,0,0,0,0,0,0
145				
146			:ERROR 15	
147				
148	003770	047720	EM15	:SPECIFIED REG. NON EXISTANT SO ABORT
149				:PROGRAM
150	003772	054455	DH15	:PC
151				:ADDR. OF REG
152	003774	060250	DT15	:\$ERRPC,TMP1
153	003776	060733	DF15	:0,0
154				
155			:ERROR 16	
156				

157	004000	047770		EM16	:WAIT LOOP FAILED
158	004002	054505		DH16	:PC
159					:WAT PC
160					:BIT WANTED
161					:REG. ADR.
162					:REG. CONT.
163	004004	060260		DT16	:SERRPC,\$TMP3,\$TMP1,\$TMP0,\$BDDAT
164	004006	060736		DF16	:0,0,0,0
165					
166					:ERROR 17
167					
168	004010	050011		EM17	:WRITE CHECK FAILING
169	004012	054642		DH17	:PC
170					:TEST NO
171					:CONTENTS OF RHBA
172					:CONTENTS OF RHDB
173					:CONTENTS OF RHWC
174					:CONTENTS OF RHCS1
175					:CONTENTS OF RHCS2
176	004014	060276		DT17	:SERRPC,\$TSTNM,\$BA,\$DB,\$WC,\$CS1,\$CS2
177	004016	060743		DF17	:0,0,0,0,0,0,0
178					
179					:ERROR 20
180					
181	004020	050035		EM20	:REGISTER FAILING
182	004022	055016		DH20	:PC
183					:TST NO
184					:CONTENTS OF RHER1
185					:CONTENTS OF RHER2
186					:CONTENTS OF RHER3
187					:CONTENTS OF RHAS
188					:CONTENTS OF RHDS1
189	004024	060316		DT20	:SERRPC,\$TSTNM,\$ER1,\$ER2,\$ER3,\$AS,\$DS1
190	004026	060752		DF20	:0,0,0,0,0,0,0
191					
192					:ERROR 21
193					
194	004030	050056		EM21	:INTERRUPT FAILING
195	004032	055171		DH21	:PC
196					:TEST NO
197					:CONTENTS OF RHCS1
198					:CONTENTS OF RHAS
199					:CONTENTS OF RHDS1
200	004034	060336		DT21	:SERRPC,\$TSTNM,\$CS1,\$AS,\$DS1
201	004036	060761		DF21	:0,0,0,0,0
202					
203					:ERROR 22
204					
205	004040	050100		EM22	:MISMATCH IN DRIVE PRESENT
206					:LOOKING AT RHAS AND RHCS2-NED(BIT#12)
207					:DRIVE PRESENT DO NOT AGREE
208					:NOTE: ON DUAL PORT SYSTEM
209					:DRIVE ON OTHER PORT WILL NOT GIVE NED
210					:HENCE THERE WILL BE A MISMATCH
211					:17777-MEANS NOT PRESENT
212	004042	055304		DH22	:PC
213					:TEST NO

214				:RHAS UNIT
215				:RHCS2 UNIT
216				:
217	004044	060352	DT22	:\$ERRPC,TSTNMS,\$GDDAT,\$BDDAT
218	004046	060766	DF22	:0,0,0,0
219				:
220			:ERROR 23	
221	004050	000000	0	:MISSMATCH IN DRIVE PRESENT
222				:LOOKING AT RHAS AND RHCS2-NED(BIT#12)
223				:DRIVE PRESENT DO NOT AGREE
224				:177777-MEANS NOT PRESENT
225	004052	000000	0	:PC
226				:TEST NO
227				:RHAS UNIT
228				:RHCS2 UNIT
229				:
230	004054	060352	DT22	:\$ERRPC,TSTNMS,\$GDDAT,\$BDDAT
231	004056	060766	DF22	:0,0,0,0
232				:
233			:ERROR 24	
234				
235	004060	050473	EM24	:LOOK AHEAD REGISTER AT THE
236				:BEGINNING OF A SECTOR IS IN
237				:ERROR
238	004062	055377	DH24	:PC
239				:RHDST
240				:BAD RHLA
241				:GOOD RHLA
242				:SECTOR NO
243				:SECTOR CLOCK
244	004064	060364	DT24	:\$ERRPC,DST,\$BDDAT,\$TMP1,\$TMP2,\$TMP3
245	004066	060772	DF24	:0,0,0,0,0
246				:
247			:ERROR 25	
248				
249	004070	050566	EM25	:LOOK AHEAD REGISTER IS
250				:IN ERROR
251				:
252	004072	055377	DH24	:PC
253				:RHDST
254				:BAD RHLA
255				:GOOD RHLA
256				:SECTOR NO
257				:SECTOR CLOCK
258	004074	060364	DT24	:\$ERRPC,DST,\$BDDAT,\$TMP1,\$TMP2,\$TMP3
259	004076	060772	DF24	:0,0,0,0,0
260				:
261			:ERROR 26	
262				
263	004100	047645	EM11	:CONTROLLER OR DRIVE STATUS
264				:
265	004102	055554	DH26	:PC
266				:PC OF JSR
267				:FAILING REGISTER ADDRESS
268				:CONTENTS OF RHCS1
269				:CONTENTS OF RHCS2
270				:CONTENTS OF RHDS1

271				:CONTENTS OF RHER1
272				
273	004104	060404	DT26	:\$ERRPC,PCJSR,\$BDADR,CS1,CS2,DS1,ER1
274	004106	061001	DF26	:0,0,0,0,0,0,
275				
276			:ERROR 27	
277				
278	004110	047474	EM1	:ERROR IN READING OR WRITING HARDWARE REGISTER
279				
280	004112	055751	DH27	:PC
281				:PC OF JSR
282				:TEST NUMBER
283				:FAILING REGISTER
284				:GOOD DATA
285				:RECEIVED DATA
286				
287	004114	060426	DT27	:\$ERRPC,PCJSR,TSTNM,REGADR,\$GDDAT,\$BDDAT
288	004116	061011	DF27	:0,0,0,0,0,0
289				
290			:ERROR 30	
291				
292	004120	050626	EM30	:CURRENT CYLINDER DOES NOT REFLECT DESIRED CYLINDER REG.
293	004122	056106	DH30	:PC
294				:PC OF JSR
295				:REGISTER ADDRESS
296				:GOOD DATA
297				:BAD DATA
298				
299	004124	060444	DT30	:\$ERRPC,PCJSR,REGADR,\$GDDAT,\$BDDAT
300	004126	061017	DF30	:0,0,0,0,0
301				
302			:ERROR 31	
303				
304	004130	050747	EM31	:ECC GENERATED IS INCORRECT
305				:EVERY WORD IN THIS SECTOR IS GIVEN IN 'DATA USED'
306				
307	004132	057046	DH34	:PC
308				:TEST NUMBER
309				:GOOD ECC1
310				:GOOD EC2C
311				:WRITTEN ECC1
312				:WRITTEN ECC2
313				:DATA USED
314				
315	004134	060552	DT34	:\$ERRPC,TSTNM,GECC1,GECC2,WECC1,WECC2,DISK
316				
317	004136	061056	DF34	:0,0,0,0,0,0,0
318				
319			:ERROR 32	
320				
321	004140	051071	EM32	:ON READ COMMAND AFTER DATA AND ECC HAVE BEEN READ
322				:ECC REGISTER OR RHER1 IS IN ERROR
323				:ONLY LOWER 11 BITS OF PATTERN REGISTER
324				:CAN BE READ
325				:THIS SHOULD MATCH LOWER 11 BITS OF ECC1
326				
327	004142	057220	DH35	:PC

328					: TEST NUMBER
329					: GOOD ECC1
330					: GOOD ECC2
331					: PATTERN REGISTER
332					: RHER1
333					
334	004144	060572	DT35		: \$ERRPC, TSTNM, GECC1, GECC2, EC2, ER1
335					
336	004146	061065	DF35		: 0,0,0,0,0,0
337					
338				: ERROR 33	
339					
340	004150	051355	EM33		: HIGH COUNT BIT NOT HIGH AFTER 38859 CLOCKS
341	004152	057414	DH36		: PC
342					: PC OF JSR
343					: TEST NUMBER
344					: RHMR
345					: POSITION REG.
346					: PATTERN REGISTER
347					
348	004154	060614	DT36		: \$ERRPC, PCJSR, TSTNM, MR, EC1, EC2
349					
350	004156	061075	DF36		: 0,0,0,0,0,0
351					
352				: ERROR 34	
353					
354	004160	051427	EM34		: ZERO DETECT BIT NOT HIGH WHEN THE
355					: 32 BIT ECC REGISTER HAS ITS 21 BITS
356					: OF ZEROS
357					: ERROR PRINTOUT WILL CONTINUE TILL
358					: ZERO DETECT BIT IS HIGH
359	004162	057414	DH36		: PC
360					: PC OF JSR
361					: TEST NUMBER
362					: RHMR
363					: POSITION REG.
364					: PATTERN REGISTER
365					
366	004164	060614	DT36		: \$ERRPC, PCJSR, TSTNM, MR, EC1, EC2
367					
368	004166	061075	DF36		: 0,0,0,0,0,0
369					
370				: ERROR 35	
371					
372	004170	051522	EM35		: POSITION REGISTER OR 11 BITS OF
373					: PATTERN REGISTER INCORRECT
374					: LOWER 11 BITS OF PATTERN REGISTER
375					: SHOULD MATCH LOWER 11 BITS OF GOOD ECC1
376					: DATA ENVELOPE AND N-CODE ZEROS ARE IN DECIMAL
377					
378	004172	057551	DH37		: PC
379					: TEST NUMBER
380					: ECC POSITION
381					: GOOD POSITION
382					: GOOD ECC1
383					: GOOD ECC2
384					: ECC PATTERN

385				:DATA ENVELOPE
386				:N-CODE ZEROS
387				
388	004174	060632	DT37	:\$ERRPC,TSTNM,EC1,POSITI,GECC1,GECC2,EC2,DATENV,ZCODE
389				
390	004176	061103	DF37	:0,0,0,0,0,0,0,0,0
391				
392				:ERROR 36
393				
394	004200	052016	EM36	:ON A READ COMMAND WITH NON CORRECTABLE
395				:ERROR INSERTED DCK AND ECH SHOULD BE SET
396	004202	057220	DH35	:PC
397				:TEST NUMBER
398				:GOOD ECC1
399				:GOOD ECC2
400				:PATTERN REGISTER
401				:POSITION REGISTER
402				:RHER1
403				
404	004204	060572	DT35	:\$ERRPC,TSTNM,GECC1,GECC2,EC2,EC1,ER1
405				
406	004206	061065	DF35	:0,0,0,0,0,0,0,0
407				
408				:ERROR 37
409				
410	004210	052203	EM37	:ERROR ON DATA COMMAND
411				:WITH A16 A17 USED
412				
413	004212	056237	DH31	:PC
414				:TEST NO
415				:WORD NO.
416				:GOOD DATA
417				:BAD DATA
418				:CONTENTS OF RHCS1
419				:CONTENTS OF RHDS1
420				:CONTENTS OF RHER1
421				
422	004214	060462	DT31	:\$ERRPC,\$TSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
423	004216	061025	DF31	:0,0,1,0,0,0,0,0,
424				
425				:ERROR 40
426				
427	004220	000000	0	:
428	004222	000000	0	:
429	004224	060462	DT31	:\$ERRPC,\$TSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1
430	004226	061025	DF31	:0,0,1,0,0,0,0,0,
431				
432				:ERROR 41
433				
434	004230	052271	EM40	:THERE WAS A READ/WRITE HEADER & DATA
435				:ERROR DURING 'DTE' TEST SETUP - THE
436				:TEST IS ABORTED AT THAT POINT
437	004232	057767	DH40	:PC
438				:TEST NO
439				:FAILING REGISTER ADDRESS
440				:CONTENTS OF RHCS1
441				:CONTENTS OF RHCS2

442				:CONTENTS OF RHDS1
443				:CONTENTS OF RHER1
444	004234	060656	DT40	:\$ERRPC,\$TSTNM,\$BDADR,CS1,CS2,DS1,ER1
445	004236	061114	DF40	:0,0,0,0,0,0
446				
447				
448				:ERROR 42
449	004240	061124	EM42	
450	004242	062570	DH42	
451	004244	063046	DT42	:ERROR PC,TEST NUMBER, REGISTER ADDRESS.
452	004246	063104	DF42	
453				
454				:ERROR 43
455				
456	004250	061207	EM43	
457	004252	062570	DH42	
458	004254	063046	DT42	:ERROR PC,TEST NUMBER, REGISTER ADDRESS.
459	004256	063104	DF42	
460				
461				:ERROR 44
462				
463	004260	061253	EM44	
464	004262	062650	DH44	
465	004264	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
466	004266	063107	DF44	
467				
468				:ERROR 45
469				
470	004270	061310	EM45	
471	004272	062650	DH44	
472	004274	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
473	004276	063107	DF44	
474				
475				:ERROR 46
476				
477	004300	061337	EM46	
478	004302	062650	DH44	
479	004304	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
480	004306	063107	DF44	
481				
482				:ERROR 47
483				
484	004310	061366	EM47	
485	004312	062570	DH42	
486	004314	063046	DT42	:ERROR PC,TEST NUMBER, REGISTER ADDRESS.
487	004316	063104	DF42	
488				
489				:ERROR 50
490				
491	004320	061423	EM50	
492	004322	062650	DH44	
493	004324	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
494	004326	063107	DF44	
495				
496				:ERROR 51
497				
498	004330	061461	EM51	

499	004332	062650	DH44	
500	004334	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
501	004336	063107	DF44	
502				
503				:ERROR 52
504				
505	004340	061502	EM52	
506	004342	062650	DH44	
507	004344	063056	DT44	
508	004346	063107	DF44	
509				
510				:ERROR 53
511				
512	004350	061542	EM53	
513	004352	062650	DH44	
514	004354	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
515	004356	063107	DF44	
516				
517				:ERROR 54
518				
519	004360	061602	EM54	
520	004362	062761	DH54	
521	004364	063072	DT54	:ERROR PC, TEST NUMBER.
522	004366	063114	DF54	
523				
524				:ERROR 55
525				
526	004370	061641	EM55	
527	004372	062650	DH44	
528	004374	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
529	004376	063107	DF44	
530				
531				:ERROR 56
532				
533	004400	061654	EM56	
534	004402	062650	DH44	
535	004404	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
536	004406	063107	DF44	
537				
538				:ERROR 57
539				
540	004410	061671	EM57	
541	004412	062650	DH44	
542	004414	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
543	004416	063107	DF44	
544				
545				:ERROR 60
546				
547	004420	061720	EM60	
548	004422	062650	DH44	
549	004424	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
550	004426	063107	DF44	
551				
552				:ERROR 61
553				
554	004430	062007	EM61	
555	004432	062650	DH44	

556	004434	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
557	004436	063107	DF44	
558				
559				:ERROR 62
560				
561	004440	062057	EM62	
562	004442	062650	DH44	
563	004444	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
564	004446	063107	DF44	
565				
566				:ERROR 63
567				
568	004450	062134	EM63	
569	004452	062650	DH44	
570	004454	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
571	004456	063107	DF44	
572				
573				:ERROR 64
574				
575	004460	062212	EM64	
576	004462	062650	DH44	
577	004464	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
578	004466	063107	DF44	
579				
580				:ERROR 65
581				
582	004470	062246	EM65	
583	004472	062650	DH44	
584	004474	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
585	004476	063107	DF44	
586				
587				:ERROR 66
588				
589	004500	062330	EM66	
590	004502	062650	DH44	
591	004504	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.
592	004506	063107	DF44	
593				
594				:ERROR 67
595				
596	004510	062402	EM67	
597	004512	062761	DH54	
598	004514	063072	DT54	:ERROR PC, TEST NUMBER.
599	004516	063114	DF54	
600				
601				:ERROR 70
602				
603	004520	062467	EM70	
604	004522	062761	DH54	
605	004524	063072	DT54	:ERROR PC, TEST NUMBER.
606	004526	063114	DF54	
607				
608				:ERROR 71
609				
610	004530	062541	EM71	
611	004532	062650	DH44	
612	004534	063056	DT44	:ERROR PC,TEST NUMBER, REGISTER ADDRESS CORRECT, ACTUAL.

613 004536 063107

DF44

ERROR POINTER TABLE

```

1      ;THIS ROUTINE HANDLES UNEXPECTED TIMEOUTS
2
3      004540 011600      BADTMO: MOV      (SP),R0      ;SAVE PC WHERE THE TIME OUT OCCURED
4      004542 005740      TST      -(R0)      ;ADJUST PC -2
5      004544 022626      CMP      (SP)+,(SP)+  ;RESTORE STACK POINTER
6      004546 104401 004554  TYPE      ,65$      ;:TYPE ASCIZ STRING
        004552 000417      BR       64$      ;:GET OVER THE ASCIZ
        ;:65$: .ASCIZ <CRLF>/UNEXPECTED BUS TIMEOUT, PC=/
        64$:
7      004612 010046      MOV      R0,-(SP)    ;SETUP FOR TYPING OUT PC
8      004614 104402      TYPOC
9      004616 000240      NOP
        ;PUT 'HALT(0)' INSTRUCTION HERE IF YOU WISH
        ;TO STOP ON UNEXPECTED TIMEOUT.
10
11
12     .SBTTL  START OF PROGRAM
13
14     004620 005037 001402  BEGIN:  CLR      SELECT      ;DO NOT SELECT UNIT, NORMAL RUN
15     004624 000403      BR       START
16
17     004626 012737 177777 001402  BEGIN2: MOV      #-1,SELECT  ;SELECT UNIT
18
19     004634 005227 000000  START:  INC      #0      ;TTY LOOP, WAIT FOR INCREMENT
20     004640 001375      BNE     -4      ;OF WORD
21     004642 000005      RESET   ;RESET THE WORLD
22
23     .SBTTL  INITIALIZE THE COMMON TAGS
        ;:CLEAR THE COMMON TAGS ($CMTAG) AREA
        MOV      #$CMTAG,R6  ;:FIRST LOCATION TO BE CLEARED
        CLR      (R6)+      ;:CLEAR MEMORY LOCATION
        CMP      #SWR,R6    ;:DONE?
        BNE     -6      ;:LOOP BACK IF NO
        MOV      #STACK,SP  ;:SETUP THE STACK POINTER
        ;:INITIALIZE A FEW VECTORS
        MOV      #$$SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
        MOV      #340,@#IOTVEC+2 ;:LEVEL 7
        MOV      #ERROR,@#EMTVEC  ;:EMT VECTOR FOR ERROR ROUTINE
        MOV      #340,@#EMTVEC+2 ;:LEVEL 7
        MOV      #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
        MOV      #340,@#TRAPVEC+2 ;:LEVEL 7
        MOV      #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
        MOV      #340,@#PWRVEC+2 ;:LEVEL 7
        CLR      $TIMES      ;:INITIALIZE NUMBER OF ITERATIONS
        CLR      $ESCAPE     ;:CLEAR THE ESCAPE ON ERROR ADDRESS
        MOVB    #1,$ERMAX    ;:ALLOW ONE ERROR PER TEST
        MOV      #.,$LPADR   ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
        MOV      #.,$LPERR   ;:SETUP THE ERROR LOOP ADDRESS
        ;:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
        ;:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
        MOV      @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
        MOV      #64$,@#ERRVEC ;:SET UP ERROR VECTOR
        MOV      #DSWR,SWR   ;:SETUP FOR A HARDWARE SWICH REGISTER
        MOV      #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
        CMP      #-1,@SWR    ;:TRY TO REFERENCE HARDWARE SWR
        BNE     66$      ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
        ;:AND THE HARDWARE SWR IS NOT = -1
        BR      65$      ;:BRANCH IF NO TIMEOUT
        64$:  MOV      #65$, (SP) ;:SET UP FOR TRAP RETURN
    
```



```

005042 000002          RTI
005044 012737 000176 001140 65$:  MOV    #SWREG,SWR      ;;POINT TO SOFTWARE SWR
005052 012737 000174 001142          MOV    #DISPRÉG,DISPLAY
005060 012637 000004          66$:  MOV    (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

24          ;SETUP "TIMEOUT" TRAP VECTOR FOR UNEXPECTED BUS TIMEOUTS
25 005064 012737 004540 000004          MOV    #BADTMO,ERRVEC  ;;SETUP FOR UNEXPECTED TIMEOUT
26 005072 012737 000300 000006          MOV    #PR6,ERRVEC+2  ;;LEVEL 6
27
31          .SBTTL  TYPE PROGRAM NAME
          ;;TYPE  THE NAME OF THE PROGRAM IF FIRST PASS
005100 005227 177777          INC    #-1              ;;FIRST TIME?
005104 001030          BNE    67$              ;;BRANCH IF NO
005106 104401 005114          TYPE  ,68$              ;;TYPE ASCIZ STRING
005112 000425          BR     67$              ;;GET OVER THE ASCIZ
          ;;68$:  .ASCIZ <CRLF>@CZRJHEG - RP04/5/6 DISKLESS TEST, PT 2@<CRLF>
          67$:
          .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
005166 005737 000042          TST    @#42              ;;ARE WE RUNNING UNDER XXDP/ACT?
005172 001006          BNE    69$              ;;BRANCH IF YES
005174 023727 001140 000176          CMP    SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
005202 001005          BNE    70$              ;;BRANCH IF NO
005204 104406          GTSWR                    ;;GET SOFT-SWR SETTINGS
005206 000403          BR     70$
005210 112737 000001 001134 69$:  MOVB  #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
005216          70$:
32 005216 005227 177777          INC    #-1              ;;FIRST TIME THRU ?
33 005222 001002          BNE    1$              ;;BR IF NO
34 005224 104401 052441          TYPE  ,PREAMB          ;;TYPE PRE-AMBLE MESSAGE
35 005230 005037 177776          1$:  CLR    PS              ;;SET PROCESSOR STATUS TO 0
36 005234 012777 043342 173764          MOV    #RPVECT,@RPVEC  ;;THIS IS FOR UNTIMELY DRIVE INTERRUPTS
37 005242 004737 044620          JSR    PC,$TKINT       ;;INITIALIZE THE TTY KEYBOARD
38
39 005246 032777 010000 173664 RH70CK: BIT    #SW12,@SWR      ;;LOOK TO SEE IF USING RH70
40 005254 001403          BEQ    1$              ;;IF SW12 = 0, SKIP NEXT
41 005256 012737 000001 003614          MOV    #1,RH70         ;;IF SW12 = 1, CU IS AN RH70
42
43 005264 005737 001402          1$:  TST    SELECT          ;;200 START?
44 005270 001434          BEQ    TST1            ;;GO TO FIRST TEST IF STARTING FROM 200
45 005272 104401 005300          TYPE  ,65$              ;;TYPE ASCIZ STRING
          64$:  BR     64$              ;;GET OVER THE ASCIZ
          ;;65$:  .ASCIZ <CRLF>/SELECT UNIT NUMBER TO BE TESTED ?/
          64$:
46 005344          RDOCT
47 005346 104412          BIC    #177770,(SP)    ;;ONLY KEEP LAST 3 BITS
48 005352 042716 177770          MOV    (SP),UNIT      ;;SAVE UNIT TO BE TESTED
49 005356 011637 001374          MOV    (SP)+,UNITSL   ;;SAVE UNIT TO BE TESTED
          001404
    
```

4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
 21
 22
 23
 24
 25
 26
 27
 28
 29
 30
 31
 32
 33
 34
 35
 36
 37
 38
 39
 40
 41
 42
 43
 005724

```

*****
*TEST 1 REFERENCE EACH REGISTER
*REFERENCE EACH REGISTER BY A MOVE INSTRUCTION
*****
TST1: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
MOV #STACK,SP ;SET UP STACK POINTER
MOV #1,TSTNM ;MOVE #1 TO TEST NUMBER
MOV #REGSA1,EMTVEC ;ERROR VECTOR SO THAT NO REGISTERS ARE SAVED
MOV ERRVEC,-(SP) ;PUSH ERRVEC ON STACK
MOV #2$,ERRVEC ;SETUP FOR BUS TIMEOUT
MOV #24,R0 ;THERE ARE 24 REG TO TEST
MOV #RHDB,R1 ;R1 NOW HAS ADDR OF ADDR OF FIRST REG.
1$: MOV @R1+,R2 ;READ HARDWARE REG.
DEC R0 ;COUNT DOWN
BNE 1$ ;BRANCH IF 24 NOT DONE
MOV (SP)+,ERRVEC ;POP STACK INTO ERRVEC
BR 5$ ;BRANCH IF 24 DONE
2$: MOV #3$, (SP) ;SETUP RETURN ADDRESS
RTI
3$: MOV (SP)+,ERRVEC ;POP STACK INTO ERRVEC
MOV -2(R1),$TMP1 ;STORE FAILING REG ADDR
EMT 15
BIT #SW13,@SWR ;INHIBIT ERROR PRINTOUT ?
BNE 4$ ;BRANCH IF YES
TYPE ,65$ ;TYPE ASCIZ STRING
BR 64$ ;GET OVER THE ASCIZ
65$: .ASCIZ <CRLF>/TO CHANGE BASE ADDRESS, RESTART AT ADDRESS /
64$: MOV #204,-(SP) ;GET READY TO TYPE STARTING ADDRESS
;OF "CHANGE OF BASE ADDRESS" ROUTINE
TYPOC
HALT
4$: JMP $EOP ;FORCE THE RESTART!
;GO TO END OF PROGRAM
5$: MOV ERRVEC,-(SP) ;PUSH ERRVEC ON STACK
MOV #6$,ERRVEC ;INITIALIZE VECTOR
CLR RH70 ;INIT RH INDICATOR ++ C.W
TST @RHBAE ;ADDRESS RPBAE (RH11/RH70?)
INC RH70 ;FOUND AN RH70-SET MASK
TYPE ,67$ ;TYPE ASCIZ STRING
BR 66$ ;GET OVER THE ASCIZ
67$: .ASCIZ <CRLF><LF>/RH70 CONTROLLER /
66$: BR 8$
6$: MOV #7$, (SP) ;SETUP RETURN ADDRESS
RTI
7$: TYPE ,69$ ;TYPE ASCIZ STRING
BR 68$ ;GET OVER THE ASCIZ
69$: .ASCIZ <CRLF><LF>/RH11 CONTROLLER /
68$:

```



```

44 005724 012737 046072 000030 8$:   MOV   # $ERROR, EMTVEC ;RESTORE ERROR VECTOR SO REGISTERS ARE SAVED
45 005732 012637 000004          MOV   (SP)+, ERRVEC  ;;POP STACK INTO ERRVEC
46
47                                     ;FIND THE SILO SIZE
48                                     ;IF ITS A RH70C MODIFY TESTS 50 AND 51
49
50 005736 004737 032054          JSR   PC, CLDISK      ;CONTROLLER CLEAR
51 005742 005037 003616          CLR   SILOSZ         ;CLEAR SILO COUNTER
52 005746 013777 003616 173254 9$:   MOV   SILOSZ, @RHDB  ;LOAD SILO
53 005754 005237 003616          INC   SILOSZ         ;KEEP COUNT
54 005760 032777 000100 173250  BIT   #IR, @RHCS2    ;IS THE SILO FULL?
55 005766 001367          BNE   9$             ;BRANCH IF NO
56 005770 022737 000406 003616  CMP   #262., SILOSZ ;RH70C?
57 005776 001031          BNE   10$           ;BRANCH IF NO
58 006000 005037 016270          CLR   VAR1+2        ;VAR1 IN TEST 50
59 006004 012737 001500 016276  MOV   #WRFROM, VAR2+2 ;VAR2 IN TEST 50
60 006012 062737 000400 016276  ADD   #256., VAR2+2
61 006020 062737 000400 016276  ADD   #256., VAR2+2
62 006026 005037 016736          CLR   VAR3+2        ;VAR3 IN TEST 51
63 006032 012737 001500 016744  MOV   #WRFROM, VAR4+2 ;VAR4 IN TEST 51
64 006040 062737 000404 016744  ADD   #260., VAR4+2
65 006046 062737 000404 016744  ADD   #260., VAR4+2
66 006054 012737 052737 016774  MOV   #52737, VAR5   ;VAR5 IN TEST 51
67 006062 004737 032054 10$:   JSR   PC, CLDISK    ;CONTROLLER CLEAR
68
73

```

```

*****
;*TEST 2      RHCS2-CONTROL AND STATUS 2
;*THIS PARTIALLY TESTS RHCS2 TO ENABLE DETERMINATION
;*OF THE NUMBER OF DRIVES PRESENT
*****

```

```

006066 000004          TST2:  SCOPE
74 006070 012737 000001 001212  MOV   #1, $TIMES    ;;DO 1 ITERATION
75 006076 012706 001100          MOV   #STACK, SP   ;RESET STACK
76 006102 012737 000002 003610  MOV   #2, TSTNM     ;MOVE #2 TO TEST NUMBER

;CHECK TO SEE IF THE PROGRAM IS RUNNING WITH AN RH70

006110 005737 003614          TST   RH70         ;TEST FLAG FOR RH70 CONTROLLER
006114 001402          BEQ   64$         ;IF FLAG = 1, THIS TEST IS SKIPPED
006116 000137 006144          JMP   TST3        ;JUMP TO NEXT TEST
006122          ;IF FLAG = 0, DO THIS TEST

77
78 006122 013737 001236 006136  MOV   RHCS2, UN+2
79 006130 004537 031622          JSR   R5, BITST    ;TEST BITS IN REGISTER
80 006134 020017          UN:   .WORD 20017  ;ONLY THESE BITS ARE TEST READ/WRITE
81 006136 000000          .WORD 0           ;ADDRESS OF REG. BEING TESTED
82 006140 104001          EMT   1
83 006142 000207          RTS   PC           ;RETURN TO BLT3 ROUTINE
84
85

```

```

*****
;*TEST 3      PARTIAL TEST OF RHAS FOR UNIT NUMBERS PRESENT
*****

```

```

006144 000004          TST3:  SCOPE
86 006146 012737 000001 001212  MOV   #1, $TIMES    ;;DO 1 ITERATION
87 006154 012737 000003 003610  MOV   #3, TSTNM     ;MOVE #3 TO TEST NUMBER
88 006162 013701 001256          MOV   RHAS, R1     ;R1 HAS ADDRESS OF RHAS

```



```

89 006166 012711 177777      MOV    #-1,@R1      ;THIS CLEARS RHAS (SURPRISED!)
90 006172 011137 001126      MOV    @R1,$BDDAT   ;TEST DATA
91 006176 105737 001126      TSTB   $BDDAT
92 006202 001405              BEQ    TST4         ;BRANCH IF GOOD
93 006204 005037 001124      CLR    $GDDAT      ;GOOD DATA
94 006210 010137 031620      MOV    R1,REGADR   ;FAILING REG. RHAS
95 006214 104001              EMT    1
96                                  ;WITH ONES MOVED INTO IT
97
98
;*****
;*TEST 4      TEST FOR DRIVES PRESENT USING RHAS AND RHCS2
;*****
TST4:  SCOPE
      MOV    #1,$TIMES      ;;DO 1 ITERATION
      RESET  ;START WITH AN INIT
      JSR    PC,$TKINT      ;INITILIZE TTY KEYBOARD
102 006234 032777 020000 172676  BIT    #SW13,@SWR      ;INHIBIT ERROR TYPEOUT ?
103 006242 001024              BNE    4$            ;SKIP NEXT IF SO
104 006244 104401 006252      TYPE   ,65$         ;;TYPE ASCIZ STRING
      BR    64$         ;;GET OVER THE ASCIZ
      ;:65$: .ASCIZ <CRLF>/LOOKING AT RHAS - DRIVES PRESENT/
      64$:
105 006314 013701 001256      4$:  MOV    RHAS,R1      ;R1 HAS ADDR. OF RHAS
106 006320 013702 001236      MOV    RHCS2,R2     ;R2 HAS ADDR. OF RHCS2
107 006324 005012              CLR    @R2          ;CLEAR RHCS2
108 006326 012700 000010      MOV    #8.,R0       ;COUNT
109 006332 013704 001242      MOV    RHER1,R4     ;R4 HAS ADDR. OF RHER1
110
111 006336 012714 177777      1$:  MOV    #-1,@R4      ;MOVE ERRORS INTO RHER1
112 006342 005212              INC    @R2          ;INCREMENT UNIT NO.
113 006344 005300              DEC    R0           ;COUNT
114 006346 001373              BNE    1$          ;BRANCH IF 8 NOT DONE
115 006350 111137 001420      MOVB   @R1,TOTALAT ;SAVE TOTAL ATTENTION
116                                  ;USED IN DRIVE CLEAR TEST
117 006354 105037 001421      CLRB   TOTALAT+1   ;CLEAR UPPER BYTE
118 006360 105711              TSTB   @R1          ;TEST FOR ANY DRIVES PRESENT
119 006362 001402              BEQ    2$          ;NONE RESPONDING - TYPE THE MESSAGE
120 006364 000137 006414      JMP    XE2         ;SOME THERE - GO FILL 'UNITS' TABLE
121
122 006370 032777 020000 172542  2$:  BIT    #SW13,@SWR      ;INHIBIT ERROR TYPE OUT?
123 006376 001402              BEQ    3$          ;'NO DRIVES' MESSAGE IF NO
124 006400 000137 006742      JMP    SELTST      ;CHECK FOR SELECTED UNIT START AND LOAD
125                                  ;'UNITS' TABLE WITH DESIRED DRIVE IF SO
126
127 006404 104401 053107      3$:  TYPE   ,NDRVAS     ;TYPE 'NO DRIVES PRESENT IN RHAS'
128 006410 000137 030660      JMP    $EOP        ;GO OUT
129
130                                  ;SET UP UNITS TABLE
131
132 006414              XE2:
133 006414 012700 000010      2$:  MOV    #8.,R0       ;COUNTER
134 006420 012703 001354      MOV    #UNITS,R3    ;POINTER
135 006424 012723 177777      3$:  MOV    #-1,(R3)+    ;PRESET BLOCK TO ALL ONES
136 006430 005300              DEC    R0           ;COUNT
137 006432 001374              BNE    3$          ;BRANCH IF 8 NOT DONE
138 006434 012703 001354      MOV    #UNITS,R3    ;POINTER

```



```

139 006440 005005          CLR      R5
140 006442 005037 001376  CLR      NOUNIT          ;NO. OF UNITS PRESENT
141 006446 012700 000010  MOV      #8,R0          ;COUNTER
142 006452 011137 001176  MOV      @R1,$TMP0     ;TEMPORARY STORAGE
143 006456 006037 001176  ROR      $TMP0         ;SET CARRY IF ONE IN 0 BIT
144
145 006462 103114          BCC      5$
146 006464 010577 172546  MOV      R5,@RHCS2     ;INSERT UNIT NUMBER
147 006470 022777 024020 172566  CMP      #24020,@RHDT  ;IS THIS A DUAL PORT RP04 ?
148 006476 001477          BEQ      6$            ;TYPE DRIVE NO. IF SO
149 006500 022777 020020 172556  CMP      #20020,@RHDT  ;IS THIS A SINGLE PORT RP04 ?
150 006506 001473          BEQ      6$            ;TYPE DRIVE NO. IF YES
151
152 006510 022777 024021 172546  CMP      #24021,@RHDT  ;IS THIS A DUAL PORT RP05 ?
153 006516 001467          BEQ      6$            ;TYPE UNIT NO. OUT
154 006520 022777 020021 172536  CMP      #20021,@RHDT  ;IS THIS A SINGLE PORT RP05 ?
155 006526 001463          BEQ      6$            ;TYPE UNIT NO. IF SO
156
157 006530 022777 024022 172526  CMP      #24022,@RHDT  ;IS THIS A DUAL PORT RP06 ?
158 006536 001457          BEQ      6$            ;TYPE THE NO IF SO
159 006540 022777 020022 172516  CMP      #20022,@RHDT  ;IS THIS A SINGLE PORT RP06 ?
160 006546 001453          BEQ      6$            ;TYPE THE NO IF SO
161
162          ;NO...IT'S NOT AN RP04/RP05/RP06 DEVICE
163          ;SO TYPE OUT THE DEVICE TYPE
164
165 006550 104401 006556          TYPE     ,65$          ;;TYPE ASCIZ STRING
    006554 000407          BR       64$          ;;GET OVER THE ASCIZ
    ;;65$: .ASCIZ <CRLF>/UNIT NUMBER /
    64$:
166 006574          MOV      R5,-(SP)      ;GET READY TO TYPE UNIT NUMBER
167 006576 010546          TYPDS
168 006600 104401 006606          TYPE     ,67$          ;;TYPE ASCIZ STRING
    006604 000405          BR       66$          ;;GET OVER THE ASCIZ
    ;;67$: .ASCIZ /, RHDT = /
    66$:
169 006620 017746 172440          MOV      @RHDT,-(SP)  ;GET READY TO TYPE RHDT
170 006624 104402          TYPOC
171 006626 104401 006634          TYPE     ,69$          ;;TYPE ASCIZ STRING
    006632 000420          BR       68$          ;;GET OVER THE ASCIZ
    ;;69$: .ASCIZ ? - NOT AN RP04/RP05/RP06 DEVICE?
    68$:
172 006674 000407          BR       5$            ;NO RP04/5/6 FOUND SO BRANCH
173 006676 010523          MOV      R5,(R3)+
174 006700 104401 001223          TYPE     , $CRLF
175 006704 010546          MOV      R5,-(SP)     ;PUT DRIVE NO. ON STACK
176 006706 104405          TYPDS          ;TYPE DRIVE NO.
177 006710 005237 001376          INC      NOUNIT       ;INCR TOTAL NO. OF UNITS
178
179 006714 005205          5$: INC      R5          ;'RHCS2' UNIT ADDRESS
180 006716 005300          DEC      R0          ;DRIVE COUNTER DOWN ONE
181 006720 001256          BNE     4$            ;TEST AND DO NEXT UNIT IF 8 NOT DONE
182
183 006722 013737 001354 001374  MOV      UNITS,UNIT    ;SET UNIT NO. TO FIRST ONE FOUND/OR 0
184 006730 013737 001376 001400  MOV      NOUNIT,NUNIT ;SAVE NO. OF UNITS
185 006736 005337 001400          DEC      NUNIT        ;IF NUNIT = 0 THEN ONLY ONE UNIT
186          ;IF NUNIT > 0 THEN MORE THAN ONE UNIT
    
```

```

187
188 006742 005737 001402      SELTST: TST      SELECT      ;STARTING ADDRESS 200 ?
189 006746 001403              BEQ      TST5          ;BRANCH IF STARTING FROM 200
190 006750 013737 001404 001374  MOV      UNITSL,UNIT ;CHANGE UNIT NUMBER TO SELECTED ONE
191
198
:*****
:*TEST 5      TYPE SERIAL NUMBER AND DRIVE TYPE
:*READ SERIAL NUMBER REGISTER AND DRIVE TYPE REGISTER
:*TYPE IT OUT AND PROCEED
:*
:*TO LOOP HERE SET SWITCH 8 AND THIS TEST NO AND RESTART
:*****
006756 000004
006760 012737 000001 001212  TST5:  SCOPE
006766 012737 007426 001106  MOV      #1,$TIMES      ;;DO 1 ITERATION
199 006774 004737 032054      MOV      #1$,SLPADR     ;;SET SCOPE LOOP ADDRESS
200 007000 005037 001416      JSR      PC,CLDISK     ;FILL UNIT NO.
201      CLR      ATTENT ;CLEAR
202
203      ;TEST FOR UNIT #0
204 007004 005737 001374      TST      UNIT          ;IS UNIT #0 NEXT IN THE UNITS TABLE ?
205 007010 001022              BNE      10$           ;IF NOT, TEST THIS UNIT
206 007012 012700 000041      MOV      #41,RO        ;IF SO, CHECK THE LOAD MEDIA LOCATION
207 007016 122710 000011      CMPB    #11,(RO)      ;WAS IS AN RP04/5/6 ?
208 007022 001015              BNE      10$           ;NO... GO AHEAD AND TEST UNIT #0
209 007024 005737 001402      TST      SELECT ;WAS UNIT #0 SELECTED ?
210      ;(IE. WAS IT A 210 START ?)
211 007030 001012              BNE      10$           ;IF SO...TEST UNIT #0
212
213      ;INCREMENT THE UNITS TABLE TO NEXT DRIVE (IF ANY)
214      ; & DECREMENT THE 'NOUNITS' PRESENT (TO BE TESTED)
215
216 007032 012700 001354      MOV      #UNITS,RO     ;LOAD UNITS TABLE POINTER
217 007036 005720              TST      (RO)+         ;SELECT THE NEXT UNIT IN THE TABLE
218      ;(DOUBLE INCREMENT THE POINTER)
219 007040 022710 177777      CMP      #-1,(RO)      ;IS THERE ANOTHER TABLE ENTRY PRESENT ?
220 007044 001404              BEQ      10$           ;IF NOT (LOC = -1) ...MUST USE UNIT #0
221 007046 011037 001374      MOV      (RO),UNIT     ;SET UP TO BE THE UNIT UNDER TEST
222 007052 005337 001376      DEC      NOUNITS ;DECREMENT BECAUSE UNIT #0 WON'T BE TESTED
223 007056 013700 001374      10$:  MOV      UNIT,RO ;RO CONTAINS THE UNIT UNDER TEST
224
236
237 007062 116037 003620 001416  MOVB    ATABLE(RO),ATTENT ;SET APPROPRIATE ATTENTION BIT
238 007070 104401 007076      TYPE    ,65$          ;;TYPE ASCIZ STRING
      007074 000413      BR      64$           ;;GET OVER THE ASCIZ
      ;;65$: .ASCIZ <CRLF>/TESTING DRIVE NUMBER/
      64$:
239 007124 013746 001374      MOV      UNIT,-(SP)    ;UNIT NO. TO STACK
240 007130 104405              TYPDS    ;TYPE DRIVE NO.
241 007132 104401 007140      TYPE    ,67$          ;;TYPE ASCIZ STRING
      007136 000410      BR      66$           ;;GET OVER THE ASCIZ
      ;;67$: .ASCIZ <CRLF>/SERIAL NO. = /
      66$:
242 007160 017746 172102      MOV      @RHSN,-(SP)   ;;SAVE @RHSN FOR TYPEOUT
      007164 104402              TYPOC    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
243 007166 104401 007174      TYPE    ,69$          ;;TYPE ASCIZ STRING
      007172 000410      BR      68$           ;;GET OVER THE ASCIZ

```



```

    007214      ::69$: .ASCIZ <CRLF>/DRIVE TYPE = /
244 007214 017746 172044      68$: MOV @RHDT,-(SP)      ;;SAVE @RHDT FOR TYPEOUT
    007220 104402      TYPOC      ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
245
246 007222 022777 024020 172034      CMP #24020,@RHDT      ;DUAL PORT RP04 ?
247 007230 001425      BEQ 4$      ;TYPE ASCII MSG OUT
248 007232 022777 020020 172024      CMP #20020,@RHDT      ;SINGLE PORT RP04 ?
249 007240 001421      BEQ 4$      ;TYPE THE MESSAGE
250
251 007242 022777 024021 172014      CMP #24021,@RHDT      ;DUAL PORT RP05 ?
252 007250 001433      BEQ 5$      ;TYPE THE MESSAGE
253 007252 022777 020021 172004      CMP #20021,@RHDT      ;SINGLE PORT RP05 ?
254 007260 001427      BEQ 5$      ;TYPE THE MESSAGE
255
256 007262 022777 024022 171774      CMP #24022,@RHDT      ;DUAL PORT RP06 ?
257 007270 001441      BEQ 6$      ;TYPE THE MESSAGE
258 007272 022777 020022 171764      CMP #20022,@RHDT      ;SINGLE PORT RP06 ?
259 007300 001435      BEQ 6$      ;TYPE IT OUT
260 007302 000451      BR 1$      ;DRIVE IS NOT RP04/5/6 - SO
261      ;DO NOT TYPE ANY MESSAGE OUT
262
263      ;-SHOULD NEVER HAPPEN AT THIS POINT
264      ;UNLESS DRIVE GOT SICK WHILE TESTING
265      ;WAS IN PROGRESS
266
267 007304      4$: TYPE 71$      ;;TYPE ASCIZ STRING
    007304 104401 007312      BR 70$      ;;GET OVER THE ASCIZ
    007310 000412      ::71$: .ASCIZ <CRLF>/DRIVE IS AN RP04/<CRLF>
268 007336      70$: BR 1$      ;SKIP NEXT ONES
    007336 000433      5$: TYPE 73$      ;;TYPE ASCIZ STRING
269 007340 007346      BR 72$      ;;GET OVER THE ASCIZ
    007344 000412      ::73$: .ASCIZ <CRLF>/DRIVE IS AN RP05/<CRLF>
270 007372      72$: BR 1$      ;SKIP NEXT
    007372 000415      6$: TYPE 75$      ;;TYPE ASCIZ STRING
271 007374 104401 007402      BR 74$      ;;GET OVER THE ASCIZ
    007374 000412      ::75$: .ASCIZ <CRLF>/DRIVE IS AN RP06/<CRLF>
    007426      74$:
272
273 007426 005777 171634      1$: TST @RHSN      ;READ SERIAL NO. AND DRIVE TYPE
274 007432 005777 171626      TST @RHDT      ;THESE TWO ARE TO HELP SCOPE LOOPS
275 007436 017737 171624 001412      MOV @RHSN,SAVSN      ;SAVE TO CHECK IF CLR RHCS2 BIT 5 CLEARS ANY BITS
276 007444 017737 171614 001410      MOV @RHDT,SAVDT      ;SAVE TO CHECK IF CLR RHCS2 BIT 5 CLEARS ANY BITS
277
283      ;*****
      ;*TEST 6 CHECK MOL TO BE LOW
      ;*MAKE SURE THAT DRIVE IS OFF LINE BEFORE STARTING PROGRAM
      ;*IF DRIVE IS ON LINE THEN AFTER TYPE OUT THE PROGRAM WILL
      ;*HANG FOR EVER WAITING FOR DRIVE TO GO OFF LINE
      ;*****
284 007452 000004      TST6: SCOPE
    007454 012737 000006 003610      MOV #6,TSTNM      ;MOVE #6 TO TEST NUMBER
  
```

```

285 007462 004737 032054 JSR PC,CLDISK ;GIVE INITILIZE
286 007466 032713 010000 BIT #MOL,@R3 ;CHECK MOL IN RHDS1
287 007472 001542 BEQ TST7 ;BRANCH IF MOL LOW
288 007474 104401 007502 TYPE ,65$ ;:TYPE ASCIZ STRING
007500 000420 BR 64$ ;:GET OVER THE ASCIZ
;:65$: .ASCIZ <CRLF>/DRIVE IS ON LINE - MOL IS HIGH/
64$:
289 007542 104401 007550 TYPE ,67$ ;:TYPE ASCIZ STRING
007546 000423 BR 66$ ;:GET OVER THE ASCIZ
;:67$: .ASCIZ <CRLF>/HIT STOP ON DRIVE TO GET IT OFF LINE/
66$:
290 007616 104401 007624 TYPE ,69$ ;:TYPE ASCIZ STRING
007622 000430 BR 68$ ;:GET OVER THE ASCIZ
;:69$: .ASCIZ <CRLF>/PROGRAM WILL HANG TESTING MOL TILL MOL IS LOW/
68$:
291 007704 032713 010000 BIT #MOL,@R3 ;CHECK MOL IN RHDS1
292 007710 001375 BNE 1$ ;BRANCH IF MOL IS HIGH
293 007712 104401 007720 TYPE ,71$ ;:TYPE ASCIZ STRING
007716 000430 BR 70$ ;:GET OVER THE ASCIZ
;:71$: .ASCIZ <CRLF><LF>/MOL IS NOW LOW. PROGRAM WILL NOW BE EXECUTED/<CRLF>
70$:

```

```

294 010000
301
;:*****
;*TEST 7 PACK ACKNOWLEDGE COMMAND TEST
;*THE PACK ACKNOWLEDGE COMMAND WILL BE LOADED INTO RHCS1 WITH GO
;*THEN ALL REGISTERS WILL BE CHECKED
;*RH CLEAR WILL BE GIVEN
;*THEN ALL REGISTERS WILL BE CHECKED
;:*****

```

```

302 010000 000004 TST7: SCOPE
303 010002 012706 MOV #STACK,SP ;RESET STACK
304 010006 012737 MOV #7,TSTNM ;MOVE #7 TO TEST NUMBER
305 010014 004737 032054 JSR PC,CLDISK ;INIT AND SET UP GENERAL REG.
306 ;AND UNIT NUMBER
307 010020 012777 000001 171232 MOV #DMD,@RHMR ;SET DIAGNOSTIC MODE
308
309 010026 013777 001470 171204 MOV PKACK,@RHCS1 ;LOAD PACK ACKNOWLEDGE COMMAND INTO RHCS1
310
311 ;SAVE REGISTERS FOR COMPARISON AFTER GO
312 010034 004037 032554 JSR R0,SAVER ;SAVE
313 010040 001232 RHCW ;FROM
314 010042 002544 REINTO ;TO
315 010044 000023 19. ;NUMBER OF REGISTERS SAVED
316
317 ;GIVE GO TO PACK ACKNOWLEDGE COMMAND
318 010046 052777 000001 171164 BIS #GO,@RHCS1 ;GO TO PACK ACKNOWLEDGE COMMAND
319
320 ;CHANGE SAVED REGISTERS TO EXPECTED VALUES
321 010054 052737 000100 002574 BIS #VV,REINTO+30 ;SAVED RHDS1
322
323 ;AFTER GO HAS BEEN GIVEN TO PACK ACKNOWLEDGE COMMAND
324 ;SAVE REGISTERS AGAIN SO THAT COMPARISONS CAN
325 ;BE DONE
326 010062 004037 032554 JSR R0,SAVER ;SAVE
327 010066 001232 RHCW ;FROM
328 010070 001500 WRFROM

```


329 010072 000023 19. ;NUMBER OF REGISTERS SAVED

330
331 :AS UPPER BYTE OF RHAS CAN BE CHANGING IN A DUAL PORT
332 :OPERATION THE UPPER BYTE OF RHAS WILL BE SAVED AS IS
333 :SO THAT THE COMPARES ARE ONLY VALID FOR THE LOWER BYTE
334 010074 113737 002571 001525 MOVB REINTO+25,WRFROM+25;SAVE UPPER RHAS

335
336
337 :COMPARE REGISTERS BEFORE PACK ACKNOWLEDGE COMMAND
338 :WITH AFTER GO

339 010102 004037 032756 JSR R0,COMPAR ;COMPARE
340 010106 002544 REINTO ;GOOD BUFFER
341 010110 001500 WRFROM ;TEST BUFFER
342 010112 000023 19. ;NUMBER
343 010114 010122 1\$;RETURN FOR ERROR
344 010116 010122 1\$;SAME
345 010120 010142 2\$;RETURN FOR GOOD COMPARISON
346 010122 013705 037030 1\$: MOV ERWORD,R5 ;GETTING READY TO INDEX
347 010126 060505 ADD R5,R5 ;DOUBLE ERROR WORD
348 010130 016537 001230 031620 MOV RHWC-2(R5),REGADR ;FAILING REGISTER ADDRESS

349
350 010136 104001 EMT 1 ;AFTER PACK ACKNOWLEDGE COMMAND
351
352 :WITH GO IS GIVEN
353 010140 000207 RTS PC ;RETURN TO COMPARISION

354
355 010142 2\$:
356
357

::*****
:*TEST 10 MAKE CURRENT CYLINDER = 0
:*****

010142 000004
010144 012706 001100
010150 012737 000010 003610
010156 004737 032054
010162 012777 000001 171070
010170 004037 034434
010174 000000
369
TST10: SCOPE
MOV #STACK,SP ;RESET STACK
MOV #10,TSTNM ;MOVE #10 TO TEST NUMBER
JSR PC,CLDISK ;INIT DRIVE
MOV #DMD,@RHMR ;SET DIAGNOSTIC MODE
JSR R0,MAKECYL ;DO SEEK COMMAND FOLLOWED BY AN INIT TO
.WORD 0 ;CHANGE RHCC TO CYLINDER 0

::*****
:*TEST 11 BCTA LEGAL REGISTER RESPONSE TEST
:*THE FOLLOWING TESTS WILL ATTEMPT TO CATCH THOSE BUGS WHICH FAULT
:*INSERTION HAS SHOWN US WE MISSED IN THE FIRST PART OF THIS TEST.
:*
:*THIS TEST WILL ASCERTAIN THAT THE ALL RH11 REGISTERS WILL
:*RESPOND TO I.E. NOT CAUSE A NON-EXISTANT MEMORY ERROR WHEN ACCESSED
:*BY A 'TST' INSRUCTION.
:*****

010176 000004
010200 012737 000001 001212
370 010206 012737 000011 003610
371 010214 012777 000040 171014
372
373 010222 012705 001240
374
375 010226 013737 000004 001176
376 010234 012737 010272 000004
377
TST11: SCOPE
MOV #1,\$TIMES ;:DO 1 ITERATION
MOV #11,TSTNM ;MOVE #11 TO TEST NUMBER
MOV #CLR,@RHCS2 ;CLEAR RH11 CONTROLLER
MOV #RHCS1,R5 ;R5=LIST POINTER.
MOV TIMOT,\$TMP0 ;SAVE TIMEOUT VECTOR.
MOV #E00,TIMOT ;SET VECTOR TO E00

```

378 010242 012706 001100 L00: MOV #STACK,SP ;SET STACK POINTER.
379
380 010246 011502 100: MOV (R5),R2 ;R2=RH11 ADDRESS.
381 010250 010237 031620 MOV R2,REGADR
382 010254 005712 TST (R2) ;DOES THE RH11 REGISTER RESPOND?
383
384 010256 062705 000002 ADD #2,R5 ;UPDATE ADDRESS
385 010262 020527 001272 CMP R5,#RHEC2 ;AT THE END OF LEGAL RH11 REGISTERS?
386 010266 101767 BLOS I00 ;NOPE
387 010270 000401 BR 000 ;YES! GOTO 000.
388
389 010272 E00: EMT 42
010272 104042
390
391 010274 013737 001176 000004 000: MOV $TMP0,TIMOT ;RESTORE TIMEOUT VECTOR.
392
405

```

```

*****
*TEST 12 BCTA MOV B LO BYTE TO WC
*THE FOLLOWING 6 TESTS WILL TEST THE BYTE LOGIC FOR THE RH11 REGISTERS
*NO ATTEMPT IS MADE TO DETERMINE IF ALL POSSIBLE DATA PATTERNS CAN BE
*LOADED, ONLY THAT THE RH11 HARDWARE WILL ALLOW THE PROGRAM TO SELECTIVLY
*MANIPULATE BYTES USING THE FOLLOWING COMMANDS:
*
* 1). MOVE BYTE BOTH TO AND FROM THE WORD COUNT REGISTER
*
* 2). BIT SET INTO THE WORD COUNT REGISTER.
*
* 3). BIT CLEAR INTO THE WORD COUNT REGISTER.
*****

```

```

010302 000004
010304 012737 000001 001212
406 010312 012737 000012 003610
407 010320 012777 000040 170710
408
409 010326 012706 001100 MOV #STACK,SP ;SET STACK POINTER.
410
411 010332 013702 001232 MOV RHWC,R2 ;R2=RH11 WC ADDRESS.
412 010336 010237 031620 MOV R2,REGADR ;REGISTER ADDRESS TO REGADR FOR TYPING.
413 010342 012737 000377 001124 MOV #377,$GDDAT ;$GDDAT=S/B.
414
415 010350 005012 L02: CLR (R2) ;CLEAR RH11 WC.
416
417 010352 112712 000377 102: MOV #377,(R2) ;SET WC TO LO BYTE.
418 010356 011237 001126 MOV (R2),$BDDAT ;$BDDAT=ACTUAL WAS
419
420 010362 023737 001126 001124 CMP $BDDAT,$GDDAT ;COMPARE RESULTS
421 010370 001401 BEQ TST13 ;NEXT TEST
422 010372 104044 EMT 44
423
424

```

```

*****
*TEST 13 BCTA MOV B HI BYTE TO WC
*****

```

```

010374 000004
010376 012737 000001 001212
425 010404 012737 000013 003610
426 010412 012777 000040 170616
427
TST13: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION
MOV #13,TSTNM ;MOVE #13 TO TEST NUMBER
MOV #CLR,@RHCS2 ;CLEAR RH11 CONTROLLER

```



```
428 010420 012706 001100      MOV      #STACK,SP      ;SET STACK POINTER.
429
430 010424 013702 001232      MOV      RHWC,R2        ;R2=RH11 WC HI BYTE ADDRESS.
431 010430 010237 031620      MOV      R2,REGADR
432 010434 012737 177400 001124  MOV      #177400,$GDDAT ;$GDDAT=S/B.
433
434 010442 005012              L03:    CLR      (R2)      ;CLEAR RH11 WC.
435
436 010444 005202              INC      R2              ;R2=HI BYTE ADDRESS.
437
438 010446 112712 000377      I03:    MOV B   #377,(R2)    ;SET WC HI BYTE.
439
440 010452 005302              DEC      R2              ;R2=FULL WORD ADDRESS.
441
442 010454 011237 001126      MOV      (R2),$BDDAT    ;$BDDAT=ACTUAL.
443
444 010460 023737 001126 001124  CMP      $BDDAT,$GDDAT  ;COMPARE RESULTS.
445 010466 001401              BEQ     TST14           ;NEXT TEST
446 010470 104044              EMT     44
447
448
```

```
:::*****
:*TEST 14      BCTA BISB LO BYTE TO WC
:::*****
```

```
TST14:  SCOPE
449 010472 000004              MOV      #1,$TIMES      ;;DO 1 ITERATION
010474 012737 000001 001212  MOV      #14,TSTNM      ;MOVE #14 TO TEST NUMBER
450 010502 012737 000014 003610  MOV      #CLR,@RHCS2    ;CLEAR RH11 CONTROLLER
451 010510 012777 000040 170520
452 010516 012706 001100      MOV      #STACK,SP      ;SET STACK POINTER.
453
454 010522 013702 001232      MOV      RHWC,R2        ;R2=RH11 WC ADDRESS
455 010526 010237 031620      MOV      R2,REGADR
456 010532 012737 000377 001124  MOV      #377,$GDDAT    ;$GDDAT=S/B.
457
458 010540 005012              L04:    CLR      (R2)      ;CLEAR RH11 WC.
459
460 010542 012712 000252      I04:    MOV      #252,(R2)    ;SET UP WC
461 010546 152712 000125      BISB   #125,(R2)        ;DO A BISB
462
463 010552 011237 001126      MOV      (R2),$BDDAT    ;$BDDAT=WAS.
464
465 010556 023737 001126 001124  CMP      $BDDAT,$GDDAT  ;COMPARE RESULTS
466 010564 001401              BEQ     TST15           ;NEXT TEST
467 010566 104045              EMT     45
468
469
```

```
:::*****
:*TEST 15      BCTA BISB HI-BYTE TO WC
:::*****
```

```
TST15:  SCOPE
470 010570 000004              MOV      #1,$TIMES      ;;DO 1 ITERATION
010572 012737 000001 001212  MOV      #15,TSTNM      ;MOVE #15 TO TEST NUMBER
471 010600 012737 000015 003610  MOV      #CLR,@RHCS2    ;CLEAR RH11 CONTROLLER
472 010606 012777 000040 170422
473 010614 012706 001100      MOV      #STACK,SP      ;SET STACK POINTER.
474
475 010620 013702 001232      MOV      RHWC,R2        ;R2=RH11 WC ADDRESS.
476 010624 010237 031620      MOV      R2,REGADR
```

```
477 010630 012737 177400 001124      MOV      #177400,$GDDAT  ;$GDDAT=S/B.
478
479 010636 005012      L05:    CLR      (R2)      ;CLEAR RH11 WC.
480
481 010640 005202      INC      R2              ;R2 =HI BYTE.
482
483 010642 112712 000125      I05:    MOV      #125,(R2)   ;SET UP RH11 WC.
484 010646 152712 000252      BISB    #252,(R2)   ;DO A BISB.
485
486 010652 005302      DEC      R2              ;R2=FULL WORD ADDRESS.
487
488 010654 011237 001126      MOV      (R2),$BDDAT    ;$BDDAT=WAS.
489
490 010660 023737 001126 001124      CMP      $BDDAT,$GDDAT  ;COMPARE RESULTS.
491 010666 001401      BEQ     TST16          ;NEXT TEST
492 010670 104045      EMT     45
493
494
```

```
::*****
:*TEST 16      BCTA BICB LO-BYTE TO WC
*****
```

```
TST16:  SCOPE
495 010672 000004      MOV      #1,$TIMES      ;;DO 1 ITERATION
496 010674 012737 000001 001212      MOV      #16,TSTNM     ;MOVE #16 TO TEST NUMBER
497 010702 012737 000016 003610      MOV      #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
498 010710 012777 000040 170320
499
500 010716 012706 001100      MOV      #STACK,SP     ;SET STACK POINTER.
501
502 010722 013702 001232      MOV      RHWC,R2       ;R2=RH11 WC ADDRESS.
503 010726 010237 031620      MOV      R2,REGADR
504 010732 012737 000252 001124      MOV      #252,$GDDAT   ;$GDDAT=S/B.
505
506 010740 005012      L06:    CLR      (R2)      ;CLEAR RH11 WC.
507
508 010742 012712 000377      MOV      #377,(R2)     ;SET WC=0000377
509
510 010746 142712 000125      I06:    BICB    #125,(R2)   ;DO A BICB
511
512 010752 011237 001126      MOV      (R2),$BDDAT   ;$BDDAT=WAS.
513
514 010756 023737 001126 001124      CMP      $BDDAT,$GDDAT  ;COMPARE RESULTS.
515 010764 001401      BEQ     TST17          ;NEXT TEST
516 010766 104046      EMT     46
```

```
::*****
:*TEST 17      BCTA BICB HI- BYTE TO WC
*****
```

```
TST17:  SCOPE
517 010770 000004      MOV      #1,$TIMES      ;;DO 1 ITERATION
518 010772 012737 000001 001212      MOV      #17,TSTNM     ;MOVE #17 TO TEST NUMBER
519 011000 012737 000017 003610      MOV      #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
520 011006 012777 000040 170222
521
522 011014 012706 001100      MOV      #STACK,SP     ;SET STACK POINTER.
523
524 011020 013702 001232      MOV      RHWC,R2       ;R2=RH11 WC ADDRESS.
525 011024 010237 031620      MOV      R2,REGADR
526 011030 012737 125000 001124      MOV      #125000,$GDDAT ;$GDDAT=S/B.
```



```

526 011036 005012          L07:  CLR      (R2)          ;CLEAR RH11 WC.
527                                INC      R2              ;R2=HI BYTE.
528 011040 005202          I07:  MOVB    #377,(R2)       ;SET WC=177400
529                                BICB    #125,(R2)       ;DO A BICB
530 011042 112712 000377          DEC      R2              ;R2=FULL WORD.
531 011046 142712 000125          MOV     (R2), $BDDAT    ;$BDDAT=WAS.
532                                CMP     $BDDAT,$GDDAT   ;COMPARE RESULTS.
533 011052 005302          BEQ     TST20          ;NEXT TEST
534                                EMT     46
535 011054 011237 001126          ;
536                                ;
537 011060 023737 001126 001124  CMP     $BDDAT,$GDDAT
538 011066 001401          BEQ     TST20
539 011070 104046          EMT     46
540
552
  
```

```

;*****
;*TEST 20          A ILLEGAL REGISTER TEST
;*THIS TEST ATTEMPT TO ACCESS AN ILLEGAL REGISTER WITHIN THE RANGE
;*OF ADDRESSES SELECTED BY THE XOR'S. THE RH11 ADDRESS DECODE LOGIC WILL ALLOW UP TO 32
;*CONTIGUOUS REGISTERS TO EXIST, IN OUR CONFIGURATION ONLY 16 WILL REALLY
;*EXIST, THIS TEST ATTEMPTS TO ACCESS THE 20(8) REGISTER - IF IT RESPONDS
;*THE TEST WILL TYPE OUT AN ERROR.
;*
;*FOR THE CASE OF THE RH70, THE CONFIGURATION ALLOWS 18 OR 32 REGISTERS, SO
;*WE WILL ATTEMPT TO ADDRESS REGISTER 23(8), 33(8) OR AS BEFORE THE LAST
;*REGISTER + 2.
;*****
  
```

```

011072 000004          TST20: SCOPE
011074 012737 000001 001212      MOV     #1,$TIMES      ;;DO 1 ITERATION
553 011102 012737 000020 003610    MOV     #20,TSTNM     ;MOVE #20 TO TEST NUMBER
554 011110 012777 000040 170120    MOV     #CLR,$RHCS2   ;CLEAR RH11 CONTROLLER
555
556 011116 005737 003614          TST     RH70          ;CHECK TO SEE IF RUNNING WITH RH70
557 011122 001403          BEQ     1$            ;IF NOT...SKIP NEXT & DO FOLLOWING
558 011124 013702 001300          MOV     RHCS3,R2      ;R2 = LAST LEGAL RH70 REG ADDRESS
559 011130 000402          BR      2$            ;SKIP NEXT
560
561 011132 013702 001272          1$:  MOV     RHEC2,R2     ;R2 = LAST LEGAL RH11 REG ADDRESS.
562 011136 062702 000002          2$:  ADD     #2,R2       ;R2 = FIRST ILLEGAL ADDRESS.
563 011142 010237 031620          MOV     R2,REGADR
564 011146 005037 001126          CLR     $BDDAT       ;$BDDAT=WAS.
565 011152 005037 001124          CLR     $GDDAT       ;$GDDAT=S/B.
566
567 011156 013737 000004 001176    MOV     TIMOT,$TMPO   ;SAVE TIMEOUT VECTOR.
568 011164 012737 011210 000004    MOV     #010,TIMOT   ;SET TIMEOUT VECTOR TO 010.
569
570 011172 012706 001100          L10:  MOV     #STACK,SP ;SET THE STACK POINTER.
571
572 011176 005712          TST     (R2)         ;TEST ADDRESS.
573
574 011200 062702 000024          ADD     #24,R2       ;THIS MIGHT BE THE CASE OF 32 REGISTERS
575 011204 005712          TST     (R2)         ;TEST IT AGAIN
576 011206 104047          EMT     47
577
578 011210 013737 001176 000004 010: MOV     $TMPO,TIMOT   ;RESTORE TIMEOUT VECTOR.
579
589
  
```

```

;*****
  
```

```

;*TEST 21      BCTB (NED) NON-EXISTANT DRIVE TEST. (SET)
;*THE FOLLOWING TWO TESTS DETERMINE IF ALL NON DRIVES SET THE
;*NED BIT AND THAT ALL EXISTING DRIVES CLEAR THE NED BIT.
;*THE TESTS LOOK AT TOTALAT TO DETERMINE WHICH DRIVES EXIST AND TEST THAT
;*THESE DRIVES WILL CLEAR THE NED BIT. AND ALSO THAT NON EXISTANT DRIVES
;*WILL SET THE NED BIT.
;*ON FIRST PASS ONLY, IF ALL DRIVES ARE ON LINE A MESSAGE WILL SO ADVISE THE
;*OPERATOR. THE TEST WILL CONTINUE REGARDLESS OF THE OPERATORS ACTION.
;*****
    
```

```

011216 000004
011220 012737 000001 001212
590 011226 012737 000021 003610
591 011234 012777 000040 167774
592
593 011242 123727 001420 000377
594 011250 001147
595
596 011252 023727 001100 000000
597
598
599 011260 001002
600 011262 000137 011672
601
602 011266
011266 104401 011274
011272 000426
011350
603 011350 104401 011356
011354 000426
011432
604 011432 104401 011440
011436 000424
011510
605 011510 104401 011516
011514 000425
011570
606
607 011570 012706 001100
608
609 011574 013702 001236
610 011600 010237 031620
611
612 011604 013700 001420
613 011610 005001
614 011612 006000
615 011614 103423
616
617 011616 010137 001124
618 011622 052737 010000 001124
619
620 011630 013705 001240
621
622 011634 010112
    
```

```

TST21:  SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      #21,TSTNM      ;;MOVE #21 TO TEST NUMBER
        MOV      #CLR,@RHCS2    ;;CLEAR RH11 CONTROLLER
        CMPB     TOTALAT,#377    ;;ARE ALL DRIVES ON LINE.
        BNE      U11            ;;NO GO RUN THE TEST.
        CMP      $PASS,#0       ;;IF PASS #0 THEN TYPE MESSAGE
        BNE      Y11            ;;ADVISING OPERATOR
        JMP      X11            ;;THAT THIS TEST WILL NOT BE RUN

Y11:
        TYPE     ,65$           ;;TYPE ASCIZ STRING
        BR       64$           ;;GET OVER THE ASCIZ
        ;;65$: .ASCIZ <CRLF>/ALL DRIVES APPEAR TO BE ON LINE THEREFORE/
        64$:
        TYPE     ,67$           ;;TYPE ASCIZ STRING
        BR       66$           ;;GET OVER THE ASCIZ
        ;;67$: .ASCIZ <CRLF>/THE NED NON-EXISTANT DRIVE TEST CANNOT BE/
        66$:
        TYPE     ,69$           ;;TYPE ASCIZ STRING
        BR       68$           ;;GET OVER THE ASCIZ
        ;;69$: .ASCIZ <CRLF>/RUN. TO RUN THIS TEST TAKE ONE OR MORE/
        68$:
        TYPE     ,71$           ;;TYPE ASCIZ STRING
        BR       70$           ;;GET OVER THE ASCIZ
        ;;71$: .ASCIZ <CRLF>/DRIVES OFF-LINE AND RESTART THE PROGRAM/
        70$:
U11:    MOV      #STACK,SP      ;;SET STACK POINTER.
        MOV      RHCS2,R2      ;;R2=RH11 RHCS2 ADDRESS.
        MOV      R2,REGADR
        MOV      TOTALAT,R0
        CLR      R1
S11:    ROR      R0
        BCS     N11
R11:    MOV      R1,$GDDAT      ;;$GDDAT=S/B.
        BIS     #NED,$GDDAT
        MOV      RHCS1,R5      ;;ADDRESS OF A DEVICE REGISTER.
L11:    MOV      R1,(R2)       ;;LOAD A NON-EXISTANT DRIVE.
    
```



```

623
624 011636 011537 001176      MOV      (R5), $TMP0      ;ATTEMPT TO READ FROM DEVICE REGISTER.
625
626 011642 011237 001126      MOV      (R2), $BDDAT    ;$BDDAT=WAS.
627 011646 042737 167770 001126  BIC      #^C<NED!US4!US2!US1>, $BDDAT; $BDDAT=SAVED DATA.
628
629 011654 023737 001126 001124  CMP      $BDDAT, $GDDAT  ;COMPARE RESULTS.
630 011662 001005      BNE      E11
631
632 011664 005201      N11:    INC      R1
633 011666 020127 000010      CMP      R1, #8.        ;TESTED ALL DRIVES YET.
634 011672      X11:    BEQ      TST22     ;NEXT TEST
        011672 001402      BR
635 011674 000746      BR      S11
636
637 011676      E11:    EMT      50
        011676 104050
638
639
  
```

```

:*****
:*TEST 22      BCTB (NED) NON-EXISTANT DRIVE TEST. (CLEARED)
:*****
  
```

```

        011700 000004
        011702 012737 000001 001212
640 011710 012737 000022 003610
641 011716 012777 000040 167312
642
643 011724 012706 001100      MOV      #STACK, SP      ;SET STACK POINTER.
644
645 011730 013702 001236      MOV      RHCS2, R2       ;R2=RH11 RHCS2 ADDRESS.
646 011734 010237 031620      MOV      R2, REGADR
647 011740 013700 001420      MOV      TOTALAT, R0
648 011744 005001      CLR      R1
649 011746 006000      S12:    ROR      R0
650 011750 103020      BCC      N12
651
652 011752 010137 001124      MOV      R1, $GDDAT
653
654 011756 013705 001240      MOV      RHCS1, R5       ;ADDRESS OF A DEVICE REGISTER.
655
656 011762 010112      L12:    MOV      R1, (R2)   ;SELECT UNIT.
657
658 011764 011537 001176      MOV      (R5), $TMP0     ;ATTEMPT TO READ FROM DEVICE.
659
660 011770 011237 001126      MOV      (R2), $BDDAT    ;$BDDAT=WAS.
661 011774 042737 167770 001126  BIC      #^C<NED!US4!US2!US1>, $BDDAT ;$BDDAT=SAVED DATA.
662
663 012002 023737 001126 001124  CMP      $BDDAT, $GDDAT  ;COMPARE RESULTS.
664 012010 001005      BNE      E12
665
666 012012 005201      N12:    INC      R1
667 012014 020127 000010      CMP      R1, #8.        ;TESTED ALL DRIVES.
668 012020 001402      BEQ      TST23     ;NEXT TEST
669
670 012022 000751      BR      S12
671 012024      E12:    EMT      50
        012024 104050
672
  
```

677

```

:*****
:*TEST 23      BCTB AS REGISTER TEST
:*TEST TO SEE IF WE CAN READ FROM THE 'AS' REGISTER AND NOT CAUSE
:*A NED NON-EXISTANT DRIVE ERROR
:*****
    
```

```

012026 000004
678 012030 012737 000001 001212
012036 012737 000023 003610
679 012044 012777 000040 167164
680
681 012052 012706 001100
682
683 012056 013702 001236
684 012062 012737 000007 001124
685
686 012070 013705 001240
687
688 012074 012712 000007
689
690 012100 013702 001256
691
692 012104 011237 001176
693
694 012110 005012
695 012112 013702 001236
696 012116 010237 031620
697
698 012122 011237 001126
699 012126 042737 167770 001126
700
701 012134 023737 001126 001124
702 012142 001401
703 012144 104051
704
710
    
```

```

TST23: SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      #23,TSTNM     ;;MOVE #23 TO TEST NUMBER
        MOV      #CLR,@RHCS2   ;;CLEAR RH11 CONTROLLER
        MOV      #STACK,SP    ;;SET STACK POINTER.
        MOV      RHCS2,R2     ;;R2=RH11 CS2 ADDRESS.
        MOV      #US4!US2!US1,$GDDAT;$GDDAT=S/B.
        MOV      RHCS1,R5     ;;ADDRESS OF A DEVICE REGISTER.
L13:    MOV      #US4!US2!US1,(R2);LOAD A NON-EXISTANT DEVICE.
        MOV      RHAS,R2      ;;R2= 'AS'.
        MOV      (R2),$TMP0   ;;ATTEMPT TO READ FROM DEVICE AS.
        CLR      (R2)        ;;CLEAR AS REGISTER.
        MOV      RHCS2,R2     ;;R2=RH11 CS2 ADDRESS.
        MOV      R2,REGADR
        MOV      (R2),$BDDAT  ;;$BDDAT=WAS.
        BIC      #^C<NED!US4!US2!US1>,$BDDAT;SAVE NED AND DEVICE SELECTED.
        CMP      $BDDAT,$GDDAT ;;COMPARE RESULTS.
        BEQ      TST24      ;;NEXT TEST
        EMT      51
    
```

```

:*****
:*TEST 24      BCTC BUS ADDRESS REGISTER
:*THE NEXT THREE TESTS WILL TEST THE BUS ADDRESS REGISTER IN BOTH WORD
:*AND BYTE MODE. THE PATTERNS ARE CHOSEN TO PICK UP ANY DROPPED OR STUCK
:*BITS.
:*****
    
```

```

012146 000004
711 012150 012737 000001 001212
012156 012737 000024 003610
712 012164 012777 000040 167044
713
714 012172 012706 001100
715
716 012176 013702 001234
717 012202 010237 031620
718 012206 012705 012246
719
720 012212 012537 001124
721
722 012216 013712 001124
723 012222 011237 001126
724
725 012226 023737 001126 001124
    
```

```

TST24: SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      #24,TSTNM     ;;MOVE #24 TO TEST NUMBER
        MOV      #CLR,@RHCS2   ;;CLEAR RH11 CONTROLLER
        MOV      #STACK,SP    ;;SET STACK POINTER.
        MOV      RHBA,R2     ;;R2=RH11 BA ADDRESS.
        MOV      R2,REGADR
        MOV      #LST14A,R5   ;;R5=TEST LIST ADDRESS.
L14:    MOV      (R5)+,$GDDAT  ;;$GDDAT=S/B.
I14:    MOV      $GDDAT,(R2)   ;;SET BUS ADDRESS REGISTER.
        MOV      (R2),$BDDAT  ;;READ BUS ADDRESS REGISTER.
        CMP      $BDDAT,$GDDAT ;;COMPARE RESULTS.
    
```


726	012234	001401		BEQ	N14		:GET NEXT TEST DATA.
727	012236	104052		EMT	52		
728							
729	012240	005715	N14:	TST	(R5)		:AT END OF LIST
730	012242	001363		BNE	L14		:NO!
731	012244	000440		BR	TST25		:NEXT TEST
732	012246	000002	LST14A:	2			
733	012250	000004		4			
734	012252	000010		10			
735	012254	000020		20			
736	012256	000040		40			
737	012260	000100		100			
738	012262	000200		200			
739	012264	000400		400			
740	012266	001000		1000			
741	012270	002000		2000			
742	012272	004000		4000			
743	012274	010000		10000			
744	012276	020000		20000			
745	012300	040000		40000			
746	012302	100000		100000			
747	012304	177776		177776			
748	012306	177774		177774			
749	012310	177772		177772			
750	012312	177766		177766			
751	012314	177756		177756			
752	012316	177736		177736			
753	012320	177676		177676			
754	012322	177576		177576			
755	012324	177376		177376			
756	012326	176776		176776			
757	012330	175776		175776			
758	012332	173776		173776			
759	012334	167776		167776			
760	012336	157776		157776			
761	012340	137776		137776			
762	012342	077776		077776			
763	012344	000000		0			
764							
765							

 :*TEST 25 BCTC BUS ADDRESS REGISTER LO-BYTE

	012346	000004			TST25: SCOPE		
	012350	012737	000001	001212	MOV	#1,\$TIMES	::DO 1 ITERATION
766	012356	012737	000025	003610	MOV	#25,TSTNM	:MOVE #25 TO TEST NUMBER
767	012364	012777	000040	166644	MOV	#CLR,@RHCS2	:CLEAR RH11 CONTROLLER
768							
769	012372	012706	001100		MOV	#STACK,SP	:RESET STACK.
770							
771	012376	013702	001234		MOV	RHBA,R2	:R2=RH11 BA ADDRESS.
772	012402	010237	031620		MOV	R2,REGADR	
773	012406	012705	012450		MOV	#LST15A,R5	:R5=TEST LIST ADDRESS.
774							
775	012412	005012			L15: CLR	(R2)	:CLEAR RH11 BA REGISTER.
776							
777	012414	012537	001124		MOV	(R5)+,\$GDDAT	:GDDAT=S/B.
778							

```

779 012420 113712 001124      I15:  MOVB  $GDDAT,(R2)      ;SET BUS ADDRESS REGISTER.
780 012424 011237 001126      MOV    (R2),$BDDAT      ;READ BUS ADDRESS REGISTER.
781
782 012430 023737 001126 001124  CMP    $BDDAT,$GDDAT    ;COMPARE RESULTS.
783 012436 001401              BEQ    R15
784
785 012440 104053              EMT    53
786
787 012442 005715      R15:  TST    (R5)          ;AT END OF TEST LIST.
788 012444 001362      BNE    L15              ;NO!
789 012446 000417      BR     TST26           ;NEXT TEST
790 ;THIS LIST WILL BE USED TO LOAD THE LOWER BYTE OF THE BA REGISTER.
791
792 012450 000002      LST15A: 2
793 012452 000004      4
794 012454 000010      10
795 012456 000020      20
796 012460 000040      40
797 012462 000100      100
798 012464 000200      200
799 012466 000376      376
800 012470 000372      372
801 012472 000366      366
802 012474 000356      356
803 012476 000336      336
804 012500 000276      276
805 012502 000176      176
806 012504 000000      0
807
808

```

 :*TEST 26 BCTC BUS ADDRESS REGISTER HI-BYTE

```

809 012506 000004      TST26: SCOPE
810 012510 012737 000001 001212  MOV    #1,$TIMES      ;;DO 1 ITERATION
811 012516 012737 000026 003610  MOV    #26,TSTNM     ;MOVE #26 TO TEST NUMBER
812 012524 012777 000040 166504  MOV    #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
813
814 012532 012706 001100      MOV    #STACK,SP     ;RESET STACK.
815
816 012536 013702 001234      MOV    RHBA,R2       ;R2=RH11 BA ADDRESS.
817 012542 010237 031620      MOV    R2,REGADR     ;R2=RH11 BA ADDRESS.
818 012546 012705 012620      MOV    #LST16A,R5    ;R5=TEST LIST ADDRESS.
819
820 012552 005012      L16:  CLR    (R2)      ;CLEAR BA REGISTER.
821
822 012554 012537 001124      MOV    (R5)+,$GDDAT  ;$GDDAT=S/B.
823
824 012560 005202      INC    R2             ;R2=HI BYTE ADDRESS.
825
826 012562 113712 001124      I16:  MOVB  $GDDAT,(R2) ;SET BUS ADDRESS REGISTER HI-BYTE.
827
828 012566 005302      DEC    R2             ;R2=FULL WORD ADDRESS.
829
830 012570 011237 001126      MOV    (R2),$BDDAT   ;READ BUS ADDRESS.
831 012574 000337 001124      SWAB  $GDDAT         ;ADJUST DATA FOR HI BYTE.
832
833 012600 023737 001126 001124  CMP    $BDDAT,$GDDAT ;COMPARE RESULTS.

```



```

832 012606 001401          BEQ    R16          ;OKAY
833
834 012610 104053          EMT    53
835
836 012612 005715          R16:   TST    (R5)          ;AT END OF TEST LIST.
837 012614 001356          BNE    L16          ;NOPE
838 012616 000420          BR     TST27        ;NEXT TEST
839
;THIS LIST WILL BE USED TO LOAD THE UPPER BYTE OF THE BA REGISTER.
840
LST16A: 2
841 012620 000002          4
842 012622 000004          10
843 012624 000010          20
844 012626 000020          40
845 012630 000040          100
846 012632 000100          200
847 012634 000200          376
848 012636 000376          374
849 012640 000374          376
850 012642 000376          366
851 012644 000366          356
852 012646 000356          336
853 012650 000336          276
854 012652 000276          176
855 012654 000176          0
856 012656 000000
857
863
    
```

```

:*****
:*TEST 27      RH11 INTERRUPT TEST
:*THIS TEST CAUSE AN RH11 INTERRUPT VIA THE SPECIAL COMMAND SEQUENCE
:*'BIS #IE,@RHCS1'. THIS TEST PROVES ONLY THAT THE HARDWARE CAN HANDLE
:*INTERRUPTS PROPERLY
:*****
    
```

```

012660 000004
012662 012737 000001 001212
864 012670 012737 000027 003610
865 012676 012777 000040 166332
866
867 012704 012706 001100          MOV    #STACK,SP          ;SET STACK POINTER.
868
869 012710 013702 001240          MOV    RHCS1,R2          ;R2=RH11 ADDRESS.
870 012714 010237 031620          MOV    R2,REGADR
871 012720 005037 001126          CLR    $BDDAT          ;$BDDAT=WAS.
872 012724 005037 001124          CLR    $GDDAT          ;$GDDAT=S/B.
873
874 012730 017737 166272 001176          MOV    @RPVEC,$TMP0      ;SAVE RH11 INTERRUPT VECTOR.
875 012736 012777 012772 166262          MOV    #021,@RPVEC      ;SET RH11 INTERRUPT VECTOR TO 021.
876
877 012744 012706 001100          L21:  MOV    #STACK,SP          ;SET STACK POINTER.
878 012750 005037 177776          CLR    PS
879
880 012754 052712 000100          BIS    #IE,(R2)          ;CAUSE INTERRUPT.
881 012760 000240          NOP          ;WAIT FOR INTERRUPT.
882 012762 000240          NOP
883 012764 011237 001124          MOV    (R2),$GDDAT      ;SAVE CONTENTS OF REGISTER.
884
885 012770 104054          EMT    54
886
    
```

```

887 012772 013777 001176 166226 021:  MOV    STMP0,@RPVEC    ;RESTORE RH11 INTERRUPT VECTOR.
888
893
;*****
; *TEST 30      BCTD CLR L TEST
; *THE PROGRAM SETS THE PORT SELECT BIT .THEN DOES A RESET
; *IF THE SIGNAL CLR L WAS GENERATED THE PORT SELECT BIT WILL CLEAR.
;*****
013000 000004
013002 012737 000001 001212
894 013010 012737 000030 003610
895 013016 012777 000040 166212
896
897 013024 013702 001240      MOV    RHCS1,R2      ;R2=RH11 CS1 ADDRESS
898 013030 010237 031620      MOV    R2,REGADR
899 013034 005037 001124      CLR    $GDDAT      ;$GDDAT=S/B
900
901 013040 012706 001100      L22:  MOV    #STACK,SP      ;SET STACK POINTER.
902 013044 012712 002000      MOV    #PSEL,(R2)    ;SET PORT SELECT FLOP.
903 013050 000005      RESET              ;DO A BUSA INIT.
904
905 013052 011237 001126      MOV    (R2),$BDDAT   ;$BDDAT=WAS.
906 013056 042737 177677 001126  BIC    #^C<IE>,$BDDAT ;SAVE ONLY I.E. BIT.
907
908 013064 023737 001126 001124  CMP    $BDDAT,$GDDAT ;COMPARE RESULTS.
909 013072 001401      BEQ    TST31        ;NEXT TEST
910 013074 104055      EMT    55
911
915
;*****
; *TEST 31      MXF TEST
; *SET THE MXF FLOP AND READ IT BACK.
;*****
013076 000004
013100 012737 000001 001212
916 013106 012737 000031 003610
917
;CHECK TO SEE IF THE PROGRAM IS RUNNING WITH AN RH70
013114 005737 003614      TST    RH70        ;TEST FLAG FOR RH70 CONTROLLER
013120 001402      BEQ    64$        ;IF FLAG = 1, THIS TEST IS SKIPPED
013122 000137 013206      JMP    TST32      ;JUMP TO NEXT TEST
013126
918 013126 012777 000040 166102 64$:  MOV    #CLR,@RHCS2  ;CLEAR RH11 CONTROLLER
919
920 013134 012706 001100      MOV    #STACK,SP   ;SET STACK POINTER.
921 013140 013702 001236      MOV    RHCS2,R2    ;R2=RH11 CS2 ADDRESS.
922 013144 010237 031620      MOV    R2,REGADR
923 013150 012737 001000 001124  MOV    #MXF,$GDDAT ;$GDDAT=S/B.
924
925 013156 012712 001000      I24:  MOV    #MXF,(R2)   ;LOAD MXF
926
927 013162 011237 001126      MOV    (R2),$BDDAT ;$BDDAT=WAS.
928
929 013166 042737 176777 001126  BIC    #^C<MXF>,$BDDAT ;SAVE ONLY MXF
930
931 013174 023737 001126 001124  CMP    $BDDAT,$GDDAT ;COMPARE RESULTS
932 013202 001401      BEQ    TST32      ;NEXT TEST
933 013204 104056      EMT    56
    
```


934
 939
 940 013206 000004
 013210 012737 000001 001212
 941 013216 012737 000032 003610
 942
 943 013224 005737 003614
 013230 001402
 013232 000137 013322
 944 013236
 945 013236 012777 000040 165772
 946 013244 012706 001100
 947 013250 013702 001236
 948 013254 010237 031620
 949 013260 012737 020000 001124
 950
 951 013266 013712 001124
 952 013272 000240
 953 013274 000240
 954
 955 013276 011237 001126
 956 013302 042737 157777 001126
 957
 958 013310 023737 001126 001124
 959 013316 001401
 960 013320 104057
 961
 966

```

:*****
:*TEST 32      CSRB UNIBUS PARITY ERROR SET TEST
:*SET THE UNIBUS PARITY ERROR FLOP DIRECTLY VIA A MOV #UPE TO RHCS2
:*ATTEMPT TO READ IT BACK.
:*****
TST32:  SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      #32,TSTNM     ;;MOVE #32 TO TEST NUMBER

;CHECK TO SEE IF THE PROGRAM IS RUNNING WITH AN RH70

        TST      RH70          ;TEST FLAG FOR RH70 CONTROLLER
        BEQ      64$          ;IF FLAG = 1, THIS TEST IS SKIPPED
        JMP      TST33        ;JUMP TO NEXT TEST
64$:    ;IF FLAG = 0, DO THIS TEST

        MOV      #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER

        MOV      #STACK,SP     ;SET STACK POINTER.

        MOV      RHCS2,R2      ;R2=RHCS2 ADDRESS.
        MOV      R2,REGADR
        MOV      #UPE,$GDDAT   ;$GDDAT=S/B.

L30:    MOV      $GDDAT,(R2)    ;ATTEMPT TO SET UPE.
        NOP
        NOP

        MOV      (R2),$BDDAT    ;$BDDAT=ACTUAL DATA.
        BIC      #^C<UPE>,$BDDAT ;SAVE ONLY UPE.

        CMP      $BDDAT,$GDDAT ;COMPARE RESULTS.
        BEQ      TST33        ;NEXT TEST
        EMT      57
    
```

967 013322 000004
 013324 012737 000001 001212
 968 013332 012737 000033 003610
 969 013340 012777 000040 165670
 970 013346 012706 001100
 971
 972 013352 013702 001236
 973 013356 010237 031620
 974 013362 005037 001124
 975
 976 013366 012712 020000
 977 013372 000240
 978 013374 000240
 979
 980 013376 012712 000040

```

:*****
:*TEST 33      CSRB UNIBUS PARITY ERROR CLEAR TEST
:*SET THE UNIBUS PARITY ERROR FLOP AND ATTEMPT TO CLEAR IT WITH A
:*CONTROLLER CLEAR.
:*****
TST33:  SCOPE
        MOV      #1,$TIMES      ;;DO 1 ITERATION
        MOV      #33,TSTNM     ;;MOVE #33 TO TEST NUMBER
        MOV      #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER

        MOV      #STACK,SP     ;SET STACK POINTER.

        MOV      RHCS2,R2      ;R2=RHCS2 ADDRESS.
        MOV      R2,REGADR
        CLR      $GDDAT        ;$GDDAT=S/B.

L31:    MOV      #UPE,(R2)     ;SET UNIBUS PARITY ERROR SET.
        NOP
        NOP

        MOV      #CLR,(R2)     ;CONTROLLER CLEAR.
    
```

981
982 013402 011237 001126
983 013406 042737 157777 001126
984
985 013414 023737 001126 001124
986 013422 001401
987 013424 104057
988
993

MOV (R2), \$BDDAT ;READ STATUS
BIC #^C<UPE>, \$BDDAT ;SAVE ERROR.
CMP \$BDDAT, \$GDDAT ;COMPARE RESULTS.
BEQ TST34 ;NEXT TEST
EMT 57

*TEST 34 CSRB UNIT SELECT CLEAR TEST
*SET THE UNIT SELECT REGISTER TO DRIVE #7 ALL BITS SET. GENERATE A
*CONTROLLER CLEAR AND VERIFY THAT THE UNIT SELECT REGISTER IS CLEARED.

013426 000004
013430 012737 000001 001212
994 013436 012737 000034 003610
995 013444 012777 000040 165564
996
997 013452 012706 001100
998
999 013456 013702 001236
1000 013462 010237 031620
1001 013466 005037 001124
1002
1003 013472 012712 000007
1004 013476 000240
1005 013500 000240
1006
1007 013502 012712 000040
1008
1009 013506 011237 001126
1010 013512 042737 177770 001126
1011
1012 013520 023737 001126 001124
1013 013526 001401
1014 013530 104060
1015
1020

TST34: SCOPE
MOV #1, \$TIMES ;:DO 1 ITERATION
MOV #34, TSTNM ;:MOVE #34 TO TEST NUMBER
MOV #CLR, @RHCS2 ;:CLEAR RH11 CONTROLLER
MOV #STACK, SP ;:SET STACK POINTER.
MOV RHCS2, R2 ;:R2=CS2 ADDRESS.
MOV R2, REGADR
CLR \$GDDAT ;:\$GDDAT=S/B.
L34: MOV #US4!US2!US1, (R2) ;:LOAD RHCS2 DRIVE SELECT
NOP
NOP
MOV #CLR, (R2) ;:GENERATE CLR.
MOV (R2), \$BDDAT ;:READ RHCS2
BIC #^C<US4!US2!US1>, \$BDDAT ;:SAVE DRIVE SELECT BITS.
CMP \$BDDAT, \$GDDAT ;:COMPARE RESULTS.
BEQ TST35 ;:NEXT TEST
EMT 60

*TEST 35 CSRB TRANSFER ERROR (TRE) - UPE
*SET THE UPE FLOP UNIBUS PARITY ERROR. VERIFY THAT THE SETTING OF (UPE)
*ALSO SETS THE TRANSFER ERROR (TRE) FLOP

013532 000004
013534 012737 000001 001212
1021 013542 012737 000035 003610
1022
013550 005737 003614
013554 001402
013556 000137 013646
013562
1023
1024 013562 012777 000040 165446
1025
1026 013570 012706 001100
1027

TST35: SCOPE
MOV #1, \$TIMES ;:DO 1 ITERATION
MOV #35, TSTNM ;:MOVE #35 TO TEST NUMBER
;CHECK TO SEE IF THE PROGRAM IS RUNNING WITH AN RH70
TST RH70 ;:TEST FLAG FOR RH70 CONTROLLER
BEQ 64\$;:IF FLAG = 1, THIS TEST IS SKIPPED
JMP TST36 ;:JUMP TO NEXT TEST
64\$: ;:IF FLAG = 0, DO THIS TEST
MOV #CLR, @RHCS2 ;:CLEAR RH11 CONTROLLER
MOV #STACK, SP ;:SET STACK POINTER.


```
1028 013574 013702 001236      MOV    RHCS2,R2      ;R2=RHCS2 ADDRESS.
1029 013600 012737 040000 001124  MOV    #TRE,$GDDAT  ;$GDDAT=S/B.
1030
1031 013606 012712 020000      L35:   MOV    #UPE,(R2) ;SET UNIBUS PARITY ERROR.
1032
1033 013612 013702 001240      MOV    RHCS1,R2      ;R2=RHCS1 ADDRESS.
1034 013616 010237 031620      MOV    R2,REGADR
1035
1036 013622 011237 001126      MOV    (R2),$BDDAT   ;READ REGISTER
1037 013626 042737 137777 001126  BIC    #^C<TRE>,$BDDAT ;SAVE TRANSFER ERROR.
1038
1039 013634 023737 001126 001124  CMP    $BDDAT,$GDDAT ;COMPARE RESULTS
1040 013642 001401      BEQ    TST36         ;NEXT TEST
1041 013644 104061      EMT    61
1042
1046
```

```
::*****
:*TEST 36      CSRB TRANSFER ERROR (TRE) NED
:*SET THE NED BIT NON EXISTANT DRIVE VERIFY THAT THIS SETS THE (TRE) BIT.
::*****
```

```
TST36:  SCOPE
1047 013646 000004      MOV    #1,$TIMES    ;;DO 1 ITERATION
1048 013650 012737 000001 001212  MOV    #36,TSTNM    ;MOVE #36 TO TEST NUMBER
1049 013656 012737 000036 003610  MOV    #CLR,@RHCS2  ;CLEAR RH11 CONTROLLER
1050 013664 012777 000040 165344  MOV    #STACK,SP    ;SET STACK POINTER.
1051
1052 013672 012706 001100      MOV    #STACK,SP    ;SET STACK POINTER.
1053 013676 013702 001236      MOV    RHCS2,R2      ;R2=CS2 ADDRESS.
1054 013702 012737 040000 001124  MOV    #TRE,$GDDAT  ;$GDDAT=S/B.
1055 013710 013700 001420      MOV    TOTALAT,R0   ;GET ALL AVAILABLE DRIVES DATA
1056 013714 005001      CLR    R1
1057 013716 006000      S36:   ROR    R0
1058 013720 103420      BCS    N36
1059 013722 010112      L36:   MOV    R1,(R2)    ;SET NED.
1060
1061 013724 013702 001240      MOV    RHCS1,R2      ;R2=RHCS1 ADDRESS.
1062 013730 010237 031620      MOV    R2,REGADR
1063
1064 013734 011237 001126      MOV    (R2),$BDDAT   ;READ REGISTER.
1065 013740 042737 137777 001126  BIC    #^C<TRE>,$BDDAT ;SAVE TRE.
1066
1067 013746 023737 001126 001124  CMP    $BDDAT,$GDDAT ;COMPARE RESULTS.
1068 013754 001406      BEQ    TST37         ;NEXT TEST
1069 013756 104062      EMT    62
1070 013760 000404      BR     TST37         ;NEXT TEST
1071
1072 013762 005201      N36:   INC    R1
1073 013764 020127 000010  CMP    R1,#8.
1074 013770 001352      RNE    S36
1075
1079
```

```
::*****
:*TEST 37      CSRA PSEL CLEAR TEST
:*SET THE PORT SELECT FLOP VERIFY THAT WE CAN READ IT BACK.
::*****
```

```
TST37:  SCOPE
1080 013772 000004      MOV    #1,$TIMES    ;;DO 1 ITERATION
1081 013774 012737 000001 001212  MOV    #37,TSTNM    ;MOVE #37 TO TEST NUMBER
1082 014002 012737 000037 003610
```

```
1081 014010 012777 000040 165220      MOV    #CLR,@RHCS2      ;CLEAR RH11 CONTROLLER
1082
1083 014016 012706 001100              MOV    #STACK,SP       ;SET STACK POINTER.
1084
1085 014022 013702 001240              MOV    RHCS1,R2        ;R2=RHCS1 ADDRESS.
1086 014026 010237 031620              MOV    R2,REGADR       ;
1087 014032 005037 001124              CLR    $GDDAT          ;$GDDAT=S/B.
1088
1089 014036 012712 002000      L40:  MOV    #PSEL,(R2)   ;SET P SELECT.
1090
1091 014042 012777 000040 165166      MOV    #CLR,@RHCS2     ;DO A CONTROLLER CLEAR.
1092
1093 014050 011237 001126              MOV    (R2),$BDDAT     ;READ BACK P SELECT BIT.
1094 014054 042737 175777 001126      BIC    #^C<PSEL>,$BDDAT;SAVE ONLY PORT SELECT.
1095
1096 014062 023737 001126 001124      CMP    $BDDAT,$GDDAT   ;COMPARE RESULTS.
1097 014070 001401                    BEQ    TST40           ;NEXT TEST
1098 014072 104064                    EMT    64
1099
1103
```

```
::*****
:*TEST 40      CSRB COMMAND REGISTER CLEAR TEST
:*VERIFY THAT CONTROLLER CLEAR WILL CLEAR THE COMMAND REGISTER.
::*****
```

```
TST40:  SCOPE
1104 014074 000004                    MOV    #1,$TIMES      ;;DO 1 ITERATION
1105 014076 012737 000001 001212      MOV    #40,TSTNM     ;MOVE #40 TO TEST NUMBER
1106 014104 012737 000040 003610      MOV    #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
1107 014112 012777 000040 165116
1108
1109 014120 012706 001100              MOV    #STACK,SP     ;SET STACK POINTER.
1110
1111 014124 013702 001240              MOV    RHCS1,R2      ;R2=RHCS1 ADDRESS.
1112 014130 010237 031620              MOV    R2,REGADR     ;
1113 014134 005037 001124              CLR    $GDDAT        ;$GDDAT=S/B.
1114
1115 014140 012712 000011      L41:  MOV    #11,(R2)   ;ISSUE A DRIVE CLR AND GO.
1116
1117 014144 012777 000040 165064      MOV    #CLR,@RHCS2   ;CONTROLLER CLEAR.
1118
1119 014152 011237 001126              MOV    (R2),$BDDAT   ;READ COMMAND REGISTER.
1120 014156 042737 177770 001126      BIC    #^C<?>,$BDDAT ;SAVE ONLY THE COMMAND BITS.
1121
1122 014164 023737 001126 001124      CMP    $BDDAT,$GDDAT ;COMPARE RESULTS.
1123 014172 001401                    BEQ    TST41         ;NEXT TEST
1124 014174 104065                    EMT    65
1125
1131
```

```
::*****
:*TEST 41      CSRB COMMAND REGISTER RESELECT CLEAR TEST
:*THE COMMAND REGISTER IS SUPPOSED TO BE SELF CLEARING THAT IS WHEN THE
:*RH11 IS SELECTED WHEATHER OR NOT WE ADDRESS IT DIRECTLY THE LOGIC
:*SHOULD CLEAR THE COMMAND REGISTER.
:*THIS IS TESTED BY LOADING A COMMAND INTO IT READING ANOTHER REGISTER
:*THAN RETURNING TO CHECK THE COMMAND REGISTER TO DETERMINE IF IT CLEARED.
::*****
```

```
TST41:  SCOPE
1132 014176 000004                    MOV    #1,$TIMES      ;;DO 1 ITERATION
1133 014200 012737 000001 001212      MOV    #41,TSTNM     ;MOVE #41 TO TEST NUMBER
1134 014206 012737 000041 003610      MOV    #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
1135 014214 012777 000040 165014
```



```

1134
1135 014222 012706 001100      MOV      #STACK,SP      ;SET STACK POINTER.
1136
1137 014226 013702 001240      MOV      RHCS1,R2      ;R2=RHCS1 ADDRESS.
1138 014232 010237 031620      MOV      R2,REGADR
1139 014236 005037 001124      CLR      $GDDAT        ;$GDDAT=S/B.
1140
1141 014242 012737 000340 177776  MOV      #PR7,PS      ;SET PRIORITY TO #7.
1142
1143 014250 012712 000011      L42:    MOV      #11,(R2) ;ISSUE A DRIVE CLR AND GO
1144
1145 014254 011237 001176      MOV      (R2),$TMP0    ;DO A RESELECT OF THE REGISTER.
1146
1147 014260 011237 001126      MOV      (R2),$BDDAT   ;READ THE COMMAND REGISTER.
1148 014264 042737 177770 001126  BIC      #^C<7>,$BDDAT ;SAVE ONLY THE COMMAND BITS.
1149
1150 014272 023737 001126 001124  CMP      $BDDAT,$GDDAT ;COMPARE RESULTS.
1151 014300 001401      BEQ      TST42         ;NEXT TEST
1152 014302 104066      EMT      66
1153
1157

```

```

*****
*TEST 42      CSRFB READY AND IE INTERRUPT TEST
*HERE WE TEST TO SEE IF SETTING (IE) AND (RDY) WILL CAUSE AN INTERRUPT.
*****

```

```

014304 000004
014306 012737 000001 001212
1158 014314 012737 000042 003610
1159 014322 012777 000040 164706
1160
1161 014330 013702 001240      MOV      RHCS1,R2      ;R2=RPCS1 ADDRESS.
1162 014334 010237 031620      MOV      R2,REGADR
1163 014340 005037 001126      CLR      $BDDAT        ;$BDDAT=WAS.
1164 014344 005037 001124      CLR      $GDDAT        ;$GDDAT=S/B.
1165
1166 014350 017737 164652 001176  MOV      @RPVEC,$TMP0  ;SAVE RH11 INTERRUPT VECTOR.
1167 014356 012777 014412 164642  MOV      #043,@RPVEC  ;SET RH11 INTERRUPT VECTOR 043
1168
1169 014364 012706 001100      L43:    MOV      #STACK,SP ;SET STACK POINTER.
1170 014370 005037 177776      CLR      PS
1171
1172 014374 052712 000300      BIS      #IE!RDY,(R2) ;THIS WILL CAUSE AN INTERRUPT.
1173 014400 000240      NOP
1174 014402 000240      NOP
1175
1176 014404 011237 001124      MOV      (R2),$GDDAT
1177 014410 104067      EMT      67
1178
1179 014412 013777 001176 164606 043:    MOV      $TMP0,@RPVEC ;RESTORE RH11 INTERRUPT VECTOR.
1180
1185

```

```

*****
*TEST 43      CSRFB BOES "IE" CLEAR AFTER AN INTERRUPT
*THE (IE) BIT IS SET CAUSING AN INTERRUPT. AT THE COMPLETION OF THE INTERRUPT
*WE CHECK TO SEE IF THE (IE) BIT HAS CLEARED.
*****

```

```

014420 000004
014422 012737 000001 001212
1186 014430 012737 000043 003610
      MOV      #1,$TIMES ;DO 1 ITERATION
      MOV      #43,TSTNM ;MOVE #43 TO TEST NUMBER

```

```

1187 014436 012777 000040 164572      MOV      #CLR,@RHCS2      ;CLEAR RH11 CONTROLLER
1188
1189 014444 013702 031240      MOV      RHCS1,R2        ;R2=RPCS1 ADDRESS.
1190 014450 010237 031620      MOV      R2,REGADR
1191 014454 005037 001124      CLR      $GDDAT          ;$GDDAT=S/B.
1192
1193 014460 017737 164542 001176      MOV      @RPVEC,$TMP0    ;SAVE RH11 INTERRUPT VECTOR.
1194 014466 012777 014516 164532      MOV      #044,@RPVEC    ;SET VECTOR TO 044.
1195
1196 014474 012706 001100      L44:    MOV      #STACK,SP  ;SET STACK POINTER.
1197 014500 005037 177776      CLR      PS
1198
1199 014504 052712 000100      BIS      #IE,(R2)       ;THIS WILL CAUSE AN INTERRUPT
1200 014510 000240      NOP
1201 014512 000240      NOP
1202
1203 014514 000411      BR       E44            ;NO INTERRUPT.
1204
1205 014516 011237 001126      044:    MOV      (R2),$BDDAT    ;READ RHCS1.
1206 014522 042737 177677 001126      BIC      #^C<IE>,$BDDAT ;SAVE ONLY THE "IE" BIT.
1207
1208 014530 023737 001126 001124      CMP      $BDDAT,$GDDAT  ;COMPARE RESULTS.
1209 014536 001401      BEQ     R44            ;OK!
1210
1211 014540      E44:    EM1      70
1212 014540 104070
1213 014542 013777 001176 164456      R44:    MOV      $TMP0,@RPVEC ;RESTORE RH11 INTERRUPT VECTOR.
1214
1219
    
```

```

*****
;*TEST 44      BCTB "AS" WRITE TEST
;*HERE WE VERIFY THAT WRITING INTO THE "AS" REGISTER OWILL NOT CAUSE
;*AN (NED) ERROR.
*****
    
```

```

014550 000004
014552 012737 000001 001212      TST44:  SCOPE
1220 014560 012737 000044 003610      MOV      #1,$TIMES      ;;DO 1 ITERATION
1221 014566 012777 000040 164442      MOV      #44,TSTNM     ;MOVE #44 TO TEST NUMBER
1222
1223 014574 012706 001100      MOV      #CLR,@RHCS2   ;CLEAR RH11 CONTROLLER
1224
1225 014600 013702 001236      MOV      #STACK,SP    ;SET STACK POINTER.
1226 014604 012737 000007 001124      MOV      RHCS2,R2      ;R2=RH11 CS2 ADDRESS.
1227
1228 014612 013705 001240      MOV      #US4!US2!US1,$GDDAT;$GDDAT=S/B.
1229
1230 014616 012712 000007      L45:    MOV      #US4!US2!US1,(R2);LOAD A NON EXISTANT DEVICE.
1231
1232 014622 013702 001256      MOV      RHAS,R2      ;R2="AS" ADDRESS.
1233
1234 014626 012712 000377      MOV      #377,(R2)    ;ATTEMPT TO LOAD "AS".
1235
1236 014632 005012      CLR      (R2)         ;CLEAR "AS".
1237
1238 014634 013702 001236      MOV      RHCS2,R2     ;R2=RH11 CS2 ADDRESS.
1239 014640 010237 031620      MOV      R2,REGADR
1240
    
```



```
1241 014644 011237 001126      MOV      (R2), $BDDAT      ;$BDDAT=WAS.  
1242 014650 042737 167770 001126  BIC      #^C<NED!US4!US2!US1>, $BDDAT;SAVE NED AND DEVICE SELECTED.  
1243  
1244 014656 023737 001126 001124  CMP      $BDDAT, $GDDAT    ;COMPARE RESULTS.  
1245 014664 001401      BEQ      TST45             ;NEXT TEST  
1246 014666 104071      EMT      71  
1247  
1248
```

```
::*****  
:*TEST 45      MAKE CURRENT CYLINDER = 0  
:*****
```

```
014670 000004  
014672 012706 001100  
014676 012737 000045 003610  
014704 004737 032054  
014710 012777 000001 164342  
014716 004037 034434  
014722 000000  
TST45: SCOPE  
MOV      #STACK, SP      ;RESET STACK  
MOV      #45, TSTNM      ;MOVE #45 TO TEST NUMBER  
JSR      PC, CLDISK      ;INIT DRIVE  
MOV      #DMD, @RHMR     ;SET DIAGNOSTIC MODE  
JSR      R0, MAKECYL     ;DO SEEK COMMAND FOLLOWED BY AN INIT TO  
.WORD    0                ;CHANGE RHCC TO CYLINDER 0
```

1249
1256

```
::*****  
:*TEST 46      RHCS1 - BITS 8 AND 9 - EXTENDED ADDR (A16 & A 17)  
:*WRITE CYLINDER 0, FORMAT 16 BITS PER WORD  
:*TRACK 0, SECTOR 0, KEYS 0, NUMBER OF WORDS 256  
:*DATA IS THE CONTENTS OF THE TTY READER STATUS REGISTER  
:*THIS WILL USE BITS A16 AND A17 WHEN THERE IS MORE THAN 28K OF MEMORY  
:*****
```

```
014724 000004  
1257 014726 012706 001100  
1258 014732 012737 000046 003610  
1259  
TST46: SCOPE  
MOV      #STACK, SP      ;RESET STACK  
MOV      #46, TSTNM      ;MOVE #46 TO TEST NUMBER
```

;CHECK TO SEE IF THE PROGRAM IS RUNNING WITH AN RH70

```
014740 005737 003614  
014744 001402  
014746 000137 015256  
014752  
64$: TST      RH70          ;TEST FLAG FOR RH70 CONTROLLER  
BEQ      64$             ;IF FLAG = 1, THIS TEST IS SKIPPED  
JMP      TST47          ;JUMP TO NEXT TEST  
;IF FLAG = 0, DO THIS TEST
```

```
1260  
1261 014752 004037 031772      JSR      R0, CLAREA      ;CLEAR SIMULATED DISK  
1262 014756 040626      .WORD    DISK           ;FROM  
1263 014760 041652      .WORD    TOLGAP+16     ;TO  
1264 014762 000000      .WORD    0              ;DATA  
1265  
1266
```

;THESE ARE SETUP FOR DISKLESS USE ONLY

```
1267  
1268 014764 012737 010000 036710  MOV      #FMT22, CYL     ;CYLINDER 0  
1269      ;16 BITS PER WORD  
1270 014772 005037 036712      CLR      SECOTR         ;SECTOR 0 TRACK 0  
1271 014776 005037 036714      CLR      KEY1           ;KEY1 0  
1272 015002 005037 036716      CLR      KEY2           ;KEY2 0  
1273 015006 012737 000400 036756  MOV      #256., NOWORD  ;NO OF DATA WORDS  
1274 015014 012737 000001 036720  MOV      #1, X           ;WRITE DATA  
1275 015022 004537 033270      JSR      R5, CRC        ;GO TO CALCULATE CRC  
1276 015026 036710      CYL  
1277 015030 040610      WCRC
```

;THESE ARE REGULAR SETUPS

```
1278  
1279  
1280  
1281 015032 004737 032054      JSR      PC, CLDISK     ;SETUP GENERAL REGISTERS
```

```

1282 015036 012777 177400 164166      MOV      #-256.,@RHWC      ;256 DATA WORDS
1283 015044 013777 001144 164162      MOV      $TKS,@RHBA      ;STARTING ADDRESS OF WRITE BUFFER
1284 015052 017737 164066 001422      MOV      @$TKS,TMPILL     ;TEMPORARY STORAGE OF DATA
1285 015060 005077 164160                CLR      @RHDS1          ;SECTOR 0 TRACK 0
1286 015064 012777 010000 164156      MOV      #FMT22,@RHOF    ;16 BITS PER WORD FORMAT
1287 015072 005077 164154                CLR      @RHCA           ;CYLINDER 0
1288 015076 004737 032116                JSR      PC,CHECKT       ;CHECK THAT DVA,RDY,DPR,DRY = 1
                                015102 104401 053407      TYPE      ,CPHALT        ;AND THAT NO OTHERS = 1. CANNOT CON-
                                ;TINUE TESTING IF BOTH AREN'T TRUE
                                015106 000000                HALT
1289 015110 013746 001452                MOV      WRIDAT,-(SP)     ;WRITE DATA=60
1290 015114 052716 001400                BIS      #A16!A17,(SP)   ;SET HIGH ORDER UNIBUS BITS
1291 015120 012611                MOV      (SP)+,@R1        ;FILL RHCS1
1292 015122 052777 000010 164106      BIS      #BAI,@RHCS2     ;SET BUS ADDRESS INHIBIT
1293 015130 005037 001406                CLR      ERFLG$          ;CLEAR ERROR FLAG
1294 015134 004737 036550                JSR      PC,COMHD        ;WRITE DATA
1295
1296                                     ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
1297                                     ;FROM THE 'COMHD' ROUTINE IT MEANS SECTOR GAP, SYNC BYTE
1298                                     ;HEADER, HEADER CRC, HEADER GAP AND SYNC BYTE HAVE GONE BY
1299                                     ;AND SYNC'S WERE CORRECTLY DETECTED.
1300                                     ;DATA IS TO BE CHECKED.
1301
1302 015140 004737 031554                JSR      PC,PUTREG        ;SAVE REGISTERS
1303 015144 005737 001406                TST      ERFLG$          ;HAVE ANY ERRORS OCCURED?
1304 015150 001042                BNE      TST47           ;BRANCH OUT IF YES
1305 015152 013700 001422                MOV      TMPILL,R0       ;GOOD DATA
1306 015156 012701 040626                MOV      #DISK,R1        ;DATA WRITTEN INTO 'DISK'
1307 015162 012702 000400                MOV      #256.,R2        ;COUNTER
1308 015166 012737 000401 037030 1$:    MOV      #257.,ERWORD    ;FOR ERROR WORD
1309 015174 020021                CMP      R0,(R1)+        ;COMPARE GOOD DATA WITH DATA ON DISK
1310 015176 001425                BEQ      3$              ;BRANCH IF GOOD
1311 015200 013737 001422 001124      MOV      TMPILL,$GDDAT   ;GOOD DATA
1312 015206 014137 001126                MOV      -(R1),$BDDAT    ;BAD DATA
1313 015212 160237 037030                SUB      R2,ERWORD       ;ERROR WORD NO
1314 015216 005737 001406                TST      ERFLG$          ;ANY ERRORS ALREADY THERE?
1315 015222 001002                BNE      2$              ;BRANCH IF YES
1316 015224 104037                EMT      37
1317                                     ;SEE NEXT ERROR COMMENTS
1318 015226 000401                BR       1064$           ;BRANCH TO AVOID PRINTING NEXT ERROR
1319 015230 2$:                          EMT      40
                                015230 104040
1320                                     ;ERROR OCCURED WHILE WRITING
1321                                     ;WITH A16 A17 OF RHCS1 SET
1322 015232 005721 163700 1064$:  TST      (R1)+           ;UNDO -(R1) FOR BAD DATA
1323 015234 017746 177177                MOV      @SWR,-(SP)       ;GET SWITCH SETTING
1324 015240 042716 177177                BIC      #177177,(SP)    ;KEEP ONLY SWITCH 7 AND 8
1325 015244 022726 000200                CMP      #SW07,(SP)+     ;IS 7 SET AND 8 RESET
1326 015250 001402                BEQ      TST47           ;BRANCH OUT IF YES
1327 015252 005302 3$:                DEC      R2               ;IF NOT COUNT 256 WORDS
1328 015254 001344                BNE      1$              ;BRANCH IF 256 NOT DONE
1329
1341

```

```

:*****
:*TEST 47      DRIVE TIMING ERROR
:*A READ HEADER AND DATA IS STARTED ON CYLINDER 0, SECTOR
:*0, TRACK 0, 260 WORDS. AFTER THE HEADER IS READ IN CORRECTLY,
:*NO SYNC BYTE (DATA SYNC) IS GIVEN.

```



```

1388                                     ;HEADER HAS BEEN CORRECTLY READ
1389
1390 015432 004737 036550             JSR      PC,COMHD             ;ISSUE 'GO', SEARCH FOR THE SECTOR
1391                                     ;AND READ THE HEADER
1392
1393 015436 017737 163576 001176 SETCK1: MOV      @RHCS1,$TMP0        ;READ CS1 TO CHECK FOR ANY READ ERRORS
1394 015444 032737 100000 001176     BIT      #SC,$TMP0        ;TEST FOR 'SPECIAL CONDITION' - 'SC'
1395 015452 001405                       BEQ      8$                    ;CONTINUE WITH TEST IF 'SC' = 0 (NO ERRORS)
1396 015454 004737 031554             JSR      PC,PUTREG          ;READ & SAVE ALL REGISTERS AGAIN IF AN
1397                                     ;UNDEFINED DATA TRANSFER ERROR OCCURRED
1398 015460 104040                       EMT      40
1399 015462 000137 015750             JMP      TST50             ;'DTE' TEST - ABORT THE TEST

```

```

1400
1401                                     ;NOW THE HEADER HAS BEEN READ OK
1402                                     ;NOW 560 SECTOR CLOCKS WILL BE GIVEN
1403                                     ;GAP 11 BYTES, SYNC 1 BYTE, DATA 512, ECC 4 BYTES
1404                                     ;GAP 2 BYTES, TOLERANCE 28 BYTES, EXTRA 2

```

```

1405
1406                                     ;THESE 560 SECTOR CLOCKS ARE DIVIDED INTO TWO GROUPS
1407                                     ;24 SECTOR CLOCKS WITH NORMAL DIAGNOSTIC DATA CLOCKS,
1408                                     ;AND 536 SECTOR CLOCKS WITHOUT ANY DIAGNOSTIC DATA CLOCKS.

```

```

1409
1410                                     ;THIS GIVES 24 SECTOR CLOCKS WITH DIAGNOSTIC DATA CLOCKS
1411                                     ;WHICH EQUALS 3 BYTES OF DATA

```

```

1412
1413 015466 012701 000030             8$:      MOV      #24.,R1             ;LOAD COUNTER
1414 015472 013700 001260             MOV      RHMR,R0 ;GET RHMR ADDRESS
1415 015476 012710 000001             MOV      #DMD,@R0             ;SET DIAGNOSTIC MODE
1416 015502 052710 000012             1$:      BIS      #MSTCK!MCLK,@R0 ;SET SECTOR CLOCK AND DATA CLOCK
1417 015506 042710 000012             BIC      #MSTCK!MCLK,@R0 ;CLEAR SECTOR CLOCK AND DATA CLOCK
1418 015512 012702 000007             MOV      #7,R2                ;LOAD COUNTER FOR DIAGNOSTIC CLOCKS
1419 015516 052710 000002             4$:      BIS      #MCLK,@R0        ;SET CLOCK
1420 015522 042710 000002             BIC      #MCLK,@R0          ;CLEAR CLOCK
1421 015526 005302                       DEC      R2                    ;COUNT TO 7
1422 015530 001372                       BNE      4$                    ;BRANCH IF 7 NOT DONE
1423 015532 005301                       DEC      R1                    ;COUNT TO 24
1424 015534 001362                       BNE      1$                    ;BRANCH IF 24 NOT DONE

```

```

1425
1426                                     ;THIS GIVES 536 SECTOR CLOCKS WITHOUT DIAGNOSTIC DATA CLOCKS

```

```

1427
1428 015536 012701 001030             5$:      MOV      #536.,R1          ;LOAD SECTOR CLOCK COUNTER
1429 015542 052710 000010             BIS      #MSTCK,@R0          ;SET SECTOR CLOCK
1430 015546 042710 000010             BIC      #MSTCK,@R0          ;CLEAR SECTOR CLOCK
1431 015552 005301                       DEC      R1                    ;COUNT
1432 015554 001372                       BNE      5$                    ;BRANCH IF 536 NOT DONE

```

```

1433
1434                                     ;NOW 'DTE' SHOULD BE SET
1435                                     ;CHANGE SAVED REGISTERS TO EXPECTED VALUES

```

```

1436
1437
1438 015556 012737 177400 001306     MOV      #-256.,WC           ;SAVED RHWC
1439 015564 012737 002554 001310     MOV      #REINT0+<4.*2>.,BA ;SAVED RHBA
1440 015572 052737 140000 001314     BIS      #SC!TRE,CS1         ;SAVED RHCS1
1441 015600 052737 010000 001316     BIS      #DTE,ER1            ;SAVED RHER1
1442 015606 012737 000401 001334     MOV      #401,MR             ;SAVED RHMR
1443 015614 052737 140000 001336     BIS      #ATA!ERR,DS1        ;SAVED RHDS1
1444 015622 012737 000100 001350     MOV      #100,LA             ;SAVED RHLA

```



```

1445 015630 012737 000001 001320      MOV   #1,DST ;SAVED RHDST
1446 015636 013737 001416 001332      MOV   ATTENT,AS ;SAVED RHAS
1447
1448
1449      ;NOW READ & SAVE REGISTERS AGAIN SO THAT COMPARISONS
1450      ;CAN BE DONE (USE THE 'WRFROM' SAVE BUFFER THIS TIME)
1451
1452 015644 004037 032554      JSR   R0,SAVER ;READ IN SEQUENCE
1453 015650 001232      RHC   ;FROM HARDWARE REGISTER
1454 015652 001500      WRFROM ;INTO CORE BUFFER
1455 015654 000023      19. ;NUMBER OF REGISTERS TO READ
1456
1457      ;FOR RHAS UPPER BYTE
1458 015656 113737 001333 001525      MOV   AS+1,WRFROM+25 ;UPPER RHAS
1459
1460      ;COMPARE THE HEADER READ
1461
1462 015664 004037 032756      JSR   R0,COMFAR ;COMPARE
1463 015670 036710      CYL   ;GOOD BUFFER
1464 015672 002544      REINTO ;TEST BUFFER
1465 015674 000004      4. ;NUMBER
1466 015676 015704      6$ ;RETURN FOR ERROR
1467 015700 015704      6$ ;SAME
1468 015702 015710      7$ ;RETURN FOR GOOD COMPARISON
1469 015704 104010      6$: EMT 10
1470
1471      ;IN ERROR
1472 015706 000207      RTS   PC ;RETURN
1473
1474 015710      7$: ;GOOD COMPARISON, CONTINUE
1475
1476      ;COMPARE REGISTERS BEFORE COMMAND WITH REGISTERS AFTER COMMAND
1477
1478
1479 015710 004037 032756      JSR   R0,COMPAR ;COMPARE
1480 015714 001306      WC   ;INITIAL SNAPSHOT BUFFER (CHANGED)
1481 015716 001500      WRFROM ;TEST SNAPSHOT BUFFER
1482 015720 000022      18. ;NUMBER OF REGISTERS
1483 015722 015730      2$ ;RETURN FOR ERROR
1484 015724 015730      2$ ;SAME
1485 015726 015750      3$ ;RETURN FOR GOOD COMPARISON
1486
1487 015730 013705 037030      2$: MOV   ERWORD,R5 ;GETTING READY TO INDEX
1488 015734 060505      ADD   R5,R5 ;DOUBLE ERROR WORD
1489 015736 016537 001230 031620      MOV   RHWC-2(R5),REGADR ;FAILING REGISTER
1490 015744 104001      EMT 1
1491
1492      ;CHANGE AFTER FORCING
1493 015746 000207      RTS   PC ;'DTE' ERROR
1494
1495 015750      3$: ;GOOD - REGISTERS OK, GO ON TO NEXT TEST
1496
1511      ;*****
      ;*TEST 50 DRIVE TIMING ERROR
      ;*A WRITE DATA COMMAND IS STARTED ON CYLINDER 0, SECTOR
      ;*0, TRACK 0, 256 WORDS.

```

```

:*
:*THE SECTOR IS SEARCHED FOR AND
:*AFTER THE HEADER IS READ IN CORRECTLY,
:*NO SYNC BYTE (DATA SYNC) IS GIVEN.
:*NORMAL DIAGNOSTIC DATA CLOCKS AND DIAGNOSTIC
:*SECTOR CLOCKS ARE GIVEN FOR 24 BYTES,
:*THEN 536 BYTES OF SECTOR CLOCKS ONLY, ARE GIVEN.
:*
:*THIS IS TO TO BRING SECTOR PULSE UP
:*WHICH SHOULD SET 'DRIVE TIMING ERROR' - 'DTE'
:*****

```

```

1512 015750 000004
1512 015752 012706 001100
1513 015756 012737 000050 003610
1514
1515
1516
1517 015764 012737 010000 036710
1518
1519 015772 005037 036712
1520
1521 015776 005037 036714
1522 016002 005037 036716
1523 016006 012737 177777 036720
1524 016014 004537 033270
1525 016020 036710
1526 016022 040610
1527
1528
1529
1530 016024 004737 032054
1531 016030 012777 177400 163174
1532 016036 012777 001500 163170
1533 016044 005077 163174
1534
1535 016050 012777 010000 163172
1536
1537 016056 005077 163170
1538 016062 004737 032116
      016066 104401 053407
1539 016072 000000
      016074 013711 001452
1540
1541
1542
1543 016100 004037 032554
1544 016104 001232
1545 016106 001306
1546 016110 000023
1547
1548
1549
1550
1551
1552
1553

```

```

TST50: SCOPE
        MOV     #STACK,SP      ;RESET STACK
        MOV     #50,TSTNM     ;MOVE #50 TO TEST NUMBER

;THESE ARE TO SETUP FOR DISKLESS USE
        MOV     #FMT22,CYL    ;16 BITS PER WORD
                                ;CYLINDER 0
        CLR     SECOTR ;SECTOR 0
                                ;TRACK 0
        CLR     KEY1          ;KEY1 = 0
        CLR     KEY2          ;KEY2 = 0
        MOV     #-1,X         ;THIS IS FOR WRITE DATA COMMAND
        JSR     R5,CRC        ;GO TO CALCULATE CRC
        CYL
        WCRC

; THESE ARE REGULAR SETUPS & CHECKS
        JSR     PC,CLDISK     ;SETUP GENERAL REGISTERS
        MOV     #-256,@RHWC   ;256 DATA WORDS
        MOV     #WRFROM,@RHBA ;STARTING ADDRESS OF BUFFER
        CLR     @RHDST        ;TRACK = 0
                                ;SECTOR = 0
        MOV     #FMT22,@RHOF  ;16 BITS PER WORD
                                ;ECC CORRECTION INHIBITED
        CLR     @RHCA         ;CYLINDER = 0
        JSR     PC,CHECKT     ;CHECK THAT DVA,RDY,DPR,DRY = 1
        TYPE    ,CPHALT      ;AND THAT NO OTHERS = 1. CANNOT CON-
                                ;TINUE TESTING IF BOTH AREN'T TRUE
        HALT
        MOV     WRIDAT,@R1    ;WRITE DATA = 60

;READ & SAVE REGISTERS FOR COMPARISON AFTER SIMULATED 'DTE'
        JSR     R0,SAVER      ;READ REGISTERS IN SEQUENCE
        RHWC          ;FROM HARDWARE REGISTER
        WC           ;INTO CORE BUFFER LOCATION
        19.           ;NUMBER OF REGISTERS TO READ

;NOW 'GO' WILL BE GIVEN, EVERYTHING WILL BE TREATED
;NORMALLY FOR THE HEADER. WHEN IT IS TIME TO WRITE
;DATA, ONLY SECTOR CLOCKS WILL BE GIVEN, NO DIAGNOSTIC DATA
;CLOCKS WILL BE GIVEN. THIS SHOULD BRING SECTOR PULSE HIGH
;WITHOUT PUTTING READ DOWN, HENCE 'DTE' WILL COME UP.

```



```

1554 016112 012737 177777 001426      MOV    #-1, TESDTE      ;SET DTE TEST
1555 016120 012737 177777 001406      MOV    #-1, ERFLG$     ;THIS WILL BRING THE READ HEADER
1556                                     ;AND DATA PROCESS OUT AFTER THE
1557                                     ;HEADER HAS BEEN CORRECTLY READ
1558
1559 016126 004737 036550                JSR    PC, COMHD       ;ISSUE 'GO', SEARCH FOR SECTOR,
1560                                     ;READ HEADER AND DATA.
1561
1562 016132 017737 163102 001176  SETCK2: MOV @RHCS1, $TMP0 ;READ CS1 TO CHECK FOR READ ERRORS
1563 016140 032737 100000 001176      BIT    #SC, $TMP0     ;TEST FOR "SPECIAL CONDITION" - 'SC'
1564 016146 001405                BEQ    6$,             ;CONTINUE TESTING IF NO ERROR
1565 016150 004737 031554                JSR    PC, PUTREG     ;READ & SAVE REGISTERS AGAIN IF UNDE-
1566                                     ;FINED ERROR HAS OCCURRED
1567 016154 104040                EMT    40
1568                                     ;DURING 'DTE' TEST SETUP
1569 016156 000137 016474                JMP    TST51         ;ABORT THE TEST
1570
1571                                     ;NOW THE HEADER HAS BEEN READ,
1572                                     ;560 SECTOR CLOCKS WILL BE GIVEN -
1573                                     ;GAP 11 BYTES, SYNC 1 BYTE, DATA 512, ECC 4 BYTES
1574                                     ;GAP 2 BYTES, TOLERANCE 28 BYTES, EXTRA 2
1575
1576                                     ;THESE 560 SECTOR CLOCKS ARE DIVIDED INTO TWO GROUPS
1577                                     ;24 SECTOR CLOCKS WITH NORMAL DIAGNOSTIC DATA CLOCKS
1578                                     ;AND 536 SECTOR CLOCKS WITHOUT ANY DIAGNOSTIC DATA CLOCKS
1579
1580                                     ;THIS GIVES 24 SECTOR CLOCKS WITH DIAGNOSTIC DATA CLOCKS
1581
1582 016162 012701 000030 6$:    MOV    #24, R1      ;LOAD SECTOR CLOCK COUNTER
1583 016166 013700 001260      MOV    RHMR, R0 ;GET RHMR ADDRESS
1584 016172 012710 000001      MOV    #DMD, @R0    ;SET DIAGNOSTIC MODE
1585 016176 052710 000012 1$:    BIS    #MSTCK!MCLK, @R0 ;SET SECTOR CLOCK AND DATA CLOCK
1586 016202 042710 000012      BIC    #MSTCK!MCLK, @R0 ;CLEAR SECTOR CLOCK AND DATA CLOCK
1587 016206 012702 000007      MOV    #7, R2      ;LOAD COUNTER FOR DIAGNOSTIC DATA CLOCKS
1588 016212 052710 000002 4$:    BIS    #MCLK, @R0   ;SET CLOCK (DATA)
1589 016216 042710 000002      BIC    #MCLK, @R0   ;CLEAR CLOCK (DATA)
1590 016222 005302                DEC    R2           ;COUNT
1591 016224 001372                BNE    4$,         ;BRANCH IF 7 NOT DONE
1592 016226 005301                DEC    R1           ;COUNT
1593 016230 001362                BNE    1$,         ;BRANCH IF 24 NOT DONE
1594
1595                                     ;THIS GIVES 536 SECTOR CLOCKS WITHOUT DIAGNOSTIC DATA CLOCKS
1596
1597 016232 012701 001030 5$:    MOV    #536, R1     ;LOAD SECTOR CLOCK COUNTER
1598 016236 052710 000010      BIS    #MSTCK, @R0 ;SET SECTOR CLOCK
1599 016242 042710 000010      BIC    #MSTCK, @R0 ;CLEAR SECTOR CLOCK
1600 016246 005301                DEC    R1           ;COUNT
1601 016250 001372                BNE    5$,         ;BRANCH IF 536 NOT DONE
1602
1603                                     ;ECC PATTERN REGISTER IS NOT CHECKED
1604 016252 017737 163014 001346      MOV    @RHEC2, EC2   ;RHEC2 IS NOT CHECKED
1605
1606                                     ;NOW 'DTE' SHOULD BE SET, CHANGE SAVED REGISTERS TO EXPECTED VALUES
1607
1608
1609 016260 005737 003614                TST    RH70         ;CHECK FOR RH70 CONTROLLER
1610 016264 001412                BEQ    JP1          ;SKIP RH70 CODE AND DO RH11 IF NOT

```

```

1611
1612 ;RHC AND RHBA WILL BE MODIFIED FOR RH70C
1613 016266 012737 177416 001306 VAR1: MOV #-242.,WC ;SAVED RHC
1614 016274 012737 001534 001310 VAR2: MOV #WRFROM+<14.*2>,BA ;SAVED RHBA
1615 016302 052737 000300 001312 BIS #IR!OR,CS2 ;SAVED RHCS2
1616 016310 000414 BR JP2 ;SKIP NEXT RH11 CODE
1617
1618 016312 012737 177511 001306 JP1: MOV #-183.,WC ;SAVED RHC
1619 016320 012737 001722 001310 MOV #WRFROM+<73.*2>,BA ;SAVED RHBA
1620 016326 042737 000100 001312 BIC #IR,CS2 ;SAVED RHCS2
1621 016334 052737 000200 001312 BIS #OR,CS2 ;SAVED RHCS2
1622
1623 016342 052737 140000 001314 JP2: BIS #SC!TRE,CS1 ;SAVED RHCS1
1624 016350 052737 010000 001316 BIS #DTE,ER1 ;SAVED RHER1
1625 016356 012737 000201 001334 MOV #DENVL!DMD,MR ;SAVED RHMR
1626 016364 052737 140000 001336 BIS #ATA!ERR,DS1 ;SAVED RHDS1
1627 016372 012737 000100 001350 MOV #100,LA ;SAVED RHLA
1628 016400 012737 000001 001320 MOV #1,DST ;SAVED RHDST
1629 016406 013737 001416 001332 MOV ATTENT,AS ;SAVED RHAS
1630
1631 ;NOW READ & SAVE REGISTERS AGAIN SO THAT COMPARISONS
1632 ;CAN BE DONE (USING 'WRFROM' BUFFER THIS TIME)
1633
1634 016414 004037 032554 JSR R0,SAVER ;READ IN SEQUENCE
1635 016420 001232 RHC ;FROM HARDWARE REGISTER
1636 016422 002544 REINTO ;INTO CORE BUFFER LOCATION
1637 016424 000023 19. ;NUMBER OF REGISTERS TO READ
1638
1639 ;FOR RHAS UPPER BYTE
1640 016426 113737 001333 002571 MOVB AS+1,REINTO+25 ;UPPER RHAS
1641
1642
1643 ;COMPARE CHANGED REGISTER SNAPSHOT BEFORE COMMAND WITH
1644 ;SNAPSHOT AFTER COMMAND
1645
1646 016434 004037 032756 JSR R0,COMPAR ;COMPARE
1647 016440 001306 WC ;CHANGED INITIAL SNAPSHOT BUFFER
1648 016442 002544 REINTO ;SNAPSHOT BUFFER AFTER COMMAND
1649 016444 000022 18. ;NUMBER OF REGISTERS TO COMPARE
1650 016446 016454 2$ ;RETURN FOR ERROR
1651 016450 016454 2$ ;SAME
1652 016452 016474 3$ ;RETURN FOR GOOD COMPARISON
1653
1654 016454 013705 037030 2$: MOV ERWORD,R5 ;GETTING READY TO INDEX
1655 016460 060505 ADD R5,R5 ;DOUBLE ERROR WORD
1656 016462 016537 001230 031620 MOV RHC-2(R5),REGADR ;FAILING REGISTER
1657 016470 104001 EMT 1
1658 ;CHANGE AFTER FORCING
1659 ;'DTE' ERROR
1660 016472 000207 RTS PC ;RETURN
1661
1662 016474 3$: ;GOOD, REGISTERS OK - GO ON TO NEXT TEST
1663
1673 ;*****
; *TEST 51 DRIVE TIMING ERROR
; *A WRITE HEADER AND DATA COMMAND IS GIVEN
; *TO CYLINDER 0, SECTOR 0, TRACK 0, 256. WORDS.

```



```

: *
: *AFTER SECTOR IS FOUND (THE SECTOR FOUND FLOP IS HIGH),
: *NO MORE DIAGNOSTIC DATA CLOCKS ARE GIVEN,
: *ONLY SECTOR CLOCKS ARE GIVEN TILL SECTOR PULSE IS HIGH.
: *THIS SHOULD SET 'DRIVE TIMING ERROR' - 'DTE'
: *****
TST51: SCOPE
1674 016474 000004 001100      MOV      #STACK,SP      ;RESET STACK
1675 016502 012737 000051 003610  MOV      #51,TSTNM      ;MOVE #51 TO TEST NUMBER
1676
1677
1678
1679 016510 012737 010000 042046      MOV      #FMT22,WCYL      ;FORMAT 22=16 BITWORDS AND
1680                                     ;CYLINDER 0
1681 016516 005037 042050      CLR      WSECTR ;TRACK=0, SECTOR=0
1682 016522 005037 042052      CLR      WKEY1 ;KEY1=0
1683 016526 005037 042054      CLR      WKEY2 ;KEY2=0
1684 016532 012737 000400 042106  MOV      #256.,FNWORD ;256 DATAWORDS
1685 016540 004537 033270      JSR      R5,CRC ;GO TO CALCULATE CRC
1686 016544 042046      WCYL
1687 016546 042056      GCRC
1688
1689
1690
1691 016550 004737 032054      JSR      PC,CLDISK ;SETUP GENERAL REGISTERS
1692 016554 012777 177374 162450  MOV      #-260.,@RHWC ;256 DATA WORDS & 4 HEADER WORDS
1693 016562 012777 001500 162444  MOV      #WRFROM,@RHBA ;STARTING ADDRESS OF BUFFER
1694 016570 005077 162450      CLR      @RHDST ;TRACK = 0
1695                                     ;SECTOR = 0
1696 016574 012777 014000 162446  MOV      #FMT22!ECI,@RHOF ;16 BITS PER WORD
1697                                     ;ECC CORRECTION INHIBITED
1698 016602 005077 162444      CLR      @RHCA ;CYLINDER = 0
1699 016606 004737 032116      JSR      PC,CHECKT ;CHECK THAT DVA,RDY,DPR,DRY = 1
1700 016612 104401 053407      TYPE ;AND THAT NO OTHERS = 1. CANNOT CON-
1701                                     ;TINUE TESTING IF BOTH AREN'T TRUE
1702                                     ;STOP THE TEST
1703                                     ;WRITE HEADER AND DATA = 62
1704 016616 000000      HALT
1705 016620 013711 001454      MOV      WRIFOR,@R1
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715 016624 004037 032554      JSR      R0,SAVER ;READ IN SEQUENCE
1716 016630 001232      RHWC ;FROM HARDWARE REGISTER
1717 016632 001306      WC ;INTO CORE BUFFER LOCATION
1718 016634 000023      19. ;NUMBER OF REGISTERS TO READ
1719
1720
: NOW 'GO' WILL BE GIVEN. EVERYTHING WILL BE TREATED
: NORMALLY TILL HEADER IS TO BE GIVEN, THEN ONLY
: SECTOR CLOCKS WILL BE GIVEN. NO DIAGNOSTIC DATA
: CLOCKS WILL BE GIVEN. THIS SHOULD BRING SECTOR PULSE HIGH
: WITHOUT PUTTING 'READ' DOWN, HENCE 'DTE' WILL COME UP.
1715 016636 012737 177777 001426  MOV      #-1,TESDTE ;SET DTE TEST
1716 016644 012737 177777 001406  MOV      #-1,ERFLGS ;THIS WILL BRING THE READ HEADER
1717                                     ;AND DATA PROCESS OUT AFTER THE
1718                                     ;HEADER HAS BEEN CORRECTLY READ
1719
1720 016652 004737 041672      JSR      PC,COMWHD ;ISSUE 'GO', SEARCH FOR SECTOR,
    
```

```

1721                                     ;WRITE HEADER AND DATA.
1722
1723 016656 017737 162356 001176 SETCK3: MOV @RHCS1,$TMP0 ;READ CS1 TO CHECK FOR ERRORS DURING WRITE
1724 016664 032737 100000 001176 BIT #SC,$TMP0 ;TEST FOR 'SPECIAL CONDITION' - 'SC'
1725 016672 001405 BEQ 4$ ;CONTINUE TEST IF NO ERROR ('SC' = 0)
1726 016674 004737 031554 JSR PC,PUTREG ;READ & SAVE REGISTERS AGAIN IF ERROR
1727 016700 104040 EMT 40
1728
1729 016702 000137 017134 JMP TST52 ;DURING 'DTE' TEST SETUP
1730 ;ABORT THE TEST AT THAT POINT
1731
1732 ;NOW SECTOR HAS BEEN FOUND OK
1733
1734 ;609 SECTOR CLOCKS WILL BE GIVEN,
1735 ;39 BYTES FOR SECTOR GAP,
1736 ;1 BYTE FOR HEADER SYNC,
1737 ;8 BYTES FOR HEADER,
1738 ;GAP 11 BYTES, SYNC 1 BYTE, DATA 512, ECC 4 BYTES
1739 ;GAP 2 BYTES, TOLERANCE 28 BYTES, EXTRA 3
1740
1741 ;THIS GIVES 609 SECTOR CLOCKS WITHOUT DIAGNOSTIC DATA CLOCKS
1742
1743 4$: MOV #609,R1 ;LOAD SECTOR CLOCK COUNTER
1744 5$: BIS #MSTCK,@R0 ;SET SECTOR CLOCK
1745 BIC #MSTCK,@R0 ;CLEAR SECTOR CLOCK
1746 DEC R1 ;COUNT
1747 BNE 5$ ;BRANCH IF 536 NOT DONE
1748
1749 ;NOW 'DTE' SHOULD BE SET, CHANGE SAVED REGISTERS
1750 ;TO EXPECTED VALUES
1751
1752 TST RH70 ;CHECK FOR RH70 CONTROLLER
1753 BEQ JP3 ;SKIP RH70 CODE AND DO RH11 IF NOT
1754
1755 ;RHWC, RHBA AND IR BIT WILL BE MODIFIED FOR RH70C
1756
1757 VAR3: MOV #-252,WC ;SAVED RHWC
1758 VAR4: MOV #WRFROM+<8.*2>,BA ;SAVED RHBA
1759 BR JP4 ;SKIP NEXT RH11 CODE
1760
1761 JP3: MOV #-193,WC ;SAVED RHWC
1762 MOV #WRFROM+<67.*2>,BA ;SAVED RHBA
1763
1764 JP4: BIS #SC!TRE,CS1 ;SAVED RHCS1
1765 VAR5: BIC #IR,CS2 ;SAVED RHCS2
1766 BIS #OR,CS2 ;SAVED RHCS2
1767 BIS #DTE,ER1 ;SAVED RHER1
1768 MOV #401,MR ;SAVED RHMR
1769 BIS #ATA!ERR,DS1 ;SAVED RHDS1
1770 MOV #100,LA ;SAVED RHLA
1771 MOV #1,DST ;SAVED RHDST
1772 MOV ATTENT,AS ;SAVED RHAS
1773
1774 ;NOW READ & SAVE REGISTERS AGAIN SO THAT COMPARISONS CAN BE DONE
1775
1776 JSR R0,SAVER ;READ IN SEQUENCE
1777 RHCW ;FROM HARDWARE REGISTER
1778 REINTO ;INTO CORE BUFFER LOCATION
    
```



```

1778 017064 000023          19.          ;NUMBER OF REGISTERS TO READ
1779
1780          ;FOR RHAS UPPER BYTE
1781 017066 113737 001333 002571      MOVB    AS+1,REINTO+25 ;UPPER RHAS
1782
1783          ;COMPARE CHANGED REGISTER SNAPSHOT BEFORE COMMAND
1784          ;WITH REGISTER SNAPSHOT AFTER COMMAND
1785
1786 017074 004037 032756          JSR     R0,COMPAR      ;COMPARE
1787 017100 001306          WC          ;CHANGED REGISTER SNAPSHOT
1788 017102 002544          REINTO      ;SNAPSHOT AFTER COMMAND
1789 017104 000022          18.          ;NUMBER OF REGISTERS TO COMPARE
1790 017106 017114          2$          ;RETURN FOR ERROR
1791 017110 017114          2$          ;SAME
1792 017112 017134          3$          ;RETURN FOR GOOD COMPARISON
1793
1794 017114 013705 037030          2$:      MOV     ERWORD,R5      ;GETTING READY TO INDEX
1795 017120 060505          ADD     R5,R5          ;DOUBLE ERROR WORD
1796 017122 016537 001230 031620      MOV     RHC-2(R5),REGADR ;FAILING REGISTER
1797 017130 104001          EMT     1
1798
1799          ;CHANGE AFTER FORCING
1800 017132 000207          RTS     PC          ;'DTE' ERROR
1801          ;RETURN
1802 017134          3$:          ;GOOD, REGISTERS COMPARE OK
1803          ;GO ON TO THE NEXT TEST
1804
1815

```

```

:*****
:*TEST 52      SECTOR SELECTION
:*THE SECTOR SELECTION LOGIC IS CHECKED HERE
:*EACH SECTOR ON TRACK ZERO IS WRITTEN INTO.
:*
:*DATA IS - 19 WORDS OF ZEROS - SYNC WORDS, 4 HEADER WORDS
:*1 CRC WORD, 5 WORDS OF ZEROS, 1 SYNC WORD, 100 ZEROS
:* (DATA), 1 SYNC WORD, 70 SECTOR NUMBER TO VARY
:*
:*THE WRITTEN DATA IS CHECKED IN MEMORY
:*****

```

```

1816 017134 000004          TST52:  SCOPE
1817 017136 012706 001100          MOV     #STACK,SP      ;RESET STACK
1818 017142 012737 000052 003610      MOV     #52,TSTNM      ;MOVE #52 TO TEST NUMBER
1819 017150 004737 032054          JSR     PC,CLDISK      ;SETUP GENERAL REGISTERS & CLEAR
1820 017154 012737 000026 001430      MOV     #22.,TAGDTE     ;THE DRIVE
1821          ;22 SECTORS
1822          ;THIS TEST REPEATS
1823          ;ITSELF 22 TIMES
1824
1825          ;THE FOLLOWING INITIALIZES FOR SECTOR 0
1826 017162 005037 017266          CLR     SS3+2          ;HEADER (SECTOR)
1827 017166 012737 000025 017272      MOV     #21.,SS4+2     ;HEADER (KEY1)
1828 017174 012737 000025 017276      MOV     #21.,SS5+2     ;HEADER (KEY2)
1829 017202 005037 017324          CLR     SS7+2          ;DATA (SECTOR)
1830 017206 005037 017406          CLR     SS10+2         ;DATA
1831 017212 005037 017436          CLR     SS12+2         ;SECTOR (SIMULATED DISK)
1832 017216 012737 000025 017444      MOV     #21.,SS13+2    ;KEY1 (SIMULATED DISK)
1833 017224 012737 000025 017452      MOV     #21.,SS14+2    ;KEY2 (SIMULATED DISK)

```

```

1834 017232 005037 017512          CLR      SS15+2 ;SECTOR (RHDST)
1835
1836          ;CLEAR SIMULATED DISK AREA
1837 017236          SS1:
1838 017236 012700 040530          1$:  MOV      #SECGAP,R0          ;POINTER
1839 017242 012701 000460          MOV      #304.,R1          ;COUNTER
1840 017246 005020          2$:  CLR      (R0)+          ;CLEAR SIMULATED DISK AREA
1841 017250 005301          DEC      R1          ;COUNT
1842 017252 001375          BNE      2$
1843
1844
1845          ;SETUP WRITE FROM BUFFER
1846 017254 012700 001500          MOV      #WRFROM,R0
1847
1848          ;HEADER
1849 017260 012720 010000          MOV      #FMT22,(R0)+      ;FORMAT 16 BITS PER WORD
1850          ;CYLINDER 0
1851 017264 012720 000000          SS3:  MOV      #0,(R0)+      ;SECTOR TO VARY
1852 017270 012720 000025          SS4:  MOV      #21.,(R0)+     ;KEY1 TO VARY
1853 017274 012720 000025          SS5:  MOV      #21.,(R0)+     ;KEY2 TO VARY
1854
1855          ;DATA IN WRITE FROM BUFFER ALTHOUGH THIS IS DATA AND NOT
1856          ;HEADER, THE SECTOR WITH SYNC BYTES WILL BE GIVEN AS DATA.
1857
1858          ;DATA IS - 19 WORDS OF ZEROS - SYNC WORDS, 4 HEADER WORDS
1859          ;1 CRC WORD, 5 WORDS OF ZEROS, 1 SYNC WORD, 100 ZEROS
1860          ;(DATA), 1 SYNC WORD, 70 SECTOR NUMBER TO VARY
1861
1862 017300 012705 000023          6$:  MOV      #19.,R5          ;COUNTER
1863 017304 005020          CLR      (R0)+          ;19 ZEROS
1864 017306 005305          DEC      R5          ;COUNT
1865 017310 001375          BNE      6$          ;19 DONE?
1866 017312 013720 037012          MOV      RSYNC,(R0)+      ;SYNC = 14400
1867 017316 012720 010000          MOV      #FMT22,(R0)+     ;CYLINDER 0
1868 017322 012720 000000          SS7:  MOV      #0,(R0)+      ;SECTOR TO VARY
1869 017326 005020          CLR      (R0)+
1870 017330 005020          CLR      (R0)+
1871 017332 004537 033270          JSR      R5,CRC          ;CALCULATE CRC FOR ABOVE 4 WORDS
1872 017336 001550          WRFROM+50          ;4 WORDS START FROM HERE
1873 017340 001560          WRFROM+60          ;PUT CRC HERE
1874
1875 017342 005720          TST      (R0)+          ;INCREMENT R0
1876
1877 017344 012705 000005          8$:  MOV      #5.,R5          ;5 WORDS OF ZEROS
1878 017350 005020          CLR      (R0)+          ;COUNT
1879 017352 005305          DEC      R5          ;BRANCH IF 5 NOT DONE
1880 017354 001375          BNE      8$
1881
1882 017356 013720 037012          MOV      RSYNC,(R0)+      ;SYNC = 14400
1883
1884 017362 012705 000144          9$:  MOV      #100.,R5         ;100 WORDS OF ZEROS
1885 017366 005020          CLR      (R0)+
1886 017370 005305          DEC      R5
1887 017372 001375          BNE      9$
1888
1889 017374 013720 037012          MOV      RSYNC,(R0)+      ;SYNC = 14400
1890 017400 012705 000106          MOV      #70.,R5
    
```



```

1891 017404 012720 000000      SS10:  MOV    #0,(R0)+      ;SECTOR TO VARY
1892 017410 005305              DEC    R5
1893 017412 001374              BNE    SS10
1894
1895                          ;CLEAR REST OF 256 WORDS THAT IS 54 WORDS OF ZEROS
1896
1897 017414 012705 000066      11$:   MOV    #54.,R5
1898 017420 005020              CLR    (R0)+
1899 017422 005305              DEC    R5
1900 017424 001375              BNE    11$
1901
1902                          ;THESE ARE TO BE SET UP FOR DISKLESS USE ONLY
1903
1904 017426 012737 010000 042046      MOV    #FMT22,WCYL      ;FORMAT = 16 BIT WORDS
1905                          ;CYLINDER = 0
1906 017434 012737 000000 042050      SS12:  MOV    #0,WSECTR    ;SECTOR TO VARY
1907 017442 012737 000025 042052      SS13:  MOV    #21.,WKEY1   ;KEY1 TO VARY
1908 017450 012737 000025 042054      SS14:  MOV    #21.,WKEY2   ;KEY2 TO VARY
1909 017456 012737 000312 042106      MOV    #202.,FNWORD    ;202 DATA WORDS
1910 017464 004537 033270              JSR    R5,CRC           ;CALCULATE CRC
1911 017470 042046              WCYL                    ;FIRST WORD
1912 017472 042056              GCRC                     ;PUT HERE
1913
1914                          ;THESE ARE REGULAR SETUPS
1915
1916 017474 012777 177400 161530      MOV    #-256.,@RHWC     ;202 DATA, 4 HEADER
1917 017502 012777 001500 161524      MOV    #WRFROM,@RHBA   ;FILL BUS ADDRESS
1918 017510 012777 000000 161526      SS15:  MOV    #0,@RHDST   ;SECTOR TO VARY
1919 017516 013777 001454 161514      MOV    WRIFOR,@RHCS1   ;GET READY TO DO
1920                          ;WRITE HEADER AND DATA
1921                          ;WITH 62 FUNCTION CODE IN RHCS1
1922 017524 012777 010000 161516      MOV    #FMT22,@RHOF    ;16 BITS PER WORD FORMAT
1923 017532 005077 161514              CLR    @RHCA           ;CYLINDER = 0
1924
1925 017536 005037 001406              CLR    ERFLG$         ;CLEAR ERROR FLAG
1926
1927 017542 004737 032116              JSR    PC,CHECKT      ;CHECK THAT DVA,RDY,DPR,DRY = 1
1928 017546 104401 053407              TYPE   ,CPHALT        ;AND THAT NO OTHERS = 1. CANNOT CON-
1929                          ;TINUE TESTING IF BOTH AREN'T TRUE
1930 017552 000000              HALT                   ;STOP THE TEST
1931
1932 017554 004737 041672              JSR    PC,COMWHD      ;ISSUE 'GO', COUNT SECTOR CLOCKS,
1933                          ;WRITE HEADER AND DATA
1934 017560 005737 001406              TST    ERFLG$         ;HAVE ANY ERRORS OCCURRED ?
1935 017564 001051              BNE    TST53           ;EXIT IF YES-----)
1936
1937 017566 004737 032306              JSR    PC,CHECKE     ;CHECK THAT BITS = 1
1938 017572 104401 053407              TYPE   ,CPHALT        ;CANNOT CONTINUE TESTING IF THEY DON'T
1939 017576 000000              HALT                   ;STOP THE TEST
1940
1941                          ;NOW COMPARE 'DISK' BUFFER WITH 'REINTO' BUFFER
1942 017600 004037 032756              JSR    R0,COMPAR     ;CHECK
1943 017604 001510              WRFROM+8.            ;GOOD BUFFER
1944 017606 040626              DISK                 ;TEST BUFFER
1945 017610 000400              256.                 ;NUMBER OF WORDS
1946 017612 017620              16$                  ;RETURN POINT FOR ERROR HEADER
1947 017614 017624              17$                  ;RETURN POINT FOR ERROR DATA
    
```

1943 017616 017630
 1944 017620
 1945 017622 104007
 1946 017624 000207
 1947 017624 104010
 1948 017626 000207
 1949
 1950
 1951
 1952 017630 005237 017266
 1953 017634 005337 017272
 1954 017640 005337 017276
 1955 017644 005237 017324
 1956 017650 005237 017406
 1957 017654 005237 017436
 1958 017660 005337 017444
 1959 017664 005337 017452
 1960 017670 005237 017512
 1961
 1962 017674 005337 001430
 1963 017700 001001
 1964 017702 000402
 1965 017704 000137 017236
 1966
 1973

18\$: 18\$;RETURN FOR GOOD COMPARISON
 16\$: EMT 7
 RTS PC
 17\$: EMT 10
 RTS PC

;THE FOLLOWING INCREMENTS ARE TO CHANGE THE ABOVE SET UP
 ;TO WRITE ON THE NEXT SECTOR

18\$: INC SS3+2 ;HEADER (SECTOR)
 DEC SS4+2 ;HEADER (KEY1)
 DEC SS5+2 ;HEADER (KEY2)
 INC SS7+2 ;DATA (SECTOR)
 INC SS10+2 ;DATA
 INC SS12+2 ;SECTOR (SIMULATED DISK)
 DEC SS13+2 ;KEY1 (SIMULATED DISK)
 DEC SS14+2 ;KEY2 (SIMULATED DISK)
 INC SS15+2 ;SECTOR (RHDST)
 SS2: DEC TAGDTE ;COUNT DOWN FOR 22 SECTORS
 BNE 1\$;BRANCH IF 22 SECTORS NOT DONE
 BR TST53 ;ALL DONE - GO TO NEXT TEST
 1\$: JMP SS1 ;GO BACK TO NEXT SECTOR

::*****
 ;*TEST 53 WRITE ECC TEST 1
 ;*THIS IS A WRITE ECC TEST
 ;*WRITE CYLINDER0, FORMAT 16 BITS PER WORD
 ;*TRACK 0, SECTOR 1, KEYS 0, NUMBER OF WORDS 256
 ;*OF ALL ZEROS.
 ;*****

1974 017710 000004
 1975 017712 012706 001100
 1976 017716 012737 000053 003610
 1977 017724 012700 040530
 1978 017730 012701 000402
 1979 017734 012720 177777
 1980 017740 005301
 1981 017742 001374
 1982 017744 004737 032054
 1983
 1984
 1985
 1986 017750 012737 177777 001424
 1987 017756 005037 034570
 1988 017762 013737 034564 034566
 1989 017770 013737 034572 034600
 1990 017776 005037 034556
 1991 020002 005037 034560
 1992 020006 005037 034574
 1993 020012 005037 034576
 1994
 1995
 1996

TST53: SCOPE
 MOV #STACK,SP ;RESET STACK
 MOV #53,TSTNM ;MOVE #53 TO TEST NUMBER
 MOV #SECGAP,R0 ;POINTER
 MOV #258,R1 ;COUNTER
 1\$: MOV #-1,(R0)+ ;FILL SIMULATER DISK WITH ONES
 DEC R1
 BNE 1\$
 JSR PC,CLDISK ;THIS IS USED TO SET GENERAL REGISTERS
 ;THESE ARE FOR ECC TEST ONLY
 MOV #-1,TSECC ;THIS IS AN ECC TEST
 CLR POSITI ;CLEAR ERROR POSITION COUNTER
 MOV NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
 MOV HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
 CLR GECC1 ;ECC LOW ORDER TO BE GENERATED
 CLR GECC2 ;ECC HIGH ORDER TO BE GENERATED
 CLR DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
 CLR ZCODE ;CLEAR LEADING ZEROS CLOCK COUNT


```

1997
1998
1999
2000 020016 012737 010000 042046      MOV      #FMT22,WCYL      ;FORMAT22=16BIT WORDS AND
2001                                     ;CYLINDER 0
2002 020024 012737 000001 042050      MOV      #1,WSECTR      ;TRACK=0, SECTOR=1
2003 020032 005037 042052               CLR      WKEY1           ;KEY1=0
2004 020036 005037 042054               CLR      WKEY2           ;KEY2=0
2005 020042 012737 000400 042106      MOV      #256.,FNWORD    ;256 DATA WORDS
2006 020050 004537 033270               JSR      R5,CRC          ;GO TO CALCULATE CRC
2007 020054 042046                       WCYL
2008 020056 042056                       GCRC
2009
2010                                     ;THESE ARE REGULAR SETUPS
2011
2012 020060 012777 177374 161144      MOV      #-260.,@RHWC    ;256 DATA WORDS 4 HEADER WORDS
2013 020066 012700 001500               MOV      #WRFROM,R0     ;THESE TWO INSTRUCTIONS GETS
2014 020072 010077 161136               MOV      R0,@RHBA       ;ADDR. OF WRFROM INTO R0 AND
2015                                     ;BUS ADDRESS REGISTER
2016 020076 012720 010000               MOV      #FMT22,(R0)+   ;FORMAT=16 BIT WORDS
2017                                     ;CYLINDER=0
2018 020102 012720 000001 2$:      MOV      #1,(R0)+       ;TRACK=0, SECTOR=1, KEYS=0
2019 020106 005020                       CLR      (R0)+          ;KEY1=0
2020 020110 005020                       CLR      (R0)+          ;KEY2=0
2021 020112 012705 000400               MOV      #256.,R5       ;COUNTER
2022 020116 012720 000000 3$:      MOV      #0,(R0)+       ;MOVE ALL ZEROS FOR DATA
2023 020122 005305                       DEC      R5
2024 020124 001374                       BNE     3$
2025 020126 012777 000001 161110      MOV      #1,@RHDST     ;BRANCH IF DATA NOT COMPLETE
2026                                     ;TRACK=0 SECTOR=1
2027 020134 004737 032116               JSR      PC,CHECKT     ;CHECK THAT DVA,RDY,DPR,DRY = 1
2028 020140 104401 053407               TYPE    ,CPHALT        ;AND THAT NO OTHERS = 1. CANNOT CON-
2029                                     ;TINUE TESTING IF BOTH AREN'T TRUE
2030 020144 000000                       HALT                    ;STOP THE TEST
2031
2032 020146 013711 001454               MOV      WRIFOR,@R1     ;GET READY FOR WRITE HEADER AND
2033                                     ;DATA WITH 62 IN RHCS1
2034 020152 005037 001406               CLR      ERFLG$        ;CLEAR ERROR FLAG
2035 020156 012777 010000 161064      MOV      #FMT22,@RHOF   ;FORMAT BIT=1 (16 BIT WORDS)
2036 020164 005077 161062               CLR      @RHCA          ;CYLINDER =0
2037 020170 004737 041672               JSR      PC,COMWHD     ;WRITE HEADER AND DATA
2038
2039                                     ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
2040                                     ;FROM THE 'COMWHD' ROUTINE THAT MEANS ALL HEADER ON DISK
2041                                     ;IS GOOD IE. ONLY DATA IS TO BE CHECKED TO SEE IF THEY ARE
2042                                     ;ALL ZEROS AND WRITE DATA GAP AND TOLERANCE GAP TO SEE IF
2043                                     ;THEY ARE ALL ZEROS
2044 020174 005737 001406               TST     ERFLG$         ;HAS ANY ERRORS OCCURED?
2045                                     ;IF WRITE ERROR OCCURS ECC IS NOT CHECKED
2046 020200 001056                       BNE     TST54          ;:BRANCH IF YES
2047
2048                                     ;COMPARE SOFTWARE GENERATED ECC WITH THAT GENERATED BY HARDWARE
2049 020202 023737 034556 041626      CMP     GECC1,WECC1    ;COMPARE SOFTWARE ECC WITH HARDWARE FCC
2050 020210 001402                       BEQ     6$             ;BRANCH IF GOOD
    
```

```

2051 020212 104031      EMT      31
2052 020214 000405      BR      7$          ;BRANCH TO CONTINUE
2053 020216 023737 034560 041630 6$:  CMP      GECC2,WECC2;COMPARE SOFTWARE ECC WITH HARDWARE ECC
2054 020224 001401      BEQ     7$          ;BRANCH IF GOOD
2055 020226 104031      EMT      31
2056
2057
2058 020230      7$:
    020230 004737 032306      JSR      PC,CHECKE      ;CHECK THAT BITS = 1
    020234 104401 053407      TYPE    ,CPHALT        ;CANNOT CONTINUE TESTING IF THEY DON'T
    020240 000000      HALT                    ;STOP THE TEST
2059
2060
2061
2062      ;FILL 'REINTO' BUFFER WITH EXPECTED DATA
2063
2064 020242 004037 031772      JSR      R0,CLAREA      ;FILL REINTO BUFFER
2065 020246 002544      REINTO      ;FROM
2066 020250 003542      REINTO+<255.*2>      ;TO
2067 020252 000000      .WORD    0            ;DATA
2068
2069 020254 013737 034556 003544      MOV      GECC1,REINTO+<256.*2>;FILL ECC1
2070 020262 013737 034560 003546      MOV      GECC2,REINTO+<257.*2>;FILL ECC2
2071 020270 004037 031772      JSR      R0,CLAREA      ;FILL REST
2072 020274 003550      REINTO+<258.*2>      ;FROM
2073 020276 003604      REINTO+<272.*2>      ;TO
2074 020300 000000      0                ;DATA
2075
2076
2077 020302 005037 001406      CLR      ERFLG$ ;CLEAR ERROR FLAG
2078
2079
2080      ;NOW COMPARE 'DISK' BUFFER WITH 'REINTO'
2081
2082 020306 004037 032756      JSR      R0,COMPAR      ;CHECK
2083 020312 002544      REINTO      ;GOOD BUFFER
2084 020314 040626      DISK      ;TEST BUFFER
2085 020316 000402      258.      ;NUMBER OF WORDS CHECKED
2086 020320 020326      4$        ;RETURN POINT FOR ERROR HEADER
2087 020322 020332      5$        ;RETURN POINT FOR ERROR DATA
2088 020324 020336      TST54     ;RETURN FOR GOOD COMPARISON
2089 020326      4$:
    020326 104007      EMT      7
2090 020330 000207      RTS      PC          ;RETURN TO COMPARE
2091 020332      5$:
    020332 104010      EMT      10
2092
2093      ;DATA WORDS
2094      ;WORD NOS 257 AND 258
2095      ;ARE ECC WHICH ARE CHECKED
2096      ;WORD NOS 259
2097      ;IS DATA GAP
2098      ;WORD NOS 260 TO 273
2099 020334 000207      RTS      PC          ;RETURN TO COMPARE
2100
2109      ;*****
    ;*TEST 54      READ ECC ENABLED 1A
    ;*****
    
```



```

: *THIS IS AN ECC READ DATA TEST
: *ERROR CORRECTION IS ENABLED
: *NO ERROR IS INSERTED
: *GOOD DATA USED IS 256 WORDS OF 0
: *COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
: *TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
: *****

```

```

TST54: SCOPE
2110 020336 000004          MOV      #STACK,SP      ;RESET STACK
2111 020340 012706 001100  MOV      #54,TSTNM      ;MOVE #54 TO TEST NUMBER
2112 020344 012737 000054 003610
2113
2114 :          SETUP FOR WHAT IS TO BE READ
2115 :          HEADER CRC IS RESTORED FROM A SUBROUTINE
2116 020352 012746 000000          MOV      #0,-(SP)      ;DATA TO BE READ
2117 020356 012705 000400          MOV      #256,R5      ;COUNTER
2118 020362 012700 040626          MOV      #DISK,R0     ;START OF SIMULATED DISK DATA
2119 020366 011620          1$: MOV      (SP),(R0)+    ;MOVE IN DATA ON TO SIMULATED DISK
2120 020370 005305          DEC      R5           ;COUNT
2121 020372 001375          BNE     1$           ;BRANCH IF 256 NOT COMPLETE
2122 020374 005726          TST     (SP)+        ;UNDO -(SP)
2123 020376 022020          CMP     (R0)+,(R0)+  ;JUMP OVER THE TWO ECC WORDS
2124 020400 012705 000017          MOV      #15,R5      ;1 DATA GAP
2125
2126 020404 005020          2$: CLR      (R0)+      ;14 TOLERANCE GAP
2127 020406 005305          DEC      R5           ;CLEAR DATA GAP, AND
2128 020410 001375          BNE     2$           ;TOLERANCE GAP
2129
2130
2131 020412 004737 035352          JSR     PC,FILLEC    ;INSERT THE TWO ECC WORDS ON THE DISK
2132
2133
2134 :          THESE ARE FOR ECC TEST ONLY
2135
2136 020416 012737 177777 001424          MOV      #-1,TSECC    ;THIS IS AN ECC TEST
2137 020424 005037 034570          CLR     POSITI      ;CLEAR ERROR POSITION COUNTER
2138 020430 013737 034564 034566          MOV      NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2139 020436 013737 034572 034600          MOV      HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2140 020444 005037 034556          CLR     GECC1      ;ECC LOW ORDER TO BE GENERATED
2141 020450 005037 034560          CLR     GECC2      ;ECC HIGH ORDER TO BE GENERATED
2142 020454 005037 034574          CLR     DATENV     ;CLEAR DATA ENVELOPE CLOCK COUNT
2143 020460 005037 034576          CLR     ZCODE      ;CLEAR LEADING ZEROS CLOCK COUNT
2144
2145
2146 :          THESE ARE TO SETUP FOR DISKLESS USE ONLY
2147
2148 020464 012737 010000 036710          MOV      #FMT22,CYL   ;16 BITS PER WORD
2149
2150 020472 112737 000000 036713          MOV     #0,SECOTR+1  ;CYLINDER 0, FORMAT 16 BITS
2151 020500 112737 000000 036712          MOV     #0,SECOTR    ;TRACK 0
2152 020506 012737 000000 036714          MOV     #0,KEY1      ;SECTOR 0
2153 020514 012737 000000 036716          MOV     #0,KEY2      ;KEY1=0
2154 020522 012737 000400 036770          MOV     #256,DAWORD  ;KEY2=0
2155 020530 005037 036720          MOV     #256,DAWORD  ;NO. OF DATA WORDS
2156 020534 004537 033270          CLR     X           ;THIS IS A READ COMMAND
2157 020540 036710          JSR     R5,CRC      ;GO TO CALCULATE CRC
2158 020542 040610          CYL
                WCRC

```

```

2159
2160
2161
2162
2163           ;THESE ARE REGULAR SETUPS
2164
2165 020544 004737 032054          JSR    PC,CLDISK      ;SETUP GENERAL REGISTERS
2166 020550 012777 177374 160454  MOV    #-256,-4,@RHWC ;256. DATA 4 HEADER WORDS
2167 020556 012777 002544 160450  MOV    #REINTO,@RHBA  ;STARTING ADDRESS OF READ BUFFER
2168 020564 112746 000000          MOV    #0,-(SP)      ;IN LOWER BYTE GET SECTOR
2169 020570 112766 000000 000001  MOV    #0,1(SP)      ;GET TRACK IN HIGHER BYTE
2170 020576 012677 160442          MOV    (SP)+,@RHDST  ;TRACK/SECTOR IN RHDST
2171 020602 012777 010000 160440  MOV    #FMT22,@RHOF  ;16 BITS PER WORD
2172                                     ;ECC CORRECTION NOT INHIBIT
2173                                     ;BECAUSE ECC IS NOT GOING
2174                                     ;TO BE CHECKED
2175 020610 005077 160436          CLR    @RHCA         ;CYLINDER 0
2176
2177 020614 004737 032116          JSR    PC,CHECKT     ;CHECK THAT DVA,RDY,DPR,DRY = 1
      020620 104401 053407          TYPE   ,CPHALT      ;AND THAT NO OTHERS = 1. CANNOT CON-
      020624 000000          HALT                ;TINUE TESTING IF BOTH AREN'T TRUE
      ;STOP THE TEST
2178
2179 020626 013711 001460          MOV    REFOR,@R1    ;READ HEADER AND DATA=72
2180 020632 005037 001406          CLR    ERFLG$      ;CLEAR ERROR FLAG
2181 020636 004737 036550          JSR    PC,COMHD     ;READ HEADER AND DATA
2182                                     ;IF THERE ARE READ ERRORS THEN
2183                                     ;ECC WILL NOT BE CHECKED
2184
2185
2186           ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
2187           ;FROM THE "COMHD" ROUTINE THAT MEANS SECTOR GAP,
2188           ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
2189           ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
2190           ;DETECTED
2191           ;HEADER AND DATA ARE TO BE CHECKED.
2192           ;IN CHECKING READ DATA THE WRITE FROM BUFFER
2193           ;"WRFROM" IS FILLED WITH EXPECTED DATA AND
2194           ;COMPARISONS ARE MADE
2195
2196 020642 005737 001406          TST    ERFLG$      ;ANY ERRORS ALREADY THERE
2197 020646 001102          BNE    TST55       ;BRANCH IF YES
2198 020650 004737 031554          JSR    PC,PUTREG    ;SAVE REGISTERS
2199 020654 005737 001316          TST    ER1         ;NO ERRORS SHOULD BE SET
2200 020660 001401          BEQ    6$          ;BRANCH IF NO ERRORS SET
2201 020662 104032          EMT    32
2202                                     ;ONLY 11 OF THE 32 BITS CAN BE SEEN
2203                                     ;IN THE PATERN REGISTER
2204                                     ;DCK SHOULD BE SET IN RHER1
2205 020664 013746 034556          6$:  MOV    GECC1,-(SP) ;GET PATTERN REGISTER
2206 020670 042716 174000          BIC    #174000,(SP) ;KEEP ONLY 11 BITS
2207 020674 022637 001346          CMP    (SP)+,EC2   ;COMPARE PATTERN REGISTER
2208 020700 001401          BEQ    7$          ;BRANCH IF GOOD
2209 020702 104032          EMT    32
2210
2211
2212
    
```



```

2213
2214           ;ADD 16 MAINTENANCE CLOCKS TO
2215           ;BRING EBL DOWN
2216
2217 020704 012700 000020      7$:   MOV     #16.,R0           ;COUNTER
2218 020710 052777 000002 160342 8$:   BIS     #MCLK,@RHMR       ;SET CLOCK
2219 020716 042777 000002 160334      BIC     #MCLK,@RHMR       ;CLEAR CLOCK
2220 020724 005300              DEC     R0                 ;COUNT
2221 020726 001370              BNE     8$                 ;BRANCH IF 16 CLOCKS NOT DONE
2222 020730 004737 032306      JSR     PC,CHECKE         ;CHECK THAT BITS = 1
                                020734 104401 053407      TYPE    ,CPHALT           ;CANNOT CONTINUE TESTING IF THEY DON'T
                                020740 000000              HALT                    ;STOP THE TEST
2223 020742 012700 001500      MOV     #WRFROM,R0        ;GETTING READY TO FILL EXPECTED DATA
2224 020746 012720 010000      MOV     #0!FMT22,(R0)+   ;CYLINDER 0
2225 020752 112746 000000      MOV     #0,-(SP)         ;IN LOWER BYTE GET SECTOR
2226 020756 112766 000000 000001      MOV     #0,1(SP)        ;GET TRACK IN HIGHER BYTE
2227 020764 012620              MOV     (SP)+,(R0)+      ;GET TRACK/SECTOR IN BUFFER
2228 020766 012720 000000      MOV     #0,(R0)+        ;KEY1 IN BUFFER
2229 020772 012720 000000      MOV     #0,(R0)+        ;KEY2 IN BUFFER
2230 020776 012701 000400      MOV     #256.,R1        ;DATA WORD COUNTER
2231 021002 012702 000000      MOV     #0,R2           ;DATA
2232 021006 010220      3$:   MOV     R2,(R0)+        ;DATA INTO BUFFER
2233 021010 005301              DEC     R1                 ;COUNT
2234 021012 001375              BNE     3$                 ;BRANCH IF 256 NOT DONE
2235
2236
2237 021014 005037 001406      CLR     ERFLG$          ;CLEAR ERROR FLAG
2238 021020 004737 031554      JSR     PC,PUTREG        ;SAVE REGISTERS
2239
2240
2241
2242           ;NOW READ DATA BUFFER WILL BE CHECKED
2243
2244 021024 004037 032756      JSR     R0,COMPAR        ;CHECK
2245 021030 001500              WRFROM                    ;GOOD BUFFER
2246 021032 002544              REINTO                    ;TEST BUFFER
2247 021034 000404              4+256.                    ;NUMBER OF WORDS CHECKED
2248 021036 021044              4$                          ;RETURN POINT FOR ERROR HEADER
2249 021040 021050              5$                          ;RETURN POINT FOR ERROR DATA
2250 021042 021054              TST55                      ;RETURN FOR GOOD COMPARISON
2251 021044      4$:   EMT     4
                                021044 104004      RTS     PC                 ;RETURN TO "COMPAR"
2252 021046 000207      5$:   EMT     5
2253 021050      RTS     5
                                021050 104005
2254
2255
2256 021052 000207      RTS     PC                 ;HEADER WORDS
2257
2266

```

```

:*****
:*TEST 55      READ ECC ENABLED 1B
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A CORRECTABLE ERROR IS INSERTED IN BIT POSITION 32
:*GOOD DATA USED IS 256 WORDS OF 0
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA

```

```

*****
2267 021054 000004          TST55: SCOPE
2268 021056 012706 001100      MOV      #STACK,SP          ;RESET STACK
2269 021062 012737 000055 003610  MOV      #55,TSTNM          ;MOVE #55 TO TEST NUMBER
2270
2271
2272          ;SETUP FOR WHAT IS TO BE READ
2273          ;HEADER CRC IS RESTORED FROM A SUBROUTINE
2274
2275 021070 012746 000000      MOV      #0,-(SP)           ;DATA TO BE READ
2276 021074 012705 000400      MOV      #256,R5           ;COUNTER
2277 021100 012700 040626      MOV      #DISK,R0          ;START OF SIMULATED DISK DATA
2278 021104 011620          1$:  MOV      (SP),(R0)+        ;MOVE IN DATA ON TO SIMULATED DISK
2279 021106 005305          DEC      R5                 ;COUNT
2280 021110 001375          BNE     1$                  ;BRANCH IF 256 NOT COMPLETE
2281 021112 005726          TST     (SP)+              ;UNDO -(SP)
2282 021114 022020          CMP     (R0)+,(R0)+        ;JUMP OVER THE TWO ECC WORDS
2283 021116 012705 000017      MOV      #15,R5 ;1 DATA GAP
2284
2285 021122 005020          2$:  CLR     (R0)+              ;14 TOLERANCE GAP
2286 021124 005305          DEC     R5                 ;CLEAR DATA GAP, AND
2287 021126 001375          BNE     2$                  ;TOLERANCE GAP
2288
2289
2290 021130 004737 035352      JSR     PC,FILLEC          ;INSERT ECC IN PROPER PLACE ON DISK
2291
2292
2293
2294          ;THESE ARE FOR ECC TEST ONLY
2295
2296 021134 012737 177777 001424  MOV      #-1,TSECC          ;THIS IS AN ECC TEST
2297 021142 005037 034570      CLR     POSITI ;CLEAR ERROR POSITION COUNTER
2298 021146 013737 034564 034566  MOV      NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2299 021154 013737 034572 034600  MOV      HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2300 021162 005037 034556      CLR     GECC1              ;ECC LOW ORDER TO BE GENERATED
2301 021166 005037 034560      CLR     GECC2              ;ECC HIGH ORDER TO BE GENERATED
2302 021172 005037 034574      CLR     DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
2303 021176 005037 034576      CLR     ZCODE              ;CLEAR LEADING ZEROS CLOCK COUNT
2304
2305
2306          ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
2307
2308 021202 012737 010000 036710  MOV      #FMT22,CYL        ;16 BITS PER WORD
2309
2310 021210 112737 000000 036713  MOV     #0,SECOTR+1        ;CYLINDER 0, FORMAT 16 BITS
2311 021216 112737 000000 036712  MOV     #0,SECOTR          ;TRACK 0
2312 021224 012737 000000 036714  MOV     #0,KEY1 ;KEY1=0    ;SECTOR 0
2313 021232 012737 000000 036716  MOV     #0,KEY2 ;KEY2=0
2314 021240 012737 000400 036770  MOV     #256,DAWORD ;NO. OF DATA WORDS
2315 021246 005037 036720      CLR     X                  ;THIS IS A READ COMMAND
2316 021252 004537 033270      JSR     R5,CRC ;GO TO CALCULATE CRC
2317 021256 036710      CYL
2318 021260 040610      WCRC
2319
2320
2321          ;THIS IS TO INSERT ERROR
    
```



```

2322      ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
2323      ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
2324      ;THIS MOVE
2325
2326 021262 012737 100000 040630      MOV      #100000,DISK+2 ;FORCE ERROR ON BIT NUMBER 32
2327      ;SO ERROR POSITION REGISTER WILL SHOW
2328      ;22
2329 021270 012737 000026 021444      MOV      #22.,8$ ;INSERT POSITION REG.
2330
2331
2332      ;THESE ARE REGULAR SETUPS
2333
2334 021276 004737 032054      JSR      PC,CLDISK      ;SETUP GENERAL REGISTERS
2335 021302 012777 177374 157722      MOV      #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
2336 021310 012777 002544 157716      MOV      #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
2337 021316 112746 000000      MOVVB   #0,-(SP) ;IN LOWER BYTE GET SECTOR
2338 021322 112766 000000 000001      MOVVB   #0,1(SP) ;GET TRACK IN HIGHER BYTE
2339 021330 012677 157710      MOV      (SP)+,@RHDST ;TRACK/SECTOR IN RHDST
2340 021334 012777 010000 157706      MOV      #FMT22,@RHOF ;16 BITS PER WORD
2341      ;ECC CORRECTION NOT INHIBIT
2342      ;BECAUSE ECC IS NOT GOING
2343      ;TO BE CHECKED
2344 021342 005077 157704      CLR      @RHCA ;CYLINDER 0
2345
2346 021346 004737 032116      JSR      PC,CHECKT ;CHECK THAT DVA,RDY,DPR,DRY = 1
2347 021352 104401 053407      TYPE    ,CPHALT ;AND THAT NO OTHERS = 1. CANNOT CON-
2348      ;TINUE TESTING IF BOTH AREN'T TRUE
2349      ;STOP THE TEST
2350 021356 000000      HALT
2351
2352 021360 013711 001460      MOV      REFOR,@R1 ;READ HEADER AND DATA=72
2353 021364 005037 001406      CLR      ERFLG$ ;CLEAR ERROR FLAG
2354 021370 004737 036550      JSR      PC,COMHD ;READ HEADER AND DATA
2355      ;IF THERE ARE READ ERRORS THEN
2356      ;ECC WILL NOT BE CHECKED
2357
2358      ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
2359      ;FROM THE 'COMHD' ROUTINE THAT MEANS SECTOR GAP,
2360      ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
2361      ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
2362      ;DETECTED
2363      ;HEADER AND DATA ARE TO BE CHECKED.
2364      ;IN CHECKING READ DATA THE WRITE FROM BUFFER
2365      ;'WRFROM' IS FILLED WITH EXPECTED DATA AND
2366      ;COMPARISONS ARE MADE
2367
2368 021374 005737 001406      TST      ERFLG$ ;ANY ERRORS ALREADY THERE
2369 021400 001077      BNE     TST56 ;BRANCH IF YES
2370 021402 004737 031554      JSR      PC,PUTREG ;SAVE REGISTERS
2371 021406 022737 100000 001316      CMP      #DCK,ER1 ;ONLY DATA CHECK ERROR SHOULD BE SET
2372 021414 001401      BEQ     6$ ;BRANCH IF YES
2373 021416 104032      EMT     32
2374      ;ZERO
2375      ;ONLY 11 OF THE 32 BITS CAN BE SEEN
2376      ;IN THE PATERN REGISTER
2377      ;DCK SHOULD BE SET IN RH1
2378      ;GET PATTERN REGISTER
2379 021420 013746 034556      6$: MOV      GECC1,-(SP)

```

```
2376 021424 042716 174000      BIC    #174000,(SP)      ;KEEP ONLY 11 BITS
2377 021430 022637 001346      CMP    (SP)+,EC2       ;COMPARE PATTERN REGISTER
2378 021434 001401              BEQ    7$              ;BRANCH IF GOOD
2379 021436 104032              EMT    32
2380
2381 021440 004037 035200      7$:   JSR    R0,ECORR   ;GO TO ECC CORRECTION PROCESS
2382 021444 000026              8$:   22.             ;EXPECTED POSITION REG. WHEN CORRECTION
2383                               ;IS COMPLETE
2384
2385
2386
2387 021446 004737 032306      JSR    PC,CHECKE      ;CHECK THAT BITS = 1
      021452 104401 053407      TYPE  ,CPHALT        ;CANNOT CONTINUE TESTING IF THEY DON'T
      021456 000000              HALT                  ;STOP THE TEST
2388 021460 012700 001500      MOV    #WRFROM,R0     ;GETTING READY TO FILL EXPECTED DATA
2389 021464 012720 010000      MOV    #0!FMT22,(R0)+ ;CYLINDER 0
2390 021470 112746 000000      MOV    #0,-(SP)      ;IN LOWER BYTE GET SECTOR
2391 021474 112766 000000 000001  MOV    #0,1(SP)      ;GET TRACK IN HIGHER BYTE
2392 021502 012620              MOV    (SP)+,(R0)+   ;GET TRACK/SECTOR IN BUFFER
2393 021504 012720 000000      MOV    #0,(R0)+      ;KEY1 IN BUFFER
2394 021510 012720 000000      MOV    #0,(R0)+      ;KEY2 IN BUFFER
2395 021514 012701 000400      MOV    #256.,R1      ;DATA WORD COUNTER
2396 021520 012702 000000      MOV    #0,R2         ;DATA
2397 021524 010220              3$:   MOV    R2,(R0)+   ;DATA INTO BUFFER
2398 021526 005301              DEC    R1             ;COUNT
2399 021530 001375              BNE    3$            ;BRANCH IF 256 NOT DONE
2400
2401                               ;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
2402                               ;NOW THE INSERTED ERROR WILL BE PUT IN
2403 021532 012737 100000 001512  MOV    #100000,WRFROM+<5*2> ;INSERTED ERROR
2404
2405
2406
2407 021540 005037 001406      CLR    ERFLG$        ;CLEAR ERROR FLAG
2408 021544 004737 031554      JSR    PC,PUTREG     ;SAVE REGISTERS
2409
2410
2411                               ;NOW READ DATA BUFFER WILL BE CHECKED
2412
2413 021550 004037 032756      JSR    R0,COMPAR     ;CHECK
2414 021554 001500              WRFROM              ;GOOD BUFFER
2415 021556 002544              REINTO              ;TEST BUFFER
2416 021560 000404              4+256.             ;NUMBER OF WORDS CHECKED
2417 021562 021570              4$                 ;RETURN POINT FOR ERROR HEADER
2418 021564 021574              5$                 ;RETURN POINT FOR ERROR DATA
2419 021566 021600              TST56              ;RETURN FOR GOOD COMPARISON
2420 021570              4$:   EMT    4
      021570 104004              RTS    PC           ;RETURN TO 'COMPAR'
2421 021572 000207              5$:   EMT    5
2422 021574              EMT    5
      021574 104005              RTS    PC           ;HEADER WORDS
      ;5 TO 260 ARE DATA WORDS
      ;RETURN TO 'COMPAR'
2423
2424
2425 021576 000207      RTS    PC
2426
2435
```

:*TEST 56 READ ECC ENABLED 1C

:*THIS IS AN ECC READ DATA TEST
 :*ERROR CORRECTION IS ENABLED
 :*A NON CORRECTABLE ERROR IS INSERTED IN BIT POSITION 21 THRU 32
 :*GOOD DATA USED IS 256 WORDS OF 0
 :*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
 :*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
 :*****

```

TST56: SCOPE
2436 021600 000004          MOV      #STACK,SP      ;RESET STACK
2437 021602 012706 001100  MOV      #56,TSTNM     ;MOVE #56 TO TEST NUMBER
2438 021606 012737 000056 003610
2439
2440 :          SETUP FOR WHAT IS TO BE READ
2441 :          HEADER CRC IS RESTORED FROM A SUBROUTINE
2442 021614 012746 000000          MOV      #0,-(SP)      ;DATA TO BE READ
2443 021620 012705 000400          MOV      #256.,R5     ;COUNTER
2444 021624 012700 040626          MOV      #DISK,R0     ;START OF SIMULATED DISK DATA
2445 021630 011620          1$: MOV      (SP),(R0)+   ;MOVE IN DATA ON TO SIMULATED DISK
2446 021632 005305          DEC      R5           ;COUNT
2447 021634 001375          BNE     1$           ;BRANCH IF 256 NOT COMPLETE
2448 021636 005726          TST     (SP)+        ;UNDO -(SP)
2449 021640 022020          CMP     (R0)+,(R0)+  ;JUMP OVER THE TWO ECC WORDS
2450 021642 012705 000017          MOV      #15.,R5     ;1 DATA GAP
2451 :          ;14 TOLERANCE GAP
2452 021646 005020          2$: CLR     (R0)+      ;CLEAR DATA GAP, AND
2453 021650 005305          DEC     R5           ;TOLERANCE GAP
2454 021652 001375          BNE     2$           ;BRANCH IF NOT COMPLETE
2455
2456
2457 021654 004737 035352          JSR     PC,FILLEC    ;INSERT THE TWO ECC WORDS ON THE DISK
2458 :          ;IN THE CORRECT PLACE
2459
2460 :THESE ARE FOR ECC TEST ONLY
2461
2462 021660 012737 177777 001424          MOV      #-1,TSECC    ;THIS IS AN ECC TEST
2463 021666 005037 034570          CLR     POSITI      ;CLEAR ERROR POSITION COUNTER
2464 021672 013737 034564 034566          MOV      NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2465 021700 013737 034572 034600          MOV      HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2466 021706 005037 034556          CLR     GECC1       ;ECC LOW ORDER TO BE GENERATED
2467 021712 005037 034560          CLR     GECC2       ;ECC HIGH ORDER TO BE GENERATED
2468 021716 005037 034574          CLR     DATENV      ;CLEAR DATA ENVELOPE CLOCK COUNT
2469 021722 005037 034576          CLR     ZCODE       ;CLEAR LEADING ZEROS CLOCK COUNT
2470
2471
2472 :THESE ARE TO SETUP FOR DISKLESS USE ONLY
2473
2474 021726 012737 010000 036710          MOV      #FMT22,CYL  ;16 BITS PER WORD
2475 :          ;CYLINDER 0, FORMAT 16 BITS
2476 021734 112737 000000 036713          MOV     #0,SECOTR+1  ;TRACK 0
2477 021742 112737 000000 036712          MOV     #0,SECOTR    ;SECTOR 0
2478 021750 012737 000000 036714          MOV     #0,KEY1 ;KEY1=0
2479 021756 012737 000000 036716          MOV     #0,KEY2 ;KEY2=0
2480 021764 012737 000400 036770          MOV     #256.,DAWORD ;NO. OF DATA WORDS
2481 021772 005037 036720          CLR     X           ;THIS IS A READ COMMAND
2482 021776 004537 033270          JSR     R5,CRC      ;GO TO CALCULATE CRC
2483 022002 036710
2484 022004 040610
    
```

```

2485
2486
2487
2488
2489
2490
2491 022006 012737 177760 040630
2492
2493
2494 022014 012737 010040 022170
2495
2496
2497
2498
2499 022022 004737 032054
2500 022026 012777 177374 157176
2501 022034 012777 002544 157172
2502 022042 112746 000000
2503 022046 112766 000000 000001
2504 022054 012677 157164
2505 022060 012777 010000 157162
2506
2507
2508
2509 022066 005077 157160
2510 022072 004737 032116
    022076 104401 053407
2511 022102 000000
2512 022104 013711 001460
2513 022110 005037 001406
2514 022114 004737 036550
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527 022120 005737 001406
2528 022124 001106
2529 022126 004737 031554 001316
2530 022132 022737 100000
2531 022140 001401
2532 022142 104032
2533
2534
2535
2536
2537 022144 013746 034556
2538 022150 042716 174000

:THIS IS TO INSERT ERROR
:THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
:THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
:THIS MOVE
    MOV #177760,DISK+2 :FORCE ERROR ON BIT NUMBER 21 THRU 32
    :SO ERROR POSITION REGISTER WILL SHOW
    :22
    MOV #4128.,8$ :INSERT POSITION REG.

:THESE ARE REGULAR SETUPS
    JSR PC,CLDISK :SETUP GENERAL REGISTERS
    MOV #-256.-4.,@RHWC :256. DATA 4 HEADER WORDS
    MOV #REINTO,@RHBA :STARTING ADDRESS OF READ BUFFER
    MOVB #0,-(SP) :IN LOWER BYTE GET SECTOR
    MOVB #0,1(SP) :GET TRACK IN HIGHER BYTE
    MOV (SP)+,@RHDST :TRACK/SECTOR IN RHDST
    MOV #FMT22,@RHOF :16 BITS PER WORD
    :ECC CORRECTION NOT INHIBIT
    :BECAUSE ECC IS NOT GOING
    :TO BE CHECKED
    CLR @RHCA :CYLINDER 0
    JSR PC,CHECKT :CHECK THAT DVA,RDY,DPR,DRY = 1
    TYPE ,CPHALT :AND THAT NO OTHERS = 1. CANNOT CON-
    :TINUE TESTING IF BOTH AREN'T TRUE
    HALT :STOP THE TEST
    MOV REFOR,@R1 :READ HEADER AND DATA=72
    CLR ERFLG$ ;CLEAR ERROR FLAG
    JSR PC,COMHD :READ HEADER AND DATA
    :IF THERE ARE READ ERRORS THEN
    :ECC WILL NOT BE CHECKED

:IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
:FROM THE 'COMHD' ROUTINE THAT MEANS SECTOR GAP,
:FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
:SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
:DETECTED
:HEADER AND DATA ARE TO BE CHECKED.
:IN CHECKING READ DATA THE WRITE FROM BUFFER
: 'WRFROM' IS FILLED WITH EXPECTED DATA AND
:COMPARISONS ARE MADE
    TST ERFLG$ ;ANY ERRORS ALREADY THERE
    BNE TST57 :BRANCH IF YES
    JSR PC,PUTREG :SAVE REGISTERS
    CMP #DCK,ER1 :ONLY DATA CHECK ERROR SHOULD BE SET
    BEQ 6$ :BRANCH IF YES
    EMT 32
    :ZERO
    :ONLY 11 OF THE 32 BITS CAN BE SEEN
    :IN THE PATTERN REGISTER
    :DCK SHOULD BE SET IN RHER1
    6$: MOV GECC1,-(SP) :GET PATTERN REGISTER
    BIC #174000,(SP) :KEEP ONLY 11 BITS
    
```



```

2539 022154 022637 001346      CMP      (SP)+,EC2      ;COMPARE PATTERN REGISTER
2540 022160 001401              BEQ      7$             ;BRANCH IF GOOD
2541 022162 104032              EMT
2542
2543 022164 004037 035200      7$:      JSR      R0,ECORR      ;GO TO ECC CORRECTION PROCESS
2544 022170 000000              8$:      .WORD
2545
2546
2547
2548
2549 022172 004737 031554      JSR      PC,PUTREG      ;SAVE REGISTERS
2550 022176 022737 100100 001316  CMP      #DCK!ECH,ER1   ;WITH ERRORS INSERTED IN BIT POSITION 21
2551
2552 022204 001401              BEQ      9$             ;THRU 32 HARD ERROR BIT SHOULD SET
2553 022206 104036              EMT      36            ;BRANCH IF GOOD
2554
2555
2556
2557
2558 022210              9$:
022210 004737 032306      JSR      PC,CHECKE      ;CHECK THAT BITS = 1
022214 104401 053407      TYPE     ,CPHALT       ;CANNOT CONTINUE TESTING IF THEY DON'T
022220 000000              HALT                    ;STOP THE TEST
2559 022222 012700 001500      MOV      #WRFROM,R0     ;GETTING READY TO FILL EXPECTED DATA
2560 022226 012720 010000      MOV      #0!FMT22,(R0)+ ;CYLINDER 0
2561 022232 112746 000000      MOV      #0,-(SP)       ;IN LOWER BYTE GET SECTOR
2562 022236 112766 000000 000001  MOV      #0,1(SP)       ;GET TRACK IN HIGHER BYTE
2563 022244 012620              MOV      (SP)+,(R0)+    ;GET TRACK/SECTOR IN BUFFER
2564 022246 012720 000000      MOV      #0,(R0)+      ;KEY1 IN BUFFER
2565 022252 012720 000000      MOV      #0,(R0)+      ;KEY2 IN BUFFER
2566 022256 012701 000400      MOV      #256.,R1      ;DATA WORD COUNTER
2567 022262 012702 000000      MOV      #0,R2         ;DATA
2568 022266 010220              3$:      MOV      R2,(R0)+      ;DATA INTO BUFFER
2569 022270 005301              DEC      R1             ;COUNT
2570 022272 001375              BNE     3$             ;BRANCH IF 256 NOT DONE
2571
2572
2573
2574
2575 022274 012737 177760 001512  MOV      #177760,WRFROM+<5*2> ;INSERTED ERROR
2576
2577
2578
2579 022302 005037 001406      CLR      ERFLG$        ;CLEAR ERROR FLAG
2580 022306 004737 031554      JSR      PC,PUTREG      ;SAVE REGISTERS
2581
2582
2583
2584
2585 022312 004037 032756      JSR      R0,COMPAR      ;CHECK
2586 022316 001500              WRFROM                  ;GOOD BUFFER
2587 022320 002544              REINTO                   ;TEST BUFFER
2588 022322 000404              4+256.                  ;NUMBER OF WORDS CHECKED
2589 022324 022332              4$                       ;RETURN POINT FOR ERROR HEADER
2590 022326 022336              5$                       ;RETURN POINT FOR ERROR DATA
2591 022330 022342              TST57                    ;RETURN FOR GOOD COMPARISON
2592 022332              4$:

```

```

2593 022332 104004      EMT 4
2594 022334 000207      RTS PC      ;RETURN TO "COMPAR"
2595 022336 104005      5$: EMT 5
2596 022336 104005      ;HEADER WORDS
2597 022340 000207      RTS PC      ;5 TO 260 ARE DATA WORDS
2598                                ;RETURN TO "COMPAR"
2605
    
```

```

*****
*TEST 57      WRITE ECC TEST 2
*THIS IS A WRITE ECC TEST
*WRITE CYLINDER 0, FORMAT 16 BITS PER WORD
*TRACK 0, SECTOR 1, KEYS 0, NUMBER OF WORDS 256
*OF ALL ONES.
*****
    
```

```

2606 022342 000004      TST57: SCOPE
2607 022344 012706 001100      MOV #STACK,SP      ;RESET STACK
2608 022350 012737 000057 003610      MOV #57,TSTNM      ;MOVE #57 TO TEST NUMBER
2609 022356 012700 040530      MOV #SECGAP,R0     ;POINTER
2610 022362 012701 000460      MOV #304.,R1       ;COUNTER
2611 022366 005020      1$: CLR (R0)+          ;CLEAR SIMULATED DISK AREA
2612 022370 005301      DEC R1
2613 022372 001375      BNE 1$
2614 022374 004737 032054      JSR PC,CLDISK      ;THIS IS USED TO SET GENERAL REGISTERS
2615
    
```

;THESE ARE FOR ECC TEST ONLY

```

2618 022400 012737 177777 001424      MOV #-1,TSECC      ;THIS IS AN ECC TEST
2619 022406 005037 034570      CLR POSITI         ;CLEAR ERROR POSITION COUNTER
2620 022412 013737 034564 034566      MOV NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2621 022420 013737 034572 034600      MOV HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2622 022426 005037 034556      CLR GECC1          ;ECC LOW ORDER TO BE GENERATED
2623 022432 005037 034560      CLR GECC2          ;ECC HIGH ORDER TO BE GENERATED
2624 022436 005037 034574      CLR DATENV         ;CLEAR DATA ENVELOPE CLOCK COUNT
2625 022442 005037 034576      CLR ZCODE          ;CLEAR LEADING ZEROS CLOCK COUNT
2626
2627
2628
2629
    
```

;THESE ARE TO BE SETUP FOR DISKLESS USE ONLY

```

2632 022446 012737 010000 042046      MOV #FMT22,WCYL    ;FORMAT22=16BIT WORDS AND
2633                                ;CYLINDER 0
2634 022454 012737 000001 042050      MOV #1,WSECTR      ;TRACK=0, SECTOR=1
2635 022462 005037 042052      CLR WKEY1          ;KEY1=0
2636 022466 005037 042054      CLR WKEY2          ;KEY2=0
2637 022472 012737 000400 042106      MOV #256.,FNWORD   ;256 DATA WORDS
2638 022500 004537 033270      JSR R5,CRC         ;GO TO CALCULATE CRC
2639 022504 042046      WCYL
2640 022506 042056      GCRC
2641
    
```

;THESE ARE REGULAR SETUPS

```

2644 022510 012777 177374 156514      MOV #-260.,@RHWC   ;256 DATA WORDS 4 HEADER WORDS
2645 022516 012700 001500      MOV #WRFROM,R0     ;THESE TWO INSTRUCTIONS GETS
2646 022522 010077 156506      MOV R0,@RHBA       ;ADDR. OF WRFROM INTO R0 AND
    
```



```

2647
2648 022526 012720 010000      MOV      #FMT22,(R0)+      ;BUS ADDRESS REGISTER
2649                                ;FORMAT=16 BIT WORDS
2650 022532 012720 000001      2$: MOV      #1,(R0)+      ;CYLINDER=0
2651 022536 005020                CLR      (R0)+            ;TRACK=0, SECTOR=1, KEYS=0
2652 022540 005020                CLR      (R0)+            ;KEY1=0
2653 022542 012705 000400      MOV      #256,R5          ;KEY2=0
2654 022546 012720 177777      3$: MOV      #-1,(R0)+     ;COUNTER
2655 022552 005305                DEC      R5                ;MOVE ALL ONES FOR DATA
2656 022554 001374                BNE     3$                 ;BRANCH IF DATA NOT COMPLETE
2657 022556 012777 000001 156460 MOV      #1,@RHDST        ;TRACK=0 SECTOR=1
2658
2659 022564 004737 032116      JSR     PC,CHECKT        ;CHECK THAT DVA,RDY,DPR,DRY = 1
      022570 104401 053407      TYPE    ,CPHALT         ;AND THAT NO OTHERS = 1. CANNOT CON-
                                ;TINUE TESTING IF BOTH AREN'T TRUE
      022574 000000                HALT                      ;STOP THE TEST
2660
2661 022576 013711 001454      MOV      WRIFOR,@R1      ;GET READY FOR WRITE HEADER AND
2662                                ;DATA WITH 62 IN RHCS1
2663 022602 005037 001406      CLR     ERFLG$ ;CLEAR ERROR FLAG
2664 022606 012777 010000 156434 MOV      #FMT22,@RHOF    ;FORMAT BIT=1 (16 BIT WORDS)
2665 022614 005077 156432      CLR     @RHCA            ;CYLINDER =0
2666 022620 004737 041672      JSR     PC,COMWHD        ;WRITE HEADER AND DATA
2667
2668                                ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
2669                                ;FROM THE "COMWHD" ROUTINE THAT MEANS ALL HEADER ON DISK
2670                                ;IS GOOD IE. ONLY DATA IS TO BE CHECKED TO SEE IF THEY ARE
2671                                ;ALL ONES AND WRITE DATA GAP AND TOLERANCE GAP TO SEE IF
2672                                ;THEY ARE ALL ZEROS
2673
2674 022624 005737 001406      TST     ERFLG$ ;HAS ANY ERRORS OCCURED?
2675                                ;IF WRITE ERROR OCCURS ECC IS NOT CHECKED
2676 022630 001056                BNE     TST60             ;;BRANCH IF YES
2677
2678                                ;COMPARE SOFTWARE GENERATED ECC WITH THAT GENERATED BY HARDWARE
2679
2680
2681 022632 023737 034556 041626  CMP     GECC1,WECC1;COMPARE SOFTWARE ECC WITH HARDWARE ECC
2682 022640 001402                BEQ     6$                 ;BRANCH IF GOOD
2683 022642 104031                EMT     31
2684 022644 000405                BR      7$                 ;BRANCH TO CONTINUE
2685 022646 023737 034560 041630 6$: CMP     GECC2,WECC2;COMPARE SOFTWARE ECC WITH HARDWARE ECC
2686 022654 001401                BEQ     7$                 ;BRANCH IF GOOD
2687 022656 104031                EMT     31
2688
2689
2690 022660                7$:
      022660 004737 032306      JSR     PC,CHECKE        ;CHECK THAT BITS = 1
      022664 104401 053407      TYPE    ,CPHALT         ;CANNOT CONTINUE TESTING IF THEY DON'T
      022670 000000                HALT                      ;STOP THE TEST
2691
2692
2693
2694
2695                                ;FILL "REINTO" BUFFER WITH EXPECTED DATA
2696
2697 022672 004037 031772      JSR     R0,CLAREA        ;FILL REINTO BUFFER
    
```

```

2698 022676 002544          REINTO          :FROM
2699 022700 003542          REINTO+<255.*2> :TO
2700 022702 177777          .WORD -1          :DATA
2701
2702 022704 013737 034556 003544 MOV GECC1,REINTO+<256.*2>;FILL ECC1
2703 022712 013737 034560 003546 MOV GECC2,REINTO+<257.*2>;FILL ECC2
2704 022720 004037 031772          JSR R0,CLAREA      :FILL REST
2705 022724 003550          REINTO+<258.*2>    :FROM
2706 022726 003604          REINTO+<272.*2>    :TO
2707 022730 000000          0                  :DATA
2708
2709
2710 022732 005037 001406          CLR ERFLG$ ;CLEAR ERROR FLAG
2711
2712
2713          ;NOW COMPARE 'DISK' BUFFER WITH 'REINTO'
2714
2715 022736 004037 032756          JSR R0,COMPAR      :CHECK
2716 022742 002544          REINTO             :GOOD BUFFER
2717 022744 040626          DISK              :TEST BUFFER
2718 022746 000402          258.             :NUMBER OF WORDS CHECKED
2719 022750 022756          4$              :RETURN POINT FOR ERROR HEADER
2720 022752 022762          5$              :RETURN POINT FOR ERROR DATA
2721 022754 022766          TST60            :RETURN FOR GOOD COMPARISON
2722 022756          4$:
2723 022756 104007          EMT 7
2724 022760 000207          RTS PC           :RETURN TO COMPARE
2725 022762          5$:
2726 022762 104010          EMT 10
2727
2728          ;DATA WORDS
2729          ;WORD NOS 257 AND 258
2730          ;ARE ECC WHICH ARE CHECKED
2731          ;WORD NOS 259
2732          ;IS DATA GAP
2733          ;WORD NOS 260 TO 273
2734          ;ARE TOLERANCE GAP
2735          ;RETURN TO COMPARE
2736
2737          ;*****
2738          ;*TEST 60 READ ECC ENABLED 2A
2739          ;*THIS IS AN ECC READ DATA TEST
2740          ;*ERROR CORRECTION IS ENABLED
2741          ;*NO ERROR IS INSERTED
2742          ;*GOOD DATA USED IS 256 WORDS OF 177777
2743          ;*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
2744          ;*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
2745          ;*****
2746          TST60: SCOPE
2747          MOV #STACK,SP :RESET STACK
2748          MOV #60,TSTNM :MOVE #60 TO TEST NUMBER
2749
2750          ;
2751          ; SETUP FOR WHAT IS TO BE READ
2752          ; HEADER CRC IS RESTORED FROM A SUBROUTINE
2753
2754          MOV #-1,-(SP) :DATA TO BE READ
2755          MOV #256.,R5 :COUNTER
2756          MOV #DISK,R0 :START OF SIMULATED DISK DATA
    
```



```

2752 023016 011620          1$:  MOV    (SP),(R0)+    ;MOVE IN DATA ON TO SIMULATED DISK
2753 023020 005305          DEC    R5              ;COUNT
2754 023022 001375          BNE   1$              ;BRANCH IF 256 NOT COMPLETE
2755 023024 005726          TST   (SP)+          ;UNDO -(SP)
2756 023026 022020          CMP   (R0)+,(R0)+    ;JUMP OVER THE TWO ECC WORDS
2757 023030 012705 000017    MOV   #15.,R5        ;1 DATA GAP
2758                                ;14 TOLERANCE GAP
2759 023034 005020          2$:  CLR   (R0)+          ;CLEAR DATA GAP, AND
2760 023036 005305          DEC   R5              ;TOLERANCE GAP
2761 023040 001375          BNE   2$              ;BRANCH IF NOT COMPLETE
2762
2763
2764 023042 004737 035352    JSR   PC,FILLEC      ;INSERT THE TWO ECC WORDS ON THE DISK
2765                                ;IN THE CORRECT PLACE
2766
2767                                ;THESE ARE FOR ECC TEST ONLY
2768
2769 023046 012737 177777 001424  MOV   #-1,TSECC      ;THIS IS AN ECC TEST
2770 023054 005037 034570          CLR   POSITI         ;CLEAR ERROR POSITION COUNTER
2771 023060 013737 034564 034566  MOV   NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2772 023066 013737 034572 034600  MOV   HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2773 023074 005037 034556          CLR   GECC1          ;ECC LOW ORDER TO BE GENERATED
2774 023100 005037 034560          CLR   GECC2          ;ECC HIGH ORDER TO BE GENERATED
2775 023104 005037 034574          CLR   DATENV         ;CLEAR DATA ENVELOPE CLOCK COUNT
2776 023110 005037 034576          CLR   ZCODE          ;CLEAR LEADING ZEROS CLOCK COUNT
2777
2778
2779                                ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
2780
2781 023114 012737 010000 036710  MOV   #FMT22,CYL     ;16 BITS PER WORD
2782                                ;CYLINDER 0, FORMAT 16 BITS
2783 023122 112737 000000 036713  MOVB  #0,SECOTR+1    ;TRACK 0
2784 023130 112737 000000 036712  MOVB  #0,SECOTR      ;SECTOR 0
2785 023136 012737 000000 036714  MOV   #0,KEY1 ;KEY1=0
2786 023144 012737 000000 036716  MOV   #0,KEY2 ;KEY2=0
2787 023152 012737 000400 036770  MOV   #256.,DAWORD  ;NO. OF DATA WORDS
2788 023160 005037 036720          CLR   X              ;THIS IS A READ COMMAND
2789 023164 004537 033270          JSR   R5,CRC         ;GO TO CALCULATE CRC
2790 023170 036710          CYL
2791 023172 040610          WCRC
2792
2793
2794
2795
2796                                ;THESE ARE REGULAR SETUPS
2797
2798 023174 004737 032054          JSR   PC,CLDISK     ;SETUP GENERAL REGISTERS
2799 023200 012777 177374 156024  MOV   #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
2800 023206 012777 002544 156020  MOV   #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
2801 023214 112746 000000          MOVB  #0,-(SP)      ;IN LOWER BYTE GET SECTOR
2802 023220 112766 000000 000001  MOVB  #0,1(SP)      ;GET TRACK IN HIGHER BYTE
2803 023226 012677 156012          MOV   (SP)+,@RHDST  ;TRACK/SECTOR IN RHDST
2804 023232 012777 010000 156010  MOV   #FMT22,@RHOF  ;16 BITS PER WORD
2805                                ;ECC CORRECTION NOT INHIBIT
2806                                ;BECAUSE ECC IS NOT GOING
2807                                ;TO BE CHECKED
2808 023240 005077 156006          CLR   @RHCA         ;CYLINDER 0
    
```

```

2809
2810 023244 004737 032116      JSR    PC,CHECKT      ;CHECK THAT DVA,RDY,DPR,DRY = 1
      023250 104401 053407      TYPE    ,CPHALT      ;AND THAT NO OTHERS = 1.  CANNOT CON-
      023254 000000      HALT                    ;TINUE TESTING IF BOTH AREN'T TRUE
      ;STOP THE TEST
2811
2812 023256 013711 001460      MOV    REFOR,@R1      ;READ HEADER AND DATA=72
2813 023262 005037 001406      CLR    ERFLG$ ;CLEAR ERROR FLAG
2814 023266 004737 036550      JSR    PC,COMHD      ;READ HEADER AND DATA
2815                                ;IF THERE ARE READ ERRORS THEN
2816                                ;ECC WILL NOT BE CHECKED
2817
2818
2819                                ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
2820                                ;FROM THE "COMHD" ROUTINE THAT MEANS SECTOR GAP,
2821                                ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
2822                                ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
2823                                ;DETECTED
2824                                ;HEADER AND DATA ARE TO BE CHECKED.
2825                                ;IN CHECKING READ DATA THE WRITE FROM BUFFER
2826                                ;"WRFROM" IS FILLED WITH EXPECTED DATA AND
2827                                ;COMPARISONS ARE MADE
2828
2829 023272 005737 001406      TST    ERFLG$ ;ANY ERRORS ALREADY THERE
2830 023276 001102      BNE    TST61          ;BRANCH IF YES
2831 023300 004737 031554      JSR    PC,PUTREG      ;SAVE REGISTERS
2832 023304 005737 001316      TST    ER1            ;NO ERRORS SHOULD BE SET
2833 023310 001401      BEQ    6$            ;BRANCH IF NO ERRORS SET
2834 023312 104032      EMT    32
2835                                ;ONLY 11 OF THE 32 BITS CAN BE SEEN
2836                                ;IN THE PATERN REGISTER
2837                                ;DCK SHOULD BE SET IN RHER1
2838 023314 013746 034556      6$:  MOV    GECC1,-(SP) ;GET PATTERN REGISTER
2839 023320 042716 174000      BIC    #174000,(SP) ;KEEP ONLY 11 BITS
2840 023324 022637 001346      CMP    (SP)+,EC2     ;COMPARE PATTERN REGISTER
2841 023330 001401      BEQ    7$            ;BRANCH IF GOOD
2842 023332 104032      EMT    32
2843
2844
2845
2846
2847                                ;ADD 16 MAINTENANCE CLOCKS TO
2848                                ;BRING EBL DOWN
2849
2850 023334 012700 000020      7$:  MOV    #16,R0      ;COUNTER
2851 023340 052777 000002 155712 8$:  BIS    #MCLK,@RHMR ;SET CLOCK
2852 023346 042777 000002 155704 8$:  BIC    #MCLK,@RHMR ;CLEAR CLOCK
2853 023354 005300      DEC    R0            ;COUNT
2854 023356 001370      BNE    8$            ;BRANCH IF 16 CLOCKS NOT DONE
2855 023360 004737 032306      JSR    PC,CHECKE      ;CHECK THAT BITS = 1
      023364 104401 053407      TYPE    ,CPHALT      ;CANNOT CONTINUE TESTING IF THEY DON'T
      023370 000000      HALT                    ;STOP THE TEST
2856 023372 012700 001500      MOV    #WRFROM,R0    ;GETTING READY TO FILL EXPECTED DATA
2857 023376 012720 010000      MOV    #0!FMT22,(R0)+ ;CYLINDER 0
2858 023402 112746 000000      MOVB   #0,-(SP)      ;IN LOWER BYTE GET SECTOR
2859 023406 112766 000000 000001      MOVB   #0,1(SP)      ;GET TRACK IN HIGHER BYTE
2860 023414 012620      MOV    (SP)+,(R0)+   ;GET TRACK/SECTOR IN BUFFER
    
```



```

2861 023416 012720 000000      MOV    #0,(R0)+      ;KEY1 IN BUFFER
2862 023422 012720 000000      MOV    #0,(R0)+      ;KEY2 IN BUFFER
2863 023426 012701 000400      MOV    #256.,R1      ;DATA WORD COUNTER
2864 023432 012702 177777      MOV    #-1,R2 ;DATA
2865
2866 023436 010220      3$:   MOV    R2,(R0)+      ;DATA INTO BUFFER
2867 023440 005301      DEC    R1              ;COUNT
2868 023442 001375      BNE    3$             ;BRANCH IF 256 NOT DONE
2869 023444 005037 001406      CLR    ERFLG$ ;CLEAR ERROR FLAG
2870 023450 004737 031554      JSR    PC,PUTREG      ;SAVE REGISTERS
2871
2872      ;NOW READ DATA BUFFER WILL BE CHECKED
2873
2874 023454 004037 032756      JSR    R0,COMPAR      ;CHECK
2875 023460 001500      WRFROM                ;GOOD BUFFER
2876 023462 002544      REINTO                ;TEST BUFFER
2877 023464 000404      4+256.              ;NUMBER OF WORDS CHECKED
2878 023466 023474      4$                   ;RETURN POINT FOR ERROR HEADER
2879 023470 023500      5$                   ;RETURN POINT FOR ERROR DATA
2880 023472 023504      TST61                ;RETURN FOR GOOD COMPARISON
2881 023474
2882 023474 104004      4$:   EMT    4
2883 023476 000207      RTS    PC             ;RETURN TO 'COMPAR'
2884 023500
2885 023500 104005      5$:   EMT    5
2886 023502 000207      RTS    PC             ;HEADER WORDS
2887      ;5 TO 260 ARE DATA WORDS
2896      ;RETURN TO 'COMPAR'

:*****
:*TEST 61 READ ECC ENABLED 2B
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A CORRECTABLE ERROR IS INSERTED IN BIT POSITION 32
:*GOOD DATA USED IS 256 WORDS OF 177777
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
:*****
TST61: SCOPE
2897 023504 000004      MOV    #STACK,SP      ;RESET STACK
2898 023506 012706 001100
2899 023512 012737 000061 003610      MOV    #61,TSTNM      ;MOVE #61 TO TEST NUMBER
2900
2901
2902      ;SETUP FOR WHAT IS TO BE READ
2903      ;HEADER CRC IS RESTORED FROM A SUBROUTINE
2904
2905 023520 012746 177777      MOV    #-1,-(SP)      ;DATA TO BE READ
2906 023524 012705 000400      MOV    #256.,R5      ;COUNTER
2907 023530 012700 040626      MOV    #DISK,R0      ;START OF SIMULATED DISK DATA
2908 023534 011620      1$:   MOV    (SP),(R0)+      ;MOVE IN DATA ON TO SIMULATED DISK
2909 023536 005305      DEC    R5              ;COUNT
2910 023540 001375      BNE    1$             ;BRANCH IF 256 NOT COMPLETE
2911 023542 005726      TST    (SP)+          ;UNDO -(SP)
2912 023544 022020      CMP    (R0)+,(R0)+    ;JUMP OVER THE TWO ECC WORDS
2913 023546 012705 000017      MOV    #15.,R5 ;1 DATA GAP
2914      ;14 TOLERANCE GAP

```

```

2915 023552 005020      2$: CLR (R0)+ ;CLEAR DATA GAP, AND
2916 023554 005305      DEC R5 ;TOLERANCE GAP
2917 023556 001375      BNE 2$ ;BRANCH IF NOT COMPLETE
2918
2919
2920 023560 004737 035352 JSR PC,FILLEC ;INSERT ECC IN PROPER PLACE ON DISK
2921
2922
2923
2924 ;THESE ARE FOR ECC TEST ONLY
2925
2926 023564 012737 177777 001424 MOV #-1,TSECC ;THIS IS AN ECC TEST
2927 023572 005037 034570 CLR POSITI ;CLEAR ERROR POSITION COUNTER
2928 023576 013737 034564 034566 MOV NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
2929 023604 013737 034572 034600 MOV HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
2930 023612 005037 034556 CLR GECC1 ;ECC LOW ORDER TO BE GENERATED
2931 023616 005037 034560 CLR GECC2 ;ECC HIGH ORDER TO BE GENERATED
2932 023622 005037 034574 CLR DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
2933 023626 005037 034576 CLR ZCODE ;CLEAR LEADING ZEROS CLOCK COUNT
2934
2935
2936 ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
2937
2938 023632 012737 010000 036710 MOV #FMT22,CYL ;16 BITS PER WORD
2939 ;CYLINDER 0, FORMAT 16 BITS
2940 023640 112737 000000 036713 MOVB #0,SECOTR+1 ;TRACK 0
2941 023646 112737 000000 036712 MOVB #0,SECOTR ;SECTOR 0
2942 023654 012737 000000 036714 MOV #0,KEY1 ;KEY1=0
2943 023662 012737 000000 036716 MOV #0,KEY2 ;KEY2=0
2944 023670 012737 000400 036770 MOV #256.,DAWORD ;NO. OF DATA WORDS
2945 023676 005037 036720 CLR X ;THIS IS A READ COMMAND
2946 023702 004537 033270 JSR R5,CRC ;GO TO CALCULATE CRC
2947 023706 036710 CYL
2948 023710 040610 WCRC
2949
2950
2951 ;THIS IS TO INSERT ERROR
2952 ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
2953 ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
2954 ;THIS MOVE
2955
2956 023712 012737 077777 040630 MOV #77777,DISK+2 ;FORCE ERROR ON BIT NUMBER 32
2957 ;SO ERROR POSITION REGISTER WILL SHOW
2958 ;22
2959 023720 012737 000026 024074 MOV #22.,8$ ;INSERT POSITION REG.
2960
2961
2962 ;THESE ARE REGULAR SETUPS
2963
2964 023726 004737 032054 JSR PC,CLDISK ;SETUP GENERAL REGISTERS
2965 023732 012777 177374 155272 MOV #-256.-4.,@RHWC ;256. DATA & HEADER WORDS
2966 023740 012777 002544 155266 MOV #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
2967 023746 112746 000000 MOVB #0,-(SP) ;IN LOWER BYTE GET SECTOR
2968 023752 112766 000000 000001 MOVB #0,1(SP) ;GET TRACK IN HIGHER BYTE
2969 023760 012677 155260 MOV (SP)+,@RHDST ;TRACK/SECTOR IN RHDST
2970 023764 012777 010000 155256 MOV #FMT22,@RHOF ;16 BITS PER WORD
2971 ;ECC CORRECTION NOT INHIBIT
    
```



```

3024 024150 012702 177777
3025 024154 010220
3026 024156 005301
3027 024160 001375
3028
3029
3030
3031
3032 024162 012737 077777 001512
3033 024170 004737 031554
3034 024174 005037 001406
3035
3036
3037
3038
3039 024200 004037 032756
3040 024204 001500
3041 024206 002544
3042 024210 000404
3043 024212 024220
3044 024214 024224
3045 024216 024230
3046 024220
3047 024222 104004
3048 024224 000207
3049
3050
3051 024226 000207
3052
3061

3062 024230 000004
3063 024232 012706 001100
3064 024236 012737 000062 003610
3065
3066
3067
3068 024244 012746 177777
3069 024250 012705 000400
3070 024254 012700 040626
3071 024260 011620
3072 024262 005305
3073 024264 001375
3074 024266 005726
3075 024270 022020
3076 024272 012705 000017
3077

3$: MOV #-1,R2 ;DATA
MOV R2,(R0)+ ;DATA INTO BUFFER
DEC R1 ;COUNT
BNE 3$ ;BRANCH IF 256 NOT DONE

;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
;NOW THE INSERTED ERROR WILL BE PUT IN

MOV #77777,WRFROM+<5*2> ;INSERTED ERROR
JSR PC,PUTREG ;SAVE REGISTERS
CLR ERFLG$ ;CLEAR ERROR FLAG

;NOW READ DATA BUFFER WILL BE CHECKED

JSR R0,COMPAR ;CHECK
WRFROM ;GOOD BUFFER
REINTO ;TEST BUFFER
4+256. ;NUMBER OF WORDS CHECKED
4$ ;RETURN POINT FOR ERROR HEADER
5$ ;RETURN POINT FOR ERROR DATA
TST62 ;RETURN FOR GOOD COMPARISON

4$: EMT 4
RTS PC ;RETURN TO "COMPAR"

5$: EMT 5

RTS PC ;HEADER WORDS
;5 TO 260 ARE DATA WORDS
;RETURN TO "COMPAR"

:*****
:*TEST 62 READ ECC ENABLED 2C
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A NON CORRECTABLE ERROR IS INSERTED IN BIT POSITION 32 AND 21
:*GOOD DATA USED IS 256 WORDS OF 177777
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
:*****
TST62: SCOPE
MOV #STACK,SP ;RESET STACK
MOV #62,TSTNM ;MOVE #62 TO TEST NUMBER

;
; SETUP FOR WHAT IS TO BE READ
; HEADER CRC IS RESTORED FROM A SUBROUTINE

MOV #-1,-(SP) ;DATA TO BE READ
MOV #256.,R5 ;COUNTER
MOV #DISK,R0 ;START OF SIMULATED DISK DATA
1$: MOV (SP),(R0)+ ;MOVE IN DATA ON TO SIMULATED DISK
DEC R5 ;COUNT
BNE 1$ ;BRANCH IF 256 NOT COMPLETE
TST (SP)+ ;UNDO -(SP)
CMP (R0)+,(R0)+ ;JUMP OVER THE TWO ECC WORDS
MOV #15.,R5 ;1 DATA GAP
;14 TOLERANCE GAP
    
```



```

3078 024276 005020      2$: CLR (R0)+ ;CLEAR DATA GAP, AND
3079 024300 005305      DEC R5 ;TOLERANCE GAP
3080 024302 001375      BNE 2$ ;BRANCH IF NOT COMPLETE
3081
3082
3083 024304 004737 035352      JSR PC,FILLEC ;INSERT THE TWO ECC WORDS ON THE DISK
3084 ;IN THE CORRECT PLACE
3085
3086 ;THESE ARE FOR ECC TEST ONLY
3087
3088 024310 012737 177777 001424      MOV #-1,TSECC ;THIS IS AN ECC TEST
3089 024316 005037 034570      CLR POSITI ;CLEAR ERROR POSITION COUNTER
3090 024322 013737 034564 034566      MOV NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
3091 024330 013737 034572 034600      MOV HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
3092 024336 005037 034556      CLR GECC1 ;ECC LOW ORDER TO BE GENERATED
3093 024342 005037 034560      CLR GECC2 ;ECC HIGH ORDER TO BE GENERATED
3094 024346 005037 034574      CLR DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
3095 024352 005037 034576      CLR ZCODE ;CLEAR LEADING ZEROS CLOCK COUNT
3096
3097
3098 ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
3099
3100 024356 012737 010000 036710      MOV #FMT22,CYL ;16 BITS PER WORD
3101 ;CYLINDER 0, FORMAT 16 BITS
3102 024364 112737 000000 036713      MOVB #0,SECOTR+1 ;TRACK 0
3103 024372 112737 000000 036712      MOVB #0,SECOTR ;SECTOR 0
3104 024400 012737 000000 036714      MOV #0,KEY1 ;KEY1=0
3105 024406 012737 000000 036716      MOV #0,KEY2 ;KEY2=0
3106 024414 012737 000400 036770      MOV #256.,DAWORD ;NO. OF DATA WORDS
3107 024422 005037 036720      CLR X ;THIS IS A READ COMMAND
3108 024426 004537 033270      JSR R5,CRC ;GO TO CALCULATE CRC
3109 024432 036710      CYL
3110 024434 040610      WCRC
3111
3112
3113 ;THIS IS TO INSERT ERROR
3114 ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
3115 ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
3116 ;THIS MOVE
3117
3118 024436 012737 077757 040630      MOV #77757,DISK+2 ;FORCE ERROR ON BIT NUMBER 32 AND 21
3119 ;SO ERROR POSITION REGISTER WILL SHOW
3120 ;22
3121 024444 012737 010040 024620      MOV #4128.,8$ ;INSERT POSITION REG.
3122
3123
3124 ;THESE ARE REGULAR SETUPS
3125
3126 024452 004737 032054      JSR PC,CLDISK ;SETUP GENERAL REGISTERS
3127 024456 012777 177374 154546      MOV #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
3128 024464 012777 002544 154542      MOV #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
3129 024472 112746 000000      MOVB #0,-(SP) ;IN LOWER BYTE GET SECTOR
3130 024476 112766 000000 000001      MOVB #0,1(SP) ;GET TRACK IN HIGHER BYTE
3131 024504 012677 154534      MOV (SP)+,@RHDST ;TRACK/SECTOR IN RHDST
3132 024510 012777 010000 154532      MOV #FMT22,@RHOF ;16 BITS PER WORD
3133 ;ECC CORRECTION NOT INHIBIT
3134 ;BECAUSE ECC IS NOT GOING
    
```

```

3135
3136 024516 005077 154530          CLR    @RHCA          ;TO BE CHECKED
3137                                     ;CYLINDER 0
3138 024522 004737 032116          JSR    PC,CHECKT     ;CHECK THAT DVA,RDY,DPR,DRY = 1
      024526 104401 053407          TYPE   ,CPHALT      ;AND THAT NO OTHERS = 1. CANNOT CON-
      024532 000000                  HALT                ;TINUE TESTING IF BOTH AREN'T TRUE
3139                                     ;STOP THE TEST
3140 024534 013711 001460          MOV    REFOR,@R1     ;READ HEADER AND DATA=72
3141 024540 005037 001406          CLR    ERFLG$       ;CLEAR ERROR FLAG
3142 024544 004737 036550          JSR    PC,COMHD     ;READ HEADER AND DATA
3143                                     ;IF THERE ARE READ ERRORS THEN
3144                                     ;ECC WILL NOT BE CHECKED
3145
3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156
3157 024550 005737 001406          TST    ERFLG$       ;ANY ERRORS ALREADY THERE
3158 024554 001106                  BNE    TST63         ;BRANCH IF YES
3159 024556 004737 031554          JSR    PC,PUTREG    ;SAVE REGISTERS
3160 024562 022737 100000 001316  CMP    #DCK,ER1     ;ONLY DATA CHECK ERROR SHOULD BE SET
3161 024570 001401                  BEQ    6$           ;BRANCH IF YES
3162 024572 104032                  EMT    32
3163
3164
3165
3166
3167 024574 013746 034556          6$:  MOV    GECC1,-(SP) ;ZERO
3168 024600 042716 174000          BIC    #174000,(SP) ;ONLY 11 OF THE 32 BITS CAN BE SEEN
3169 024604 022637 001346          CMP    (SP)+,EC2   ;IN THE PATERN REGISTER
3170 024610 001401                  BEQ    7$           ;DCK SHOULD BE SET IN RHER1
3171 024612 104032                  EMT    32          ;GET PATTERN REGISTER
3172                                     ;KEEP ONLY 11 BITS
3173 024614 004037 035200          7$:  JSR    R0,ECORR ;COMPARE PATTERN REGISTER
3174 024620 000000          8$:  .WORD ;BRANCH IF GOOD
3175                                     ;GO TO ECC CORRECTION PROCESS
3176                                     ;EXPECTED POSITION REG. WHEN CORRECTION
3177 024622 004737 031554          JSR    PC,PUTREG    ;IS COMPLETE
3178                                     ;SAVE REGISTERS
3179
3180 024626 022737 100100 001316  CMP    #DCK!ECH,ER1 ;WITH ERRORS INSERTED IN BIT POSITION 21
3181                                     ;AND 32 HARD ERROR BIT SHOULD SET
3182 024634 001401                  BEQ    9$           ;BRANCH IF GOOD
3183 024636 104036                  EMT    36
3184                                     ;32 HCE SHOULD SET
3185
3186
3187
3188
    
```



```

3189 024640          9$:      JSR      PC,CHECKE      ;CHECK THAT BITS = 1
      024640 004737 032306      TYPE      ,CPHALT      ;CANNOT CONTINUE TESTING IF THEY DON'T
      024644 104401 053407      HALT          ;STOP THE TEST
      024650 000000      MOV      #WRFROM,R0      ;GETTING READY TO FILL EXPECTED DATA
3190 024652 012700 001500      MOV      #0!FMT22,(R0)+ ;CYLINDER 0
3191 024656 012720 010000      MOV      #0,-(SP)      ;IN LOWER BYTE GET SECTOR
3192 024662 112746 000000      MOV      #0,1(SP)      ;GET TRACK IN HIGHER BYTE
3193 024666 112766 000000 000001  MOV      (SP)+,(R0)+    ;GET TRACK/SECTOR IN BUFFER
3194 024674 012620      MOV      #0,(R0)+      ;KEY1 IN BUFFER
3195 024676 012720 000000      MOV      #0,(R0)+      ;KEY2 IN BUFFER
3196 024702 012720 000000      MOV      #256,R1      ;DATA WORD COUNTER
3197 024706 012701 000400      MOV      #-1,R2      ;DATA
3198 024712 012702 177777      MOV      R2,(R0)+     ;DATA INTO BUFFER
3199 024716 010220      DEC      R1            ;COUNT
3200 024720 005301      BNE     3$            ;BRANCH IF 256 NOT DONE
3201 024722 001375
3202
3203      ;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
3204      ;NOW THE INSERTED ERROR WILL BE PUT IN
3205
3206 024724 012737 077757 001512  MOV      #77757,WRFROM+<5*2> ;INSERTED ERROR
3207 024732 004737 031554      JSR      PC,PUTREG     ;SAVE REGISTERS
3208 024736 005037 001406      CLR      ERFLG$ ;CLEAR ERROR FLAG
3209
3210      ;NOW READ DATA BUFFER WILL BE CHECKED
3211
3212
3213 024742 004037 032756      JSR      R0,COMPAR     ;CHECK
3214 024746 001500      WRFROM      ;GOOD BUFFER
3215 024750 002544      REINTO     ;TEST BUFFER
3216 024752 000404      4+256.    ;NUMBER OF WORDS CHECKED
3217 024754 024762      4$        ;RETURN POINT FOR ERROR HEADER
3218 024756 024766      5$        ;RETURN POINT FOR ERROR DATA
3219 024760 024772      TST63     ;RETURN FOR GOOD COMPARISON
3220 024762
3221 024764 104004      4$:      EMT      4
3222 024766 000207      RTS      PC            ;RETURN TO 'COMPAR'
3223 024766 104005      5$:      EMT      5
3224
3225 024770 000207      RTS      PC            ;HEADER WORDS
3226
3227
3228
3229
3230
3231
3232
3233
3234 024772 000004      ;*****
3235 024774 012706 001100      ;*TEST 63      WRITE ECC TEST 3
3236 025000 012737 000063 003610  ;*THIS IS A WRITE ECC TEST
3237 025006 012700 040530      ;*WRITE CYLINDER 0, FORMAT 16 BITS PER WORD
3238 025012 012701 000460      ;*TRACK 0, SECTOR 1, KEYS 0, NUMBER OF WORDS 256
3239 025016 005020      ;*OF ALL 52525.
3240
3241      ;*****
3242      TST63: SCOPE
3243      MOV      #STACK,SP      ;RESET STACK
3244
3245      MOV      #63,TSTNM     ;MOVE #63 TO TEST NUMBER
3246      MOV      #SECGAP,R0    ;POINTER
3247      MOV      #304.,R1     ;COUNTER
3248      CLR      (R0)+        ;CLEAR SIMULATED DISK AREA
    
```

```

3240 025020 005301          DEC      R1
3241 025022 001375          BNE     1$
3242 025024 004737 032054  JSR     PC,CLDISK      ;THIS IS USED TO SET GENERAL REGISTERS
3243
3244                      ;THESE ARE FOR ECC TEST ONLY
3245
3246 025030 012737 177777 001424  MOV     #-1,TSECC      ;THIS IS AN ECC TEST
3247 025036 005037 034570          CLR     POSITI        ;CLEAR ERROR POSITION COUNTER
3248 025042 013737 034564 034566  MOV     NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
3249 025050 013737 034572 034600  MOV     HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
3250 025056 005037 034556          CLR     GECC1         ;ECC LOW ORDER TO BE GENERATED
3251 025062 005037 034560          CLR     GECC2         ;ECC HIGH ORDER TO BE GENERATED
3252 025066 005037 034574          CLR     DATENV        ;CLEAR DATA ENVELOPE CLOCK COUNT
3253 025072 005037 034576          CLR     ZCODE         ;CLEAR LEADING ZEROS CLOCK COUNT
3254
3255
3256
3257
3258                      ;THESE ARE TO BE SETUP FOR DISKLESS USE ONLY
3259
3260 025076 012737 010000 042046  MOV     #FMT22,WCYL    ;FORMAT22=16BIT WORDS AND
3261                                ;CYLINDER 0
3262 025104 012737 000001 042050  MOV     #1,WSECTR      ;TRACK=0, SECTOR=1
3263 025112 005037 042052          CLR     WKEY1         ;KEY1=0
3264 025116 005037 042054          CLR     WKEY2         ;KEY2=0
3265 025122 012737 000400 042106  MOV     #256.,FNWORD   ;256 DATA WORDS
3266 025130 004537 033270          JSR     R5,CRC        ;GO TO CALCULATE CRC
3267 025134 042046          WCYL
3268 025136 042056          GCRC
3269
3270                      ;THESE ARE REGULAR SETUPS
3271
3272 025140 012777 177374 154064  MOV     #-260.,@RHWC   ;256 DATA WORDS 4 HEADER WORDS
3273 025146 012700 001500          MOV     #WRFROM,R0    ;THESE TWO INSTRUCTIONS GETS
3274 025152 010077 154056          MOV     R0,@RHBA      ;ADDR. OF WRFROM INTO R0 AND
3275                                ;BUS ADDRESS REGISTER
3276 025156 012720 010000          MOV     #FMT22,(R0)+  ;FORMAT=16 BIT WORDS
3277                                ;CYLINDER=0
3278 025162 012720 000001 2$:    MOV     #1,(R0)+      ;TRACK=0, SECTOR=1, KEYS=0
3279 025166 005020          CLR     (R0)+        ;KEY1=0
3280 025170 005020          CLR     (R0)+        ;KEY2=0
3281 025172 012705 000400          MOV     #256.,R5      ;COUNTER
3282 025176 012720 052525 3$:    MOV     #52525,(R0)+  ;MOVE ALL 52525 FOR DATA
3283 025202 005305          DEC     R5
3284 025204 001374          BNE     3$
3285 025206 012777 000001 154030  MOV     #1,@RHDST     ;BRANCH IF DATA NOT COMPLETE
3286                                ;TRACK=0 SECTOR=1
3287 025214 004737 032116          JSR     PC,CHECKT    ;CHECK THAT DVA,RDY,DPR,DRY = 1
3288 025220 104401 053407          TYPE    ,CPHALT     ;AND THAT NO OTHERS = 1. CANNOT CON-
3289                                ;TINUE TESTING IF BOTH AREN'T TRUE
3290                                ;STOP THE TEST
3291 025224 000000          HALT
3292
3293 025226 013711 001454          MOV     WRIFOR,@R1    ;GET READY FOR WRITE HEADER AND
3294                                ;DATA WITH 62 IN RHCS1
3295 025232 005037 001406          CLR     ERFLG$       ;CLEAR ERROR FLAG
3296 025236 012777 010000 154004  MOV     #FMT22,@RHOF  ;FORMAT BIT=1 (16 BIT WORDS)
3297 025244 005077 154002          CLR     @RHCA        ;CYLINDER =0
    
```



```

3294 025250 004737 041672      JSR    PC,COMWHD      ;WRITE HEADER AND DATA
3295
3296      ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
3297      ;FROM THE "COMWHD" ROUTINE THAT MEANS ALL HEADER ON DISK
3298      ;IS GOOD IE. ONLY DATA IS TO BE CHECKED TO SEE IF THEY ARE
3299      ;ALL 52525 AND WRITE DATA GAP AND TOLERANCE GAP TO SEE IF
3300      ;THEY ARE ALL ZEROS
3301
3302 025254 005737 001406      TST    ERFLG$ ;HAS ANY ERRORS OCCURED?
3303      ;IF WRITE ERROR OCCURS ECC IS NOT CHECKED
3304 025260 001056      BNE    TST64      ;:BRANCH IF YES
3305
3306
3307      ;COMPARE SOFTWARE GENERATED ECC WITH THAT GENERATED BY HARDWARE
3308 025262 023737 034556 041626  CMP    GECC1,WECC1;COMPARE SOFTWARE ECC WITH HARDWARE ECC
3309 025270 001402      BEQ    6$      ;BRANCH IF GOOD
3310 025272 104031      EMT    31
3311 025274 000405      BR     7$      ;BRANCH TO CONTINUE
3312 025276 023737 034560 041630 6$:  CMP    GECC2,WECC2 ;COMPARE SOFTWARE ECC WITH HARDWARE ECC
3313 025304 001401      BEQ    7$      ;BRANCH IF GOOD
3314 025306 104031      EMT    31
3315
3316
3317 025310      7$:
3318 025310 004737 032306      JSR    PC,CHECKE    ;CHECK THAT BITS = 1
3319 025314 104401 053407  TYPE    ,CPHALT    ;CANNOT CONTINUE TESTING IF THEY DON'T
3320 025320 000000      HALT           ;STOP THE TEST
3321
3322      ;FILL "REINTO" BUFFER WITH EXPECTED DATA
3323
3324 025322 004037 031772      JSR    R0,CLAREA    ;FILL REINTO BUFFER
3325 025326 002544      REINTO        ;FROM
3326 025330 003542      REINTO+<255.*2> ;TO
3327 025332 052525      .WORD 52525      ;DATA
3328
3329 025334 013737 034556 003544  MOV    GECC1,REINTO+<256.*2>;FILL ECC1
3330 025342 013737 034560 003546  MOV    GECC2,REINTO+<257.*2>;FILL ECC2
3331 025350 004037 031772      JSR    R0,CLAREA    ;FILL REST
3332 025354 003550      REINTO+<258.*2>    ;FROM
3333 025356 003604      REINTO+<272.*2>    ;TO
3334 025360 000000      0              ;DATA
3335
3336
3337 025362 005037 001406      CLR    ERFLG$ ;CLEAR ERROR FLAG
3338
3339
3340      ;NOW COMPARE "DISK" BUFFER WITH "REINTO"
3341
3342 025366 004037 032756      JSR    R0,COMPAR    ;CHECK
3343 025372 002544      REINTO        ;GOOD BUFFER
3344 025374 040626      DISK          ;TEST BUFFER
3345 025376 000402      258.         ;NUMBER OF WORDS CHECKED
3346 025400 025406      4$           ;RETURN POINT FOR ERROR HEADER
3347 025402 025412      5$           ;RETURN POINT FOR ERROR DATA
    
```

```

3348 025404 025416          TST64          ;RETURN FOR GOOD COMPARISON
3349 025406          4$: EMT 7
      025406 104007          RTS PC          ;RETURN TO COMPARE
3350 025410 000207          EMT 10
3351 025412 104010          ;DATA WORDS
3352          ;WORD NOS 257 AND 258
3353          ;ARE ECC WHICH ARE CHECKED
3354          ;WORD NOS 259
3355          ;IS DATA GAP
3356          ;WORD NOS 260 TO 273
3357          ;ARE TOLERANCE GAP
3358          ;RETURN TO COMPARE
3359 025414 000207          RTS PC
3360
3369
    
```

```

*****
*TEST 64 READ ECC ENABLED 3A
*THIS IS AN ECC READ DATA TEST
*ERROR CORRECTION IS ENABLED
*NO ERROR IS INSERTED
*GOOD DATA USED IS 256 WORDS OF 52525
*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
*****
    
```

```

3370 025416 000004 001100 003610 TST64: SCOPE
3371 025420 012706 000064          MOV #STACK,SP ;RESET STACK
3372 025424 012737          MOV #64,TSTNM ;MOVE #64 TO TEST NUMBER
3373
3374 ;SETUP FOR WHAT IS TO BE READ
3375 ;HEADER CRC IS RESTORED FROM A SUBROUTINE
3376 025432 012746 052525          MOV #52525,-(SP) ;DATA TO BE READ
3377 025436 012705 000400          MOV #256,R5 ;COUNTER
3378 025442 012700 040626          MOV #DISK,R0 ;START OF SIMULATED DISK DATA
3379 025446 011620          1$: MOV (SP),(R0)+ ;MOVE IN DATA ON TO SIMULATED DISK
3380 025450 005305          DEC R5 ;COUNT
3381 025452 001375          BNE 1$ ;BRANCH IF 256 NOT COMPLETE
3382 025454 005726          TST (SP)+ ;UNDO -(SP)
3383 025456 022020          CMP (R0)+,(R0)+ ;JUMP OVER THE TWO ECC WORDS
3384 025460 012705 000017          MOV #15,R5 ;1 DATA GAP
3385          ;14 TOLERANCE GAP
3386 025464 005020          2$: CLR (R0)+ ;CLEAR DATA GAP, AND
3387 025466 005305          DEC R5 ;TOLERANCE GAP
3388 025470 001375          BNE 2$ ;BRANCH IF NOT COMPLETE
3389
3390 025472 004737 035352          JSR PC,FILLEC ;INSERT THE TWO ECC WORDS ON THE DISK
3391          ;IN THE CORRECT PLACE
3392
3393 ;THESE ARE FOR ECC TEST ONLY
3394
3395 025476 012737 177777 001424          MOV #-1,TSECC ;THIS IS AN ECC TEST
3396 025504 005037 034570          CLR POSITI ;CLEAR ERROR POSITION COUNTER
3397 025510 013737 034564 034566          MOV NCODE,NCOUNT ;TEMPORARY N-CODE COUNTER
3398 025516 013737 034572 034600          MOV HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
3399 025524 005037 034556          CLR GECC1 ;ECC LOW ORDER TO BE GENERATED
3400 025530 005037 034560          CLR GECC2 ;ECC HIGH ORDER TO BE GENERATED
3401 025534 005037 034574          CLR DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
    
```



```

3402 025540 005037 034576          CLR      ZCODE          ;CLEAR LEADING ZEROS CLOCK COUNT
3403
3404
3405          ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
3406
3407 025544 012737 010000 036710    MOV      #FMT22,CYL      ;16 BITS PER WORD
3408                                     ;CYLINDER 0, FORMAT 16 BITS
3409 025552 112737 000000 036713    MOV      #0,SECOTR+1     ;TRACK 0
3410 025560 112737 000000 036712    MOV      #0,SECOTR      ;SECTOR 0
3411 025566 012737 000000 036714    MOV      #0,KEY1 ;KEY1=0
3412 025574 012737 000000 036716    MOV      #0,KEY2 ;KEY2=0
3413 025602 012737 000400 036770    MOV      #256.,DAWORD    ;NO. OF DATA WORDS
3414 025610 005037 036720          CLR      X              ;THIS IS A READ COMMAND
3415 025614 004537 033270          JSR      R5,CRC          ;GO TO CALCULATE CRC
3416 025620 036710          CYL
3417 025622 040610          WCRC
3418
3419
3420
3421
3422          ;THESE ARE REGULAR SETUPS
3423
3424 025624 004737 032054          JSR      PC,CLDISK      ;SETUP GENERAL REGISTERS
3425 025630 012777 177374 153374    MOV      #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
3426 025636 012777 002544 153370    MOV      #REINTO,@RHBA  ;STARTING ADDRESS OF READ BUFFER
3427 025644 112746 000000          MOV      #0,-(SP)       ;IN LOWER BYTE GET SECTOR
3428 025650 112766 000000 000001    MOV      #0,1(SP)       ;GET TRACK IN HIGHER BYTE
3429 025656 012677 153362          MOV      (SP)+,@RHDST    ;TRACK/SECTOR IN RHDST
3430 025662 012777 010000 153360    MOV      #FMT22,@RHOF ;16 BITS PER WORD
3431                                     ;ECC CORRECTION NOT INHIBIT
3432                                     ;BECAUSE ECC IS NOT GOING
3433                                     ;TO BE CHECKED
3434 025670 005077 153356          CLR      @RHCA          ;CYLINDER 0
3435
3436 025674 004737 032116          JSR      PC,CHECKT      ;CHECK THAT DVA,RDY,DPR,DRY = 1
3437 025700 104401 053407          TYPE     ,CPHALT       ;AND THAT NO OTHERS = 1. CANNOT CON-
3438          025704 000000          HALT                    ;TINUE TESTING IF BOTH AREN'T TRUE
3439                                     ;STOP THE TEST
3440 025706 013711 001460          MOV      REFOR,@R1      ;READ HEADER AND DATA=72
3441 025712 005037 001406          CLR      ERFLG$ ;CLEAR ERROR FLAG
3442 025716 004737 036550          JSR      PC,COMHD       ;READ HEADER AND DATA
3443                                     ;IF THERE ARE READ ERRORS THEN
3444                                     ;ECC WILL NOT BE CHECKED
3445
3446          ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
3447          ;FROM THE "COMHD" ROUTINE THAT MEANS SECTOR GAP,
3448          ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
3449          ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
3450          ;DETECTED
3451          ;HEADER AND DATA ARE TO BE CHECKED.
3452          ;IN CHECKING READ DATA THE WRITE FROM BUFFER
3453          ;"WRFROM" IS FILLED WITH EXPECTED DATA AND
3454          ;COMPARISONS ARE MADE
3455 025722 005737 001406          TST      ERFLG$ ;ANY ERRORS ALREADY THERE
    
```

```

3456 025726 001102          BNE      TST65      ;BRANCH IF YES
3457 025730 004737 031554   JSR      PC,PUTREG ;SAVE REGISTERS
3458 025734 005737 001316   TST      ER1       ;NO ERRORS SHOULD BE SET
3459 025740 001401          BEQ      6$        ;BRANCH IF NO ERRORS SET
3460 025742 104032          EMT      32
3461
3462
3463
3464 025744 013746 034556   6$:     MOV      GECC1,-(SP) ;ONLY 11 OF THE 32 BITS CAN BE SEEN
3465 025750 042716 174000   BIC      #174000,(SP) ;IN THE PATERN REGISTER
3466 025754 022637 001346   CMP      (SP)+,EC2 ;DCK SHOULD BE SET IN RHER1
3467 025760 001401          BEQ      7$        ;GET PATTERN REGISTER
3468 025762 104032          EMT      32        ;KEEP ONLY 11 BITS
                          ;COMPARE PATTERN REGISTER
                          ;BRANCH IF GOOD
3469
3470
3471
3472
3473
3474
3475
3476 025764 012700 000020   7$:     MOV      #16.,R0    ;COUNTER
3477 025770 052777 000002 153262 8$:     BIS      #MCLK,@RHMR ;SET CLOCK
3478 025776 042777 000002 153254   BIC      #MCLK,@RHMR ;CLEAR CLOCK
3479 026004 005300          DEC      R0        ;COUNT
3480 026006 001370          BNE      8$        ;BRANCH IF 16 CLOCKS NOT DONE
3481 026010 004737 032306   JSR      PC,CHECKE ;CHECK THAT BITS = 1
      026014 104401 053407   TYPE    .CPHALT   ;CANNOT CONTINUE TESTING IF THEY DON'T
      026020 000000          HALT              ;STOP THE TEST
3482 026022 012700 001500   MOV      #WRFROM,R0 ;GETTING READY TO FILL EXPECTED DATA
3483 026026 012720 010000   MOV      #0!FMT22,(R0)+ ;CYLINDER 0
3484 026032 112746 000000   MOV      #0,-(SP)   ;IN LOWER BYTE GET SECTOR
3485 026036 112766 000000 000001  MOV      #0,1(SP)   ;GET TRACK IN HIGHER BYTE
3486 026044 012620          MOV      (SP)+,(R0)+ ;GET TRACK/SECTOR IN BUFFER
3487 026046 012720 000000   MOV      #0,(R0)+  ;KEY1 IN BUFFER
3488 026052 012720 000000   MOV      #0,(R0)+  ;KEY2 IN BUFFER
3489 026056 012701 000400   MOV      #256.,R1  ;DATA WORD COUNTER
3490 026062 012702 052525   MOV      #52525.,R2 ;DATA
3491 026066 010220          3$:     MOV      R2,(R0)+  ;DATA INTO BUFFER
3492 026070 005301          DEC      R1        ;COUNT
3493 026072 001375          BNE      3$        ;BRANCH IF 256 NOT DONE
3494 026074 005037 001406   CLR      ERFLG$    ;CLEAR ERROR FLAG
3495 026100 004737 031554   JSR      PC,PUTREG ;SAVE REGISTERS
3496
3497
3498
3499 026104 004037 032756          JSR      R0,COMPAR ;CHECK
3500 026110 001500          WRFROM          ;GOOD BUFFER
3501 026112 002544          REINTO         ;TEST BUFFER
3502 026114 000404          4+256.        ;NUMBER OF WORDS CHECKED
3503 026116 026124          4$           ;RETURN POINT FOR ERROR HEADER
3504 026120 026130          5$           ;RETURN POINT FOR ERROR DATA
3505 026122 026134          TST65        ;RETURN FOR GOOD COMPARISON
3506 026124
3507 026126 104004 000207   4$:     EMT      4
      026124 000207   RTS      PC        ;RETURN TO "COMPAR"
3508 026130
      026130 104005   5$:     EMT      5
    
```



```

3509                                     :HEADER WORDS
3510                                     :5 TO 260 ARE DATA WORDS
3511 026132 000207                       RTS    PC      :RETURN TO 'COMPAR'
3512
3522                                     :*****
:*****
:*TEST 65      READ ECC ENABLED 3B
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A CORRECTABLE ERROR IS INSERTED IN BIT POSITION 4128
:*THIS IS THE LAST BIT OF THE ECC
:*GOOD DATA USED IS 256 WORDS OF 52525
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
:*****
3523 026134 000004 001100 003610  TST65:  SCOPE
3524 026136 012706 000065          MOV    #STACK,SP      :RESET STACK
3525 026142 012737 000065          MOV    #65,TSTNM      :MOVE #65 TO TEST NUMBER
3526
3527                                     :SETUP FOR WHAT IS TO BE READ
3528                                     :HEADER CRC IS RESTORED FROM A SUBROUTINE
3529 026150 012746 052525          MOV    #52525,-(SP)   :DATA TO BE READ
3530 026154 012705 000400          MOV    #256,R5       :COUNTER
3531 026160 012700 040626          MOV    #DISK,R0      :START OF SIMULATED DISK DATA
3532 026164 011620 1$:          MOV    (SP),(R0)+    :MOVE IN DATA ON TO SIMULATED DISK
3533 026166 005305          DEC    R5            :COUNT
3534 026170 001375          BNE    1$           :BRANCH IF 256 NOT COMPLETE
3535 026172 005726          TST    (SP)+        :UNDO -(SP)
3536 026174 022020          CMP    (R0)+,(R0)+  :JUMP OVER THE TWO ECC WORDS
3537 026176 012705 000017          MOV    #15,R5 ;1 DATA GAP
3538                                     :14 TOLERANCE GAP
3539 026202 005020 2$:          CLR    (R0)+        :CLEAR DATA GAP, AND
3540 026204 005305          DEC    R5           :TOLERANCE GAP
3541 026206 001375          BNE    2$           :BRANCH IF NOT COMPLETE
3542
3543
3544 026210 004737 035352          JSR    PC,FILLEC    :INSERT ECC !N PROPER PLACE ON DISK
3545
3546
3547
3548                                     :THESE ARE FOR ECC TEST ONLY
3549
3550 026214 012737 177777 001424  MOV    #-1,TSECC     :THIS IS AN ECC TEST
3551 026222 005037 034570          CLR    POSITI       :CLEAR ERROR POSITION COUNTER
3552 026226 013737 034564 034566  MOV    NCODE,NCOUNT :TEMPORARY N-CODE COUNTER
3553 026234 013737 034572 034600  MOV    HARDER,HADTMP :TEMPORARY HARD ERROR COUNTER
3554 026242 005037 034556          CLR    GECC1        :ECC LOW ORDER TO BE GENERATED
3555 026246 005037 034560          CLR    GECC2        :ECC HIGH ORDER TO BE GENERATED
3556 026252 005037 034574          CLR    DATENV       :CLEAR DATA ENVELOPE CLOCK COUNT
3557 026256 005037 034576          CLR    ZCODE        :CLEAR LEADING ZEROS CLOCK COUNT
3558
3559
3560                                     :THESE ARE TO SETUP FOR DISKLESS USE ONLY
3561
3562 026262 012737 010000 036710  MOV    #FMT22,CYL   :16 BITS PER WORD
3563                                     :CYLINDER 0, FORMAT 16 BITS
3564 026270 112737 000000 036713  MOVB   #0,SECOTR+1  :TRACK 0
    
```

```

3565 026276 112737 000000 036712      MOVB   #0,SECOTR      ;SECTOR 0
3566 026304 012737 000000 036714      MOV    #0,KEY1 ;KEY1=0
3567 026312 012737 000000 036716      MOV    #0,KEY2 ;KEY2=0
3568 026320 012737 000400 036770      MOV    #256.,DAWORD  ;NO. OF DATA WORDS
3569 026326 005037 036720      CLR    X              ;THIS IS A READ COMMAND
3570 026332 004537 033270      JSR    R5,CRC        ;GO TO CALCULATE CRC
3571 026336 036710      CYL
3572 026340 040610      WCRC
3573
3574
3575      ;THIS IS TO INSERT ERROR
3576      ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
3577      ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
3578      ;THIS MOVE
3579      ;THIS CHANGES THE LAST BIT OF THE ECC
3580
3581 026342 013746 041630      MOV    WECC2,-(SP)   ;GET LAST ECC
3582 026346 005116      COM    (SP)          ;INVERT ALL BITS OF WECC2
3583 026350 042716 077777      BIC    #^C100000,(SP) ;KEEP BIT 16
3584 026354 042737 100000 041630      BIC    #100000,WECC2 ;CLEAR BIT 16 IN ECC
3585 026362 052637 041630      BIS    (SP)+,WECC2  ;THIS WILL SET BIT 16 IF IT WAS 0
3586                                     ;OR WILL SET NOTHING IF IT WAS A 1
3587
3588
3589 026366 012737 010026 026542      MOV    #4118.,8$    ;INSERT POSITION REG.
3590
3591
3592      ;THESE ARE REGULAR SETUPS
3593
3594 026374 004737 032054      JSR    PC,CLDISK    ;SETUP GENERAL REGISTERS
3595 026400 012777 177374 152624      MOV    #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
3596 026406 012777 002544 152620      MOV    #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
3597 026414 112746 000000      MOVVB #0,-(SP)      ;IN LOWER BYTE GET SECTOR
3598 026420 112766 000000 000001      MOVVB #0,1(SP)      ;GET TRACK IN HIGHER BYTE
3599 026426 012677 152612      MOV    (SP)+,@RHDST ;TRACK/SECTOR IN RHDST
3600 026432 012777 010000 152610      MOV    #FMT22,@RHOF ;16 BITS PER WORD
3601                                     ;ECC CORRECTION NOT INHIBIT
3602                                     ;BECAUSE ECC IS NOT GOING
3603                                     ;TO BE CHECKED
3604 026440 005077 152606      CLR    @RHCA        ;CYLINDER 0
3605
3606 026444 004737 032116      JSR    PC,CHECKT    ;CHECK THAT DVA,RDY,DPR,DRY = 1
    026450 104401 053407      TYPE   ,CPHALT      ;AND THAT NO OTHERS = 1. CANNOT CON-
    026454 000000      HALT                ;TINUE TESTING IF BOTH AREN'T TRUE
    ;STOP THE TEST
3607
3608 026456 013711 001460      MOV    REFOR,@R1    ;READ HEADER AND DATA=72
3609 026462 005037 001406      CLR    ERFLG$ ;CLEAR ERROR FLAG
3610 026466 004737 036550      JSR    PC,COMHD     ;READ HEADER AND DATA
3611                                     ;IF THERE ARE READ ERRORS THEN
3612                                     ;ECC WILL NOT BE CHECKED
3613
3614
3615      ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
3616      ;FROM THE "COMHD" ROUTINE THAT MEANS SECTOR GAP,
3617      ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
3618      ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
    
```



```

3619      ;DETECTED
3620      ;HEADER AND DATA ARE TO BE CHECKED.
3621      ;IN CHECKING READ DATA THE WRITE FROM BUFFER
3622      ;'WRFROM' IS FILLED WITH EXPECTED DATA AND
3623      ;COMPARISONS ARE MADE
3624
3625 026472 005737 001406      TST      ERFLG$ ;ANY ERRORS ALREADY THERE
3626 026476 001074      BNE      TST66 ;BRANCH IF YES
3627 026500 004737 031554      JSR      PC,PUTREG ;SAVE REGISTERS
3628 026504 022737 100000 001316  CMP      #DCK,ER1 ;ONLY DATA CHECK ERROR SHOULD BE SET
3629 026512 001401      BEQ      6$ ;BRANCH IF YES
3630 026514 104032      EMT      32
3631
3632      ;ZERO
3633      ;ONLY 11 OF THE 32 BITS CAN BE SEEN
3634      ;IN THE PATERN REGISTER
3635 026516 013746 034556      6$:     MOV      GECC1,-(SP) ;DCK SHOULD BE SET IN RHER1
3636 026522 042716 174000      BIC      #174000,(SP) ;GET PATTERN REGISTER
3637 026526 022637 001346      CMP      (SP)+,EC2 ;KEEP ONLY 11 BITS
3638 026532 001401      BEQ      7$ ;COMPARE PATTERN REGISTER
3639 026534 104032      EMT      32 ;BRANCH IF GOOD
3640
3641 026536 004037 035200      7$:     JSR      RO,ECORR ;GO TO ECC CORRECTION PROCESS
3642 026542 010026      8$:     4118. ;EXPECTED POSITION REG. WHEN CORRECTION
3643      ;IS COMPLETE
3644
3645
3646
3647 026544 004737 032306      JSR      PC,CHECKE ;CHECK THAT BITS = 1
3648 026550 104401 053407      TYPE    ,CPHALT ;CANNOT CONTINUE TESTING IF THEY DON'T
3649 026554 000000      HALT    ;STOP THE TEST
3650 026556 012700 001500      MOV      #WRFROM,RO ;GETTING READY TO FILL EXPECTED DATA
3651 026562 012720 010000      MOV      #0!FMT22,(RO)+ ;CYLINDER 0
3652 026566 112746 000000      MOV      #0,-(SP) ;IN LOWER BYTE GET SECTOR
3653 026572 112766 000000 000001  MOV      #0,1(SP) ;GET TRACK IN HIGHER BYTE
3654 026600 012620      MOV      (SP)+,(RO)+ ;GET TRACK/SECTOR IN BUFFER
3655 026602 012720 000000      MOV      #0,(R0)+ ;KEY1 IN BUFFER
3656 026606 012720 000000      MOV      #0,(R0)+ ;KEY2 IN BUFFER
3657 026612 012701 000400      MOV      #256,R1 ;DATA WORD COUNTER
3658 026616 012702 052525      MOV      #52525,R2 ;DATA
3659 026622 010220      3$:     MOV      R2,(R0)+ ;DATA INTO BUFFER
3660 026624 005301      DEC      R1 ;COUNT
3661 026626 001375      BNE      3$ ;BRANCH IF 256 NOT DONE
3662
3663      ;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
3664      ;NOW THE INSERTED ERROR WILL BE PUT IN
3665      ;BUT INSERTED ERROR IS IN ECC SO DATA IS NOT WRONG
3666 026630 004737 031554      JSR      PC,PUTREG ;SAVE REGISTERS
3667 026634 005037 001406      CLR      ERFLG$ ;CLEAR ERROR FLAG
3668
3669
3670      ;NOW READ DATA BUFFER WILL BE CHECKED
3671
3672 026640 004037 032756      JSR      RO,COMPAR ;CHECK
3673 026644 001500      WRFROM ;GOOD BUFFER

```

```

3674 026646 002544          REINTO          ;TEST BUFFER
3675 026650 000404          4+256.        ;NUMBER OF WORDS CHECKED
3676 026652 026660          4$            ;RETURN POINT FOR ERROR HEADER
3677 026654 026664          5$            ;RETURN POINT FOR ERROR DATA
3678 026656 026670          TST66         ;RETURN FOR GOOD COMPARISON
3679 026660                4$:
026660 104004          EMT          4
3680 026662 000207          RTS          PC          ;RETURN TO "COMPAR"
3681 026664                5$:
026664 104005          EMT          5
3682                                ;HEADER WORDS
3683                                ;5 TO 260 ARE DATA WORDS
3684 026666 000207          RTS          PC          ;RETURN TO "COMPAR"
3685
3695
:*****
:*TEST 66          READ ECC ENABLED 3C
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A NON CORRECTABLE ERROR IS INSERTED IN BIT POSITION 296 THRU 308
:*THIS IS IN WORD NUMBER 19 AND 20
:*GOOD DATA USED IS 256 WORDS OF 52525
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
:*****
TST66:  SCOPE
3696 026670 000004          MOV          #STACK,SP      ;RESET STACK
3697 026672 012706 001100  MOV          #66,TSTNM      ;MOVE #66 TO TEST NUMBER
3698
3699
3700          ;SETUP FOR WHAT IS TO BE READ
3701          ;HEADER CRC IS RESTORED FROM A SUBROUTINE
3702 026704 012746 052525          MOV          #52525,-(SP)    ;DATA TO BE READ
3703 026710 012705 000400          MOV          #256.,R5      ;COUNTER
3704 026714 012700 040626          MOV          #DISK,R0      ;START OF SIMULATED DISK DATA
3705 026720 011620                1$:  MOV          (SP),(R0)+      ;MOVE IN DATA ON TO SIMULATED DISK
3706 026722 005305          DEC          R5            ;COUNT
3707 026724 001375          BNE          1$            ;BRANCH IF 256 NOT COMPLETE
3708 026726 005726          TST          (SP)+         ;UNDO -(SP)
3709 026730 022020          CMP          (R0)+,(R0)+    ;JUMP OVER THE TWO ECC WORDS
3710 026732 012705 000017          MOV          #15.,R5 ;1 DATA GAP
3711
3712 026736 005020                2$:  CLR          (R0)+         ;14 TOLERANCE GAP
3713 026740 005305          DEC          R5            ;CLEAR DATA GAP, AND
3714 026742 001375          BNE          2$            ;TOLERANCE GAP
3715
3716
3717 026744 004737 035352          JSR          PC,FILLEC      ;INSERT THE TWO ECC WORDS ON THE DISK
3718                                ;IN THE CORRECT PLACE
3719
3720          ;THESE ARE FOR ECC TEST ONLY
3721
3722 026750 012737 177777 001424          MOV          #-1,TSECC      ;THIS IS AN ECC TEST
3723 026756 005037 034570          CLR          POSITI        ;CLEAR ERROR POSITION COUNTER
3724 026762 013737 034564 034566          MOV          NCODE,NCOUNT  ;TEMPORARY N-CODE COUNTER
3725 026770 013737 034572 034600          MOV          HARDER,HADTMP ;TEMPORARY HARD ERROR COUNTER
3726 026776 005037 034556          CLR          GECC1         ;ECC LOW ORDER TO BE GENERATED
3727 027002 005037 034560          CLR          GECC2         ;ECC HIGH ORDER TO BE GENERATED
    
```


3728 027006 005037 034574 CLR DATENV ;CLEAR DATA ENVELOPE CLOCK COUNT
 3729 027012 005037 034576 CLR ZCODE ;CLEAR LEADING ZEROS CLOCK COUNT

3730
 3731
 3732 ;THESE ARE TO SETUP FOR DISKLESS USE ONLY
 3733

3734 027016 012737 010000 036710 MOV #FMT22,CYL ;16 BITS PER WORD
 ;CYLINDER 0, FORMAT 16 BITS
 3735
 3736 027024 112737 000000 036713 MOVB #0,SECOTR+1 ;TRACK 0
 3737 027032 112737 000000 036712 MOVB #0,SECOTR ;SECTOR 0
 3738 027040 012737 000000 036714 MOV #0,KEY1 ;KEY1=0
 3739 027046 012737 000000 036716 MOV #0,KEY2 ;KEY2=0
 3740 027054 012737 000400 036770 MOV #256.,DAWORD ;NO. OF DATA WORDS
 3741 027062 005037 036720 CLR X ;THIS IS A READ COMMAND
 3742 027066 004537 033270 JSR R5,CRC ;GO TO CALCULATE CRC
 3743 027072 036710 CYL
 3744 027074 040610 WCRC

3745
 3746
 3747 ;THIS IS TO INSERT ERROR
 3748 ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
 3749 ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
 3750 ;THIS MOVE

3751
 3752 027076 012737 152652 040672 MOV #152652,DISK+44;INSERT ERROR IN POSITION 296 THRU 304
 ;IN WORD NUMBER 19
 3753
 3754 027104 012737 052532 040674 MOV #52532,DISK+46;INSERT ERROR IN POSITION 305 THRU 308
 ;IN WORD NUMBER 20
 3755
 3756 027112 012737 010040 027266 MOV #4128.,8\$;INSERT POSITION REG.

3757
 3758
 3759 ;THESE ARE REGULAR SETUPS
 3760

3761 027120 004737 032054 JSR PC,CLDISK ;SETUP GENERAL REGISTERS
 3762 027124 012777 177374 152100 MOV #-256.-4.,@RHWC ;256. DATA 4 HEADER WORDS
 3763 027132 012777 002544 152074 MOV #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
 3764 027140 112746 000000 MOVB #0,-(SP) ;IN LOWER BYTE GET SECTOR
 3765 027144 112766 000000 000001 MOVB #0,1(SP) ;GET TRACK IN HIGHER BYTE
 3766 027152 012677 152066 MOV (SP)+,@RHDST ;TRACK/SECTOR IN RHDST
 3767 027156 012777 010000 152064 MOV #FMT22,@RHOF ;16 BITS PER WORD
 ;ECC CORRECTION NOT INHIBIT
 ;BECAUSE ECC IS NOT GOING
 ;TO BE CHECKED
 3768
 3769
 3770
 3771 027164 005077 152062 CLR @RHCA ;CYLINDER 0
 3772 027170 004737 032116 JSR PC,CHECKT ;CHECK THAT DVA,RDY,DPR,DRY = 1
 027174 104401 053407 TYPE ,CPHALT ;AND THAT NO OTHERS = 1. CANNOT CON-
 ;TINUE TESTING IF BOTH AREN'T TRUE
 ;STOP THE TEST
 3773 027200 000000 HALT ;READ HEADER AND DATA=72
 3774 027202 013711 001460 MOV REFOR,@R1
 3775 027206 005037 001406 CLR ERFLG\$;CLEAR ERROR FLAG
 3776 027212 004737 036550 JSR PC,COMHD ;READ HEADER AND DATA
 ;IF THERE ARE READ ERRORS THEN
 ;ECC WILL NOT BE CHECKED

3777
 3778
 3779
 3780 ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
 3781 ;FROM THE 'COMHD' ROUTINE THAT MEANS SECTOR GAP,

```

3782 :FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
3783 :SYNC BYTE HAVE GONÉ BY AND SYNCs WERE CORRECTLY
3784 :DETECTED
3785 :HEADER AND DATA ARE TO BE CHECKED.
3786 :IN CHECKING READ DATA THE WRITE FROM BUFFER
3787 :'WRFROM' IS FILLED WITH EXPECTED DATA AND
3788 :COMPARISONS ARE MADE
3789
3790 027216 005737 001406 TST ERFLG$ ;ANY ERRORS ALREADY THERE
3791 027222 001111 BNE TST67 ;BRANCH IF YES
3792 027224 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
3793 027230 022737 100000 001316 CMP #DCK,ER1 ;ONLY DATA CHECK ERROR SHOULD BE SET
3794 027236 001401 BEQ 6$ ;BRANCH IF YES
3795 027240 104032 EMT 32
3796
3797 :ZERO
3798 :ONLY 11 OF THE 32 BITS CAN BE SEEN
3799 :IN THE PATERN REGISTER
3800 027242 013746 034556 6$: MOV CECC1,-(SP) ;DCK SHOULD BE SET IN RHER1
3801 027246 042716 174000 BIC #174000,(SP) ;GET PATTERN REGISTER
3802 027252 022637 001346 CMP (SP)+,EC2 ;KEEP ONLY 11 BITS
3803 027256 001401 BEQ 7$ ;COMPARE PATTERN REGISTER
3804 027260 104032 EMT 32 ;BRANCH IF GOOD
3805
3806 027262 004037 035200 7$: JSR R0,ECORR ;GO TO ECC CORRECTION PROCESS
3807 027266 000000 8$: .WORD ;EXPECTED POSITION REG. WHEN CORRECTION
3808 ;IS COMPLETE
3809
3810
3811
3812 027270 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
3813 027274 022737 100100 001316 CMP #DCK!ECH,ER1 ;WITH ERRORS INSERTED IN BIT POSITION 21
3814 ;THRU 32 HARD ERROR BIT SHOULD SET
3815 027302 001401 BEQ 9$ ;BRANCH IF GOOD
3816 027304 104036 EMT 36
3817 ;32 HCE SHOULD SET
3818
3819
3820
3821 027306 9$: JSR PC,CHECKE ;CHECK THAT BITS = 1
027306 004737 032306 TYPE ,CPHALT ;CANNOT CONTINUE TESTING IF THEY DON'T
027312 104401 053407 HALT ;STOP THE TEST
027316 000000 MOV #WRFROM,R0 ;GETTING READY TO FILL EXPECTED DATA
3822 027320 012700 001500 MOV #0!FMT22,(R0)+ ;CYLINDER 0
3823 027324 012720 010000 MOV #0,-(SP) ;IN LOWER BYTE GET SECTOR
3824 027330 112746 000000 MOV #0,1(SP) ;GET TRACK IN HIGHER BYTE
3825 027334 112766 000000 000001 MOV (SP)+,(R0)+ ;GET TRACK/SECTOR IN BUFFER
3826 027342 012620 MOV #0,(R0)+ ;KEY1 IN BUFFER
3827 027344 012720 000000 MOV #0,(R0)+ ;KEY2 IN BUFFER
3828 027350 012720 000000 MOV #0,(R0)+ ;DATA WORD COUNTER
3829 027354 012701 000400 MOV #256,R1 ;DATA
3830 027360 012702 052525 MOV #52525,R2 ;DATA INTO BUFFER
3831 027364 010220 3$: MOV R2,(R0)+ ;COUNT
3832 027366 005301 DEC R1 ;BRANCH IF 256 NOT DONE
3833 027370 001375 BNE 3$
3834
3835 ;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
    
```



```
3836 ;NOW THE INSERTED ERROR WILL BE PUT IN
3837
3838 027372 012737 152652 001554 MOV #152652,WRFROM+54;INSERT ERROR IN POSITION 296 THRU 304
3839 ;IN WORD NUMBER 19 IN DATA
3840 027400 012737 052532 001556 MOV #52532,WRFROM+56;INSERT ERROR IN POSITION 305 THRU 308
3841 ;IN WORD NUMBER 20 IN DATA
3842
3843 027406 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
3844 027412 005037 001406 CLR ERFLG$ ;CLEAR ERROR FLAG
3845
3846
```

```
3847 ;NOW READ DATA BUFFER WILL BE CHECKED
3848
3849 027416 004037 032756 JSR R0,COMPAR ;CHECK
3850 027422 001500 WRFROM ;GOOD BUFFER
3851 027424 002544 REINTO ;TEST BUFFER
3852 027426 000404 4+256. ;NUMBER OF WORDS CHECKED
3853 027430 027436 4$ ;RETURN POINT FOR ERROR HEADER
3854 027432 027442 5$ ;RETURN POINT FOR ERROR DATA
3855 027434 027446 TST67 ;RETURN FOR GOOD COMPARISON
3856 027436 104004 4$: EMT 4
3857 027440 000207 RTS PC ;RETURN TO 'COMPAR'
3858 027442 104005 5$: EMT 5
3859 ;HEADER WORDS
3860 ;5 TO 260 ARE DATA WORDS
3861 027444 000207 RTS PC ;RETURN TO 'COMPAR'
3862
3872
```

```
::*****
:*TEST 67 READ ECC ENABLED 3D
:*THIS IS AN ECC READ DATA TEST
:*ERROR CORRECTION IS ENABLED
:*A NON CORRECTABLE ERROR IS INSERTED IN BIT POSITION 32 AND 4096
:*4096 IS THE LAST DATA BIT
:*GOOD DATA USED IS 256 WORDS OF 52525
:*COMMAND IS GIVEN FOR CYLINDER 0 FORMAT 16 BITS PER WORD
:*TRACK 0, SECTOR 0 KEYS 0 READ HEADER AND DATA
:*****
```

```
3873 027446 000004 001100 TST67: SCOPE
3874 027450 012706 001100 MOV #STACK,SP ;RESET STACK
3875 027454 012737 000067 003610 MOV #67,TSTNM ;MOVE #67 TO TEST NUMBER
3876
3877
3878 ;SETUP FOR WHAT IS TO BE READ
3879 ;HEADER CRC IS RESTORED FROM A SUBROUTINE
3880
3881 027462 012746 052525 MOV #52525,-(SP) ;DATA TO BE READ
3882 027466 012705 000400 MOV #256,R5 ;COUNTER
3883 027472 012700 040626 MOV #DISK,R0 ;START OF SIMULATED DISK DATA
3884 027476 011620 1$: MOV (SP),(R0)+ ;MOVE IN DATA ON TO SIMULATED DISK
3885 027500 005305 DEC R5 ;COUNT
3886 027502 001375 BNE 1$ ;BRANCH IF 256 NOT COMPLETE
3887 027504 005726 TST (SP)+ ;UNDO -(SP)
3888 027506 022020 CMP (R0)+,(R0)+ ;JUMP OVER THE TWO ECC WORDS
3889 027510 012705 000017 MOV #15,R5 ;1 DATA GAP
```

```

3890
3891 027514 005020      2$:   CLR      (R0)+      :14 TOLERANCE GAP
3892 027516 005305      DEC      R5          :CLEAR DATA GAP, AND
3893 027520 001375      BNE      2$         :TOLERANCE GAP
3894
3895
3896 027522 004737 035352      JSR      PC,FILLEC  :INSERT THE TWO ECC WORDS ON THE DISK
3897
3898
3899
3900
3901 027526 012737 177777 001424      MOV      #-1,TSECC  :THIS IS AN ECC TEST
3902 027534 005037 034570      CLR      POSITI    :CLEAR ERROR POSITION COUNTER
3903 027540 013737 034564 034566      MOV      NCODE,NCOUNT :TEMPORARY N-CODE COUNTER
3904 027546 013737 034572 034600      MOV      HARDER,HADTMP :TEMPORARY HARD ERROR COUNTER
3905 027554 005037 034556      CLR      GECC1     :ECC LOW ORDER TO BE GENERATED
3906 027560 005037 034560      CLR      GECC2     :ECC HIGH ORDER TO BE GENERATED
3907 027564 005037 034574      CLR      DATENV    :CLEAR DATA ENVELOPE CLOCK COUNT
3908 027570 005037 034576      CLR      ZCODE     :CLEAR LEADING ZEROS CLOCK COUNT
3909
3910
3911
3912
3913 027574 012737 010000 036710      MOV      #FMT22,CYL :16 BITS PER WORD
3914
3915 027602 112737 000000 036713      MOV      #0,SECOTR+1 :CYLINDER 0, FORMAT 16 BITS
3916 027610 112737 000000 036712      MOV      #0,SECOTR  :TRACK 0
3917 027616 012737 000000 036714      MOV      #0,KEY1    :KEY1=0
3918 027624 012737 000000 036716      MOV      #0,KEY2    :KEY2=0
3919 027632 012737 000400 036770      MOV      #256.,DAWORD :NO. OF DATA WORDS
3920 027640 005037 036720      CLR      X          :THIS IS A READ COMMAND
3921 027644 004537 033270      JSR      R5,CRC    :GO TO CALCULATE CRC
3922 027650 036710
3923 027652 040610
3924
3925
3926
3927
3928
3929
3930
3931 027654 012737 152525 040630      MOV      #152525,DISK+2 :FORCE ERROR ON BIT NUMBER 32
3932 027662 012737 152525 041624      MOV      #152525,DISK+<255.*2> :FORCE ERROR IN BIT 4096
3933 027670 012737 010040 030044      MOV      #4128.,8$    :INSERT POSITION REG.
3934
3935
3936
3937
3938 027676 004737 032054      JSR      PC,CLDISK  :SETUP GENERAL REGISTERS
3939 027702 012777 177374 151322      MOV      #-256.-4.,@RHWC :256. DATA 4 HEADER WORDS
3940 027710 012777 002544 151316      MOV      #REINTO,@RHBA :STARTING ADDRESS OF READ BUFFER
3941 027716 112746 000000      MOV      #0,-(SP)    :IN LOWER BYTE GET SECTOR
3942 027722 112766 000000 000001      MOV      #0,1(SP)   :GET TRACK IN HIGHER BYTE
3943 027730 012677 151310      MOV      (SP)+,@RHDST :TRACK/SECTOR IN RHDST
3944 027734 012777 010000 151306      MOV      #FMT22,@RHOF :16 BITS PER WORD
3945
3946

```

;THESE ARE FOR ECC TEST ONLY

;THESE ARE TO SETUP FOR DISKLESS USE ONLY

;THIS IS TO INSERT ERROR
 ;THE DISK DATA IS IN LOCATION STARTING FROM 'DISK'
 ;THE POSITION OF THE ERROR CAN BE CHANGED BY CHANGING
 ;THIS MOVE

;THESE ARE REGULAR SETUPS

;ECC CORRECTION NOT INHIBIT
 ;BECAUSE ECC IS NOT GOING


```

3947
3948 027742 005077 151304 CLR @RHCA ;TO BE CHECKED
3949 ;CYLINDER 0
3950 027746 004737 032116 JSR PC,CHECKT ;CHECK THAT DVA,RDY,DPR,DRY = 1
027752 104401 053407 TYPE ,CPHALT ;AND THAT NO OTHERS = 1. CANNOT CON-
;TINUE TESTING IF BOTH AREN'T TRUE
027756 000000 HALT ;STOP THE TEST
3951
3952 027760 013711 001460 MOV REFOR,@R1 ;READ HEADER AND DATA=72
3953 027764 005037 001406 CLR ERFLG$ ;CLEAR ERROR FLAG
3954 027770 004737 036550 JSR PC,COMHD ;READ HEADER AND DATA
;IF THERE ARE READ ERRORS THEN
;ECC WILL NOT BE CHECKED
3955
3956
3957
3958
3959 ;IF THE PROGRAM COMES BACK HERE WITHOUT ERROR PRINTOUTS
3960 ;FROM THE "COMHD" ROUTINE THAT MEANS SECTOR GAP,
3961 ;FIRST SYNC, HEADER, HEADER CRC, HEADER GAP AND
3962 ;SYNC BYTE HAVE GONE BY AND SYNC'S WERE CORRECTLY
3963 ;DETECTED
3964 ;HEADER AND DATA ARE TO BE CHECKED.
3965 ;IN CHECKING READ DATA THE WRITE FROM BUFFER
3966 ;"WRFROM" IS FILLED WITH EXPECTED DATA AND
3967 ;COMPARISONS ARE MADE
3968
3969 027774 005737 001406 TST ERFLG$ ;ANY ERRORS ALREADY THERE
3970 030000 001111 BNE TST70 ;BRANCH IF YES
3971 030002 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
3972 030006 022737 100000 001316 CMP #DCK,ER1 ;ONLY DATA CHECK ERROR SHOULD BE SET
3973 030014 001401 BEQ 6$ ;BRANCH IF YES
3974 030016 104032 EMT 32
3975 ;ZERO
3976 ;ONLY 11 OF THE 32 BITS CAN BE SEEN
3977 ;IN THE PATTERN REGISTER
3978 ;DCK SHOULD BE SET IN RHER1
3979 030020 013746 034556 6$: MOV GECC1,-(SP) ;GET PATTERN REGISTER
3980 030024 042716 174000 BIC #174000,(SP) ;KEEP ONLY 11 BITS
3981 030030 022637 001346 CMP (SP)+,EC2 ;COMPARE PATTERN REGISTER
3982 030034 001401 BEQ 7$ ;BRANCH IF GOOD
3983 030036 104032 EMT 32
3984
3985 030040 004037 035200 7$: JSR R0,ECORR ;GO TO ECC CORRECTION PROCESS
3986 030044 000000 8$: .WORD ;EXPECTED POSITION REG. WHEN CORRECTION
;IS COMPLETE
3987
3988
3989
3990
3991 030046 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
3992 030052 022737 100100 001316 CMP #DCK!ECH,ER1 ;WITH ERRORS INSERTED IN BIT POSITION 32
;AND 4096 HARD ERROR BIT SHOULD SET
3993
3994 030060 001401 BEQ 9$ ;BRANCH IF GOOD
3995 030062 104036 EMT 36
3996 ;32 HCE SHOULD SET
3997
3998
3999
4000
    
```

```

4001 030064          9$:      JSR      PC,CHECKE      ;CHECK THAT BITS = 1
      030064 004737 032306      TYPE      ,CPHALT      ;CANNOT CONTINUE TESTING IF THEY DON'T
      030070 104401 053407      HALT          ;STOP THE TEST
      030074 000000          MOV      #WRFROM,R0      ;GETTING READY TO FILL EXPECTED DATA
4002 030076 012700 001500      MOV      #0,!FMT22,(R0)+ ;CYLINDER 0
4003 030102 012720 010000      MOV      #0,-(SP)      ;IN LOWER BYTE GET SECTOR
4004 030106 112746 000000      MOV      #0,1(SP)      ;GET TRACK IN HIGHER BYTE
4005 030112 112766 000000 000001  MOV      (SP)+,(R0)+    ;GET TRACK/SECTOR IN BUFFER
4006 030120 012620 000000      MOV      #0,(R0)+      ;KEY1 IN BUFFER
4007 030122 012720 000000      MOV      #0,(R0)+      ;KEY2 IN BUFFER
4008 030126 012720 000000      MOV      #256,R1      ;DATA WORD COUNTER
4009 030132 012701 000400      MOV      #52525,R2     ;DATA
4010 030136 012702 052525      MOV      R2,(R0)+      ;DATA INTO BUFFER
4011 030142 010220 3$:      MOV      R1            ;COUNT
4012 030144 005301      DEC      R1            ;COUNT
4013 030146 001J75      BNE      3$           ;BRANCH IF 256 NOT DONE
4014
4015      ;ONLY GOOD DATA HAS BEEN PUT IN 'WRFROM'
4016      ;NOW THE INSERTED ERROR WILL BE PUT IN
4017
4018 030150 012737 152525 001512      MOV      #152525,WRFROM+<5*2> ;INSERTED ERROR IN BIT 32
4019 030156 012737 152525 002506      MOV      #152525,WRFROM+<259.*2> ;INSERT ERROR IN BIT 4096
4020
4021 030164 005037 001406      CLR      ERFLG$ ;CLEAR ERROR FLAG
4022 030170 004737 031554      JSR      PC,PUTREG    ;SAVE REGISTERS
4023
4024      ;NOW READ DATA BUFFER WILL BE CHECKED
4025
4026 030174 004037 032756      JSR      R0,COMPAR    ;CHECK
4027 030200 00150G      WRFROM      ;GOOD BUFFER (CHANGED)
4028 030202 002544      REINTO      ;TEST BUFFER
4029 030204 000404      4+256.     ;NUMBER OF WORDS CHECKED
4030 030206 030214      4$         ;RETURN POINT FOR ERROR HEADER
4031 030210 030220      5$         ;RETURN POINT FOR ERROR DATA
4032 030212 030224      TST70      ;RETURN FOR GOOD COMPARISON
4033 030214          4$:      EMT      4
4034 030216 104004      RTS      PC          ;RETURN TO "COMPAR"
4035 030220          5$:      EMT      5
4036          ;HEADER WORDS
4037          ;5 TO 260 ARE DATA WORDS
4038 030222 000207      RTS      PC          ;RETURN TO "COMPAR"
4039
4046      ;*****
      ;*TEST 70 PROGRAM INTERRUPT
      ;*PROGRAM INTERRUPT IS TESTED BY SETTING RDY AND IE
      ;*IN RHCS1 AT THE SAME TIME
      ;*THIS SHOULD INTERRUPT THROUGH LOCATION 254
      ;*THE PROCESSOR PRIORITY IS SET TO 4
      ;*****
4047 030224 000004      TST70:  SCOPE
4048 030226 012737 000070 003610      MOV      #70,TSTM     ;MOVE #70 TO TEST NUMBER
4049 030234 012706 001100      MOV      #STACK,SP    ;RESET STACK
4050 030240 004737 032054      JSR      PC,CLDISK    ;CLEAR DISK
4051 030244 013700 001226      MOV      RPVEC,R0     ;GET VECTOR ADDRESS
    
```



```
4052 030250 012720 030316      MOV      #RPTRP1,(R0)+      ;SET INTERRUPT VECTOR
4053 030254 012710 000340      MOV      #PR7,(R0)         ;SET SERVICE ROUTINE PRIORITY
4054 030260 012737 000200 177776  MOV      #PR4,PS           ;SET PROCESSOR PRIORITY
4055 030266 012711 000300      MOV      #RDY!IE,@R1       ;RDY, IE IN RHSC1 SHOULD CAUSE INTERRUPT
4056 030272 013737 032432 001200  MOV      TIMCNT,$TMP1      ;COUNTER
4057 030300 005337 001200      1$:     DEC      $TMP1      ;WAIT FOR INTERRUPT
4058 030304 001375      BNE      1$                ;BRANCH IF NOT ZERO
4059                                     ;BEFORE THIS IS ZERO INTERRUPT SHOULD
4060                                     ;OCCUR
4061 030306 004737 031554      JSR      PC,PUTREG         ;SAVE REGISTERS
4062 030312 104021      EMT      21
4063
4064 030314 000410      BR       TST71            ;BRANCH TO NEXT TEST
4065
4066 030316 022626      RPTRP1: CMP      (SP)+,(SP)+ ;RESTORE STACK
4067 030320 004737 031554      JSR      PC,PUTREG         ;SAVE REGISTERS
4068 030324 022737 004200 001314  CMP      #DVA!RDY,CS1     ;'IE' SHOULD BE LOW
4069 030332 001401      BEQ      TST71            ;BRANCH IF GOOD
4070 030334 104021      EMT      21
4071                                     ;'IE' FAILED TO RESET
4072
4077
```

```
::*****
:*TEST 71      INTERRUPT AT PROCESSOR AND DISK PRIORITY SAME
:*PROCESSOR PRIORITY IS SET AT 5 (SAME AS THE DISK)
:*IE AND RDY IS SET. THIS SHOULD NOT INTERRUPT
:*****
```

```
4078 030336 000004      TST71: SCOPE
4079 030340 012737 000071 003610  MOV      #71,TSTNM        ;MOVE #71 TO TEST NUMBER
4080 030346 012706 001100      MOV      #STACK,SP       ;RESET STACK
4081 030352 004737 032054      JSR      PC,CLDISK        ;CLEAR DISK
4082 030356 013700 001226      MOV      RPVEC,R0         ;GET VECTOR ADDRESS
4083 030362 012720 030422      MOV      #RPTRP2,(R0)+   ;SET INTERRUPT VECTOR
4084 030366 012710 000340      MOV      #PR7,(R0)       ;SET SERVICE ROUTINE PRIORITY
4085 030372 012737 000240 177776  MOV      #PR5,PS           ;SET PROCESSOR PRIORITY
4086 030400 012711 000300      MOV      #RDY!IE,@R1     ;RDY, IE IN RHSC1 SHOULD CAUSE INTERRUPT
4087 030404 013737 032432 001200  MOV      TIMCNT,$TMP1;COUNTER
4088 030412 005337 001200      1$:     DEC      $TMP1      ;WAIT FOR INTERRUPT
4089 030416 001375      BNE      1$                ;BRANCH IF NOT ZERO
4090                                     ;BEFORE THIS IS ZERO INTERRUPT SHOULD
4091                                     ;OCCUR
4092 030420 000404      BR       TST72            ;NO INTERRUPT SO BRANCH
4093
4094 030422 022626      RPTRP2: CMP      (SP)+,(SP)+ ;RESTORE STACK
4095 030424 004737 031554      JSR      PC,PUTREG         ;SAVE REGISTERS
4096 030430 104021      EMT      21
4097                                     ;PROCESSOR STATUS SAME
4098                                     ;AS DISK
4099
4106
```

```
::*****
:*TEST 72      END OF DRIVE
:*THIS IS THE END OF TEST FOR ONE DRIVE
:*IF THERE ARE MORE DRIVES THEN THE PROGRAM
:*JUMPS TO TEST 5 FOR NEXT DRIVE TEST
:*END PASS IS REACHED ONLY AFTER ALL DRIVES ARE COMPLETE
:*****
```

030432 000004

TST72: SCOPE

```

4107 030434 012737 000001 001212      MOV    #1,$TIMES      ;;DO 1 ITERATION
4108 030442 004737 032054                JSR    PC,CLDISK
4109 030446 012737 000000 177776      MOV    #0,PS          ;REINSTATE PS TO 0
4110 030454 104401 030462                TYPE  ,65$           ;;TYPE ASCIZ STRING
030460 000424                BR     64$           ;;GET OVER THE ASCIZ
                                ;;65$: .ASCIZ <CRLF>/TOTAL ERRORS FOR THIS PASS ON UNIT NO./
                                64$:
4111 030532 013746 001374      MOV    UNIT,-(SP)     ;GET READY TO TYPE UNIT NUMBER
4112 030536 104405                TYPDS
4113 030540 104401 030546      TYPE  ,67$           ;;TYPE ASCIZ STRING
030544 000402                BR     66$           ;;GET OVER THE ASCIZ
                                ;;67$: .ASCIZ /= /
                                66$:
4114 030552 013746 001112      MOV    $ERTTL,-(SP)   ;GET READY TO TYPE NUMBER OF ERRORS
4115 030556 104405                TYPDS
4116 030560 005037 001112      CLR    $ERTTL         ;CLEAR TOTAL NUMBER OF ERRORS
4117 030564 005037 001102      CLR    $STNM         ;CLEAR TEST NUMBER
4118 030570 005737 001402      TST   SELECT         ;STARTING FROM 200 ?
4119 030574 001413      BEQ   3$             ;TEST NEXT DRIVE IF SO
4120
4121 030576 005237 001100      INC    $PASS         ;INCREASE PASS COUNT
4122 030602 104401 030765      TYPE  ,SENDMG        ;TYPE END PASS #
4123 030606 013746 001100      MOV    $PASS,-(SP)
4124 030612 104405                TYPDS
4125 030614 104401 030762      TYPE  ,SENULL
4126 030620 000137 006756      JMP   TST5           ;CONTINUE TESTING THIS DRIVE
4127
4128 030624 005337 001376      3$:  DEC    NOUNITS    ;NO. OF UNITS PRESENT DECREMENTED
4129 030630 001413      BEQ   $EOP          ;BRANCH IF ALL DRIVES COMPLETE
4130 030632 013700 001374      MOV    UNIT,R0       ;UNIT UNDER TEST
4131 030636 012701 001354      MOV    #UNITS,R1    ;TABLE
4132 030642 022100                CMP    (R1)+,R0     ;IS THIS UNIT JUST TESTED
4133 030644 001401                BEQ   2$           ;BRANCH IF YES
4134 030646 000775                BR     1$          ;BRANCH IF NO
4135 030650 011137 001374      2$:  MOV    (R1),UNIT   ;THIS IS NEXT UNIT
4136 030654 000137 006756      JMP   TST5           ;TEST NEXT DRIVE
  
```


1

.SBTTL END OF PASS ROUTINE

```

:*****
:*INCREMENT THE PASS NUMBER ($PASS)
:*TYPE 'END PASS #XXXXX' (WHERE X IS A DECIMAL NUMBER)
:*IF THERES A MONITOR GO TO IT
:*IF THERE ISN'T JUMP TO TST1
  
```

030660				\$EOP:			
030660	000004			SCOPE			
030662	005037	001102		CLR	\$TSTNM	::ZERO THE TEST NUMBER	
030666	005037	001212		CLR	\$TIMES	::ZERO THE NUMBER OF ITERATIONS	
030672	005237	001100		INC	\$PASS	::INCREMENT THE PASS NUMBER	
030676	042737	100000	001100	BIC	#100000,\$PASS	::DON'T ALLOW A NEG. NUMBER	
030704	005327			DEC	(PC)+	::LOOP?	
030706	000001			\$EOPCT:	.WORD	1	
030710	003022			BGT	\$DOAGN	::YES	
030712	012737			MOV	(PC)+,@(PC)+	::RESTORE COUNTER	
030714	000001			\$ENDCT:	.WORD	1	
030716	030706			\$EOPCT			
030720	104401	030765		TYPE	,\$SENDMG	::TYPE 'END PASS #'	
030724	013746	001100		MOV	\$PASS,-(SP)	::SAVE \$PASS FOR TYPEOUT	
030730	104405			TYPDS		::GO TYPE--DECIMAL ASCII WITH SIGN	
030732	104401	030762		TYPE	,\$ENULL	::TYPE A NULL CHARACTER	
030736	013700	000042		\$GET42:	MOV	@#42,R0	::GET MONITOR ADDRESS
030742	001405			BEQ	\$DOAGN	::BRANCH IF NO MONITOR	
030744	000005			RESET		::CLEAR THE WORLD	
030746	004710			\$ENDAD:	JSR	PC,(R0)	::GO TO MONITOR
030750	000240			NOP		::SAVE ROOM	
030752	000240			NOP		::FOR	
030754	000240			NOP		::ACT11	
030756				\$DOAGN:			
030756	000137			JMP	@(PC)+	::RETURN	
030760	005362			\$RTNAD:	.WORD	TST1	
030762	377	377	000	\$ENULL:	.BYTE	-1,-1,0	::NULL CHARACTER STRING
030765	015	012	105	\$ENDMG:	.ASCIZ	<15><12>/END PASS #/	

2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22 031002 000000
23
24 031004
25 031004 005037 177776
26 031010 104401 031016
031014 000421
031060
27 031060 013746 003610
28 031064 104402
29 031066 104401 031074
031072 000414
031124
30 031124 013746 001110
31 031130 104402
32 031132 104401 001223
33 031136 104401 031144
031142 000425
031216
34 031216 104401 031224
031222 000420
031264
35 031264 104401 031272
031270 000422
031336
36 031336 104412
37 031340 062716 000002
38 031344 012637 001106
39 031350 104401 031356
031354 000416
031412

```

*****
*HERE IS A DETAILED EXPLANATION OF HOW THE LOOP ON ERROR WORKS.
*ON HITTING AN ERROR IF THE LOOP ON ERROR SWITCH IS SET, THE
*PROGRAM GOES BACK - USUALLY BACK TO THE BEGINNING OF THE TEST.

*WHEN THIS OPERATOR SELECTABLE SCOPE LOOP IS USED THEN THE POINT
*THE PROGRAM GOES BACK TO CAN BE CHANGED.
*THE RESTRICTIONS TO THE POINT WHERE THE PROGRAM CAN GO ARE: -
*1. IT MUST BE WITHIN THE TEST UNDER CONSIDERATION
*2. LOOP ON ERROR SWITCH MUST BE SET
*3. THE ERROR MUST OCCUR WITHIN THE TEST UNDER CONSIDERATION
*IF THE ERROR DOES NOT OCCUR WITHIN THE TEST UNDER CONSIDERATION
*THE PROGRAM WILL REVERT TO NORMAL OPERATION. HOWEVER, IF LOOP ON
*TEST SWITCH IS SET AND THIS OPERATOR SELECTABLE SCOPE LOOP IS USED
*THEN THE PROGRAM WILL LOOP BACK TO THE SELECTED POINT WHEN IT
*COMES TO THE END OF THE TEST UNDER CONSIDERATION.
*
*AFTER LOOPING FOR SOME TIME IF THE LOOP SWITCH IS PUT DOWN THEN
*NORMAL OPERATION WILL CONTINUE.
*****

TESTAD: 0 ;FIRST ADDRESS OF TEST

OPERSEL:
CLR PS ;MAKE PROCESSOR STATUS ZERO
TYPE ,65$ ;;TYPE ASCIZ STRING
BR 64$ ;;GET OVER THE ASCIZ
65$: .ASCIZ <CRLF>/THE PROGRAM WAS IN TEST NUMBER /
64$:
MOV TSTNM,-(SP) ;GET READY TO TYPE TEST
TYPOC ;NUMBER
TYPE ,67$ ;;TYPE ASCIZ STRING
BR 66$ ;;GET OVER THE ASCIZ
67$: .ASCIZ <CRLF>/THE LOOP BACK PC WAS /
66$:
MOV $LPERR,-(SP) ;GET READY TO TYPE LOOP BACK PC
TYPOC
TYPE ,69$ ;;TYPE ASCIZ STRING
BR 68$ ;;GET OVER THE ASCIZ
69$: .ASCIZ <CRLF>/SET LOOP ON ERROR OR LOOP ON TEST SWITCH/
68$:
TYPE ,71$ ;;TYPE ASCIZ STRING
BR 70$ ;;GET OVER THE ASCIZ
71$: .ASCIZ <CRLF>/TYPE THE FIRST PC OF THE TEST/
70$:
TYPE ,73$ ;;TYPE ASCIZ STRING
BR 72$ ;;GET OVER THE ASCIZ
73$: .ASCIZ <CRLF>/ FOLLOWED BY A CARRIAGE RETURN /<CRLF>
72$:
RDOCT
ADD #2,(SP) ;GET LPADR
MOV (SP)+,$LPADR
TYPE ,75$ ;;TYPE ASCIZ STRING
BR 74$ ;;GET OVER THE ASCIZ
75$: .ASCIZ <CRLF>/TYPE THE PC WHERE YOU WANT/
74$:
    
```


END OF PASS ROUTINE

```
40 031412 104401 031420      TYPE      77$      ;;TYPE ASCIZ STRING
   031416 000440      BR      76$      ;;GET OVER THE ASCIZ
   031520      ;;77$: .ASCIZ <CRLF>/ THE PROGRAM TO LOOP BACK TO FOLLOWED BY A CARRIAGE RETURN /<CRLF>
   031520      76$:
41 031520 104412      RDOCT
42 031522 012637 001110      MOV      (SP)+,$LPERR      ;GET LPERR
43 031526 013746 001106      MOV      $LPADR,-(SP)
44
45      ;THIS CLEARS UP GARBAGE
46 031532 005037 037024      CLR      NOSYNC      ;CLEAR FLAG FOR HEADER ERROR COMMANDS
47 031536 005037 001424      CLR      TSECC      ;CLEAR FLAG FOR ECC TEST
48                                ;WHEN =177777 IT IS AN ECC TEST
49                                ;WHEN =0 IT IS NOT AN ECC TEST
50
51 031542 005037 034562      CLR      TSECCG      ;EVEN IN AN ECC TEST EVERY CLOCK
52                                ;IS NOT TO GENERATE ECC
53                                ;IF =177777 GENERATE ECC
54                                ;IF =0 DO NOT GENERATE ECC
55 031546 005037 001426      CLR      TESDTE      ;DRIVE TIMING ERROR TEST
56 031552 000002      RTI
```

```

1
2
3
4
5
6
7
8
9
10
11
12
13 031554
    031554 010046
    031556 010146
    031560 010246
14 031562 012700 001232
15 031566 012701 001306
16 031572 012702 000023
17 031576 013021
18 031600 005302
19 031602 001375
20 031604 012602
    031606 012601
    031610 012600
21 031612 000207
    
```

```

.SBTTL SAVE REGISTERS ROUTINE
:*****
:THIS SAVES THE CONTENTS OF ALL HARDWARE REGISTERS
:IN MEMORY LOCATIONS TAGED FROM 'WC' TO 'EC2'
:
:THIS IS DONE SO THAT COMPARES ARE DONE WITH SAVED LOCATIONS
:AND NOT THE REGISTERS THEMSELVES. THIS WILL MAKE
:ERROR PRINTOUTS FOR GOOD AND BAD DATA ALWAYS DIFFRENT
:*****
PUTREG:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R2,-(SP) ;;PUSH R2 ON STACK
MOV #RHWC,R0 ;;STARTING ADDRESS OF REG
MOV #WC,R1 ;;STARTING ADDRESS OF SAVE LOCATIONS
MOV #RHCC-RHWC+2/2,R2 ;;NUMBER OF REG. INTO R2
10$: MOV @ (R0)+,(R1)+ ;SAVE HARDWARE REG.
DEC R2
BNE 10$
MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTS PC
    
```



```

1          .SBTTL  FLOAT 1 AND 0
2
3          ;*****
4          ;*FLOAT A ONE AND A ZERO THRU A DESIGNATED REGISTER
5          ;*ABSOLUTE ADDRESS OF REG. UNDER TEST IS IN R4
6          ;*****
7
8 031614 000000      MASK: 0          ;BITS UNDER TEST
9 031616 000000      LERR: 0         ;ERROR HLT ADDRESS
10 031620 000000     REGADR: 0
11
12 031622 012537 031614  BITST: MOV      (R5)+,MASK      ;FETCH DATA MASK
13 031626 012504          MOV      (R5)+,R4          ;GET ADDRESS OF REG. UNDER TEST
14 031630 010437 031620  MOV      R4,REGADR
15 031634 010537 031616  MOV      R5,LERR ;GET ERROR RETURN ADDR.
16 031640 062705 000004  ADD      #4,R5  ;MODIFY RETURN ADDR. TO JUMP OVER RTS
17 031644 012703 000001  MOV      #1,R3  ;INITIALIZE DATA PATTERN
18 031650 004737 031672  BLT1:  JSR      PC,BLT2 ;OUTPUT FLOATING ZERO
19 031654 004737 031672  JSR      PC,BLT2 ;OUTPUT FLOATING ONE
20 031660 000241          CLC
21 031662 006103          ROL      R3          ;SHIFT PATTERN
22 031664 005703          TST      R3
23 031666 001370          BNE     BLT1      ;BRANCH IF NOT COMPLETE
24 031670 000205          RTS
25 031672 005103          BLT2:  COM      R3          ;COMPLEMENT PATTERN
26 031674 012737 031702 032034  MOV      #BLT3,LAD ;SET SCOPE LOOP
27 031702 010337 001124  BLT3:  MOV      R3,$GDDAT ;STORE GOOD DATA
28 031706 005137 031614          COM      MASK      ;AND MASK WITH PATTERN
29 031712 043737 031614 001124  BIC      MASK,$GDDAT;CLEAR THE REST
30 031720 005137 031614          COM      MASK      ;RESTORE MASK
31 031724 013714 001124  MOV      $GDDAT,(R4) ;OUTPUT TO REGISTER
32 031730 011437 001126  MOV      (R4),$BDDAT ;INPUT FROM REGISTER
33 031734 005137 031614          COM      MASK
34 031740 043737 031614 001126  BIC      MASK,$BDDAT ;AND MASK OUT RECEIVED DATA
35 031746 005137 031614          COM      MASK      ;RESTORE MASK
36 031752 023737 001124 001126  CMP      $GDDAT,$BDDAT;IS DATA CORRECT
37 031760 001403          BEQ     1$          ;BRANCH IF GOOD
38 031762 004777 177630  JSR      PC,@LERR ;GO TO REPORT ERROR
39 031766 104413          SCOP1
40 031770 000207          RTS      PC
    
```

```

1      .SBTTL  CLEAR MEMORY ROUTINE
2
3      ;THIS CLEARS ANY BLOCK OF MEMORY
4      ;FILLING IT WITH ANY DATA
5
6      ;CALL
7
8      JSR    R0,CLAREA
9      .WORD  X           ;STARTING ADDRESS OF BLOCK
10     .WORD  Y           ;DATA TO BE FILLED
11     .WORD  Z
12
13     ;R1 WILL HAVE STARTING ADDRESS OF BLOCK TO BE FILLED
14     ;R2 AFTER SUBTRACTION WILL HAVE TWICE NUMBER OF LOCATIONS
15     ;R3 WILL HAVE DATA TO BE FILLED
16
17     ;TO AVOID DIVIDE ROUTINE TWO DECREMENT R2 WILL BE USED
18
19     CLAREA:
20     MOV    R1,-(SP)    ;:PUSH R1 ON STACK
21     MOV    R2,-(SP)    ;:PUSH R2 ON STACK
22     MOV    R3,-(SP)    ;:PUSH R3 ON STACK
23     MOV    (R0)+,R1    ;:FROM
24     MOV    (R0)+,R2    ;:TO
25     MOV    (R0)+,R3    ;:DATA
26     SUB    R1,R2      ;:NO. OF LOCATIONS MINUS TWO
27     ADD    #2,R2      ;:GET TWICE NO OF LOCATIONS
28     1$:   MOV    R3,(R1)+ ;:MOVE IN DATA
29     DEC    R2
30     DEC    R2
31     BNE    1$         ;:BRANCH IF NOT COMPLETE
32     MOV    (SP)+,R3    ;:POP STACK INTO R3
33     MOV    (SP)+,R2    ;:POP STACK INTO R2
34     MOV    (SP)+,R1    ;:POP STACK INTO R1
35     RTS    R0         ;:RETURN

```

```

18 031772
031772 010146
031774 010246
031776 010346
19 032000 012001
20 032002 012002
21 032004 012003
22 032006 160102
23 032010 062702 000002
24 032014 010321
25 032016 005302
26 032020 005302
27 032022 001374
28 032024 012603
032026 012602
032030 012601
29 032032 000200

```



```

1
2 032034 000000
3
4 032036 032777 001000 147074 T.SCOPI: BIT #SW09,@SWR
5 032044 001402 BEQ 1$
6 032046 013716 032034 MOV LAD,(SP)
7 032052 000002 1$: RTI
8
9 ;EXAMPLE OF THE USE OF THE ABOVE
10 ;THIS WILL LOOP BETWEEN X: AND SCOP1 PROVIDED THERE IS NO 'NEWTST'
11 ;:
12 ;: MOV #X,LAD
13 ;X: --- ---
14 ;: --- ---
15 ;: --- ---
16 ;: SCOP1
17
18 .SBTTL CLEAR DISK ROUTINE
19 032054 013701 001240 CLDISK: MOV RHCS1,R1 ;R1 WILL BE CONTROL AND STATUS1
20 032060 013702 001236 MOV RHCS2,R2 ;R2 WILL BE CONTROL AND STATUS2
21 032064 013703 001262 MOV RHDS1,R3 ;R3 WILL BE DISK STATUS REGISTER1
22 032070 013704 001242 MOV RHER1,R4 ;R4 WILL BE ERROR REGISTER #1
23
24 032074 012712 000040 MOV #CLR,@R2 ;CLEAR ALL REG.
25 032100 013712 001374 MOV UNIT,@R2 ;REINSTATE UNIT NO.
26 032104 005011 CLR @R1 ;CLEAR FUNCTION BITS
27 032106 012777 177777 147142 MOV #-1,@RHAS ;CLEAR ATTENTION BITS
28 032114 000207 RTS PC
    
```

```

1          .SBTTL  CHECK DISK STATUS ROUTINES
2
3          ;*****
4          ;*THIS CHECKS THAT DEVICE AVAILABLE (DVA) AND READY (RDY) IN RHCS1 = 1
5          ;*AND CHECKS THAT DEVICE PRESENT (DPR), DEVICE READY (DRY) IN RHDS1 = 1
6          ;*IT ALSO CHECKS THAT ALL OTHER BITS IN THESE REGISTERS = 0
7          ;*****
8
9 032116 011637 001414 CHECKT: MOV    (SP),PCJSR    ;SAVE PC OF JSR+4
10 032122 162737 000004 001414 SUB     #4,PCJSR    ;GET PC OF JSR
11 032130 004737 031554 JSR     PC,PUTREG  ;SAVE REGISTERS
12 032134 022737 004200 001314 CMP     #DVA!RDY,CS1 ;RHCS1 SHOULD HAVE DEVICE AVAILABLE
13                                     ;AND BE READY
14 032142 001423 BEQ     3$        ;BRANCH IF GOOD TO RHDS1 CHECK
15
16 032144 032737 004000 001314 BIT     #DVA,CS1    ;BAD SO TEST DEVICE AVAILABLE
17 032152 001004 BNE     1$        ;TEST READY IF DVA THERE
18 032154 010137 001122 MOV     R1,$BDADR ;ADDRESS OF BAD REGISTER (RHCS1)
19 032160 104026 EMT     26
20
21 032162 000413 BR      3$        ;AVAILABLE AT START OF TEST
22                                     ;BRANCH TO RHDS1 CHECK
23 032164 032737 000200 001314 1$: BIT   #RDY,CS1    ;TEST READY
24 032172 001003 BNE     2$        ;IF RDY THERE BRANCH
25 032174 010137 001122 MOV     R1,$BDADR ;ADDRESS OF BAD REGISTER (RHCS1)
26 032200 104026 EMT     26
27
28 032202 000403 2$: BR      3$        ;AT THE START OF TEST
29 032204 010137 001122 MOV     R1,$BDADR ;BRANCH TO NEXT COMPARE
30 032210 104026 EMT     26 ;ADDRESS OF BAD REGISTER (RHCS1)
31
32                                     ;THAN DVA AND RDY SET
33                                     ;ALL OTHER BITS SHOULD BE 0
34                                     ;AT START OF TEST
35 032212 013746 001336 3$: MOV   DS1,-(SP)    ;GET RHDS1
36 032216 042716 001100 BIC     #VV!PROG,(SP) ;CLEAR VV AND PROGRAMABLE BIT
37 032222 022726 000600 CMP     #DPR!DRY,(SP)+ ;RHDS1 SHOULD HAVE THESE SET
38 032226 001424 BEQ     8$        ;RETURN TO TEST IF GOOD
39
40 032230 032737 000400 001336 4$: BIT   #DPR,DS1    ;BAD SO TEST DRIVE PRESENT
41 032236 001004 BNE     5$        ;CHECK DRY IF GOOD
42 032240 010337 001122 MOV     R3,$BDADR ;ADDRESS OF BAD REGISTER (RHDS1)
43 032244 104026 EMT     26
44 032246 000413 BR      7$        ;BRANCH OUT
45 032250 032737 000200 001336 5$: BIT   #DRY,DS1    ;TEST DRIVE READY
46 032256 001004 BNE     6$        ;IF DPR WAS THERE SO BRANCH
47 032260 010337 001122 MOV     R3,$BDADR ;ADDRESS OF BAD REGISTER (RHDS1)
48 032264 104026 EMT     26
49 032266 000403 BR      7$        ;BRANCH OUT
50 032270 010337 001122 6$: MOV   R3,$BDADR ;ADDRESS OF BAD REGISTER (RHDS1)
51 032274 104026 EMT     26
52
53                                     ;THAN MOL, DRY, DPR, SET
54 032276 000207 7$: RTS    PC        ;ALL OTHER BITS SHOULD BE 0
55                                     ;RETURN TO TEST AND HALT - FATAL ERROR
56 032300 062716 000006 8$: ADD   #6,(SP)    ;ADJUST STACK PTR TO GET OVER HALT IN TEST
57 032304 000207 RTS     PC        ;RETURN TO TEST AND CONTINUE TESTING
    
```



```

58
59
60
61
62
63
64
65 032306 011637 001414 CHECKE: MOV (SP),PCJSR ;SAVE PC OF JSR+4
66 032312 162737 000004 001414 SUB #4,PCJSR ;GET PC OF JSR
67 032320 004737 031554 JSR PC,PUTREG ;SAVE REGISTERS
68 032324 032737 000200 001314 BIT #RDY,CS1 ;RHCS1 SHOULD HAVE DEVICE AVAILABLE
69 ;AND BE READY
70 032332 001004 BNE 1$ ;BRANCH IF GOOD
71 032334 010137 001122 MOV R1,$BDADR ;FAILING REGISTER
72 032340 104026 EMT 26
73 ;DOES NOT HAVE DVA, RDY
74 032342 000427 BR 4$ ;BRANCH
75
76 032344 032737 004000 001314 1$: BIT #DVA,CS1 ;RHCS1 SHOULD HAVE DEVICE AVAILABLE
77 ;AND BE READY
78 032352 001004 BNE 2$ ;BRANCH IF GOOD
79 032354 010137 001122 MOV R1,$BDADR ;FAILING REGISTER
80 032360 104026 EMT 26
81 ;DOES NOT HAVE DVA, RDY
82 032362 000417 BR 4$ ;BRANCH OUT
83 032364 032737 000200 001336 2$: BIT #DRY,DS1 ;RHDS1 SHOULD HAVE DPR,DRY
84 032372 001004 BNE 3$ ;BRANCH IF THERE
85 032374 010337 001122 MOV R3,$BDADR ;FAILING REGISTER RHDS1
86 032400 104026 EMT 26
87 032402 000407 BR 4$ ;BRANCH OUT
88 032404 032737 000400 001336 3$: BIT #DPR,DS1 ;RHDS1 SHOULD HAVE DPR,DRY
89 032412 001004 BNE 5$ ;BRANCH IF THERE
90 032414 010337 001122 MOV R3,$BDADR ;FAILING REGISTER RHDS1
91 032420 104026 EMT 26
92 032422 000207 4$: RTS PC ;RETURN TO TEST AND HALT - FATAL ERROR
93
94 032424 062716 000006 5$: ADD #6,(SP) ;ADJUST STACK TO GET OVER HALT IN TEST
95 032430 000207 RTS PC ;RETURN TO TEST AND CONTINUE TESTING
96
97 .SBTTL WAIT LOOP
98 ;*****
99 ;*WAIT LOOP
100 ;*ONE LOOP OR ONE COUNT = 5.15 MICROSEC WITH BIPOLAR MEMORY (MIN)
101 ;*ONE LOOP OR ONE COUNT = 11.86 MICROSEC WITH CORE (MIN)
102 ;*WITH CORE ERROR IS INDICATED AFTER ABOUT 650 MILLISEC (MIN)
103 ;*****
104
105 032432 177777 TIMCNT: 177777 ;WAITING COUNT
106
107 032434 010046 WAIT.T: MOV R0,-(SP) ;SAVE R0
108 032436 016600 000002 MOV 2(SP),R0 ;GET ADDRESS OF REG. ADDRESS
109 032442 010037 001204 MOV R0,$TMP3 ;WAT PC+2 IN $TMP3
110 032446 162737 000002 001204 SUB #2,$TMP3 ;WAT PC FOR TYPEOUT
111 032454 012037 001176 MOV (R0)+,$TMP0 ;WAIT REGISTER ADDRESS
112 032460 012037 001200 MOV (R0)+,$TMP1 ;WAIT ON BIT
113 032464 010066 000002 MOV R0,2(SP) ;RESTORE RETURN ON STACK
114 032470 012600 MOV (SP)+,R0 ;RESTORE R0
    
```

```
115 032472 013737 032432 001202      MOV      TIMCNT,$TMP2;TEMPORARY COUNT
116
117 032500 033777 001200 146470 1$:   BIT      $TMP1,@$TMP0      ;IS REQUIRED BIT THERE?
118 032506 001021                BNE      2$                ;BRANCH IF YES
119 032510 005337 001202                DEC      $TMP2              ;COUNT
120 032514 001371                BNE      1$                ;BRANCH IF NOT TIME UP
121 032516 013737 032432 001202      MOV      TIMCNT,$TMP2;TEMPORARY COUNT
122 032524 033777 001200 146444 3$:   BIT      $TMP1,@$TMP0      ;IS REQUIRED BIT THERE?
123 032532 001007                BNE      2$                ;BRANCH IF YES
124 032534 005337 001202                DEC      $TMP2              ;COUNT
125 032540 001371                BNE      3$                ;BRANCH IF NOT TIME UP
126 032542 017737 146430 001126      MOV      @$TMP0,$BDDAT     ;REGISTER CONTENTS
127 032550 104016                EMT      16
128 032552 000002                2$:   RTI
129
130      ;CALL FOR THE ABOVE WAITLOOP IS
131      :
132      :
133      :       MOV      @A,X$      ;A CONTAINS REGISTER ADDRESS
134      :       -       -       -   ;HENCE X$ WILL HAVE ABSOLUTE REG. ADR.
135      :       -       -       -
136      :       WAT
137      :X$:  0       ;ABSOLUTE REG. ADDRESS UNDER WAIT
138      :       .WORD  0       ;BIT WAITED FOR
139      :       ;CONTINUE
```



```

1      .SBTTL  SAVE ROUTINE
2
3      ;*****
4      ;THIS IS A SUBROUTINE TO READ & SAVE REGISTERS
5      ;IN THE REGISTER TABLE TO ANY LOCATION
6      ;THE CALL IS
7      ;JSR   RO,SAVER
8      ;FROM
9      ;TO
10     ;NUMBER OF WORDS SAVED
11
12     032554  SAVER:
13     032554  010146      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
14     032556  010246      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
15     032560  010346      MOV     R3,-(SP)      ;;PUSH R3 ON STACK
16     032562  012001      MOV     (R0)+,R1      ;FROM
17     032564  012002      MOV     (R0)+,R2      ;TO
18     032566  012003      MOV     (R0)+,R3      ;NUMBER
19     032570  013122      1$:   MOV     @ (R1)+,(R2)+ ;SAVE REGISTER CONTENTS
20     032572  005303      DEC     R3            ;COUNT
21     032574  001375      BNE    1$            ;BRANCH IF NOT DONE
22     032576  012603      MOV     (SP)+,R3      ;;POP STACK INTO R3
23     032600  012602      MOV     (SP)+,R2      ;;POP STACK INTO R2
24     032602  012601      MOV     (SP)+,R1      ;;POP STACK INTO R1
25     032604  000200      RTS     R0
26
27     .SBTTL  WRITE CHECK ROUTINE
28
29     ;*****
30     ;THIS IS A SUBROUTINE TO DO WRITE CHECK HEADER AND DATA
31     ;CYLINDER 0, TRACK 1, SECTOR 1, KEYS 0
32     ;*****
33     ;THESE ARE TO SET UP FOR DISKLESS USE ONLY
34
35     032606  012737  010000  036710  WRCHHD: MOV     #FMT22,CYL      ;CYLINDER 0 FORMAT 16 BIT WORDS
36     032614  112737  000001  036713      MOVB    #1,SECOTR+1     ;TRACK=1
37     032622  112737  000001  036712      MOVB    #1,SECOTR      ;SECTOR=1
38     032630  005037  036714      CLR     KEY1           ;KEY1=0
39     032634  005037  036716      CLR     KEY2           ;KEY2=0
40     032640  012737  000044  036770      MOV     #36.,DAWORD    ;NO OF DATA WORDS
41     032646  005037  036720      CLR     X              ;THIS IS A READ OPERATION
42     032652  004537  033270      JSR     R5,CRC        ;GO TO CALCULATE CRC
43     032656  036710      CYL
44     032660  040610      WCRC
45
46     ;THESE ARE REGULAR SETUPS
47
48     032662  004737  032054      JSR     PC,CLDISK     ;SET UP GENERAL REGISTERS
49     ;AND CLEAR DISK REGISTERS
50     032666  012777  177730  146336      MOV     #-40.,@RHWC    ;36 DATA WORDS 4 HEADER WORDS
51     032674  012777  002544  146332      MOV     #REINTO,@RHBA ;STARTING ADDRESS OF READ BUFFER
52     032702  112746  000001      MOVB    #1,-(SP)      ;SECTOR=1
53     032706  112766  000001  000001      MOVB    #1,1(SP)      ;TRACK=1 IN UPPER BYTE
54     032714  012677  146324      MOV     (SP)+,@RHDST  ;TRACK=1, SECTOR=1 IN RHDST
55     032720  012777  014000  146322      MOV     #FMT22!ECI,@RHOF ;16 BIT WORDS
56     ;ECC CORRECTION INHIBIT BECAUSE
    
```



```

1      .SBTTL COMPARE ROUTINE
2
3      :THIS IS A SUBROUTINE TO COMPARE TWO BLOCKS IN MEMORY
4      :
5      : R1 HAS GOOD DATA BUFFER ADDRESS
6      : R2 HAS TEST DATA BUFFER ADDRESS
7      : $TMP0 HAS ADDRESS OF RETURN ON ERROR TO PRINT HEADER
8      : $TMP1 HAS ADDRESS OF RETURN ON ERROR TO PRINT DATA
9      : R3 HAS NUMBER OF WORDS TO BE COMPARED
10     : R4 HAS ONE MORE THAN NUMBER OF WORDS TO BE COMPARED
11
11 032756 COMPAR:
12 032756 010146 MOV R1,-(SP)      ;;PUSH R1 ON STACK
13 032760 010246 MOV R2,-(SP)      ;;PUSH R2 ON STACK
14 032762 010346 MOV R3,-(SP)      ;;PUSH R3 ON STACK
15 032764 010446 MOV R4,-(SP)      ;;PUSH R4 ON STACK
16 032766 010546 MOV R5,-(SP)      ;;PUSH R5 ON STACK
17 032770 012001 MOV (R0)+,R1      :ADDRESS OF GOOD DATA BUFFER
18 032772 012002 MOV (R0)+,R2      :ADDRESS OF TEST DATA BUFFER
19 032774 012003 MOV (R0)+,R3      :NO OF WORDS TO BE COMPARED
20 032776 012037 001176 MOV (R0)+,$TMP0  :RETURN ON ERROR TO PRINT HEADER
21 033002 012037 001200 MOV (R0)+,$TMP1  :RETURN ON ERROR TO PRINT DATA
22 033006 011000 MOV (R0),R0      :RETURN ON NO ERROR
23 033010 010304 MOV R3,R4        :NO OF WORDS TO BE COMPARED
24 033012 005204 INC R4
25 033014 010437 037030 1$: MOV R4,ERWORD    :FOR ERROR WORD NO
26 033020 022122 CMP (R1)+,(R2)+  :COMPARE GOOD WITH TEST DATA
27 033022 001426 BEQ 3$      :BRANCH IF GOOD
28
29 033024 014137 001124 MOV -(R1),$GDDAT :GOOD DATA
30 033030 014237 001126 MOV -(R2),$BDDAT :BAD DATA
31 033034 160337 037030 SUB R3,ERWORD    :ERROR WORD NO.
32 033040 005737 001406 TST ERFLG$ ;ANY ERRORS ALREAY THERE
33 033044 001003 BNE 2$          :BRANCH IF YES
34 033046 004777 146124 JSR PC,@$TMP0    :RETURN TO PRINT HEADER
35 033052 000402 BR 5$           :BRANCH TO AVOID PRINTING NEXT ERROR
36 033054 004777 146120 2$: JSR PC,@$TMP1    :RETURN TO PRINT DATA
37 033060 022122 5$: CMP (R1)+,(R2)+  :UNDO -(R1) AND -(R2) FOR ERRORS
38 033062 017746 146052 MOV @SWR,-(SP)   :GET SWITCH SETTING
39 033066 042716 177177 BIC #^C600,(SP) :KEEP ONLY SWITCH 7 AND 8
40 033072 022726 000200 CMP #SW07,(SP)+ :IS 7 SET AND 8 RESET
41 033076 001402 BEQ 4$          :BRANCH OUT IF YES
42 033100 005303 3$: DEC R3              :COUNT
43 033102 001344 BNE 1$          :BRANCH IF ALL NOT DEVICE
44 033104 4$:
45 033104 012605 MOV (SP)+,R5     ;;POP STACK INTO R5
46 033106 012604 MOV (SP)+,R4     ;;POP STACK INTO R4
47 033110 012603 MOV (SP)+,R3     ;;POP STACK INTO R3
48 033112 012602 MOV (SP)+,R2     ;;POP STACK INTO R2
49 033114 012601 MOV (SP)+,R1     ;;POP STACK INTO R1
50 033116 000200 RTS R0      :RETURN TO MAIN PROGRAM

```

```

1      .SBTTL  WRITE CHECK DATA
2
3      ::*****
4      :THIS IS A SUBROUTINE TO DO WRITE CHECK DATA
5      :CYLINDER 0, TRACK 1, SECTOR 1, KEYS 0
6      ::*****
7
8      ;THESE ARE TO SET UP FOR DISKLESS USE ONLY
9
10     033120 012737 010000 036710 WRCHDA: MOV    #FMT22,CYL    ;CYLINDER 0 FORMAT 16 BIT WORDS
11     033126 112737 000001 036713      MOVB   #1,SECOTR+1  ;TRACK=1
12     033134 112737 000001 036712      MOVB   #1,SECOTR    ;SECTOR=1
13     033142 005037 036714      CLR    KEY1         ;KEY1=0
14     033146 005037 036716      CLR    KEY2         ;KEY2=0
15     033152 012737 000040 036770      MOV    #32.,DAWORD  ;NO OF DATA WORDS
16     033160 005037 036720      CLR    X            ;THIS IS A READ OPERATION
17
18     033164 004537 033270      JSR    R5,CRC      ;GO TO CALCULATE CRC
19     033170 036710      CYL
20     033172 040610      WCRC
21
22     ;THESE ARE REGULAR SETUPS
23
24     033174 004737 032054      JSR    PC,CLDISK   ;SET UP GENERAL REGISTERS
25                                     ;AND CLEAR DISK REGISTERS
26
27     033200 012777 177740 146024      MOV    #-32.,@RHWC  ;36 DATA WORDS 4 HEADER WORDS
28     033206 012777 002544 146020      MOV    #REINT0,@RHBA ;STARTING ADDRESS OF READ BUFFER
29     033214 112746 000001          MOVB   #1,-(SP)     ;SECTOR=1
30     033220 112766 000001 000001      MOVB   #1,1(SP)    ;TRACK=1 IN UPPER BYTE
31     033226 012677 146012          MOV    (SP)+,@RHDST ;TRACK=1, SECTOR=1 IN RHDST
32     033232 012777 014000 146010      MOV    #FMT22!ECI,@RHOF ;16 BIT WORDS
33                                     ;ECC CORRECTION INHIBIT BECAUSE
34                                     ;ECC LOGIC IS NOT CHECKED YET
35     033240 005077 146006          CLR    @RHCA       ;CYLINDER=0
36     033244 004737 032116          JSR    PC,CHECKT   ;CHECK THAT DVA,RDY,DPR,DRY = 1
37     033250 104401 053407          TYPE   ,CPHALT    ;AND THAT NO OTHERS = 1. CANNOT CON-
38                                     ;TINUE TESTING IF BOTH AREN'T TRUE
39     033254 000000          HALT
40     033256 013711 001446          MOV    WRCHK,@R1  ;WRITE CHECK DATA=50 INTO RHCS1
41     033262 004737 036550          JSR    PC,COMHD   ;WRITE CHECK HEADER AND DATA
42                                     ;SAME AS READ HEADER AND DATA
43
44     033266 000207          RTS    PC         ;RETURN TO WRITE CHECK TEST

```



```

1      .SBTTL  CRC GENERATION ROUTINE
2
3      :THIS IS A SUBROUTINE TO CALCULATE CRC FOR THE FOUR
4      :HEADER WORDS AND STORE THEM IN 'WCRC' AND 'GCRC'
5      :
6      :R1 - REGISTER FOR CRC, INCREMENTED CRC VALUE IS HERE
7      :R2 - THIS HAS BIT POSITION 2 VALUE C
8      :R3 - THIS HAS BIT POSITION 16 I.E. OUTPUT BIT VALUE B
9      :R4 - THIS HAS BIT POSITION 15 VALUE E
10     :
11     :$TMP0 - NUMBER OF WORDS
12     :$TMP2 - NUMBER OF BITS PER WORD = 16
13     :$TMP3 - TEMPORARY REG.
14     :$TMP4 - TEMPORARY REG TO TRANSFER CARRY
15     :$TMP5 - THIS HAS DATA BIT VALUE D
16
17     :FETCH DATA BIT D
18     :B = D XOR 16
19     :C = B XOR 2
20     :E = B XOR 15
21     :ROTATE RIGHT ONE POSITION
22     :B GOES TO POSITION 1
23     :C GOES TO POSITION 3
24     :E GOES TO POSITION 16
25     :REPEAT 64 TIMES
26
27     :CALL
28     :
29     :      JSR      R5,CRC
30     :      .WORD   X      :FIRST LOCATION AT
31     :      .WORD   Y      :PUT CRC IN WCRC FOR READ GCRC FOR WRITE
32
33     CRC:
34     MOV      R0,-(SP)      ;;PUSH R0 ON STACK
35     MOV      (R5)+,R0     :GET POINTER TO CYL NO.
36     MOV      R1,-(SP)     ;;PUSH R1 ON STACK
37     MOV      R2,-(SP)     ;;PUSH R2 ON STACK
38     MOV      R3,-(SP)     ;;PUSH R3 ON STACK
39     MOV      R4,-(SP)     ;;PUSH R4 ON STACK
40     CLR      R1           :CLEAR WORKING LOCATION
41     CLR      $TMP5
42     MOV      #4,$TMP0     :WORD COUNT
43     MOV      (R0)+,$TMP3  :TEMPORARY WORD STORAGE
44     MOV      #16,$TMP2   :BIT COUNT
45     MOV      $TMP3,$TMP4 :TEMPORARY WORD STORAGE
46     ROR      $TMP3       :GET LSB INTO 'C'
47     ROR      $TMP5       :GET ABOVE 'C' INTO $TMP5
48     BIT      #BIT0,R1    :IS POSITION 15 HIGH
49     BEQ      1$         :BRANCH IF POSITION 16 LOW
50     MOV      #BIT15,R3   :GET POSITION 16
51     BR      2$
52     CLR      R3         :GET POSITION 16
53     ADD     $TMP5,R3     :XOR POSITION 16 WITH D
54     :TO GIVE B
55     BIT      #BIT14,R1   :IS POSITION 2 HIGH
56     BEQ      3$         :BRANCH IF POSITION 2 LOW
57     MOV      #BIT15,R2   :GET POSITION 2
58     BR      4$
59     CLR      R2         :GET POSITION 2
60     ADD     R3,R2       :XOR B WITH POSITION 2

```

```

30 033270
31 033270 010046
32 033272 012500
33 033274 010146
34 033276 010246
35 033300 010346
36 033302 010446
37 033304 005001
38 033306 005037 001210
39 033312 012737 000004 001176
40 033320 012037 001204
41 033324 012737 000020 001202
42 033332 013737 001204 001206
43 033340 006037 001204
44 033344 006037 001210
45 033350 032701 000001
46 033354 001403
47 033356 012703 100000
48 033362 000401
49 033364 005003
50 033366 063703 001210
51 033372 032701 040000
52 033376 001403
53 033400 012702 100000
54 033404 000401
55 033406 005002
56 033410 060302

```



```

1      .SBTTL  SIMULATED DISK SETUP
2
3      :THIS IS A SUBROUTINE TO SET UP THE SIMULATOR DISK FOR
4      :CYLINDER 0 (16 BITS PER WORD)
5      :TRACK 1, SECTOR 1
6      :KEY1 1
7      :KEY2 1
8      :CRC THROUGH THE JSR R5,CRC
9      :256 WORDS OF 177400
10
11     ;CALL
12     ;      JSR      PC,SETDSK
13
14     SETDSK:
15     033544      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
16     033544      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
17     033546      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
18     033550      MOV      #177400,R0    ;;DATA IN THE DISK
19     033552      MOV      #256.,R1     ;;COUNTER
20     033556      MOV      #DISK,R2     ;;START OF SIMULATOR DISK
21     033562      1$:      MOV      R0,(R2)+  ;;MOVE IN DATA
22     033566      DEC      R1           ;;COUNT FOR 256
23     033570      BNE     1$           ;;BRANCH IF 256 NOT COMPLETE
24     033574      MOV      #17.,R1     ;;2 ECC WORDS, 1 DATA GAP
25     033600      2$:      CLR      (R2)+  ;;14 TOLERANCE GAP
26     033602      DEC      R1           ;;CLEAR ECC,DATA GAP AND
27     033604      BNE     2$           ;;TOLERANCE GAP
28     033606      ;;COUNT
29     033608      ;;BRANCH IF NOT COMPLETE
30     ;NOW SET UP FOR DISKLESS USE
31     033606      MOV      #FMT22,CYL    ;;CYLINDER 0 (16 BIT WORDS)
32     033614      MOV      #1,SECOTR+1  ;;TRACK=1
33     033622      MOV      #1,SECOTR    ;;SECTOR=1
34     033630      MOV      #1,KEY1 ;KEY1=1
35     033636      MOV      #1,KEY2 ;KEY2=1
36     033644      MOV      256.,DAWORD  ;;NO. OF DATA WORDS
37     033652      JSR      R5,CRC      ;;GO TO CALCULATE CRC
38     033656      CYL      ;;FIRST CRC WORD
39     033660      WCRC     ;;PUT CALCULATED CRC
40     033662      MOV      (SP)+,R2    ;;POP STACK INTO R2
41     033664      MOV      (SP)+,R1    ;;POP STACK INTO R1
42     033666      MOV      (SP)+,R0    ;;POP STACK INTO R0
43     033670      RTS      PC

```



```

108 034146 012737 000200 001124      MOV    #HCE,$GDDAT    ;GOOD DATA
109 034154 104027                    EMT    27
110                                ;HEADER ONLY HCE SHOULD
111 034156 000460                    BR     7$             ;BE SET
112 034160                                11$:
113 034160 017737 145056 001126      MOV    @RHER1,$BDDAT  ;TEST DATA
114 034166 022737 100200 001126      CMP    #DCK!HCE,$BDDAT ;ONLY HEADER COMPARE BIT?
115                                ;SHOULD BE SET
116                                ;DCK IS SET BECAUSE ECC IS NOT READ
117 034174 001451                    BEQ    7$             ;BRANCH IF GOOD
118 034176 013737 001242 031620      MOV    RHER1,REGADR  ;REGISTER ADDRESS RHER1
119 034204 012737 100200 001124      MOV    #DCK!HCE,$GDDAT ;GOOD DATA
120 034212 104027                    EMT    27
121                                ;HEADER ONLY HCE SHOULD
122 034214 000441                    BR     7$             ;BE SET
123 034216 022737 000072 001210 6$:   CMP    #72,$TMP5     ;IS THIS A READ COMMAND?
124 034224 001417                    BEQ    12$           ;BRANCH IF A READ
125 034226 017737 145010 001126      MOV    @RHER1,$BDDAT  ;TEST DATA
126 034234 022737 000400 001126      CMP    #HCRC,$BDDAT  ;ONLY CRC ERROR SHOULD BE THERE
127 034242 001426                    BEQ    7$
128 034244 013737 001242 031620      MOV    RHER1,REGADR  ;REG. ADDR = RHER1
129 034252 012737 000400 001124      MOV    #HCRC,$GDDAT  ;GOOD DATA
130 034260 104027                    EMT    27
131                                ;SHOULD BE SET
132 034262 000416                    BR     7$             ;BRANCH OUT
133 034264 017737 144752 001126 12$:  MOV    @RHER1,$BDDAT  ;TEST DATA
134
135 034272 022737 100400 001126      CMP    #DCK!HCRC,$BDDAT ;HCRC AND DCK SHOULD BE SET
136                                ;DCK IS SET BECAUSE ECC IS NOT READ
137 034300 001407                    BEQ    7$             ;BRANCH IF GOOD
138 034302 012737 100400 001124      MOV    #DCK!HCRC,$GDDAT ;GOOD DATA
139 034310 013737 001242 031620      MOV    RHER1,REGADR  ;FAILING REGISTER RHER1
140 034316 104027                    EMT    27
141                                ;DCK AND HCRC SHOULD BE SET
142                                ;DCK IS SET BECAUSE ECC IS NOT READ
143 034320 000200                    7$:   RTS    R0          ;RETURN TO MAIN TEST
    
```



```

1
2
3
4
5
6
7
8
9
10 034322          .SBTTL  EXIT WRT HEADER & DATA ROUTINE
    034322 010046   ;THIS IS A SUBROUTINE TO LEAVE AT THE MIDDLE OF
    034324 010146   ;A WRITE HEADER AND DATA COMMAND
11 034326 013777 001454 144704 ;IT TRYS TO GET SECTOR 10, TRACK 0, CYLINDER 0
12                                     ;BUT COMES OUT AFTER ONE SECTOR
13                                     ;THE COMMAND OS JSR PC,MIDDLE
14                                     ;BAI IS SET
    034334 012777 177766 144670   MIDDLE:
    034342 012777 001500 144664   MOV     R0,-(SP)      ;;PUSH R0 ON STACK
    034350 012777 000010 144666   MOV     R1,-(SP)      ;;PUSH R1 ON STACK
    034356 052777 000010 144652   MOV     WRIFOR,@RHCS1 ;WRITE HEADER AND DATA=62
    034364 012777 010000 144656   MOV     #-10,@RHWC   ;IN RHCS1
    034372 005077 144654          MOV     #WRFROM,@RHBA ;10 WORDS
    034376 012737 000001 034424   MOV     #10,@RHDST  ;BUS ADDRESS=WRFRCM
    034404 012777 000001 144646   BIS     #BAI,@RHCS2  ;DESIRED TRACK=0 SECTOR=10
    034412 052777 000001 144620   MOV     #FMT22,@RHOF ;BUS ADDRESS INCREMENT INHIBIT
    034420 004137 042756          CLR     @RHCA        ;FORMAT 16 BIT WORDS
    034424 000000          MOV     #1,MID ;SECTOR ;CYLINDER=0
    034426 012601          ;IS SET TO 1 SO THAT
    034430 012600          ;WE CAN GET OUT AT THE
    034432 000207          ;MIDDLE OF AN OPERATION
    034404 012777 000001 144646   MOV     #DMD,@RHMR   ;LOOKING FOR SECTOR 10
    034412 052777 000001 144620   BIS     #GO,@RHCS1  ;SET DIAGNOSTIC MODE
    034420 004137 042756          JSR     R1,SEARCH    ;GO TO RHCS1 WITH 62
    034424 000000          MID:  .WORD 0      ;SECTOR
    034426 012601          MOV     (SP)+,R1    ;;POP STACK INTO R1
    034430 012600          MOV     (SP)+,R0    ;;POP STACK INTO R0
    034432 000207          RTS     PC
  
```

```

1      .SBTTL  JAM CURRENT CYLINDER ROUTINE
2
3      :THIS SUBROUTINE WILL CHANGE THE CURRENT CYLINDER REGISTER
4      :THIS IS DONE BY GIVING A SEEK COMMAND THEN AN INIT
5      :WHICH WILL LOAD THE CURRENT CYLINDER WITH THE DESIRED CYLINDER VALUE
6
7      :CALL
8
9      :      JSR      R0,MAKECYL
10     :      .WORD   XC           ;DESIRED VALUE OF CURRENT CYLINDER
11
12     MAKECYL:
13     MOV      R5,-(SP)           ;;PUSH R5 ON STACK
14     MOV      R0,PCJSR          ;PC OF JSR+4
15     SUB      #4,PCJSR          ;SAVE PC OF JSR
16     MOV      (R0)+,R5          ;GETTING READY TO FILL DESIRED CYLINDER
17     MOV      R5,@RHCA          ;FILL DESIRED CYLINDER REGISTER
18     CLR      @RHDST            ;MAKE SURE DESIRED SECTOR TRACK IS NOT ILLEGAL
19     MOV      SEECOM,@RHCS1     ;FILL SEEK COMMAND
20     MOV      #DMD,@RHMR        ;SET DIAGNOSTIC MODE
21     BIS      #GO,@RHCS1        ;GO TO SEEK
22     NOP
23     NOP                         ;ALLOW TIME FOR SEEK TO HANG UP
24     NOP                         ;ALLOW TIME FOR SEEK TO HANG UP
25     NOP                         ;ALLOW TIME FOR SEEK TO HANG UP
26     NOP                         ;ALLOW TIME FOR SEEK TO HANG UP
27     JSR      PC,CLDISK          ;GIVE INIT
28     MOV      @RHCC,$BDDAT       ;TEST DATA
29     CMP      R5,$BDDAT          ;COMPARE CURRENT CYLINDER
30     BEQ      1$                 ;BRANCH IF GOOD
31     MOV      R5,$GDDAT          ;GOOD VALUE OF RHCC
32     MOV      RHCC,REGADR        ;FAILING REGISTER ADDRESS
33     EMT      30
34
35     ;REGISTER AFTER A SEEK AND AN INIT
36
37     1$:
38     MOV      (SP)+,R5           ;;POP STACK INTO R5
39     RTS      R0
    
```



```

1      .SBTTL  ECC GENERATION AND COMPARISON ROUTINE
2
3      ;THIS SUBROUTINE GENERATES AND TESTS ECC
4      ;CALL
5      ;      JSR      PC,ECTEST
6
7      100000  PIE1   =100000
8      040000  PIE2   =40000
9      020000  PIE3   =20000
10     010000  PIE4   =10000
11     004000  PIE5   =4000
12     002000  PIE6   =2000
13     001000  PIE7   =1000
14     000400  PIE8   =400
15     000200  PIE9   =200
16     000100  PIE10  =100
17     000040  PIE11  =40
18     000020  PIE12  =20
19     000010  PIE13  =10
20     000004  PIE14  =4
21     000002  PIE15  =2
22     000001  PIE16  =1
23     100000  PIE17  =100000
24     040000  PIE18  =40000
25     020000  PIE19  =20000
26     010000  PIE20  =10000
27     004000  PIE21  =4000
28     002000  PIE22  =2000
29     001000  PIE23  =1000
30     000400  PIE24  =400
31     000200  PIE25  =200
32     000100  PIE26  =100
33     000040  PIE27  =40
34     000020  PIE28  =20
35     000010  PIE29  =10
36     000004  PIE30  =4
37     000002  PIE31  =2
38     000001  PIE32  =1
39
40 034554 000000  ECDATA: 0      ;DATA BIT FOR ECC
41                                     ;IF ALL ONES THEN CURRENT BIT IS A ONE
42                                     ;IF ZERO THEN CURRENT BIT IS A ZERO
43
44 034556 000000  GECC1: 0      ;LOW ORDER ECC WORD TO BE GENERATED HERE
45                                     ;=R1
46
47 034560 000000  GECC2: 0      ;HIGH ORDER ECC WORD TO BE GENERATED HERE
48                                     ;=R2
49
50 034562 000000  TSECCG: 0     ;IF =177777 GENERATE AND TEST ECC FOR THIS BIT
51                                     ;IF =0 DO NOT GENERATE AND TEST ECC FOR THIS BIT
52
53 034564 113713  NCODE: 38859.   ;N-CODE WORD
54 034566 000000  NCOUNT: 0      ;TEMPORARY N CODE
55 034570 000000  POSITI: 0      ;POSITION REGISTER
56 034572 010041  HARDER: 4129.   ;HARD ERROR COUNT
57                                     ;TRUE COUNT IS 4128 BUT AS COMPARES ARE
  
```

```

58                                     ;DONE ONE STAGE LATER SO 4129
59 034574 000000      DATENV: 0        ;DATA ENVELOPE FOR TYPE OUT
60                                     ;MAX FOR WRITE IS 4096
61                                     ;MAX FOR READ IS 4128
62 034576 000000      ZCODE: 0         ;LEADING ZEROS ENVELOPE FOR TYPE OUT
63                                     ;THIS IS SHUT OFF WHEN POSITION COUNTER
64                                     ;IN ENABLED
65                                     ;MAX COUNT IS 38859
66
67
68
69 034600 000000      HADTMP: 0         ;TEMPORARY HARD ERROR COUNT
70 034602 000000      P3: 0
71 034604 000000      P12: 0
72 034606 000000      P22: 0
73 034610 000000      P24: 0
74
75
76
77
78
79 034612                ECTEST:
   034612 010046          MOV     R0,-(SP)      ;;PUSH R0 ON STACK
   034614 010146          MOV     R1,-(SP)      ;;PUSH R1 ON STACK
   034616 010246          MOV     R2,-(SP)      ;;PUSH R2 ON STACK
   034620 010346          MOV     R3,-(SP)      ;;PUSH R3 ON STACK
   034622 010446          MOV     R4,-(SP)      ;;PUSH R4 ON STACK
   034624 010546          MOV     R5,-(SP)      ;;PUSH R5 ON STACK
80 034626 013701 034556  MOV     GECC1,R1      ;ECC1 WORD
81 034632 013702 034560  MOV     GECC2,R2      ;ECC2 WORD
82 034636 005737 034554  TST     ECDATA      ;IS CURRENT BIT A ONE
83 034642 001406          BEQ     2$          ;BRANCH IF CURRENT DATA D=0
84
85                                     ;IF CARRY IS NOT ZERO THEN D=1
86                                     ;INVERT X32 TO GIVE R0
87
88 034644 010103          1$:  MOV     R1,R3
89 034646 052703 177776  BIS     #^CPIE32,R3
90 034652 005103          COM     R3
91 034654 010300          MOV     R3,R0
92 034656 000404          BR      3$
93
94                                     ;IF CARRY IS ZERO THEN D=0, X32 BECOMES R0
95
96 034660 010103          2$:  MOV     R1,R3
97 034662 042703 177776  BIC     #^CPIE32,R3
98 034666 010300          MOV     R3,R0
99
100 034670 000241          3$:  CLC
101 034672 006000          ROR     R0
102 034674 006000          ROR     R0
103 034676 005700          TST     R0
104 034700 001462          BEQ     10$          ;BRANCH IF R0=0
105
106                                     ;INVERT X2
107
108 034702 010203          MOV     R2,R3
    
```



```

109 034704 052703 137777      BIS      #^CPIE2,R3
110 034710 005103              COM      R3
111 034712 010337 034602      MOV      R3,P3
112 034716 006237 034602      ASR      P3
113
114          ;INVERT X11
115
116
117 034722 010203              MOV      R2,R3
118 034724 052703 177737      BIS      #^CPIE11,R3
119 034730 005103              COM      R3
120 034732 010337 034604      MOV      R3,P12
121 034736 006237 034604      ASR      P12
122
123          ;INVERT X21
124
125 034742 010103              MOV      R1,R3
126 034744 052703 173777      BIS      #^CPIE21,R3
127 034750 005103              COM      R3
128 034752 010337 034606      MOV      R3,P22
129 034756 006237 034606      ASR      P22
130
131          ;INVERT X23
132
133 034762 010103              MOV      R1,R3
134 034764 052703 176777      BIS      #^CPIE23,R3
135 034770 005103              COM      R3
136 034772 010337 034610      MOV      R3,P24
137 034776 006237 034610      ASR      P24
138
139          ;NOW THAT R0 FOR POSITION 1
140          ;          P3 FOR POSITION 3
141          ;          P12 FOR POSITION 12
142          ;          P22 FOR POSITION 22
143          ;          P24 FOR POSITION 24
144          ;ARE KNOWN THE ROTATE WILL BE DONE AND
145          ;THESE BITS JAMED IN
146
147 035002 006002              ROR      R2
148 035004 006001              ROR      R1
149 035006 053700 034602      BIS      P3,R0
150 035012 053700 034604      BIS      P12,R0
151 035016 042702 120020      BIC      #PIE1!PIE3!PIE12,R2
152 035022 050002              BIS      R0,R2
153
154 035024 005000              CLR      R0
155 035026 053700 034606      BIS      P22,R0
156 035032 053700 034610      BIS      P24,R0
157 035036 042701 002400      BIC      #PIE22!PIE24,R1
158 035042 050001              BIS      R0,R1
159 035044 000404              BR      12$
160
161          ;THE PROGRAM COMES HERE IF R0=0
162          ;SO AFTER ROTATE R0 GETS PUT INTO POSITION 1
163
164 035046 006002              10$:    ROR      R2
165 035050 006001              ROR      R1
    
```

```

166 035052 042702 100000      BIC    #PIE1,R2
167 035056 010137 034556      12$:  MOV    R1,GECC1          :SAVE ECC1
168 035062 010237 034560      MOV    R2,GECC2          :SAVE ECC2
169 035066 005737 034562      TST    TSECCG ;IS HARDWARE TO BE CHECKED
170                                     :IF =1777777 TEST HARDWARE
171                                     :IF = 0 DO NOT TEST HARDWARE
172 035072 001432      BEQ    14$                :BRANCH IF HARDWARE NOT TO BE CHECKED
173
174
175                                     ;*CHECK HARDWARE
176 035074 032777 000400 144036  BIT    #SW8,@SWR          :IS SWITCH 8 SET
177 035102 001005      BNE    15$                :BRANCH IF SW8 IS SET
178 035104 032777 000100 144026  BIT    #SW6,@SWR          :IS SWITCH 6 SET
179 035112 001401      BEQ    15$                :BRANCH IF SW6 IS NOT SET
180 035114 000421      BR     14$                :IF SWITCH 8 IS NOT SET AND
181                                     :SWITCH 6 IS SET THEN
182                                     :DO NOT DO COMPARES
183 035116 010146      15$:  MOV    R1,-(SP)          :GOOD PATTERN REGISTER
184 035120 042716 174000      BIC    #174000,(SP)      :GET ONLY PATTERN BITS
185 035124 022677 144142      CMP    (SP)+,@RHEC2      :COMPARE PATTERN REGISTER
186 035130 001404      BEQ    13$                :BRANCH IF GOOD
187
188                                     ;TO SAVE TIME
189 035132 004737 031554      JSR    PC,PUTREG          :SAVE REGISTERS
190 035136 104035      EMT
191 035140 000407      BR     14$                :BRANCH OUT
192 035142 023777 034570 144120  13$:  CMP    POSITI,@RHEC1    :COMPARE POSITION REGISTER
193 035150 001403      BEQ    14$                :BRANCH IF GOOD
194
195                                     ;TO SAVE TIME
196 035152 004737 031554      JSR    PC,PUTREG          :SAVE REGISTERS
197 035156 104035      EMT    35
198                                     :'DATA ENVLOP'' GIVES NUMBER OF CLOCK
199                                     :PULSES FROM BEGINING OF COMMAND
200                                     :THAT IS THE CLOCKS IN THE R/W DATA FIELD ENVELOPE
201                                     :
202                                     :IN A WRITE THERE ARE 10000 OCTAL CLOCKS
203                                     :IN A READ THERE ARE 10040 OCTAL CLOCKS
204                                     :
205                                     :
206                                     :'N-CODE ZEROS'' GIVE THE NUMBER OF CLOCKS
207                                     :GIVEN FOR THE LEADING ZEROS FIELD
208                                     :MAX COUNT IS 113713 OCTAL
209                                     :
210                                     :'GOOD POSITION'' GIVES NUMBER OF CLOCKS
211                                     :GIVEN AFTER LEADING ZEROS WHICH IS FOR THE DATA
212                                     :FIELD
213                                     :MAX COUNT IS 10040 OR 10041 OCTAL
214
215
216 035160      14$:  MOV    (SP)+,R5          ::POP STACK INTO R5
      035160 012605      MOV    (SP)+,R4          ::POP STACK INTO R4
      035162 012604      MOV    (SP)+,R3          ::POP STACK INTO R3
      035164 012603      MOV    (SP)+,R2          ::POP STACK INTO R2
      035166 012602      MOV    (SP)+,R1          ::POP STACK INTO R1
      035170 012601      MOV    (SP)+,R0          ::POP STACK INTO R0
      035172 012600
    
```



```

217 035174 000207          RTS      PC
218
219          .SBTTL  ECC GENERATION CONTROL ROUTINE
220
221          ;THIS SUBROUTINE WILL CONTROL THE ECC GENERATION ROUTINE
222          ;FOR ERROR CORRECTION PROCESS
223          ;CALL
224          ;
225          ;      JSR      PC,ECORR          ;EXPECTED POSITION REGISTER WHEN CORRECTION IS COMPLETE
226          ;      .WORD  XP
227 035176 000000          ERPOS:  0          ;POSITION REG. WHEN CORRECTION IS COMPLETE
228
229
230
231 035200 010037 001414          ECORR:  MOV      R0,PCJSR          ;SAVE PC OF JSR + 4
232 035204 162737 000004 001414  SUB      #4,PCJSR          ;SAVE PC OF JSR
233 035212 012037 035176          MOV      (R0)+,ERPOS      ;GET POSITION REG. WHEN CORRECTION IS COMPLETE
234 035216 010146          MOV      R1,-(SP)        ;PUSH R1 ON STACK
235 035220 013701 001260          MOV      RHMR,R1 ;MAINTENANCE REGISTER
236 035224 012711 000001          MOV      #DMD,@R1      ;SET DIAGNOSTIC MODE BIT
237 035230 005037 034554          CLR      ECDATA ;ECC DATA IS ZERO
238
239
240
241 035234 005737 034570          1$:    TST      POSITI ;IS SOFTWARE POSITION NON ZERO
242 035240 001007          BNE      2$          ;BRANCH IF N-CODE S COMPLETE
243 035242 005337 034566          DEC      NCOUNT ;DECREMENT N-CODE
244 035246 001001          BNE      6$          ;BRANCH IF N-CODE IS NOT COMPLETE
245 035250 000403          BR       2$          ;BRANCH AS N-CODE IS COMPLETE
246 035252 005237 034576          6$:    INC      ZCODE      ;INCREMENT CLOCKS GIVEN FOR LEADING ZEROS
247 035256 000420          BR       3$          ;BRANCH AS N-CODE IS NOT COMPLETE
248          ;GO TO GIVE CLOCK AND TEST ECC
249 035260 005237 034570          2$:    INC      POSITI ;INCREMENT SOFTWARE POSITION
250 035264 023737 035176 034570  CMP      ERPOS,POSITI;HAVE ENOUGH CLOCKS BEEN GIVEN TO DETECT ERROR
251 035272 103012          BHS      3$          ;BRANCH IF MORE CLOCKS TO BE GIVEN
252 035274 023737 034600 034570  CMP      HADTMP,POSITI;HAVE ENOUGH CLOCKS BEEN GIVEN FOR HARD ERROR
253          ;THAT IS HAVE 4128 MORE CLOCKS BEEN GIVEN
254 035302 001415          BEQ      5$          ;BRANCH IF YES
255 035304 032711 000400          BIT      #ZER,@R1      ;CHECK ZERO DETECT BIT IN RHMR
256 035310 001016          BNE      4$          ;BRANCH IS ZER SET
257
258          ;TO SAVE TIME
259 035312 004737 031554          JSR      PC,PUTREG      ;SAVE REGISTERS
260 035316 104034          EMT      34
261          ;WHEN 21 BITS IN ECC 32 BIT REGISTER IS 0
262
263
264 035320 052711 000002          3$:    BIS      #MCLK,@R1      ;SET CLOCK
265 035324 042711 000002          BIC      #MCLK,@R1      ;CLEAR CLOCK
266 035330 004737 034612          JSR      PC,ECTEST      ;GO TO GENERATE AND TEST ECC
267 035334 000737          BR       1$          ;CONTINUE
268
269          ;THIS EXTRA CLOCK IS TO BRING ECH HIGH
270          ;AFTER THIS CLOCK POSITION REGISTER MAY BE 10040 OR 10041 OCTAL
271
272 035336 052711 000002          5$:    BIS      #MCLK,@R1      ;SET CLOCK
273 035342 042711 000002          BIC      #MCLK,@R1      ;CLEAR CLOCK
    
```

```
274  
275 035346  
035346 012601  
276 035350 000200  
4S: MOV (SP)+,R1  
RTS R0 ::POP STACK INTO R1
```



```

1      .SBTTL  SOFTWARE DISK DATA ECC GEN. ROUTINE
2
3      ;THIS SUBROUTINE GENERATES THE ECC FOR WHAT IS ON DISK AND INSERTS THEM
4      ;ON LOCATIONS 'DISK+1000' AND 'DISK+1002'
5
6      FILLEC:
7      MOV     R0,-(SP)      ;;PUSH R0 ON STACK
8      MOV     R1,-(SP)      ;;PUSH R1 ON STACK
9      MOV     R2,-(SP)      ;;PUSH R2 ON STACK
10     MOV     R3,-(SP)      ;;PUSH R3 ON STACK
11     MOV     R4,-(SP)      ;;PUSH R4 ON STACK
12     MOV     R5,-(SP)      ;;PUSH R5 ON STACK
13     CLR     POSITI ;CLEAR POSITION
14     CLR     GECC1 ;CLEAR GECC1
15     CLR     GECC2 ;CLEAR
16     MOV     #DISK,R1      ;POINTER TO DATA FOR ECC GENERATION
17     MOV     #256.,R2      ;COUNTER FOR NUMBER OF DATA WORDS
18     MOV     #16.,R3       ;COUNTER FOR NUMBER OF BITS PER WORD
19     MOV     (R1)+,R4      ;DATA IN R4
20     ROR     R4            ;GET ONE DATA BIT IN CARRY
21     BCC     11$          ;BRANCH IF DATA BIT IS ZERO
22     MOV     #-1,ECDATA    ;ECC DATA BIT IS A ONE
23     BR      12$          ;BRANCH TO GENERATE ECC
24     CLR     ECDATA ;ECC DATA BIT IS A ZERO
25     JSR     PC,ECCTEST    ;GO TO GENERATE ECC
26     DEC     R3           ;DECREMENT BIT COUNT
27     BNE     10$         ;BRANCH IF 16 BITS NOT DONE
28     DEC     R2           ;DECREMENT WORD COUNT
29     BNE     9$          ;BRANCH IF 256 WORDS NOT DONE
30     MOV     GECC1,DISK+<256.*2>;INSERT ECC1 ON DISK
31     MOV     GECC2,DISK+<257.*2>;INSERT ECC2 ON DISK
32     MOV     (SP)+,R5      ;;POP STACK INTO R5
33     MOV     (SP)+,R4      ;;POP STACK INTO R4
34     MOV     (SP)+,R3      ;;POP STACK INTO R3
35     MOV     (SP)+,R2      ;;POP STACK INTO R2
36     MOV     (SP)+,R1      ;;POP STACK INTO R1
37     MOV     (SP)+,R0      ;;POP STACK INTO R0
38     RTS     PC
    
```

```

6 035352 010046
6 035352 010046
6 035354 010146
6 035356 010246
6 035360 010346
6 035362 010446
6 035364 010546
7 035366 005037 034570
8 035372 005037 034556
9 035376 005037 034560
10 035402 012701 040626
11 035406 012702 000400
12 035412 012703 000020
13 035416 012104
14 035420 006004
15 035422 103004
16 035424 012737 177777 034554
17 035432 000402
18 035434 005037 034554
19 035440 004737 034612
20 035444 005303
21 035446 001364
22 035450 005302
23 035452 001357
24 035454 013737 034556 041626
25 035462 013737 034560 041630
26 035470 012605
26 035472 012604
26 035474 012603
26 035476 012602
26 035500 012601
26 035502 012600
27 035504 000207
    
```

```

1      .SBTTL  RH BASE ADDRESS CHANGE ROUTINE
2
3      ;THIS ROUTINE WILL ALLOW THE CHANGE OF THE BASE
4      ;ADDRESS FROM 176700 TO ANY TYPED VALUE
5
6 035506 012706 001100  BASECH: MOV  #STACK,SP      ;RESET STACK
7 035512 104401 035520  TYPE  ,65$          ;::TYPE ASCIZ STRING
   035516 000415  BR    64$          ;::GET OVER THE ASCIZ
   ;:65$: .ASCIZ <CRLF>/PRESENT BASE ADDRESS IS /
   64$:
8 035552 013746 001240  MOV  RHCS1,-(SP)      ;GET READY TO TYPE OLD BASE
9 035556 104402  TYPOC
10 035560 104401 035566  TYPE  ,67$          ;::TYPE ASCIZ STRING
   035564 000425  BR    66$          ;::GET OVER THE ASCIZ
   ;:67$: .ASCIZ <CRLF>/TYPE NEW BASE ADDRESS FOLLOWED BY 'CR' /
   66$:
11 035640 004737 044620  JSR  PC,$TKINT      ;INITIALIZE THE TTY KEYBOARD
12 035644 104412  RDOCT
13 035646 012700 001230  MOV  #RHDB,R0       ;GET STARTING ADDRESS OF REGISTERS
14 035652 012701 000026  MOV  #22,R1         ;NUMBER OF REGISTERS
15 035656 012737 036436  MOV  #ADTMO,ERRVEC  ;SET UP TO CHECK NEW ADDRESS
16 035664 021637 001240  CMP  (SP),RHCS1     ;NEW CSR?
17 035670 001407  BEQ  1$           ;NO-SKIP NEXT
18 035672 005776 000000  TST  @ (SP)         ;ACCESS THE NEW ADDRESS
19 035676 163716 001240  SUB  RHCS1,(SP)     ;GET THE ADDRESS OFFSET
20 035702 061620  2$: ADD  (SP),(R0)+   ;AND PLUG IT IN
21 035704 005301  DEC  R1             ;DONE ALL OF THEM YET?
22 035706 001375  BNE  2$           ;NOT YET, SO DO MORE
23 035710 104401 035716  1$: TYPE ,69$      ;::TYPE ASCIZ STRING
   035714 000416  BR    68$          ;::GET OVER THE ASCIZ
   ;:69$: .ASCIZ <CRLF>/PRESENT VECTOR ADDRESS IS /
   68$:
24 035752 013746 001226  MOV  RPVEC,-(SP)    ;GET READY TO TYPE OLD VECTOR ADDRESS
25 035756 104402  TYPOC
26 035760 104401 035766  TYPE  ,71$          ;::TYPE ASCIZ STRING
   035764 000437  BR    70$          ;::GET OVER THE ASCIZ
   ;:71$: .ASCIZ <CRLF>/TYPE NEW VECTOR ADDRESS OR RETYPE OLD ONE FOLLOWED BY 'CR' /
   70$:
27 036064 104412  RDOCT
28 036066 012637 001226  MOV  (SP)+,RPVEC    ;SETUP VECTOR ADDRESS
29 036072 104401 036100  TYPE  ,73$          ;::TYPE ASCIZ STRING
   036076 000413  BR    72$          ;::GET OVER THE ASCIZ
   ;:73$: .ASCIZ <CRLF>/NEW BASE ADDRESS IS /
   72$:
30 036126 013746 001240  MOV  RHCS1,-(SP)
31 036132 104402  TYPOC
32 036134 104401 036142  TYPE  ,75$          ;::TYPE ASCIZ STRING
   036140 000414  BR    74$          ;::GET OVER THE ASCIZ
   ;:75$: .ASCIZ <CRLF>/NEW VECTOR ADDRESS IS /
   74$:
33 036172 013746 001226  MOV  RPVEC,-(SP)
34 036176 104402  TYPOC
35 036200 104401 036206  TYPE  ,77$          ;::TYPE ASCIZ STRING
   036204 000416  BR    76$          ;::GET OVER THE ASCIZ
   ;:77$: .ASCIZ <CRLF>/UNTIL PROGRAM IS RELOADED./
   76$:
036242
  
```

000004

RH	BASE	ADDRESS	CHANGE	ROUTINE			
36	036242	104401	036250		TYPE 79\$::TYPE ASCIZ STRING	
	036246	000402			BR 78\$::GET OVER THE ASCIZ	
					::79\$: .ASCIZ <CRLF>/ /		
	036254				78\$:		
37	036254	104401	036262		TYPE 81\$::TYPE ASCIZ STRING	
	036260	000424			BR 80\$::GET OVER THE ASCIZ	
					::81\$: .ASCIZ <CRLF>/UNLESS HALTED AND MANUALLY RESTARTED./		
	036332				80\$:		
38	036332	104401	036340		TYPE 83\$::TYPE ASCIZ STRING	
	036336	000425			BR 82\$::GET OVER THE ASCIZ	
					::83\$: .ASCIZ <CRLF>/PROGRAM WILL AUTOMATICALLY RESTART FROM /		
	036412				82\$:		
39	036412	012746	000200		MOV #200,-(SP)		
40	036416	104402			TYPOC		
41	036420	104401	036426		TYPE 85\$::TYPE ASCIZ STRING	
	036424	000402			BR 84\$::GET OVER THE ASCIZ	
					::85\$: .ASCIZ <CRLF>/ /		
	036432				84\$:		
42	036432	000137	004620		JMP BEGIN	:OK, NOW START OVER WITH NEW ADDRESS	
43							
44	036436				ADTIMO:		
	036436	104401	036444		TYPE 65\$::TYPE ASCIZ STRING	
	036442	000423			BR 64\$::GET OVER THE ASCIZ	
					::65\$: .ASCIZ <CRLF><377>/SPECIFIED ADDRESS DID NOT RESPOND. /		
	036512				64\$:		
45	036512	022626			CMP (SP)+,(SP)+	:RESTORE THE STACK	
46	036514	000137	035506		JMP BASECH	:AND DO IT AGAIN.	
47							
48					:*THIS IS A LITTLE ROUTINE THAT TESTS NED BIT 11 IN PHCS2		
49					:*THIS LOOPS HERE FOR EVER		
50					:*TO BE USED ONLY IF DRIVES PRESENT LOOKING AT NED DOES NOT AGREE		
51					:*WITH WHAT IS REALY THERE		
52	036520	000000			ERUNIT: 0	:UNIT UNDER MANUAL TEST	
53	036522	004737	032054		ERSTART:JSR PC,CLDISK	:SET GENERAL REG.	
54	036526	013712	036520		MOV ERUNIT,@R2	:SELECT UNIT	
55	036532	005714			1\$: TST @R4	:TEST RHER1	
56	036534	032712	010000		BIT #NED,@R2	:TEST NED	
57	036540	001401			BEQ 2\$:BRANCH IF GOOD	
58	036542	000773			BR 1\$:NED NOT SET	
59	036544	000772			2\$: BR 1\$:NED SET	

DISK SIMULATION

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

.SBTTL DISK SIMULATION

:IN A WRITE HEADER AND DATA COMMAND FILL THE FOLLOWING
WCLY=WITH CYLINDER TO BE ON DISK
WSECTR=WITH SECTOR AND TRACK TO BE ON DISK
WKEY1= WITH KEY1 TO BE ON DISK
WKEY2= WITH KEY2 TO BE ON DISK
FNWORD= NO OF DATA WORDS TO BE WRITTEN ON DISK
:THE COMMAND THEN IS JSR PC,COMWHD

:IN A WRITE DATA COMMAND FILL THE FOLLOWING
CYL=WITH CYLINDER TO BE FOUND ON DISK
SECOTR= WITH SECTOR AND TRACK TO BE FOUND ON DISK
KEY1= WITH KEY1 TO BE FOUND ON DISK
KEY2= WITH KEY2 TO BE FOUND ON DISK
X= 1 MUST BE ONE
NOWORD= WITH NUMBER OF DATA WORDS TO BE WRITTEN
:THE COMMAND THEN IS JSR PC,COMHD

:IN A READ HEADER AND DATA COMMAND FILL THE FOLLOWING
CYL= WITH CYLINDER TO BE FOUND ON DISK
SECOTR= WITH SECTOR AND TRACK TO BE FOUND ON DISK
KEY1= WITH KEY1 TO BE FOUND ON DISK
KEY2=WITH KEY2 TO BE FOUND ON DISK
DAWORD= WITH NUMBER OF WORDS TO BE FOUND ON DISK
X=0 MUST BE ZERO
:THE COMMAND THEN IS JSR PC,COMHD

:IN A READ DATA COMMAND FILL THE FOLLOWING
CYL= WITH CYLINDER TO BE FOUND ON DISK
SECOTR= WITH SECTOR AND TRACK TO BE FOUND ON DISK
KEY1= WITH KEY1 TO BE FOUND ON DISK
KEY2=WITH KEY2 TO BE FOUND ON DISK
DAWORD= WITH NUMBER OF WORDS TO BE FOUND ON DISK
X=0 MUST BE ZERO
:THE COMMAND THEN IS JSR PC,COMHD


```

53
54 ;DUMMY ERROR CALL LOCATIONS FOR THE READ HEADER OPERATION
55
56 036722 000240 SSSYN: NOP ;IF 'ERROR 2' INSERTED BY RDHEAD
57 ;SUBROUTINE, THEN THE FIRST SYNC
58 ;IS NOT DETECTED. NO BAD DATA
59 ;IS GIVEN BECAUSE SYNC=144000
60 ;CANNOT BE READ. WORD NUMBER
61 ;IS '1' BECAUSE THIS IS THE FIRST
62 ;WORD TESTED.
63
64 036724 000240 HEDGAP: NOP ;IF 'ERROR 3' INSERTED BY
65 ;RDHEAD SUBROUTINE, THEN THE
66 ;HEADER GAP 0'S WERE NOT
67 ;WRITTEN RIGHT.
68
69 ;IF 'WORD NO' CONTAINS, SAY
70 ;3(8), THEN IT IS THE THIRD
71 ;WORD OF A 5 WORD HEADER
72 ;GAP THAT IS WRONG.
73
74 ;'BAD DATA' CONTAINS WHAT IS
75 ;GOING ON THE DISK.
76
77 036726 000240 HEDSYN: NOP ;IF 'ERROR 3' INSERTED BY RDHEAD
78 ;SUBROUTINE, THEN THE HEADER SYNC
79 ;GENERATED BY DCL IS WRONG,
80 ;OR THE LAST BYTE
81 ;OF THE HEADER GAP 0'S IS WRONG.
82
83 ;IN EITHER CASE WORD NO=6,
84 ;RIGHT BYTE IS HEADER 0,
85 ;LEFT BYTE IS SYNC.
86
87 ;'BAD DATA' HAS WHAT IS GOING
88 ;ON DISK.
89
90
91 036730 005737 001406 TST ERFLGS ;WERE ANY ERRORS DETECTED ?
92 036734 001017 BNE OUT ;IF YES, EXIT ----->
93
94 036736 005737 036720 TST X ;IS IT A DATA WRITE OPERATION ?
95 036742 001410 BEQ DAREAD ;NO...THEN DO A DATA READ
96 036744 005737 037024 TST NOSYNC ;IS THIS FORCED HEADER ERROR COMMAND ?
97 ;IF YES NOSYNC=-1 THEN WRITE OR READ
98 ;IS SHUT OFF, SO BRANCH OUT
99 ;IF NOSYNC=0 THEN CONTINUE
100 036750 001011 BNE OUT ;EXIT IF SET ----->
101
102 036752 004137 040276 JSR R1,WRDATA ;WRITE DATA <----->
103 036756 000000 NOWORD: .WORD 0 ;NO OF WORDS TO BE WRITTEN
104 036760 000000 Y: .WORD 0 ;
105 036762 000404 BR OUT ;EXIT ----->
106
107 036764 004137 043232 DAREAD: JSR R1,REDATA ;READ DATA <----->
108 036770 000000 DAWORD: .WORD 0 ;NO OF WORDS TO BE READ
109 036772 000000 .WORD 0
    
```


110					
111	036774		OUT:		
	036774	012605	MOV	(SP)+,R5	::POP STACK INTO R5
	036776	012604	MOV	(SP)+,R4	::POP STACK INTO R4
	037000	012603	MOV	(SP)+,R3	::POP STACK INTO R3
	037002	012602	MOV	(SP)+,R2	::POP STACK INTO R2
	037004	012601	MOV	(SP)+,R1	::POP STACK INTO R1
	037006	012600	MOV	(SP)+,R0	::POP STACK INTO R0
112	037010	000207	RTS	PC	::EXIT

1
2
3
4
5
6
7
8 037012 014400
9 037014 000000
10 037016 000000
11 037020 000000
12 037022 000000
13
14
15
16
17
18
19
20
21

```
*****  
:*THE DISK SECTOR IS DEVIDED AS FOLLOWS  
  
:*19 WORDS OF 0, ONE WORD 144000  
:*THESE MAKE 39 BYTES FOR SECTOR GAP AND ONE SYNC. BYTE  
  
RSYNC: 14400  
RCYL: 0  
RSETR: 0  
RKEY1: 0  
RKEY2: 0  
  
:*5 WORDS OF 0'S, ONE WORD 144000  
:*THESE MAKE 11 BYTES FOR HEADER GAP AND ONE SYNC. BYTE  
:*THESE ARE DCL GENERATED  
  
:*THERE ARE 256 WORDS OF DATA  
:*THERE ARE 2 WORDS FOR ECC GENERATED BY DCL  
:*15 WORDS OF 0'S FOR DATA GAP AND TOLERANCE GAP  
*****
```



```

1          ::*****
2          ::*READ DISK HEADER
3          ::*****
4
5 037024 000000 NOSYNC: 0 ;FORCED HEADER ERROR = -1
6          ;NORMAL = 0
7 037026 000000 TY: 0 ;ERROR TYPE NO.
8 037030 000000 ERWORD: 0 ;ERROR WORD NO.
9
10 037032 012137 037014 RDHEAD: MOV (R1)+,RCYL ;STORE CYLINDER ADDRESS
11 037036 012137 037016 MOV (R1)+,RSETR ;STORE SECTOR AND TRACK ADDRESS
12 037042 012137 037020 MOV (R1)+,RKEY1 ;STORE KEY1
13 037046 012137 037022 MOV (R1)+,RKEY2 ;STORE KEY2
14 037052 012137 037622 MOV (R1)+,COMPA ;STORE COMPARE OR NOT
15 037056 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
16
17 037060 013700 001260 MOV RHMR,RO ;RO CONTAINS MAINTANENCE REG.
18 037064 012705 000002 MOV #2,R5 ;R5 IS A COUNTER FOR WORDS
19 037070 012710 000001 MOV #DMD,@RO ;SET DIAG. MODE
20 037074 052710 000010 BIS #MSTCK,@RO ;SET SECTOR CLOCK FOR FIRST WORD
21 037100 052710 000002 BIS #MCLK,@RO ;SET MAINT.CLOCK FOR FIRST WORD
22 037104 042710 000012 BIC #MSTCK!MCLK,@RO ;RESET THEM
23
24 037110 000404 BR 2$ ;DON'T GIVE SECTOR CLOCK FIRST TIME
25 037112 012710 000013 1$: MOV #MSTCK!MCLK!DMD,@RO ;SET SECTOR, CLOCK, DIAG. MODE, RESET INDEX
26 037116 042710 000012 BIC #MSTCK!MCLK,@RO ;RESET SECTOR & MAINT.CLOCK
27
28 037122 012702 000007 2$: MOV #7,R2 ;LOAD BYTE COUNTER
29 037126 052710 000002 3$: BIS #MCLK,@RO ;SET MAINT. CLOCK
30 037132 042710 000002 BIC #MCLK,@RO ;RESET IT
31 037136 005302 DEC R2 ;BYTE COUNTER
32 037140 001372 BNE 3$ ;CONTINUE IF BYTE NOT COMPLETE
33 037142 005305 DEC R5 ;WORD CCUNTER
34 037144 001362 BNE 1$ ;CONTINUE IF WORD NOT COMPLETE
35
36 037146 012702 000022 4$: MOV #18.,R2 ;LOAD NO OF WORDS OF ZEROS
37 037152 005037 037620 CLR WORD ;DO 0'S
38 037156 004737 037624 JSR PC,READ ;READ 0'S <----->
39 037162 005302 DEC R2 ;COUNT DOWN WORDS
40 037164 001372 BNE 4$ ;CONTINUE
41
42 037166 013737 037012 037620 MOV RSYNC,WORD ;SYNC. WORD
43 037174 004737 037624 JSR PC,READ ;READ IT <----->
44 037200 032710 001000 BIT #DTSY,@RO ;SYNC. BYTE DETECTED?
45 037204 001012 BNE 5$ ;CONTINUE IF SYNC DETECTED
46 037206 012737 000001 037030 MOV #1,ERWORD ;ERROR WORD NO
47 037214 013737 037012 001124 MOV RSYNC,$GDDAT;SYNC WORD
48 037222 012737 104002 036722 MOV #104002,SSYN ;INSERT 'ERROR 2' IN SSYN
49 037230 000571 BR 13$ ;BRANCH OUT ----->
50
51 037232 013737 037014 037620 5$: MOV RCYL,WORD ;SETUP CYLINDER
52 037240 004737 037624 JSR PC,READ ;READ IT <----->
53 037244 013737 037016 037620 MOV RSETR,WORD ;SETUP SECTOR/TRACK
54 037252 004737 037624 JSR PC,READ ;READ THEM <----->
55 037256 013737 037020 037620 MOV RKEY1,WORD ;SETUP KEY1
56 037264 004737 037624 JSR PC,READ ;READ IT <----->
57 037270 013737 037022 037620 MOV RKEY2,WORD ;SETUP KEY2
    
```

```

58 037276 004737 037624      JSR    PC,READ ;READ IT <----->
59 037302 013737 040610 037620  MOV    WCRC,WORD ;SETUP CRC
60 037310 004737 037624      JSR    PC,READ ;READ IT <----->
61
62 037314 005737 001426      TST    TESDTE ;IS THIS A DRIVE TIMING ERROR ?
63 037320 001135      BNE    13$ ;BRANCH OUT IF YES ----->
64 037322 005737 037622      TST    COMPA ;IS THIS A READ OR WRITE COMMAND ?
65 037326 001472      BEQ    11$ ;DO READ IF = 0
66
67 ;*CONTINUE WITH DIAGNOSTIC WRITE COMMAND
68
69 037330 012705 040612      MOV    #HEGAP,R5 ;POINTER FOR HEADER GAP
70 037334 012702 000005      MOV    #5,R2 ;NO OF WORDS OF ZEROS
71 037340 012737 000006 037030 6$:  MOV    #6,ERWORD ;ERROR WORD NO SET
72 037346 004737 040056      JSR    PC,WRITE ;FOR HEADER GAP
73 037352 005737 040054      TST    WWORD ;TEST WRITTEN WORD
74 037356 001413      BEQ    7$ ;CONTINUE IF GOOD, THAT IS = 0
75 037360 160237 037030      SUB    R2,ERWORD ;WORD NO IN ERROR
76 037364 005037 001124      CLR    $GDDAT ;GOOD WORD SHOULD BE 0
77 037370 013737 040054 001126  MOV    WWORD,$BDDAT ;BAD DATA
78 037376 012737 104003 036724  MOV    #104003,HEDGAP;'ERROR 2' GOES IN HEDGAP
79 037404 000503      BR     13$ ;BRANCH OUT ----->
80
81 037406 013725 040054      7$:  MOV    WWORD,(R5)+ ;SAVE HEADER GAP
82 037412 005302      DEC    R2
83 037414 001351      BNE    6$
84 037416 004737 040056      JSR    PC,WRITE ;WRITE HEADER (DATA) GAP SYNC
85 037422 023737 037012 040054  CMP    RSYNC,WWORD
86 037430 001426      BEQ    10$
87 037432 005737 037024      TST    NOSYNC ;IS THIS FORCED HEADER ERROR COMMAND ?
88 ;IF YES NOSYNC=-1 THEN WRITE OR READ
89 ;IS SHUT OFF SO BRANCH OUT
90 ;IF NO NOSYNC=0 THEN CONTINUE
91 037436 001406      BEQ    14$ ;PRINT IT IF TRUE ERROR
92
93 037440 005737 040054      TST    WWORD ;IS IT = 0 ?
94 037444 001420      BEQ    10$ ;CONTINUE IF GOOD
95 037446 005037 001124      CLR    $GDDAT ;IT SHOULD BE ZERO
96 037452 000403      BR     15$ ;BRANCH TO TYPE ERROR
97 037454 013737 037012 001124 14$:  MOV    RSYNC,$GDDAT;GOOD DATA
98 037462 013737 040054 001126 15$:  MOV    WWORD,$BDDAT;BAD DATA
99 037470 012737 000006 037030  MOV    #6,ERWORD
100 037476 012737 104003 036726  MOV    #104003,HEDSYN
101 037504 000443      BR     13$ ;BRANCH OUT ----->
102
103 037506 013725 040054      10$:  MOV    WWORD,(R5)+ ;SAVE DATA SYNC.
104 037512 000440      BR     13$ ;EXIT ----->
105
106 ;*READ COMMAND START FROM HERE
107
108 037514 012702 000005      11$:  MOV    #5,R2
109 037520 005037 037620      12$:  CLR    WORD
110 037524 004737 037624      JSR    PC,READ ;READ HEADER GAP <----->
111 037530 005302      DEC    R2 ;ARE 5 HEADER GAP ZEROS COMPLETE ?
112 037532 001372      BNE    12$ ;IF NOT CONTINUE
113 037534 013737 037012 037620  MOV    RSYNC,WORD ;SYNC WORD
114 037542 004737 037624      JSR    PC,READ ;READ HEADER (DATA) SYNC)
    
```


115	037546	005737	037024			TST	NOSYNC ;FORCED	SYNC ERROR ?
116	037552	001404				BEQ	16\$:IF NOT ERROR COMMAND CONTINUE
117	037554	032710	001000			BIT	#DTSY,@R0	:SYNC. DETECTED
118	037560	001415				BEQ	13\$:IF ZERO BRANCH OUT ----->
119	037562	000403				BR	17\$:IF NOT ZERO BRANCH TO ERROR
120								
121	037564	032710	001000		16\$:	BIT	#DTSY,@R0	:SYNC. DETECTED ?
122	037570	001011				BNE	13\$:EXIT IF YES ----->
123	037572	012737	000006	037030	17\$:	MOV	#6,ERWORD	:ERROR WORD NO.
124	037600	013737	037012	001124		MOV	RSYNC,\$GDDAT;SYNC WORD	
125	037606	012737	104002	036726		MOV	#104002,HEDSYN;MOVE 'ERROR 2'	
126	037614				13\$:			
	037614	012601				MOV	(SP)+,R1	::POP STACK INTO R1
127	037616	000201				RTS	R1	:EXIT ----->

DISK SIMULATION

```

1
2
3
4
5 037620 000000
6 037622 000000
7
8 037624
   037624 010246
9 037626 012705 000002
10 037632 012710 000001
11 037636 006037 037620
12 037642 103002
13 037644 052710 000020
14 037650 012702 000007
15 037654 052710 0C0012
16 037660 005737 034562
17 037664 001411
18 037666 032710 000020
19 037672 001404
20 037674 012737 177777 034554
21 037702 000402
22 037704 005037 034554
23 037710 012746 000001
24 037714 006037 037620
25 037720 103002
26 037722 012716 000021
27 037726 012610
28 037730 005737 034562
29 037734 001404
30 037736 005237 034574
31 037742 004737 034612
32 037746 052710 000002
33 037752 005737 034562
34 037756 001411
35 037760 032710 000020
36 037764 001404
37 037766 012737 177777 034554
38 037774 000402
39 037776 005037 034554
40 040002 012746 000001
41 040006 006037 037620
42 040012 103002
43 040014 012716 000021
44 040020 012610
45 040022 005737 034562
46 040026 001404
47 040030 005237 034574
48 040034 004737 034612
49
50 040040 005302
51 040042 001341
52 040044 005305
53 040046 001300
54 040050 012602
55 040052 000207

*****
*READ ONE WORD IN 'WORD'
*****

WORD: 0
COMPA: 0

READ:
MOV R2,-(SP)      ;;PUSH R2 ON STACK
MOV #2,R5        ;;WORD COUNTER
MOV #DMD,@R0     ;;SET DIAG. MODE
ROR WORD        ;;CHECKING IF THERE IS A ONE
BCC 1$          ;;IF NO ONE BRANCH
BIS #MRD,@R0     ;;SET BIT 4 IF DATA HAS ONE
1$: MOV #7,R2     ;;BYTE COUNTER
BIS #MSTCK!MCLK,@R0 ;;SET CLOCK,DATA IF ANY, SECTOR
TST TSECCG ;IS THIS BIT TO GENERATE AND TEST ECC ?
BEQ 6$          ;;BRANCH IF NO
BIT #MRD,@R0    ;;IS DATA BIT A ONE ?
BEQ 5$          ;;BRANCH IF DATA BIT IS 0
MOV #-1,ECDATA ;;ECC DATA BIT IS A ONE
BR 6$          ;;BRANCH
5$: CLR ECDATA ;ECC DATA BIT IS A 0
6$: MOV #DMD,-(SP) ;;KEEP ONLY DIAG. MODE
ROR WORD        ;;CHECKING IF THERE IS A ONE
BCC 2$          ;;IF NO ONE BRANCH
MOV #MRD!DMD,(SP) ;;KEEP DATA AND DIAG. MODE
2$: MOV (SP)+,@R0 ;;PUT IN DATA,RESET CLOCK, SECTOR
TST TSECCG ;IS ECC TO BE GENERATED FOR THIS BIT
BEQ 3$          ;;BRANCH IF NO
INC DATENV ;NUMBER OF CLOCKS GIVEN FOR DATA ENVELOPE
JSR PC,ECTEST   ;;GO TO GENERATE AND TEST ECC
3$: BIS #MCLK,@R0 ;;SET CLOCK
TST TSECCG ;IS THIS BIT TO GENERATE ECC
BEQ 8$          ;;BRANCH IF NO
BIT #MRD,@R0    ;;IS DATA BIT A ONE
BEQ 7$          ;;BRANCH IF DATA BIT IS = 0
MOV #-1,ECDATA ;;ECC DATA BIT IS A ONE
BR 8$          ;;BRANCH
7$: CLR ECDATA ;ECC DATA BIT IS = 0
8$: MOV #DMD,-(SP) ;;KEEP DIAG. MODE
ROR WORD        ;;CHECKING IF THERE IS A ONE
BCC 4$          ;;BRANCH IF NO ONE
MOV #MRD!DMD,(SP) ;;KEEP DIAG. MODE AND DATA
4$: MOV (SP)+,@R0 ;;SET DATA, DIAG. MODE, CLEAR CLOCK
TST TSECCG ;IS THIS BIT TO GENERATE ECC
BEQ 9$          ;;BRANCH IF NO
INC DATENV ;NUMBER OF CLOCKS FOR DATA ENVELOPE
JSR PC,ECTEST   ;;GO TO GENERATE AND TEST ECC
9$: DEC R2      ;;BYTE COUNTER
BNE 3$        ;;CONTINUE IF ONE BYTE NOT COMPLETE
DEC R5       ;;WORD COUNTER
BNE 1$       ;;CONTINUE IF ONE WORD NOT COMPLETE
MOV (SP)+,R2 ;;POP STACK INTO R2
RTS PC      ;;EXIT ----->

```



```

1
2
3
4
5 040054 000000
6
7 040056
   040056 010046
   040060 010246
   040062 010346
   040064 010546
8 040066 012705 000002
9 040072 012710 000001
10 040076 012702 000007
11 040102 012710 000013
12 040106 032710 000040
13 040112 001406
14 040114 012737 177777 034554
15 040122 000261
16 040124 006003
17 040126 000404
18 040130 005037 034554
19 040134 000241
20 040136 006003
21 040140 012710 000001
22 040144 005737 034562
23 040150 001404
24 040152 005237 034574
25 040156 004737 034612
26
27 040162 052710 000002
28 040166 032710 000040
29 040172 001406
30 040174 012737 177777 034554
31 040202 000261
32 040204 006003
33 040206 000404
34 040210 005037 034554
35 040214 000241
36 040216 006003
37 040220 012710 000001
38 040224 005737 034562
39 040230 001404
40 040232 005237 034574
41 040236 004737 034612
42
43 040242 005302
44 040244 001346
45 040246 005305
46 040250 001312
47 040252 010337 040054
48 040256 012605
   040260 012603
   040262 012602
   040264 012600
49 040266 000207

*****
*WRITE ONE WORD WHICH COMES BACK IN 'WORD'
*****
WORD: 0

WRITE:
   MOV R0,-(SP)      ;;PUSH R0 ON STACK
   MOV R2,-(SP)      ;;PUSH R2 ON STACK
   MOV R3,-(SP)      ;;PUSH R3 ON STACK
   MOV R5,-(SP)      ;;PUSH R5 ON STACK
   MOV #2,R5         ;WORD COUNTER
   MOV #1,@R0        ;SET DIAG. MODE
1$:  MOV #7,R2         ;BYTE COUNTER
   MOV #MSTCK!MCLK!DMD,@R0 ;SET SECTOR & MANT. CLOCKS
   BIT #MWR,@R0      ;CHECK WRITEBIT IN MAINT. REG.
   BEQ 2$            ;BRANCH IF ZERO
   MOV #-1,ECDATA    ;ECC DATA BIT IS A ONE
   SEC              ;SET CARRY
   ROR R3            ;MOVE 1 FORWARD
   BR 3$
2$:  CLR ECDATA      ;ECC DATA BIT IS = 0
   CLC              ;CLEAR CARRY
   ROR R3            ;MOVE 0 FOR WORD
3$:  MOV #DMD,@R0    ;CLEAR SECTOR AND CLOCK
   TST TSECCG       ;IS THIS BIT TO GENERATE ECC ?
   BEQ 4$           ;BRANCH IF NO
   INC DATENV       ;NUMBER OF CLOCKS FOR DATA ENVELOPE
   JSR PC,ECTEST    ;GO TO GENERATE AND TEST ECC <----->
26
4$:  BIS #MCLK,@R0   ;SET CLOCK
   BIT #MWR,@R0     ;CHECK WRITE BIT IN MAINT. REG.
   BEQ 5$           ;BRANCH IF ZERO
   MOV #-1,ECDATA   ;ECC DATA BIT IS A ONE
   SEC              ;SET CARRY
   ROR R3            ;MOVE 1 FOR WORD
   BR 6$
5$:  CLR ECDATA      ;ECC DATA BIT IS ZERO
   CLC              ;CLEAR CARRY
   ROR R3            ;MOVE 0 FOR WORD
6$:  MOV #DMD,@R0    ;CLEAR CLOCK
   TST TSECCG       ;IS THIS BIT TO GENERATE ECC ?
   BEQ 7$           ;BRANCH IF NO
   INC DATENV       ;NUMBER OF CLOCKS FOR DATA ENVELOPE
   JSR PC,ECTEST    ;GO TO GENERATE AND TEST ECC <----->
42
7$:  DEC R2          ;COUNT FOR BYTE END
   BNE 4$           ;IF NOT BYTE END CONTINUE
   DEC R5           ;COUNT FOR WORD END
   BNE 1$          ;IF NOT WORD END CONTINUE
   MOV R3,WORD      ;STORE THE WORD
   MOV (SP)+,R5     ;;POP STACK INTO R5
   MOV (SP)+,R3     ;;POP STACK INTO R3
   MOV (SP)+,R2     ;;POP STACK INTO R2
   MOV (SP)+,R0     ;;POP STACK INTO R0
   RTS             ;EXIT ----->

```

```

1
2
3
4
5 040270 000000
6 040272 000400
7 040274 000000
8 040276
9 040276 011137 040270
10 040302 012102
11 040304 012137 037622
12 040310 010046
    040312 010146
    040314 010246
    040316 010346
    040320 010446
13 040322 012701 000016
14 040326 012703 041634
15 040332 012723 177777
16 040336 005301
17 040340 001374
18
19 040342 013700 001260
20 040346 013746 040272
21 040352 163716 040270
22 040356 011637 040274
23 040362 012604
24 040364 005737 001424
25 040370 001403
26 040372 012737 177777 034562
27 040400 012703 040626
28 040404 004737 040056
29 040410 013723 040054
30 040414 005302
31 040416 001372
32 040420 005704
33 040422 001406
34
35 040424 004737 040056
36 040430 013723 040054
37 040434 005304
38 040436 001372
39 040440 005037 034562
40 040444 012701 000002
41 040450 004737 040056
42 040454 013723 040054
43 040460 005301
44 040462 001372
45 040464 004737 040056
46 040470 013723 040054
47 040474 012701 000016
48 040500 004737 040056
49 040504 013723 040054
50 040510 005301
51 040512 001372
52 040514 012604
    040516 012603
    
```

```

*****
*WRITE DATA HOUSEKEEPING ROUTINE
*****
COUNTD: 0
FORMAT: 256.
ZWORDS: 0
WRDATA:
MOV (R1),COUNTD ;STORE NO. OF WORDS TO BE WRITTEN
MOV (R1)+,R2 ;SAME IN R2
MOV (R1)+,COMP ;COMPARE OR NOT
MOV R0,-(SP) ;PUSH R0 ON STACK
MOV R1,-(SP) ;PUSH R1 ON STACK
MOV R2,-(SP) ;PUSH R2 ON STACK
MOV R3,-(SP) ;PUSH R3 ON STACK
MOV R4,-(SP) ;PUSH R4 ON STACK
MOV #14.,R1 ;NO. OF TOLERANCE GAP WORDS
MOV #TOLGAP,R3 ;START OF TOLERANCE GAP TABLE
1$: MOV #-1,(R3)+ ;MAKE IT 177777
DEC R1 ;IS 14 COMPLETED ?
BNE 1$ ;IF NOT, CONTINUE

MOV RHMR,R0 ;R0 CONTAINS MAINTANENCE REG.
MOV FORMAT,-(SP)
SUB COUNTD,(SP)
MOV (SP),ZWORDS ;NO. OF ZERO WORDS TO BE WRITTEN
MOV (SP)+,R4
TST TSECC ;IS THIS AN ECC TEST
BEQ 7$ ;BRANCH IF NO
MOV #-1,TSECCG ;THESE BITS ARE TO GENERATE ECC
7$: MOV #DISK,R3 ;SIMULATED DISK AREA
2$: JSR PC,WRITE ;WRITE ON SIMULATED DISK
MOV WWORD,(R3)+ ;STORE ON SIMULATED DISK
DEC R2
BNE 2$
TST R4 ;ANY ZEROS TO BE WRITTEN ?
BEQ 4$ ;BRANCH IF NONE TO BE WRITTEN

3$: JSR PC,WRITE ;WRITE ZEROS ON SIMULATED DISK
MOV WWORD,(R3)+ ;STORE THEM
DEC R4
BNE 3$
4$: CLR TSECCG ;NO MORE ECC TO BE GENERATED
MOV #2,R1
5$: JSR PC,WRITE ;WRITE ECC1 AND ECC2 ON SIMULATED DISK
MOV WWORD,(R3)+ ;STORE IN WEEC1 AND WEEC2
DEC R1
BNE 5$
JSR PC,WRITE ;WRITE DATA GAP
MOV WWORD,(R3)+ ;STORE IT
MOV #14.,R1
6$: JSR PC,WRITE ;WRITE TOLERANCE GAP ZEROS
MOV WWORD,(R3)+ ;STORE THEM
DEC R1
BNE 6$
MOV (SP)+,R4 ;POP STACK INTO R4
MOV (SP)+,R3 ;POP STACK INTO R3
    
```


040520 012602
040522 012601
040524 012600
53 040526 000201

MOV (SP)+,R2
MOV (SP)+,R1
MOV (SP)+,R0
RTS R1

::POP STACK INTO R2
::POP STACK INTO R1
::POP STACK INTO R0
:EXIT ----->

1
2
3
4
5 040530
6 040576
7 040600
8 040610
9 040612
10 040624
11 040626
12 040626
13 041626
14 041630
15 041632
16 041634

```
*****  
: *THIS IS THE SIMULATED DISK, ONLY ONE SECTOR OF SPACE IS ALLOCATED  
: *****  
SECGAP: .BLKW 19.          ;SECTOR GAP 38 BYTES OF 0  
WSSYNC: .BLKW 1           ;SECTOR GAP 1 BYTE OF 0 ONE SYNC BYTE  
HEADER: .BLKW 4           ;HEADER = CYL, SECTOR/TRACK, KEY1, KEY2  
WCRC: .BLKW 1            ;CRC  
HEGAP: .BLKW 5           ;HEADER GAP 10 BYTES OF 0  
HDWSYN: .BLKW 1          ;HEADER GAP 1 BYTE OF 0 ONE SYNC. BYTE  
SILOTB:                   ;(ALSO USED IN SILO TEST AS SILO TABLE)  
DISK: .BLKW 256.         ;DATA SPACE  
WECC1: .BLKW 1           ;ECC1  
WECC2: .BLKW 1           ;ECC2  
DTAGAP: .BLKW 1          ;DATA GAP 2 BYTES OF 0  
TOLGAP: .BLKW 14.        ;TOLERANCE GAP 28 BYTES OF 0
```


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

```

:*****
:*WRITE HEADER AND DATA
:*****
:*THIS SUBROUTINE IS THE FIRST IN A SERIES OF NESTED SUBROUTINES
:*IT ISSUES DIAGNOSTIC MODE, AN EXTRA DIAGNOSTIC INDEX, AND THE
:*'GO' BIT
:*IT THEN CALLS THREE OTHER SUBROUTINES, WHICH IN TURN CALL OTHER
:*SUBROUTINES. THE THREE SUBROUTINES CALLED HERE ARE:
:*SEARCH          :ISSUES SECTOR CLOCKS TO SET SECTOR FOUND FLOP
:*WRHEAD          :WRITES THE SECTOR HEADER
:*WRDATA          :WRITES THE ACTUAL SECTOR DATA
:*ALL OF THE ABOVE MENTIONED 'WRITING' IS ACTUALLY DONE INTO A CORE
:*BUFFER AREA CALLED 'DISK' VIA THE MAINTENANCE REGISTER (RHMR)
    
```

```

25 041670 000000          RNCTR1: .WORD 0          ;'RUN' LINE STALL COUNTER
26
27 041672 011637 001414  COMWHD: MOV (SP),PCJSR      ;SAVE PC OF JSR + 4
28 041676 162737 000004 001414  SUB #4,PCJSR      ;SAVE PC OF JSR
29 041704 010046          MOV R0,-(SP)      ;:PUSH R0 ON STACK
    041706 010146          MOV R1,-(SP)      ;:PUSH R1 ON STACK
    041710 010246          MOV R2,-(SP)      ;:PUSH R2 ON STACK
    041712 010346          MOV R3,-(SP)      ;:PUSH R3 ON STACK
    041714 010446          MOV R4,-(SP)      ;:PUSH R4 ON STACK
    041716 010546          MOV R5,-(SP)      ;:PUSH R5 ON STACK
30
31 041720 012777 000001 137332  MOV #DMD,@RHMR    ;SET DIAGNOSTIC MODE
32 041726 052777 000004 137324  BIS #MINX,@RHMR  ;SET DIAGNOSTIC INDEX
33 041734 042777 000004 137316  BIC #MINX,@RHMR  ;CLEAR DIAGNOSTIC INDEX
34 041742 052777 000001 137270  BIS #GO,@RHCS1   ;SET 'GO' BIT & STALL 'TILL 'RUN'
35                                     ;(FUNCTION CODE WAS SET UP BY THE TEST)
36 041750 012737 000113 041670  RNWAT1: MOV #75.,RNCTR1 ;LOAD STALL COUNTER = APPROX 450US
37                                     ;FOR 11/50 CPU
38 041756 005337 041670          1$: DEC RNCTR1 ;COUNT DOWN 1 TIME
39 041762 001375          BNE 1$          ;CONTINUE UNTIL = 0
40
41 041764 013746 042050          MOV WSECTR,-(SP)  ;GET DESIRED SECTOR/TRACK
42 041770 042716 177740          BIC #177740,(SP) ;MAKE ONLY SECTOR
43 041774 012637 042004          MOV (SP)+,WTRK   ;SAVE SECTOR
44 042000 004137 042756          2$: JSR R1,SEARCH ;ISSUE SECTOR CLOCKS TO GET TO
45                                     ;DESIRED SECTOR <----->
46
47 042004 000000          WTRK: .WORD 0          ;SECTOR NO.
48 042006 012701 000240          MOV #+NOP,R1     ;GOING TO MOVE NOPS
49 042012 010137 042060          MOV R1,SEGPER   ;NOP INTO SEGAP
50 042016 010137 042062          MOV R1,FSYNER   ;NOP INTO FSYNER
51 042022 010137 042064          MOV R1,ERHEAD   ;NOP INTO ERHEAD
52 042026 010137 042066          MOV R1,ERCRC    ;NOP INTO ERCRC
    
```



```

110
111
112
113
114
115
116 042070 000240          ERHDGP: NOP
117
118
119
120
121
122
123
124
125
126
127
128
129 042072 000240          HDESYN: NOP
130
131
132
133
134
135
136
137
138
139
140
141
142
143 042074 005737 001406          TST      ERFLG$ ;ARE ANY ERRORS DETECTED ?
144 042100 001004          BNE      FOUT    ;IF YES BRANCH
145 042102 004137 040276          JSR      R1,WRDATA ;WRITE THE DATA
146
147 042106 000000          FNWORD: .WORD 0    ;FORMAT COMMAND NO. OF DATA
148 042110 000000          .WORD 0
149
150 042112          FOUT:
    042112 012605          MOV      (SP)+,R5    ;;POP STACK INTO R5
    042114 012604          MOV      (SP)+,R4    ;;POP STACK INTO R4
    042116 012603          MOV      (SP)+,R3    ;;POP STACK INTO R3
    042120 012602          MOV      (SP)+,R2    ;;POP STACK INTO R2
    042122 012601          MOV      (SP)+,R1    ;;POP STACK INTO R1
    042124 012600          MOV      (SP)+,R0    ;;POP STACK INTO R0
151 042126 000207          RTS      PC          ;EXIT
    
```

```

;GOOD DATA IS WHAT SHOULD BE ON DISK
;BAD DATA IS WHAT IS GOING ON DISK.

;WORD NO IS 5

;IF 'ERROR 6' INSERTED BY
;WRHEAD SUBROUTINE, THEN HEADER
;GAP GOING ON DISK IS WRONG.

;WORD NO GIVES WHICH OF
;THE HEADER GAP WORDS
;ARE WRONG. FOR EXAMPLE:

;WORD NO 1 = FIRST HEADER
;                GAP WORD
;BAD WORD IS WHAT IS GOING ON DISK

;IF 'ERROR 6' INSERTED BY
;WRHEAD SUBROUTINE, THEN LAST
;HEADER GAP BYTE OR HEADER
;SYNC BYTE GOING ON DISK IS WRONG.

;WORD NO = 5

;BAD DATA IS WHAT IS GOING
;ON DISK, RIGHT BYTE IS HEADER
;GAP 0'S BYTE, LEFT BYTE IS HEADER
;GAP SYNC.
    
```

```

1
2
3
4
5
6
7
8
9
10
11
12 042130 000000
13 042132 000000
14 042134 000000
15 042136 000000
16 042140 000000
17
18 042142 012137 042130
19 042146 012137 042132
20 042152 012137 042134
21 042156 012137 042136
22 042162 012137 042140
23 042166 010146
24
25 042170 012701 040530
26 042174 013700 001260
27 042200 012710 000001
28 042204 012705 000002
29 042210 052710 000010
30 042214 012710 000013
31
32 042220 032710 000040
33 042224 001403
34 042226 000261
35 042230 006003
36 042232 000402
37 042234 000241
38 042236 006003
39 042240 012710 000001
40 042244 012702 000007
41 042250 052710 000002
42 042254 032710 000040
43 042260 001403
44
45 042262 000261
46 042264 006003
47 042266 000402
48 042270 000241
49 042272 006003
50 042274 012710 000001
51 042300 005302
52 042302 001362
53 042304 005305
54 042306 001342
55 042310 010321
56 042312 005703
57 042314 001414

*****
*WRITE HEADER
*****

;*R0 = MAINT.REG.
;*R1 = SIMULATED DISK
;*R2 = BYTE COUNT
;*R3 = WRITE WORD
;*R5 = WORD COUNT

SCYL: 0
SSECTR: 0
SKEY1: 0
SKEY2: 0
SCRC: 0

WRHEAD: MOV (R1)+,SCYL
MOV (R1)+,SSECTR
MOV (R1)+,SKEY1
MOV (R1)+,SKEY2
MOV (R1)+,SCRC
MOV R1,-(SP) ;;PUSH R1 ON STACK

MOV #SECGAP,R1 ;;SIMULATED DISK INDICATOR
MOV RHM,R0 ;R0 NOW HAS MAINT. REG. ADDR.
MOV #DMD,@R0 ;;SET DIAG. MODE
MOV #2,R5 ;;WORD COUNTER
BIS #MSTCK,@R0 ;;SET SECTOR FOR FIRST BYTE
1$: MOV #MSTCK!MCLK!DMD,@R0;SET SECTOR, CLOCK, DIAG.
;;MODE, RESET INDEX
;;CHECK WRITE BIT IN MAINT. REG.
BIT #MWR,@R0
BEQ 2$

2$: CLC ;;CLEAR CARRY
ROR R3 ;;MOVE ONE FORWARD
BR 3$

3$: MOV #DMD,@R0 ;;CLEAR CLOCK, SECTOR
MOV #7,R2 ;;BYTE COUNTER
4$: BIS #MCLK,@R0 ;;SET BIT CLOCK
BIT #MWR,@R0 ;;CHECK WRITE BIT IN MAINT.REG.
BEQ 5$

5$: SEC ;;SET CARRY
ROR R3 ;;MOVE ONE FORWARD
BR 6$

6$: MOV #DMD,@R0
DEC R2
BNE 4$
DEC R5
BNE 1$
MOV R3,(R1)+
TST R3
BEQ 7$
    
```



```

58
59 042316 012737 000001 037030      MOV    #1,ERWORD
60 042324 005037 001124              CLR    $GDDAT
61 042330 010337 001126              MOV    R3,$BDDAT
62 042334 012737 104006 042060      MOV    #104006,SEGP
63 042342 000137 042750              JMP    17$      ;BRANCH OUT ----->
64
65 042346 012702 000022 7$:      MOV    #18.,R2      ;COUNT NO. OF SECTOR GAP
66 042352 012737 000024 037030 10$:     MOV    #20.,ERWORD ;COUNT TO GIVE ERROR WORD
67 042360 004737 040056              JSR    PC,WRITE    ;WRITE SECTOR GAP
68 042364 013721 040054              MOV    WWORD,(R1)+ ;STORE SECTOR GAP WORD
69 042370 001413              BEQ    11$
70 042372 160237 037030              SUB    R2,ERWORD   ;IF NOT GET ERROR WORD NO.
71 042376 005037 001124              CLR    $GDDAT     ;GOOD WORD
72 042402 013737 040054 001126      MOV    WWORD,$BDDAT;BAD WORD
73 042410 012737 104006 042060      MOV    #104006,SEGP;STORE 'ERROR 6' IN SEGP
74 042416 000554              BR     17$        ;BRANCH OUT
75 042420 005302 11$:      DEC    R2          ;HAVE 18 WORDS OF ZEROS BEEN WRITTEN ?
76 042422 001353              BNE    10$        ;IF NOT CONTINUE
77
78 ;AT THIS POINT THE SECTOR FOUND FLOP SHOULD
79 ;BE HIGH, SO THAT THE HEADER SYNC BYTE CAN BE GIVEN.
80
81 ;HOWEVER, IN THE DRIVE TIMING ERROR TEST THE REST OF THE ROUTINE
82 ;IS ABORTED
83
84 042424 005737 001426              TST    TESDTE     ;IS THIS A DRIVE TIMING ERROR TEST ?
85 042430 001147              BNE    17$        ;BRANCH OUT IF YES ----->
86
87 042432 004737 040056              JSR    PC,WRITE    ;WRITE ONE SECTOR GAP 0 BYTE
88 ;AND ONE SYNC. BYTE = 230
89 042436 013711 040054              MOV    WWORD,(R1) ;SAVE 0 BYTE AND SYNC BYTE
90 042442 023721 037012              CMP    RSYNC,(R1)+ ;IF SYNC. BYTE RIGHT
91 042446 001414              BEQ    12$        ;IF YES CONTINUE OPERATION
92 042450 012737 000024 037030      MOV    #20.,ERWORD ;IF NOT GET READY FOR ERROR PRINT
93
94 042456 013737 037012 001124      MOV    RSYNC,$GDDAT;GOOD WORD
95 042464 014137 001126              MOV    -(R1),$BDDAT ;BAD WORD
96 042470 012737 104006 042062      MOV    #104006,FSYNER;INSERT 'ERROR 6' IN FSYNER
97 042476 000524              BR     17$        ;BRANCH OUT ----->
98
99 042500 012702 000004 12$:      MOV    #4,R2      ;FOUR HEADER WORDS
100 042504 012703 042130              MOV    #SCYL,R3   ;POINTER FOR HEADER TABLE
101 042510 012737 000005 037030 13$:     MOV    #5,ERWORD  ;ERROR WORD NO SET
102 042516 004737 040056              JSR    PC,WRITE    ;WRITE 4 HEADER WORDS
103 042522 013711 040054              MOV    WWORD,(R1) ;STORE WRITTEN WORD
104 042526 022321              CMP    (R3)+,(R1)+ ;IS IT CORRECT ?
105 042530 001412              BEQ    14$        ;IF GOOD CONTINUE OPERATION
106 ;IF NOT GET READY FOR ERROR PRINT
107
108 042532 160237 037030              SUB    R2,ERWORD  ;WORD NO
109 042536 014337 001124              MOV    -(R3),$GDDAT ;GOOD DATA
110 042542 014137 001126              MOV    -(R1),$BDDAT ;BAD DATA
111 042546 012737 104006 042064      MOV    #104006,ERHEAD;INSERT 'ERROR 6'
112 042554 000475              BR     17$        ;BRANCH OUT ----->
113
114 042556 005302 14$:      DEC    R2          ;ARE 4 HEADER WORDS DONE?
    
```

```

115 042560 001353          BNE      13$          ;IF NOT CONTINE
116 042562 004737 040056   JSR      PC,WRITE    ;WRITE CRC
117 042566 013711 040054   MOV      WWORD,(R1)  ;STORE CRC
118 042572 022137 042056   CMP      (R1)+,GCRC  ;COMPARE GOOD CRC
119 042576 001414          BEQ      20$          ;IF GOOD CONTINUE OPERATION
120                                ;IF NOT GET READY FOR ERROR PRINT
121
122 042600 014137 001126   MOV      -(R1),SBDDATA ;BAD CRC WRITTEN
123 042604 013737 042056 001124   MOV      GCRC,$GDDAT  ;GOOD CRC
124 042612 012737 000005 037030   MOV      #5,ERWORD    ;ERROR WORD NO
125 042620 012737 104006 042066   MOV      #104006,ERCRC ;INSERT ERROR 6
126 042626 000450          BR       17$          ;EXIT ----->
127
128 042630 012702 000005          MOV      #5,R2        ;NO OF HEADER GAP
129 042634 012737 000006 037030 15$:   MOV      #6,ERWORD    ;ERROR WORD NO SET
130 042642 004737 040056          JSR      PC,WRITE    ;WRITE HEADER GAP
131 042646 013721 040054          MOV      WWORD,(R1)+ ;STORE
132 042652 001412          BEQ      16$          ;IF GOOD CONTINUE OPERATION
133                                ;IF NOT GET READY FOR PRINT
134
135 042654 160237 037030          SUB      R2,ERWORD    ;ERROR WORD NO
136 042660 005037 001124          CLR      $GDDAT      ;GOOD DATA
137 042664 014137 001126          MOV      -(R1),SBDDAT ;BAD DATA
138 042670 012737 104006 042070   MOV      #104006,ERHDGP;STORE 'ERROR 6'
139 042676 000424          BR       17$          ;BRANCH OUT ----->
140
141 042700 005302          16$:   DEC      R2          ;ARE 5 HEADER GAP ZEROS DONE ?
142 042702 001354          BNE      15$          ;IF NOT CONTINUE
143 042704 004737 040056   JSR      PC,WRITE
144 042710 013711 040054   MOV      WWORD,(R1)
145 042714 023721 037012   CMP      RSYNC,(R1)+
146 042720 001413          BEQ      17$          ;A-OK, EXIT ----->
147
148 042722 012737 000005 037030   MOV      #5,ERWORD    ;IF NOT GET READY FOR ERROR PRINT
149 042730 014137 001126   MOV      -(R1),SBDDAT
150 042734 013737 037012 001124   MOV      RSYNC,$GDDAT
151 042742 012737 104006 042072   MOV      #104006,HDESYN
152
153 042750          17$:   MOV      (SP)+,R1     ;:POP STACK INTO R1
154 042752 012601 000201   RTS      R1          ;:EXIT ----->
    
```


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

```

:*****
:*SEARCH SECTOR
:*****
    
```

```

:*R0=RHMR ADDRESS
:*R1=PASSED ARGUMENT (SECTOR SEARCHED FOR)
:*R2=CLOCK COUNT (PER BYTE)
:*R3=SECTOR COUNTER FROM R1
:*R5=BYTES PER WORD COUNT
    
```

```

:*BEFORE INDEX IS GIVEN TWO SECTOR CLOCKS ARE GIVEN TO RESET
:*SECTOR PULSE IN CASE IT IS SET.
    
```

```

:*AT THE BEGINNING OF EACH SECTOR, ONE SECTOR CLOCK HAS TO RISE
:*BEFORE MAINT. CLOCK, THEN EVERY EIGHT MAINT.CLOCKS, ONE SECTOR CLOCK
:*IS IDENTICAL WITH THE MAINT. CLOCK
:*THE SECTOR CLOCKS ARE NUMBERED AS FOLLOWS:
    
```

```

:*THE SECTOR CLOCK UNDER INDEX - 0
:*THE NEXT - 1
:*THE NEXT - 2
:*ETC.
:*THEN THE LAST SECTOR CLOCK IN ONE SECTOR HAS NUMBER - 608
:*THE NEXT SECTOR WILL HAVE 608 SECTOR CLOCKS
:*THE NEXT SECTOR THEN HAS ANOTHER 608 SECTOR CLOCKS
:*AND SO ON
    
```

```

SECTR: 0 ;SECTOR SEARCHED FOR
    
```

```

SEARCH: MOV (R1)+,SECTR ;SAVE SECTOR SEARCHED FOR
        MOV R0,-(SP) ;:PUSH R0 ON STACK
        MOV R1,-(SP) ;:PUSH R1 ON STACK
        MOV R2,-(SP) ;:PUSH R2 ON STACK
        MOV R3,-(SP) ;:PUSH R3 ON STACK
        MOV R4,-(SP) ;:PUSH R4 ON STACK
        MOV R5,-(SP) ;:PUSH R5 ON STACK
        MOV RHMR,R0 ;NOW R0 HAS MAINTENANCE REG. ADR.
        MOV SECTR,R3 ;LOAD SECTOR COUNTER
    
```

```

        MOV #DMD,@R0 ;SET DIAGNOSTIC MODE
        BIS #MSTCK,@R0 ;SET SECTOR CLOCK
        BIC #MSTCK,@R0 ;CLEAR SECTOR CLOCK
        BIS #MSTCK,@R0 ;SET SECTOR CLOCK
        BIC #MSTCK,@R0 ;CLEAR SECTOR CLOCK
        ;THE ABOVE TWO SECTOR CLOCKS ARE GIVEN FOR
        ;RESETTING SECTOR PULSE
        ;IN CASE IT STARTS SET
        BIS #MINX!MSTCK,@R0 ;SET INDEX AND SECTOR CLOCK
        BIC #MINX!MSTCK,@R0 ;RESET INDEX AND SECTOR CLOCK
        TST R3 ;TEST FOR SECTOR 0
        BEQ 7$ ;IF SECTOR 0 IS REQUIRED, EXIT ----->
    
```

```

;NOW 304 WORDS WILL START (608 BYTES)
    
```

```

:*FOR FIRST BYTE MAINT. SECTOR CLOCK WILL GO HIGH, THEN MAINT. CLOCK WILL GO HIGH
:*BOTH WILL COME DOWN TOGETHER, THEN SEVEN MAINT. CLOCKS WILL BE GIVEN
:*FOR SECOND BYTE, AND ALL OTHER BYTES TILL NEXT SECTOR, SECTOR CLOCK
    
```

```
53                                     ;*WILL BE IDENTICAL WITH ONE MAINT. CLOCK
54                                     ;ONE WORD ONLY
55
56 043046 012702 000010                1$:   MOV     #8.,R2      ;BYTE COUNTER
57 043052 012705 000002                MOV     #2.,R5      ;BYTES PER WORD
58 043056 052710 000010                BIS     #MSTCK,@RO  ;SET SECTOR CLOCK
59 043062 052710 000002                BIS     #MCLK,@RO   ;SET MAINT. CLOCK
60 043066 000402                        BR      3$          ;BRANCH TO CLEAR SECTOR AND CLOCK
61 043070 052710 000012                2$:   BIS     #MSTCK!MCLK,@RO ;SET BOTH CLOCKS
62 043074 042710 000012                3$:   BIC     #MSTCK!MCLK,@RO ;CLEAR BOTH CLOCKS
63 043100 052710 000002                8$:   BIS     #MCLK,@RO   ;SET MAINT. CLOCK
64 043104 042710 000002                BIC     #MCLK,@RO   ;CLEAR MAINT.CLOCK
65 043110 005302                        DEC     R2          ;BYTE COUNTER
66 043112 001372                        BNE     8$          ;CONTINUE IF BYTE NOT COMPLETE
67
68 043114 012702 000007                MOV     #7.,R2      ;SETUP FOR SECOND BYTE
69 043120 005305                        DEC     R5          ;IS WORD COMPLETE?
70 043122 001362                        BNE     2$          ;CONTINUE IF NOT COMPLETE
71                                     ;TO GIVE SECTOR CLOCK AND MAINT. CLOCK
72
73
74
75
```

;NOW 303 WORDS ARE LEFT AND ALL ARE IDENTICAL

```
76 043124 012701 000457                MOV     #303.,R1    ;WORDS PER SECTOR COUNTER
77 043130 012705 000002                4$:   MOV     #2.,R5      ;BYTES PER WORD COUNTER
78 043134 012702 000007                5$:   MOV     #7.,R2      ;BIT COUNTER (MAINT.CLOCK COUNTER)
79 043140 052710 000012                BIS     #MSTCK!MCLK,@RO ;SET SECTOR CLOCK AND MAINT. CLOCK
80 043144 042710 000012                BIC     #MSTCK!MCLK,@RO ;CLEAR CLOCKS
81 043150 052710 000002                6$:   BIS     #MCLK,@RO   ;SET MAINT. CLOCK
82 043154 042710 000002                BIC     #MCLK,@RO   ;RESET MAINT. CLOCK
83 043160 005302                        DEC     R2          ;IS BYTE COMPLETE ?
84 043162 001372                        BNE     6$          ;CONTINUE IF NOT COMPLETE
85
86 043164 005305                        DEC     R5          ;IS WORD COMPLETE ?
87 043166 001362                        BNE     5$          ;CONTINUE IF NOT
88
89 043170 005301                        DEC     R1          ;IS SECTOR COMPLETE ?
90 043172 001356                        BNE     4$          ;CONTINUE IF NOT
91
92 043174 052710 000010                BIS     #MSTCK,@RO  ;SET SECTOR CLOCK 1 MORE TIME (FOR 609)
93 043200 042710 000010                BIC     #MSTCK,@RO  ;CLEAR SECTOR CLOCK
94 043204 005303                        DEC     R3          ;IS REQUIRED NO OF SECTORS COMPLETE ?
95 043206 001317                        BNE     1$          ;CONTINUE IF NOT
96
97 043210                                7$:   MOV     (SP)+,R5    ;:POP STACK INTO R5
    043210 012605                        MOV     (SP)+,R4    ;:POP STACK INTO R4
    043212 012604                        MOV     (SP)+,R3    ;:POP STACK INTO R3
    043214 012603                        MOV     (SP)+,R2    ;:POP STACK INTO R2
    043216 012602                        MOV     (SP)+,R1    ;:POP STACK INTO R1
    043220 012601                        MOV     (SP)+,R0    ;:POP STACK INTO R0
    043222 012600                        RTS     R1
98 043224 000201
```

```
::*****
:*READ ONE SECTOR OF DATA
:*****
```

100
101
102
103

1

.SBTTL SCOPE HANDLER ROUTINE

```

:*****
:*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
:*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
:*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
:*SW14=1      LOOP ON TEST
:*SW11=1      INHIBIT ITERATIONS
:*SW09=1      LOOP ON ERROR
:*SW08=1      LOOP ON TEST IN SWR<7:0>
:*CALL
:*          SCOPE          ;;SCOPE=IOT
    
```

```

043422          $SCOPE:
043422 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
043424 005037 037024  CLR          NOSYNC      ;CLEAR FLAG FOR HEADER ERROR COMMANDS
043430 005037 001424  CLR          TSECC        ;CLEAR FLAG FOR ECC TEST
                                          ;WHEN =177777 IT IS AN ECC TEST
                                          ;WHEN =0 IT IS NOT AN ECC TEST

043434 005037 034562          CLR          TSECCG      ;EVEN IN AN ECC TEST EVERY CLOCK
                                          ;IS NOT TO GENERATE ECC
                                          ;IF =177777 GENERATE ECC
                                          ;IF =0 DO NOT GENERATE ECC
                                          ;DRIVE TIMING ERROR TEST

043440 005037 001426          CLR          TESDTE
043444          1$:
043444 032777 040000 135466 1$:      BIT          #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
043452 001402          BEQ          9$          ;;NO IF SW14=0
043454 000137 044044          JMP          $OVER        ;;JUMP OVER SCOPE ROUTINE
043460          9$:
:#####START OF CODE FOR THE XOR TESTER#####
043460 000416          $XTSTR: BR          6$
043462 013746 000004          MOV          @#ERRVEC,-(SP)      ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
043466 012737 043506 000004          MOV          #5$,@#ERRVEC      ;;THIS INSTRUCTION TO A 'NOP' (NOP=240)
043474 005737 177060          TST          @#177060      ;;SAVE THE CONTENTS OF THE ERROR VECTOR
043500 012637 000004          MOV          (SP)+,@#ERRVEC      ;;SET FOR TIMEOUT
043504 000544          BR          $$VLAD      ;;TIME OUT ON XOR?
043506 022626          5$:      CMP          (SP)+,(SP)+      ;;RESTORE THE ERROR VECTOR
043510 012637 000004          MOV          (SP)+,@#ERRVEC      ;;GO TO THE NEXT TEST
043514 000504          BR          7$          ;;CLEAR THE STACK AFTER A TIME OUT
043516          6$:;#####END OF CODE FOR THE XOR TESTER#####
043516 032777 000400 135414          BIT          #BIT08,@SWR      ;;RESTORE THE ERROR VECTOR
043524 001404          BEQ          2$          ;;LOOP ON SPEC. TEST?
043526 127737 135406 001102          CMPB         @SWR,$TSTNM      ;;BR IF NO
043534 001543          BEG          $OVER        ;;ON THE RIGHT TEST? SWR<7:0>
043536 105737 001103          2$:      TSTB         $ERFLG      ;;BR IF YES
043542 001502          BEQ          3$          ;;HAS AN ERROR OCCURRED?
043544 022737 177777 046430          CMP          #-1,CPSAVE      ;;BR IF NO
043552 001455          BEQ          2003$      ;;SEE IF TIMEOUT WAS PREVIOUSLY RECORDED
043554 013746 000004          MOV          ERRVEC,-(SP)      ;;KICK AROUND ROUTINE IF SO
043560 012737 043576 000004          MOV          #2000$,ERRVEC      ;;SAVE CONTENTS OF ERROR VECTOR
043566 013737 177766 046430          MOV          177766,CPSAVE      ;;SETUP 'TRAP' RETURN ADDRESS
043574 000406          BR          2001$      ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
043576 012737 177777 046430 2000$:  MOV          #-1,CPSAVE      ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
043604 012716 043612          MOV          #2001$,(SP)      ;;SETUP RETURN ADDRESS
    
```



```

043610 000002 RTI
043612 012637 000004 2001$: MOV (SP)+,ERRVEC ;;RESTORE CONTENTS OF ERROR VECTOR

043616 022737 177777 046430 2002$: CMP #-1,CPSAVE ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
043624 001430 BEQ 2003$ ;;BRANCH IF SO
043626 032737 000001 046430 BIT #BIT00,CPSAVE ;;SEE IF THE POWER MONITOR BIT IS ON
043634 001424 BEQ 2003$ ;;BRANCH TO CONTINUE ROUTINE IF CLEAR
043636 042737 000001 177766 BIC #BIT00,177766 ;;CLEAR THE BIT FOUND TO BE SET
043644 013746 001140 MOV SWR,-(SP) ;;SAVE SWR ADDRESS
043650 017646 000000 MOV @ (SP),-(SP) ;;SAVE SWR VALUE
043654 012737 000176 001140 MOV #176,SWR ;;GET SOFTWARE SWR ADDRESS
043662 011677 135252 MOV (SP),@SWR ;;GET CURRENT SWR VALUE
043666 042777 001000 135244 BIC #BIT09,@SWR ;;DON'T ALLOW LOOP ON ERROR ON THIS ERROR
043674 104177 EMT 177 ;;CALL SPECIAL POWER FAIL BIT ERROR CALL
043676 012676 000C00 MOV (SP)+,@(SP) ;;RESTORE SWR TO ORIGINAL VALUE
043702 012637 001140 MOV (SP)+,SWR ;;RESTORE SWR ADDRESS
043706 2003$:
043706 123737 001115 001103 CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
043714 101015 BHI 3$ ;;BR IF NO
043716 032777 001000 135214 BIT #BIT09,@SWR ;;LOOP ON ERROR?
043724 001404 BEQ 4$ ;;BR IF NO
043726 013737 001110 001106 7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
043734 000443 BR $OVER
043736 105037 001103 4$: CLRB $ERFLG ;;ZERO THE ERROR FLAG
043742 005037 001212 CLR $TIMES ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
043746 000415 BR 1$ ;;ESCAPE TO THE NEXT TEST
043750 032777 004000 135162 3$: BIT #BIT11,@SWR ;;INHIBIT ITERATIONS?
043756 001011 BNE 1$ ;;BR IF YES
043760 005737 001100 TST $PASS ;;IF FIRST PASS OF PROGRAM
043764 001406 BEQ 1$ ;; INHIBIT ITERATIONS
043766 005237 001104 INC $ICNT ;;INCREMENT ITERATION COUNT
043772 023737 001212 001104 CMP $TIMES,$ICNT ;;CHECK THE NUMBER OF ITERATIONS MADE
044000 002021 BGE $OVER ;;BR IF MORE ITERATION REQUIRED
044002 012737 000001 001104 1$: MOV #1,$ICNT ;;REINITIALIZE THE ITERATION COUNTER
044010 013737 044060 001212 MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
044016 105237 001102 $SVLAD: INCB $STNM ;;COUNT TEST NUMBERS
044022 011637 001106 MOV (SP),$LPADR ;;SAVE SCOPE LOOP ADDRESS
044026 011637 001110 MOV (SP),$LPERR ;;SAVE ERROR LOOP ADDRESS
044032 005037 001214 CLR $ESCAPE ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
044036 112737 000001 001115 MOVB #1,$ERMAX ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
044044 013777 001102 135070 $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
044052 013716 001106 MOV $LPADR,(SP) ;;FUDGE RETURN ADDRESS
044056 000002 RTI ;;FIXES PS
044060 000004 $MXCNT: 4 ;;MAX. NUMBER OF ITERATIONS
    
```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

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

```

```

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

```

```

044062
044062 010046
044064 010146
044066 010246
044070 010346
044072 010546
044074 012746 020200
044100 016605 000020
044104 100004
044106 005405
044110 112766 000055 000001
044116 005000
044120 012703 044276
044124 112723 000040
044130 005002
044132 016001 044266
044136 160105
044140 002402
044142 005202
044144 000774
044146 060105
044150 005702
044152 001002
044154 105716
044156 100407
044160 106316
044162 103003
044164 116663 000001 177777
044172 052702 000060
044176 052702 000040
044202 110223
044204 005720
044206 020027 000010
044212 002746
044214 003002
044216 010502
044220 000764
044222 105726
044224 100003
044226 116663 177777 177776
044234 105013
044236 012605
044240 012603
044242 012602
044244 012601

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5     ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB     #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
CLR      R0            ;;ZERO THE CONSTANTS INDEX
MOV      #SDBLK,R3    ;;SETUP THE OUTPUT POINTER
MOVB     #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2    ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
3$:      SUB      R1,R5  ;;FORM THIS BCD DIGIT
BLT      4$            ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      AD      R1,R5  ;;ADD BACK THE CONSTANT
TST      R2            ;;CHECK IF BCD DIGIT=0
BNE      5$            ;;FALL THROUGH IF 0
TSTB     (SP)          ;;STILL DOING LEADING 0'S?
BMI      7$            ;;BR IF YES
5$:      ASLB     (SP)  ;;MSD?
BCC      6$            ;;BR IF NO
MOVB     1(SP),-1(R3)  ;;YES--SET THE SIGN
6$:      BIS      #'0,R2 ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2 ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB     R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST      (R0)+        ;;JUST INCREMENTING
CMP      R0,#10      ;;CHECK THE TABLE INDEX
BLT      2$            ;;GO DO THE NEXT DIGIT
BGT      8$            ;;GO TO EXIT
MOV      R5,R2        ;;GET THE LSD
BR       6$            ;;GO CHANGE TO ASCII
8$:      TSTB     (SP)+  ;;WAS THE LSD THE FIRST NON-ZERO?
BPL      9$            ;;BR IF NO
MOVB     -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
9$:      CLRB     (R3)  ;;SET THE TERMINATOR
MOV      (SP)+,R5     ;;POP STACK INTO R5
MOV      (SP)+,R3     ;;POP STACK INTO R3
MOV      (SP)+,R2     ;;POP STACK INTO R2
MOV      (SP)+,R1     ;;POP STACK INTO R1

```


044246	012600			MOV	(SP)+,R0	::POP STACK INTO R0
044250	104401	044276		TYPE	\$DBLK	::NOW TYPE THE NUMBER
044254	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
044262	012616			MOV	(SP)+,(SP)	
044264	000002			RTI		::RETURN TO USER
044266	023420			\$DTBL:	10000.	
044270	001750				1000.	
044272	000144				100.	
044274	000012				10.	
044276				\$DBLK:	.BLKW 4	

TYPE ROUTINE

.SBTTL TYPE ROUTINE

```

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

```

```

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

```

```

044306 105737 001157 $TYPE: TSTB $TPFLG ;;IS THERE A TERMINAL?
044312 100002 BPL 1$ ;;BR IF YES
044314 000000 HALT ;;HALT HERE IF NO TERMINAL
044316 000407 BR 3$ ;;LEAVE
044320 010046 1$: MOV R0,-(SP) ;;SAVE R0
044322 017600 000002 MOV @2(SP),R0 ;;GET ADDRESS OF ASCIZ STRING
044326 112046 2$: MOVB (R0)+,-(SP) ;;PUSH CHARACTER TO BE TYPED ONTO STACK
044330 001005 BNE 4$ ;;BR IF IT ISN'T THE TERMINATOR
044332 005726 TST (SP)+ ;;IF TERMINATOR POP IT OFF THE STACK
044334 012600 60$: MOV (SP)+,R0 ;;RESTORE R0
044336 062716 000002 3$: ADD #2,(SP) ;;ADJUST RETURN PC
044342 000002 RTI ;;RETURN
044344 122716 000011 4$: CMPB #HT,(SP) ;;BRANCH IF <HT>
044350 001430 BEQ 8$
044352 122716 000200 CMPB #CRLF,(SP) ;;BRANCH IF NOT <CRLF>
044356 001006 BNE 5$
044360 005726 TST (SP)+ ;;POP <CR><LF> EQUIV
044362 104401 TYPE ;;TYPE A CR AND LF
044364 001223 $CRLF
044366 105037 044574 CLRB $CHARCNT ;;CLEAR CHARACTER COUNT
044372 000755 BR 2$ ;;GET NEXT CHARACTER
044374 004737 044456 5$: JSR PC,$TYPEC ;;GO TYPE THIS CHARACTER
044400 123726 001156 6$: CMPB $FILLC,(SP)+ ;;IS IT TIME FOR FILLER CHARS.?
044404 001350 BNE 2$ ;;IF NO GO GET NEXT CHAR.
044406 013746 001154 MOV $NULL,-(SP) ;;GET # OF FILLER CHARS. NEEDED
;;AND THE NULL CHAR.
044412 105366 000001 7$: DECB 1(SP) ;;DOES A NULL NEED TO BE TYPED?
044416 002770 BLT 6$ ;;BR IF NO--GO POP THE NULL OFF OF STACK
044420 004737 044456 JSR PC,$TYPEC ;;GO TYPE A NULL
044424 105337 044574 DECB $CHARCNT ;;DO NOT COUNT AS A COUNT
044430 000770 BR 7$ ;;LOOP

```

;HORIZONTAL TAB PROCESSOR

```

044432 112716 000040 8$: MOVB #' ,(SP) ;;REPLACE TAB WITH SPACE
044436 004737 044456 9$: JSR PC,$TYPEC ;;TYPE A SPACE
044442 132737 000007 044574 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
044450 001372 BNE 9$ ;;TAB STOP
044452 005726 TST (SP)+ ;;POP SPACE OFF STACK
044454 000724 BR 2$ ;;GET NEXT CHARACTER

```


044456				\$TYPEC:	TSTB	@\$TKS	::CHAR IN KYBD BUFFER?
044456	105777	134462			BPL	10\$::BR IF NOT
044462	100022				MOV	@\$TKB,-(SP)	::GET CHAR
044464	017746	134456			BIC	#177600,(SP)	::STRIP EXTRANEIOUS BITS
044470	042716	177600			CMPB	#\$XOFF,(SP)	::WAS CHAR XOFF
044474	122716	000023			BNE	102\$::BR IF NOT
044500	001012						
044502				101\$:	TSTB	@\$TKS	::WAIT FOR CHAR
044502	105777	134436			BPL	101\$	
044506	100375				MOVB	@\$TKB,(SP)	::GET CHAR
044510	117716	134432			BIC	#177600,(SP)	::STRIP IT
044514	042716	177600			CMPB	#\$XON,(SP)	::WAS IT XON?
044520	122716	000021			BNE	101\$::BR IF NOT
044524	001366						
044526				102\$:	TST	(SP)+	::FIX STACK
044526	005726						
044530				10\$:	TSTB	@\$TPS	::WAIT UNTIL PRINTER IS READY
044530	105777	134414			BPL	10\$	
044534	100375				MOVB	2(SP),@\$TPB	::LOAD CHAR TO BE TYPED INTO DATA REG.
044536	116677	000002	134406		CMPB	#CR,2(SP)	::IS CHARACTER A CARRIAGE RETURN?
044544	122766	000015	000002		BNE	1\$::BRANCH IF NO
044552	001003				CLRB	\$CHARCNT	::YES--CLEAR CHARACTER COUNT
044554	105037	044574			BR	\$TYPEX	::EXIT
044560	000406				CMPB	#LF,2(SP)	::IS CHARACTER A LINE FEED?
044562	122766	000012	000002	1\$:	BEQ	\$TYPEX	::BRANCH IF YES
044570	001402				INCB	(PC)+	::COUNT THE CHARACTER
044572	105227						::CHARACTER COUNT STORAGE
044574	000000				\$CHARCNT:	.WORD	
044576	000207				\$TYPEX:	RTS	
						PC	

.SBTTL TTY INPUT ROUTINE

```

:*****
. ENABL LSB
044600 000000 $TKCNT: .WORD 0          ;;NUMBER OF ITEMS IN QUEUE
044602 000000 $TKQIN: .WORD 0          ;;INPUT POINTER
044604 000000 $TKQOUT: .WORD 0         ;;OUTPUT POINTER
044606      044617 $TKQSRT: .BLKB 9.         ;;TTY KEYBOARD QUEUE
$TKQEND=.
.EVEN

;*TK INITIALIZE ROUTINE
;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
:
;*CALL:
:
:      JSR      PC,$TKINT
:      RETURN
:
044620 005037 044600 $TKINT: CLR      $TKCNT          ;;CLEAR COUNT OF ITEMS IN QUEUE
044624 012737 044606 044602 MOV      # $TKQSRT,$TKQIN      ;;MOVE THE STARTING ADDRESS OF THE
044632 013737 044602 044604 MOV      $TKQIN,$TKQOUT        ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
044640 012737 044670 000060 MOV      # $TKSRV,@TKVEC      ;;INITIALIZE THE KEYBOARD VECTOR
044646 012737 000200 000062 MOV      #200,@TKVEC+2        ;;'BR' LEVEL 4
044654 005777 134266 TST      @ $TKB              ;;CLEAR DONE FLAG
044660 012777 000100 134256 MOV      #100,@ $TKS          ;;ENABLE TTY KEYBOARD INTERRUPT
044666 000207      RTS      PC              ;;RETURN TO CALLER

;*TK SERVICE ROUTINE
;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
;*IT IN THE QUEUE.
;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (OPERSEL)
:
$TKSRV: MOVB     @ $TKB,-(SP)        ;;PICKUP THE CHARACTER
      BIC      #^C177,(SP)        ;;STRIP THE JUNK
      CMP      (SP),#$XON         ;;IS IT A RANDOM XON?
      BNE     30$                 ;;BRANCH IF NO
      TST     (SP)+               ;;CLEAN RANDOM XON OFF STACK
      RTI
30$:
      CMP     (SP),#3             ;;IS IT A CONTROL C?
      BNE     1$                 ;;BRANCH IF NO
      TYPE   ,SCNTLC             ;;TYPE A CONTROL-C (^C)
      JSR    PC,$TKINT           ;;INIT THE KEYBOARD
      TST   (SP)+               ;;CLEAN UP STACK
      JMP   OPERSEL             ;;CONTROL C RESTART
1$:
      CMP     (SP),#7             ;;IS IT A CONTROL G?
      BNE     2$                 ;;BRANCH IF NO
      CMP     #SWREG,SWR         ;;IS SOFT-SWR SELECTED?
      BEQ     6$                 ;;GO TO SWR CHANGE
2$:
      CMP     #9.,$TKCNT         ;;IS THE QUEUE FULL?
      BNE     3$                 ;;BRANCH IF NO
      TYPE   ,SBELL              ;;RING THE TTY BELL
044670 117746 134252 $TKSRV: MOVB     @ $TKB,-(SP)
044674 042716 177600 BIC      #^C177,(SP)
044700 021627 000021 CMP      (SP),#$XON
044704 001002 BNE     30$
044706 005726 TST     (SP)+
044710 000002 RTI
044712
044712 021627 000003 30$: CMP     (SP),#3
044716 001007 BNE     1$
044720 104401 045671 TYPE   ,SCNTLC
044724 004737 044620 JSR    PC,$TKINT
044730 005726 TST   (SP)+
044732 000137 031004 JMP   OPERSEL
044736 021627 000007 1$: CMP     (SP),#7
044742 001004 BNE     2$
044744 022737 000176 001140 CMP     #SWREG,SWR
044752 001500 BEQ     6$
044754
044754 022737 000011 044600 2$: CMP     #9.,$TKCNT
044762 001004 BNE     3$
044764 104401 001216 TYPE   ,SBELL
  
```



```

044770 005726          TST      (SP)+      ;;CLEAN CHARACTER OFF OF STACK
044772 000451          BR        5$          ;;EXIT
044774 021627 000023  3$:    CMP      (SP),#23      ;;IS IT A CONTROL-S?
045000 001021          BNE      32$          ;;BRANCH IF NO
045002 005077 134136  CLR      @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
045006 005726          TST      (SP)+      ;;CLEAN CHAR OFF STACK
045010 105777 134130  31$:  TSTB     @STKS        ;;WAIT FOR A CHAR
045014 100375          BPL      31$          ;;LOOP UNTIL ITS THERE
045016 117746 134124  MOVB     @STKB,-(SP)  ;;GET THE CHARACTER
045022 042716 177600  BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
045026 022627 000021  CMP      (SP)+,#21   ;;IS IT A CONTROL-Q?
045032 001366          BNE      31$          ;;BRANCH IF NO
045034 012777 000100 134102 MOV      #100,@STKS  ;;REENABLE TTY KEYBOARD INTERRUPTS
045042 000002          RTI                    ;;RETURN
045044 005237 044600  32$:  INC      $TKCNT      ;;COUNT THIS CHARACTER
045050 021627 000140  CMP      (SP),#140   ;;IS IT UPPER CASE?
045054 002405          BLT      4$          ;;BRANCH IF YES
045056 021627 000175  CMP      (SP),#175   ;;IS IT A SPECIAL CHAR?
045062 003002          BGT      4$          ;;BRANCH IF YES
045064 042716 000040  BIC      #40,(SP)    ;;MAKE IT UPPER CASE
045070 112677 177506  4$:    MOVB     (SP)+,@STKQIN ;;AND PUT IT IN QUEUE
045074 005237 044602  INC      $TKQIN      ;;UPDATE THE POINTER
045100 023727 044602 044617 CMP      $TKQIN,$STKQEND ;;GO OFF THE END?
045106 001003          BNE      5$          ;;BRANCH IF NO
045110 012737 044606 044602 MOV      #$STKQSRT,$STKQIN ;;RESET THE POINTER
045116 000002          RTI                    ;;RETURN
  
```

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

```

045120 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR    ;;IS THE SOFT-SWR SELECTED
045126 001124          BNE      15$          ;;EXIT IF NOT
045130 105777 134010  TSTB     @STKS        ;;IS A CHAR WAITING?
045134 100121          BPL      15$          ;;IF NOT, EXIT
045136 117746 134004  MOVB     @STKB,-(SP)  ;;YES
045142 042716 177600  BIC      #^C177,(SP) ;;MAKE IT 7-BIT ASCII
045146 021627 000007  CMP      (SP),#?     ;;IS IT A CONTROL-G?
045152 001300          BNE      2$          ;;IF NOT, PUT IT IN THE TTY QUEUE
                          ;;AND EXIT
  
```

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

```

045154 123727 001134 000001 6$:    CMPB     $AUTOB,#1    ;;ARE WE RUNNING IN AUTO-MODE?
045162 001674          BEQ      2$          ;;BRANCH IF YES
045164 005726          TST      (SP)+      ;;CLEAR CONTROL-G OFF STACK
045166 004737 044620  JSR      PC,$TKINT   ;;FLUSH THE TTY INPUT QUEUE
045172 005077 133746  CLR      @STKS        ;;DISABLE TTY KEYBOARD INTERRUPTS
045176 112737 000001 001135 MOVB     #1,$INTAG    ;;SET INTERRUPT MODE INDICATOR

045204 104401 045703          TYPE     ,SCNTLG     ;;ECHO THE CONTROL-G (^G)
045210 104401 045710  $GTSWR: TYPE     ,SMSWR   ;;TYPE CURRENT CONTENTS
045214 013746 000176  MOV      SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT
045220 104402          TYPOC          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
  
```

```
045222 104401 045721          TYPE      ,SMNEW      ;;PROMPT FOR NEW SWR
045226 005046          CLR      -(SP)      ;;CLEAR COUNTER
045230 005046          CLR      -(SP)      ;;THE NEW SWR
045232 105777 133706      7$:      TSTB     @STKS     ;;CHAR THERE?
045236 100375          BPL      7$         ;;IF NOT TRY AGAIN

045240 117746 133702      MOVB     @STKB, -(SP)  ;;PICK UP CHAR
045244 042716 177600      BIC     #^C177, (SP) ;;MAKE IT 7-BIT ASCII

045250 021627 000003      CMP     (SP), #3     ;;IS IT A CONTROL-C?
045254 001015          BNE     9$          ;;BRANCH IF NOT
045256 104401 045671      TYPE     ,SCNTLC    ;;YES, ECHO CONTROL-C (^C)
045262 062706 000006      ADD     #6, SP      ;;CLEAN UP STACK
045266 123727 001135 000001  CMPB    $INTAG, #1   ;;REENABLE TTY KEYBOARD INTERRUPTS?
045274 001003          BNE     8$          ;;BRANCH IF NO
045276 012777 000100 133640  MOV     #100, @STKS  ;;ALLOW TTY KEYBOARD INTERRUPTS
045304 000137 031004      8$:      JMP     OPERSEL    ;;CONTROL-C RESTART

045310 021627 000025      9$:      CMP     (SP), #25   ;;IS IT A CONTROL-U?
045314 001005          BNE     10$         ;;BRANCH IF NOT
045316 104401 045676      TYPE     ,SCNTLU    ;;YES, ECHO CONTROL-U (^U)
045322 062706 000006      20$:     ADD     #6, SP      ;;IGNORE PREVIOUS INPUT
045326 000737          BR      19$         ;;LET'S TRY IT AGAIN

045330 021627 000015      10$:     CMP     (SP), #15   ;;IS IT A <CR>?
045334 001022          BNE     16$         ;;BRANCH IF NO
045336 005766 000004      TST     4(SP)       ;;YES, IS IT THE FIRST CHAR?
045342 001403          BEQ     11$         ;;BRANCH IF YES
045344 016677 000002 133566  MOV     2(SP), @SWR  ;;SAVE NEW SWR
045352 062706 000006      11$:     ADD     #6, SP      ;;CLEAN UP STACK
045356 104401 001223      14$:     TYPE     ,SCRLF  ;;ECHO <CR> AND <LF>
045362 123727 001135 000001  CMPB    $INTAG, #1   ;;RE-ENABLE TTY KBD INTERRUPTS?
045370 001003          BNE     15$         ;;BRANCH IF NOT
045372 012777 000100 133544  MOV     #100, @STKS  ;;RE-ENABLE TTY KBD INTERRUPTS
045400 000002          RTI                    ;;RETURN
045402 004737 044456      15$:     JSR     PC, $TYPEC ;;ECHO CHAR
045406 021627 000060      16$:     CMP     (SP), #60 ;;CHAR < 0?
045412 002420          BLT     18$         ;;BRANCH IF YES
045414 021627 000067      CMP     (SP), #67   ;;CHAR > 7?
045420 003015          BGT     18$         ;;BRANCH IF YES
045422 042726 000060      BIC     #60, (SP)+  ;;STRIP-OFF ASCII
045426 005766 000002      TST     2(SP)       ;;IS THIS THE FIRST CHAR
045432 001403          BEQ     17$         ;;BRANCH IF YES
045434 006316          ASL     (SP)        ;;NO, SHIFT PRESENT
045436 006316          ASL     (SP)        ;;CHAR OVER TO MAKE
045440 006316          ASL     (SP)        ;;ROOM FOR NEW ONE.
045442 005266 000002      17$:     INC     2(SP)    ;;KEEP COUNT OF CHAR
045446 056616 177776      BIS     -2(SP), (SP) ;;SET IN NEW CHAR
045452 000667          BR      7$         ;;GET THE NEXT ONE
045454 104401 001222      18$:     TYPE     ,SQUES  ;;TYPE ?<CR><LF>
045460 000720          BR      20$        ;;SIMULATE CONTROL-U

.DSABL  LSB
```

```

    ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
    ;*CALL:
    ;*      RDCHR          ;;GET A CHARACTER FROM THE QUEUE
    ;*      RETURN HERE   ;;CHARACTER IS ON THE STACK
    ;*                   ;;WITH PARITY BIT STRIPPED OFF
    ;
045462 011646                $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC AND
045464 016666 000004 000002  MOV      4(SP),2(SP)    ;;THE PS
045472 005066 000004                CLR      4(SP)          ;;GET READY FOR A CHARACTER
045476 005046                CLR      -(SP)           ;;PUT NEW PS ON STACK
045500 012746 045506                MOV      #64$,-(SP)      ;;PUT NEW PC ON STACK
045504 000002                RTI                          ;;POP NEW PC AND PS
045506
045506 005737 044600          64$:  TST      $TKCNT          ;;WAIT ON A CHARACTER
045512 001775          1$:  BEQ      1$
045514 005337 044600          DEC      $TKCNT          ;;DECREMENT THE COUNTER
045520 117766 177060 000004  MOVB   @ $TKQOUT,4(SP)  ;;GET ONE CHARACTER
045526 005237 044604          INC      $TKQOUT          ;;UPDATE THE POINTER
045532 023727 044604 044617  CMP    $TKQOUT,$TKQEND ;;DID IT GO OFF OF THE END?
045540 001003          BNE      2$
045542 012737 044606 044604  MOV    # $TKQSRT,$TKQOUT ;;RESET THE POINTER
045550 000002          2$:  RTI                          ;;RETURN
    ;*****
    ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
    ;*CALL:
    ;*      RDLIN         ;;INPUT A STRING FROM THE TTY
    ;*      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
    ;*                   ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
    ;
045552 010346                $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
045554 012703 045660          1$:  MOV      # $TTYIN,R3      ;;GET ADDRESS
045560 022703 045671          2$:  CMP      # $TTYIN+9.,R3    ;;BUFFER FULL?
045564 101405                BLOS    4$
045566 104410                RDCHR          ;;GO READ ONE CHARACTER FROM THE TTY
045570 112613                MOVB   (SP)+,(R3)      ;;GET CHARACTER
045572 122713 000177          10$: CMPB   #177,(R3)    ;;IS IT A RUBOUT
045576 001003          BNE      3$
045600 104401 001222          4$:  TYPE    $QUES          ;;TYPE A '?'
045604 000763          BR      1$
045606 111337 045656          3$:  MOVB   (R3),9$      ;;ECHO THE CHARACTER
045612 104401 045656          TYPE    ,9$
045616 122723 000015          CMPB   #15,(R3)+    ;;CHECK FOR RETURN
045622 001356          BNE      2$
045624 105063 177777          CLRB   -1(R3)      ;;CLEAR RETURN (THE 15)
045630 104401 001224          TYPE    ,9LF      ;;TYPE A LINE FEED
045634 012603                MOV    (SP)+,R3      ;;RESTORE R3
045636 011646                MOV    (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
045640 016666 000004 000002  MOV    4(SP),2(SP)    ;;FIRST ASCII CHARACTER ON IT
045646 012766 045660 000004  MOV    # $TTYIN,4(SP)
045654 000002                RTI
045656 000          9$:  .BYTE   0
045657 000          .BYTE   0
045660                $TTYIN: .BLKB   9
045671 136 103 015 $CNTLC: .ASCIZ /^C/<15><12> ;;CONTROL 'C'
045676 136 125 015 $CNTLU: .ASCIZ /^U/<15><12> ;;CONTROL 'U'
045703 136 107 015 $CNTLG: .ASCIZ /^G/<15><12> ;;CONTROL 'G'
  
```

045710	015	012	123	\$MSWR:	.ASCIZ	<15><12>/SWR = /
045721	040	040	116	\$MNEW:	.ASCIZ	/ NEW = /

.SBTTL READ AN OCTAL NUMBER FROM THE TTY

```

:*****
:*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
:*CHANGE IT TO BINARY.
:*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
:*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
:*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
:*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
:*CALL:
:*      RDOCT          ::READ AN OCTAL NUMBER
:*      RETURN HERE   ::LOW ORDER BITS ARE ON TOP OF THE STACK
:*                  ::HIGH ORDER BITS ARE IN $HIOCT
  
```

045732	011646			\$RDOCT: MOV	(SP),-(SP)	::PROVIDE SPACE FOR THE
045734	016666	000004	000002	MOV	4(SP),2(SP)	::INPUT NUMBER
045742	010046			MOV	R0,-(SP)	::PUSH R0 ON STACK
045744	010146			MOV	R1,-(SP)	::PUSH R1 ON STACK
045746	010246			MOV	R2,-(SP)	::PUSH R2 ON STACK
045750	104411			1\$: RDLIN		::READ AN ASCII LINE
045752	012600			MOV	(SP)+,R0	::GET ADDRESS OF 1ST CHARACTER
045754	010037	046060		MOV	R0,5\$::AND SAVE IT
045760	005001			CLR	R1	::CLEAR DATA WORD
045762	005002			CLR	R2	
045764	112046			2\$: MOVB	(R0)+,-(SP)	::PICKUP THIS CHARACTER
045766	001420			BEQ	3\$::IF ZERO GET OUT
045770	122716	000060		CMPB	#'0,(SP)	::MAKE SURE THIS CHARACTER
045774	003026			BGT	4\$::IS AN OCTAL DIGIT
045776	122716	000067		CMPB	#'7,(SP)	
046002	002423			BLT	4\$	
046004	006301			ASL	R1	::*2
046006	006102			ROL	R2	
046010	006301			ASL	R1	::*4
046012	006102			ROL	R2	
046014	006301			ASL	R1	::*8
046016	006102			ROL	R2	
046020	042716	177770		BIC	#^C7,(SP)	::STRIP THE ASCII JUNK
046024	062601			ADD	(SP)+,R1	::ADD IN THIS DIGIT
046026	000756			BR	2\$::LOOP
046030	005726			3\$: TST	(SP)+	::CLEAN TERMINATOR FROM STACK
046032	010166	000012		MOV	R1,12(SP)	::SAVE THE RESULT
046036	010237	046070		MOV	R2,\$HIOCT	
046042	012602			MOV	(SP)+,R2	::POP STACK INTO R2
046044	012601			MOV	(SP)+,R1	::POP STACK INTO R1
046046	012600			MOV	(SP)+,R0	::POP STACK INTO R0
046050	000002			RTI		::RETURN
046052	005726			4\$: TST	(SP)+	::CLEAN PARTIAL FROM STACK
046054	105010			CLRB	(R0)	::SET A TERMINATOR
046056	104401			TYPE		::TYPE UP THRU THE BAD CHAR.
046060	000000			5\$: .WORD	0	
046062	104401	001222		TYPE	,\$QUES	::'"?' 'CR' & 'LF'
046066	000730			BR	1\$::TRY AGAIN
046070	000000			\$HIOCT: .WORD	0	::HIGH ORDER BITS GO HERE

.SBTTL ERROR HANDLER ROUTINE

```

:*****
:*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
:*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
:*AND GO TO $ERRTYP ON ERROR
:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
:*SW15=1      HALT ON ERROR
:*SW13=1      INHIBIT ERROR TYPEOUTS
:*SW10=1      BELL ON ERROR
:*SW09=1      LOOP ON ERROR
:*CALL
:*          ERROR      N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
    
```

```

046072 105037 046432 $ERROR: CLRB      IBSAVE          ;;CLEAR THE ITEM BYTE SAVE LOCATION
046076 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
046100 012737 177777 001406      MOV          #-1,ERFLG$        ;;SET ERROR FLAG
046106          REGSA1:
046106 105237 001103 7$:      INCB      $ERFLG          ;;SET THE ERROR FLAG
046112 001775          BEQ          7$          ;;DON'T LET THE FLAG GO TO ZERO
046114 013777 001102 133020      MOV          $TSTNM,@DISPLAY    ;;DISPLAY TEST NUMBER AND ERROR FLAG
046122 032777 002000 133010      BIT          #BIT10,@SWR        ;;BELL ON ERROR?
046130 001402          BEQ          1$          ;;NO - SKIP
046132 104401 001216          TYPE      ,SBELL          ;;RING BELL
046136 005237 001112 1$:      INC          $ERTTL          ;;COUNT THE NUMBER OF ERRORS
046142 011637 001116          MOV          (SP),$ERRPC        ;;GET ADDRESS OF ERROR INSTRUCTION
046146 162737 000002 001116      SUB          #2,$ERRPC
046154 117737 132736 001114      MOVB        @$ERRPC,$ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
046162 032777 001000 132750      BIT          #BIT09,@SWR        ;;SEE IF LOOP ON ERROR IS SET
046170 001060          BNE          1004$          ;;BRANCH AROUND ROUTINE IF SO
046172 122737 000177 001114      CMPB        #177,$ITEMB        ;;SEE IF THIS IS THE POWER FAIL CALL
046200 001454          BEQ          1004$          ;;BRANCH AROUND ROUTINE IF IT IS
046202 105737 046432          TSTB      IBSAVE          ;;SEE IF THIS IS THE 2ND ERROR CALL IN THIS ROUTINE
046206 001047          BNE          1003$          ;;BRANCH IF SO
046210 022737 177777 046430      CMP          #-1,CPSAVE        ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
046216 001445          BEQ          1004$          ;;BRANCH IF SO
046220 013746 000004          MOV          ERRVEC,-(SP)      ;;SAVE CONTENTS OF ERROR VECTOR
046224 012737 046242 000004      MOV          #1000$,ERRVEC     ;;SETUP 'TRAP' RETURN ADDRESS
046232 013737 177766 046430      MOV          177766,CPSAVE     ;;MOVE CPU ERROR REGISTER TO CPSAVE FOR TEST
046240 000406          BR          1001$
046242 012737 77777 046430 1000$:  MOV          #-1,CPSAVE        ;;SET CPU ERROR REGISTER TIMEOUT INDICATOR
046250 012716 046256          MOV          #1001$, (SP)     ;;SETUP RETURN ADDRESS
046254 000002          RTI
046256 012637 000004          1001$:  MOV          (SP)+,ERRVEC      ;;RESTORE CONTENTS OF ERROR VECTOR

046262 022737 177777 046430 1002$:  CMP          #-1,CPSAVE        ;;SEE IF CPSAVE HAS CPU ERR REG TIMEOUT INDICATION
046270 001420          BEQ          1004$          ;;BRANCH IF SO
046272 032737 000001 046430      BIT          #BIT00,CPSAVE     ;;SEE IF POWER MONITOR BIT IS SET IN CPU ERR REG
046300 001414          BEQ          1004$          ;;BRANCH IF OK
046302 042737 000001 177766      BIC          #BIT00,177766     ;;CLEAR THE BIT FOUND SET
046310 113737 001114 046432      MOVB        $ITEMB,IBSAVE     ;;MAKE IBSAVE NON-ZERO FOR DUAL ERROR CALL
046316 112737 000177 001114      MOVB        #177,$ITEMB      ;;SET $ITEMB TO SPECIAL POWER FAIL POINTER
046324 000402          BR          1004$          ;;BRANCH OVER IBSAVE CLEARING

046326 105037 046432          1003$:  CLRB      IBSAVE          ;;CLEAR IBSAVE SO 2ND TIME THROUGH EXITS
046332          1004$:
046332 032777 020000 132600      BIT          #BIT13,@SWR        ;;SKIP TYPEOUT IF SET
    
```


046340	001004			BNE	20\$::SKIP TYPEOUTS
046342	004737	046434		JSR	PC,\$ERRTYP	::GO TO USER ERROR ROUTINE
046346	104401	001223		TYPE	,\$CRLF	
046352			20\$:			
046352	105737	046432	2\$:	TSTB	IBSAVE	::SEE IF IBSAVE IS LOADED
046356	001005			BNE	3\$::BRANCH IF NOT - NO HALT ON PWR MON BIT ERROR
046360	005777	132554		TST	@SWR	::HALT ON ERROR
046364	100002			BPL	3\$::SKIP IF CONTINUE
046366	000000			HALT		::HALT ON ERROR!
046370	104407			CKSWR		::TEST FOR CHANGE IN SOFT-SWR
046372			3\$:			
046372	032777	001000	132540	BIT	#BIT09,@SWR	::LOOP ON ERROR SWITCH SET?
046400	001402			BEQ	4\$::BR IF NO
046402	013716	001110		MOV	\$LPERR,(SP)	::FUDGE RETURN FOR LOOPING
046406	005737	001214	4\$:	TST	\$ESCAPE	::CHECK FOR AN ESCAPE ADDRESS
046412	001402			BEQ	5\$::BR IF NONE
046414	013716	001214		MOV	\$ESCAPE,(SP)	::FUDGE RETURN ADDRESS FOR ESCAPE
046420			5\$:			
046420	105737	046432		TSTB	IBSAVE	::SEE IF ITEM BYTE SAVE LOCATION HAS AN ERROR CALL
046424	001230			BNE	7\$::BRANCH BACK TO CALL ORIGINAL ERROR
046426	000002			RTI		::RETURN
046430	000000			CPSAVE: .WORD	0	::LOCATION TO SAVE CPU ERROR REG CONTENTS
046432	000000			IBSAVE: .WORD	0	::LOCATION TO SAVE ITEM BYTE

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

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

```

046434          046434 104401 001223
046440          046440 010046
046442          046442 005000
046444          046444 153700 001114
046450          046450 001004

046452          046452 013746 001116

046456          046456 104402
046460          046460 000456
046462          046462 122700 000177
046466          046466 001006
046470          046470 113737 001102 046772
046476          046476 012700 046632
046502          046502 000406
046504          046504 005300
046506          046506 006300
046510          046510 006300
046512          046512 006300
046514          046514 062700 003630
046520          046520 012037 046530
046524          046524 001404
046526          046526 104401
046530          046530 000000
046532          046532 104401 001223
046536          046536 012037 046546
046542          046542 001404
046544          046544 104401
046546          046546 000000
046550          046550 104401 001223
046554          046554 010146
046556          046556 012001
046560          046560 001415
046562          046562 012000
046564          046564 105720
046566          046566 001003
046570          046570 013146
046572          046572 104402
046574          046574 000402
046576          046576
046576          046576 013146
046600          046600 104405
046602          046602 005711
046604          046604 001403
046606          046606 104401 046626
046612          046612 000764

046614          046614 012601
046616          046616 012600

$ERRTYP:
          TYPE      ,SCRLF      ::'CARRIAGE RETURN' & 'LINE FEED'
          MOV      R0,-(SP)      ::SAVE R0
          CLR      R0            ::PICKUP THE ITEM INDEX
          BISB     @#$ITEMB,R0
          BNE     1$            ::IF ITEM NUMBER IS ZERO, JUST
                                ::TYPE THE PC OF THE ERROR
                                ::SAVE $ERRPC FOR TYPEOUT
                                ::ERROR ADDRESS
                                ::GO TYPE--OCTAL ASCII(ALL DIGITS)
                                ::GET OUT
                                ::SEE IF THIS ERROR CALL IS SPECIAL POWER FAIL CALL
                                ::BRANCH IF NOT
                                ::GET TEST NUMBER
                                ::MOVE POWER FAIL ERROR CALL TABLE TO R0
                                ::BRANCH TO CALL ERROR
                                ::ADJUST THE INDEX SO THAT IT WILL
                                ::      WORK FOR THE ERROR TABLE

          MOV      $ERRPC,-(SP)

          TYPDC
          BR      10$
          CMPB     #177,R0
          BNE     1000$
          MOVB     $STNM,PFTSTN
          MOV      #PFECH,R0
          BR      1001$
          DEC     R0
          ASL     R0
          ASL     R0
          ASL     R0
          ADD     #$ERRTB,R0      ::FORM TABLE POINTER
          MOV     (R0)+,2$      ::PICKUP 'ERROR MESSAGE' POINTER
          BEQ     3$            ::SKIP TYPEOUT IF NO POINTER
          TYPE     'ERROR MESSAGE'
                                ::'ERROR MESSAGE' POINTER GOES HERE
                                ::'CARRIAGE RETURN' & 'LINE FEED'
          MOV     (R0)+,4$      ::PICKUP 'DATA HEADER' POINTER
          BEQ     5$            ::SKIP TYPEOUT IF 0
          TYPE     'DATA HEADER'
                                ::'DATA HEADER' POINTER GOES HERE
                                ::'CARRIAGE RETURN' & 'LINE FEED'
          MOV     R1,-(SP)      ::SAVE R1
          MOV     (R0)+,R1      ::PICKUP 'DATA TABLE' POINTER
          BEQ     9$            ::BR IF NO DATA TO BE TYPED
          MOV     (R0)+,R0      ::PICKUP 'DATA FORMAT' POINTER
          TSTB    (R0)+
                                ::'OCTAL' OR 'DECIMAL'
          BNE     7$            ::BR IF DECIMAL
          MOV     @R1+,-(SP)    ::SAVE @R1+ FOR TYPEOUT
          TYPDC
          BR      8$            ::GO TYPE--OCTAL ASCII(ALL DIGITS)

          MOV     @R1+,-(SP)    ::SAVE @R1+ FOR TYPEOUT
          TYPDS
          TST     (R1)
                                ::GO TYPE--DECIMAL ASCII WITH SIGN
                                ::IS THERE ANOTHER NUMBER?
          BEQ     9$            ::BR IF NO
          TYPE    ,11$
                                ::TYPE TWO(2) SPACES
          BR      6$            ::LOOP

          MOV     (SP)+,R1      ::RESTORE R1
          MOV     (SP)+,R0      ::RESTORE R0
  
```


046620	104401	001223			TYPE	,SCLF	::'CARRIAGE RETURN' & 'LINE FEED'
046624	000207				RTS	PC	::RETURN
046626	040	040	000	11\$:	.ASCIZ	/ /	::TWO(2) SPACES
					.EVEN		
046632	046642	046724	046756	PFECH:	PFECH1,PFECH2,PFECH3,PFECH4	::WORDS DEFINING TABLES BELOW	
046642	120	117	127	PFECH1:	.ASCIZ	?POWER MONITOR BIT IN CPU ERROR REGISTER FOUND SET?	
046724	124	105	123	PFECH2:	.ASCIZ	?TESTNO ERR PC CPUERREG?	
					.EVEN		
046756	046772	001116	046430	PFECH3:	.WORD	PFTSTN,\$ERRPC,CPSAVE,0	
046766	000	000	000	PFECH4:	.BYTE	0,0,0,0	
046772	000000			PFTSTN:	.WORD	0	
						::CONTAINS TEST NUMBER FOR PF BIT ERROR	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

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

046774	017646	000000		\$TYPOS: MOV	@(SP),-(SP)	;;PICKUP THE MODE
047000	116637	000001	047217	MOV	1(SP),SOFILL	;;LOAD ZERO FILL SWITCH
047006	112637	047221		MOV	(SP)+,SOMODE+1	;;NUMBER OF DIGITS TO TYPE
047012	062716	000002		ADD	#2,(SP)	;;ADJUST RETURN ADDRESS
047016	000406			BR	\$TYPON	
047020	112737	000001	047217	\$TYPOC: MOV	#1,SOFILL	;;SET THE ZERO FILL SWITCH
047026	112737	000006	047221	MOV	#6,SOMODE+1	;;SET FOR SIX(6) DIGITS
047034	112737	000005	047216	\$TYPON: MOV	#5,SOCNT	;;SET THE ITERATION COUNT
047042	010346			MOV	R3,-(SP)	;;SAVE R3
047044	010446			MOV	R4,-(SP)	;;SAVE R4
047046	010546			MOV	R5,-(SP)	;;SAVE R5
047050	113704	047221		MOV	SOMODE+1,R4	;;GET THE NUMBER OF DIGITS TO TYPE
047054	005404			NEG	R4	
047056	062704	000006		ADD	#6,R4	;;SUBTRACT IT FOR MAX. ALLOWED
047062	110437	047220		MOV	R4,SOMODE	;;SAVE IT FOR USE
047066	113704	047217		MOV	SOFILL,R4	;;GET THE ZERO FILL SWITCH
047072	016605	000012		MOV	12(SP),R5	;;PICKUP THE INPUT NUMBER
047076	005003			CLR	R3	;;CLEAR THE OUTPUT WORD
047100	006105		1\$:	ROL	R5	;;ROTATE MSB INTO 'C'
047102	000404			BR	3\$;;GO DO MSB
047104	006105		2\$:	ROL	R5	;;FORM THIS DIGIT
047106	006105			ROL	R5	
047110	006105			ROL	R5	
047112	010503			MOV	R5,R3	
047114	006103		3\$:	ROL	R3	;;GET LSB OF THIS DIGIT
047116	105337	047220		DECB	SOMODE	;;TYPE THIS DIGIT?
047122	100016			BPL	7\$;;BR IF NO
047124	042703	177770		BIC	#177770,R3	;;GET RID OF JUNK
047130	001002			BNE	4\$;;TEST FOR 0
047132	005704			TST	R4	;;SUPPRESS THIS 0?
047134	001403			BEQ	5\$;;BR IF YES
047136	005204		4\$:	INC	R4	;;DON'T SUPPRESS ANYMORE 0'S

047140	052703	000060		BIS	#'0,R3	::MAKE THIS DIGIT ASCII
047144	052703	000040	5\$:	BIS	#',R3	::MAKE ASCII IF NOT ALREADY
047150	110337	047214		MOVB	R3,8\$::SAVE FOR TYPING
047154	104401	047214		TYPE	8\$::GO TYPE THIS DIGIT
047160	105337	047216	7\$:	DECB	\$OCNT	::COUNT BY 1
047164	003347			BGT	2\$::BR IF MORE TO DO
047166	002402			BLT	6\$::BR IF DONE
047170	005204			INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
047172	000744			BR	2\$::GO DO THE LAST DIGIT
047174	012605		6\$:	MOV	(SP)+,R5	::RESTORE R5
047176	012604			MOV	(SP)+,R4	::RESTORE R4
047200	012603			MOV	(SP)+,R3	::RESTORE R3
047202	016666	000002 000004		MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
047210	012616			MOV	(SP)+,(SP)	
047212	000002			RTI		::RETURN
047214	000		8\$:	.BYTE	0	::STORAGE FOR ASCII DIGIT
047215	000			.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
047216	000		\$OCNT:	.BYTE	0	::OCTAL DIGIT COUNTER
047217	000		\$OFILL:	.BYTE	0	::ZERO FILL SWITCH
047220	000000		\$OMODE:	.WORD	0	::NUMBER OF DIGITS TO TYPE

.SBTTL TRAP DECODER

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

047222	010046		\$TRAP:	MOV	R0,-(SP)	::SAVE R0
047224	016600	000002		MOV	2(SP),R0	::GET TRAP ADDRESS
047230	005740			TST	-(R0)	::BACKUP BY 2
047232	111000			MOVB	(R0),R0	::GET RIGHT BYTE OF TRAP
047234	006300			ASL	R0	::POSITION FOR INDEXING
047236	016000	047256		MOV	\$TRPAD(R0),R0	::INDEX TO TABLE
047242	000200			RTS	R0	::GO TO ROUTINE

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

047244	011646		\$TRAP2:	MOV	(SP),-(SP)	::MOVE THE PC DOWN
047246	016666	000004		MOV	4(SP),2(SP)	::MOVE THE PSW DOWN
047254	000002	000002		RTI		::RESTORE THE PSW

.SBTTL TRAP TABLE

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

			:	ROUTINE		
			:	-----		
047256	047244		\$TRPAD:	.WORD	\$TRAP2	
047260	044306			\$TYPE	::CALL=TYPE	TRAP+1(104401) TTY TYPEOUT ROUTINE
047262	047020			\$TYPOC	::CALL=TYPOC	TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
047264	046774			\$TYPOS	::CALL=TYPOS	TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZERO?)
047266	047034			\$TYPON	::CALL=TYPON	TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
047270	044062			\$TYPDS	::CALL=TYPDS	TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
047272	045210			\$GTSWR	::CALL=GTSWR	TRAP+6(104406) GET SOFT-SWR SETTING
047274	045120			\$CKSWR	::CALL=CKSWR	TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
047276	045462			\$RDCHR	::CALL=RDCHR	TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
047300	045552			\$RDLIN	::CALL=RDLIN	TRAP+11(104411) TTY TYPEIN STRING ROUTINE
047302	045732			\$RDOCT	::CALL=RDOCT	TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
2 047304	032036			T.SCOPI	::CALL=SCOPI	TRAP+13(104413) MY LOCAL SCOPES
3 047306	032116			CHECKD	::CALL=CHECKD	TRAP+14(104414) CHECK DVA,RDY,DPR,DRY
4 047310	032434			WAIT.WAT	::CALL=WAT	TRAP+15(104415) WAIT LOOP

1

.SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
047312 012737 047456 000024 $PWRDN: MOV $IILLUP,@#PWRVEC ;;SET FOR FAST UP
047320 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
047326 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
047330 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
047332 010246 MOV R2,-(SP) ;;PUSH R2 ON STACK
047334 010346 MOV R3,-(SP) ;;PUSH R3 ON STACK
047336 010446 MOV R4,-(SP) ;;PUSH R4 ON STACK
047340 010546 MOV R5,-(SP) ;;PUSH R5 ON STACK
047342 017746 131572 MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
047346 010637 047462 MOV SP,$SAVR6 ;;SAVE SP
047352 012737 047364 000024 MOV #PWRUP,@#PWRVEC ;;SET UP VECTOR
047360 000000 HALT
047362 000776 BR -2 ;;HANG UP

*****
:POWER UP ROUTINE
047364 012737 047456 000024 $PWRUP: MOV $IILLUP,@#PWRVEC ;;SET FOR FAST DOWN
047372 013706 047462 MOV $SAVR6,SP ;;GET SP
047376 005037 047462 CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
047402 005237 047462 1$: INC $SAVR6 ;;WAIT FOR THE INC
047406 001375 BNE 1$ ;;OF WORD
047410 012677 131524 MOV (SP)+,@SWR ;;POP STACK INTO @SWR
047414 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
047416 012604 MOV (SP)+,R4 ;;POP STACK INTO R4
047420 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
047422 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
047424 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
047426 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
047430 012737 047312 000024 MOV #PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
047436 012737 000340 000026 MOV #340,@#PWRVEC+2 ;;PRIO:7
047444 104401 TYPE ;;REPORT THE POWER FAILURE
047446 047464 $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
047450 012716 MOV (PC)+,(SP) ;;RESTART AT BEGIN
047452 004620 $PWRAD: .WORD BEGIN ;;RESTART ADDRESS
047454 000002 RTI
047456 000000 $IILLUP: HALT ;;THE POWER UP SEQUENCE WAS STARTED
047460 000776 BR -2 ;; BEFORE THE POWER DOWN WAS COMPLETE
047462 000000 $SAVR6: 0 ;;PUT THE SP HERE
047464 015 012 120 $POWER: .ASCIZ <15><12>'POWER'
.EVEN
    
```

.SBTTL ERROR AND MESSAGE TABLES

2						
3	047474	127	122	117	EM1:	.ASCIZ /WRONG DATA IN READING OR WRITING HARDWARE REGISTER/
4	047557	105	122	122	EM2:	.ASCIZ /ERROR ON DATA COMMAND/
5	047606	105	122	122	EM6:	.ASCIZ /ERROR ON WRITE HEADER AND DATA/
6	047645	103	117	116	EM11:	.ASCIZ /CONTROLLER OR DRIVE STATUS/
7	047700	122	105	107	EM14:	.ASCIZ /REGISTER FAILED/
8	047720	116	117	116	EM15:	.ASCIZ /NON EXISTENT REGISTER, PROGRAM ABORTED./
9	047770	127	101	111	EM16:	.ASCIZ /WAIT LOOP FAILED/
10	050011	127	122	111	EM17:	.ASCIZ /WRITE CHECK FAILING/
11	050035	122	105	107	EM20:	.ASCIZ /REGISTER FAILING/
12	050056	111	114	124	EM21:	.ASCIZ /INTERRUPT FAILING/
13	050100	105	122	122	EM22:	.ASCIZ /ERROR ON DRIVE PRESENT/<CRLF>
14	050127	124	110	105		.ASCIZ /THE UNIT NO'S FOUND BY SETTING RHAS/<CRLF>
15	050173	104	117	040		.ASCIZ /DO NOT AGREE WITH THE UNIT NO. FOUND FROM/<CRLF>
16	050245	122	110	103		.ASCIZ /RHCS2-'NED' BIT #12/<CRLF>
17	050271	061	067	067		.ASCIZ /177777-MEANS NO UNIT FOUND/<CRLF>
18	050324	116	117	124		.ASCIZ /NOTE: ON DUAL PORT SYSTEM, DRIVE ON OTHER PORT WILL NOT GIVE/<CRLF>
19	050421	047	116	105		.ASCIZ /'NED', HENCE THERE WILL BE AN EXTRA DRIVE/
20	050473	114	117	117	EM24:	.ASCIZ /LOOK AHEAD REGISTER AT THE BEGINNING OF SECTOR IS IN ERROR/
21	050566	114	117	117	EM25:	.ASCIZ /LOOK AHEAD REGISTER IS IN ERROR/
22	050626	103	125	122	EM30:	.ASCIZ /CURRENT CYLINDER DOES NOT MATCH DESIRED CYLINDER REGISTER/<CRLF>
23	050721	101	106	124		.ASCIZ /AFTER A SEEK AND INIT/
24	050747	105	103	103	EM31:	.ASCIZ /ECC GENERATED IS INCORRECT/<CRLF>
25	051002	105	126	105		.ASCIZ /EVERY WORD ON THIS SECTOR IS THAT GIVEN IN 'DATA USED'/
26	051071	117	116	040	EM32:	.ASCIZ /ON READ COMMAND, AFTER DATA AND ECC HAVE BEEN READ,/<CRLF>
27	051155	105	103	103		.ASCIZ /ECC REGISTERS OR RHER1 ARE IN ERROR/<CRLF>
28	051221	117	116	114		.ASCIZ /ONLY LOWER 11 BITS OF PATTERN REG. CAN BE READ/<CRLF>
29	051300	124	110	111		.ASCIZ /THIS SHOULD MATCH LOWER 11 BITS OF GOOD ECC1/
30	051355	110	111	107	EM33:	.ASCIZ /HIGH COUNT BIT NOT SET AFTER 38859 CLOCKS/
31	051427	132	105	122	EM34:	.ASCIZ /ZERO DETECT BIT NOT HIGH WHEN 32 BIT ECC REG. HAS 21 ZEROS/
32	051522	120	117	123	EM35:	.ASCIZ /POSITION REGISTER OR 11 BITS OF PATTERN REGISTER INCORRECT/<CRLF>
33	051615	114	117	127		.ASCIZ /LOWER 11 BITS OF PATTERN REGISTER SHOULD MATCH LOWER/<CRLF>
34	051702	061	061	040		.ASCIZ /11 BITS OF GOOD ECC1/<CRLF>
35	051727	104	101	124		.ASCIZ /DAT ENVELOP GOOD POSITION AND N-CODE ZEROS ARE IN OCTAL/
36	052016	117	116	040	EM36:	.ASCIZ /ON READ COMMAND WITH NON-CORRECTABLE ERROR DCK AND ECH SHOULD BE SET/<CRLF>
37	052123	111	106	040		.ASCIZ /IF POSITION REGISTER =10040 OR 10041 IT IS GOOD/
38	052203	127	122	111	EM37:	.ASCIZ /WRITING WITH BUS ADDRESS HIGHER THAN 28K CAUSED ERROR/
39	052271	124	110	105	EM40:	.ASCIZ /THERE WAS A READ-WRITE HEADER & DATA ERROR DURING 'DTE'/<CRLF>
40	052361	124	105	123		.ASCIZ /TEST SETUP - THE TEST WAS ABORTED AT THAT POINT/

1	052441	200	101	114	PREAMB: .ASCII	<CRLF>/ALL DCL'S UNDER TEST MUST BE LOCKED ON CORRECT PORT./
2	052526	200	111	106	.ASCII	<CRLF>/IF CHANGES ARE REQUIRED ON PORT SWITCH, THEN A CYCLE/
3	052613	200	125	120	.ASCII	<CRLF>/UP SEQUENCE IS REQUIRED FOR STROBING THE PORT SELECT/
4	052700	200	106	114	.ASCII	<CRLF>/FLOP./
5	052706	200			.ASCII	<CRLF>
6	052707	200	101	114	.ASCII	<CRLF>/ALL DCL'S NOT UNDER TEST MUST BE SWITCHED OFF./
7	052774	200	111	106	.ASCII	<CRLF>/IF THIS IS NOT DONE, ERRORS WILL RESULT IN THE 'NED'./
8	053061	200	124	105	.ASCIZ	<CRLF>/TESTS (T21 AND T36)./
9						
10	053107	200	116	117	NDRVAS: .ASCII	<CRLF>/NO DRIVES PRESENT, RHAS=0/
11	053141	200			.ASCII	<CRLF>
12	053142	200	127	122	.ASCII	<CRLF>/WRITTING ONES INTO RHER1 FOR ALL UNIT NUMBERS DOES/
13	053227	200	116	117	.ASCII	<CRLF>/NOT SET ANY BIT IN RHAS, SO ABORT PROGRAM./
14	053302	200			.ASCII	<CRLF>
15	053303	200	124	117	.ASCII	<CRLF>/TO LOOP ON THIS TEST WITHOUT PRINTOUT, SET SWITCHES/
16	053370	200	061	063	.ASCIZ	<CRLF>/13, 8 AND 2./<CRLF>
17						
18	053407	106	101	124	CPHALT: .ASCII	/FATAL ERROR - SEE DOCUMENT LISTING/<CRLF>
19	053452	040	200	207	.ASCII	/ /<CRLF><207><377><377><207><377><377><207><377><377>
20	053465	124	110	105	.ASCII	/THE CONTROLLER OR DEVICE HAS GONE OFFLINE, LOST/<CRLF>
21	053545	047	122	105	.ASCII	/'READY', BECOME UNAVAILABLE, OR HAS STATUS BITS/<CRLF>
22	053625	127	110	111	.ASCIZ	/WHICH CANNOT BE CLEARED/

1	060144	001116	003610	031620	DT1:	.WORD	\$ERRPC,TSTNM,REGADR,\$GDDAT,\$BDDAT,0
2	060160	001116	003610	037030	DT2:	.WORD	\$ERRPC,TSTNM,ERWORD,\$GDDAT,0
3	060172	001116	003610	037030	DT3:	.WORD	\$ERRPC,TSTNM,ERWORD,\$GDDAT,\$BDDAT,0
4							
5	060206	001116	003610	001122	DT11:	.WORD	\$ERRPC,TSTNM,\$BDADR,CS1,CS2,DS1,ER1,0
6	060226	001116	003610	001122	DT14:	.WORD	\$ERRPC,TSTNM,\$BDADR,\$BDDAT,CS1,CS2,DS1,ER1,0
7	060250	001116	003610	001200	DT15:	.WORD	\$ERRPC,TSTNM,\$TMP1,0
8	060260	001116	003610	001204	DT16:	.WORD	\$ERRPC,TSTNM,\$TMP3,\$TMP1,\$TMP0,\$BDDAT,0
9	060276	001116	003610	001310	DT17:	.WORD	\$ERRPC,TSTNM,BA,DB,WC,CS1,CS2,0
10							
11	060316	001116	003610	001316	DT20:	.WORD	\$ERRPC,TSTNM,ER1,ER2,ER3,AS,DS1,0
12	060336	001116	003610	001314	DT21:	.WORD	\$ERRPC,TSTNM,CS1,AS,DS1,0
13	060352	001116	003610	001124	DT22:	.WORD	\$ERRPC,TSTNM,\$GDDAT,\$BDDAT,0
14	060364	001116	003610	001320	DT24:	.WORD	\$ERRPC,TSTNM,DST,\$BDDAT,\$TMP1,\$TMP2,\$TMP3,0
15	060404	001116	003610	001414	DT26:	.WORD	\$ERRPC,TSTNM,PCJSR,\$BDADR,CS1,CS2,DS1,ER1,0
16	060426	001116	003610	001414	DT27:	.WORD	\$ERRPC,TSTNM,PCJSR,REGADR,\$GDDAT,\$BDDAT,0
17							
18	060444	001116	003610	001414	DT30:	.WORD	\$ERRPC,TSTNM,PCJSR,REGADR,\$GDDAT,\$BDDAT,0
19	060462	001116	003610	037030	DT31:	.WORD	\$ERRPC,TSTNM,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1,0
20	060504	001116	003610	001414	DT32:	.WORD	\$ERRPC,TSTNM,PCJSR,ERWORD,\$GDDAT,\$BDDAT,CS1,DS1,ER1,0
21	060530	001116	003610	001414	DT33:	.WORD	\$ERRPC,TSTNM,PCJSR,ERWORD,\$GDDAT,CS1,DS1,ER1,0
22	060552	001116	003610	034556	DT34:	.WORD	\$ERRPC,TSTNM,GECC1,GECC2,WECC1,WECC2,DISK,0
23	060572	001116	003610	034556	DT35:	.WORD	\$ERRPC,TSTNM,GECC1,GECC2,EC2,EC1,POSITI,ER1,0
24	060614	001116	003610	001414	DT36:	.WORD	\$ERRPC,TSTNM,PCJSR,MR,EC1,EC2,0
25	060632	001116	003610	001344	DT37:	.WORD	\$ERRPC,TSTNM,EC1,POSITI,GECC1,GECC2,EC2,DATENV,ZCODE,0
26							
27	060656	001116	003610	001122	DT40:	.WORD	\$ERRPC,TSTNM,\$BDADR,CS1,CS2,DS1,ER1,0
28							
29	060676	000	000	000	DF1:	.BYTE	0,0,0,0,0
30	060703	000	000	001	DF2:	.BYTE	0,0,1,0
31	060707	000	000	001	DF3:	.BYTE	0,0,1,0,0
32							
33	060714	000	000	000	DF11:	.BYTE	0,0,0,0,0,0,0
34	060723	000	000	000	DF14:	.BYTE	0,0,0,0,0,0,0,0
35	060733	000	000	000	DF15:	.BYTE	0,0,0
36	060736	000	000	000	DF16:	.BYTE	0,0,0,0,0
37	060743	000	000	000	DF17:	.BYTE	0,0,0,0,0,0,0
38	060752	000	000	000	DF20:	.BYTE	0,0,0,0,0,0,0
39							
40	060761	000	000	000	DF21:	.BYTE	0,0,0,0,0
41	060766	000	000	000	DF22:	.BYTE	0,0,0,0
42	060772	000	000	000	DF24:	.BYTE	0,0,0,0,0,0,0
43	061001	000	000	000	DF26:	.BYTE	0,0,0,0,0,0,0,0
44	061011	000	000	000	DF27:	.BYTE	0,0,0,0,0,0
45	061017	000	000	000	DF30:	.BYTE	0,0,0,0,0,0
46							
47	061025	000	000	001	DF31:	.BYTE	0,0,1,0,0,0,0,0
48	061035	000	000	000	DF32:	.BYTE	0,0,0,1,0,0,0,0,0
49	061046	000	000	000	DF33:	.BYTE	0,0,0,1,0,0,0,0
50	061056	000	000	000	DF34:	.BYTE	0,0,0,0,0,0,0
51	061065	000	000	000	DF35:	.BYTE	0,0,0,0,0,0,0,0
52	061075	000	000	000	DF36:	.BYTE	0,0,0,0,0,0
53	061103	000	000	000	DF37:	.BYTE	0,0,0,0,0,0,0,0,0
54	061114	000	000	000	DF40:	.BYTE	0,0,0,0,0,0,0
55						.EVEN	

1	061124	122	110	061	EM42:	.ASCIZ	/RH11 REGISTER FAILED TO RESPOND TO A "TST" COMMAND/
2	061207	122	110	061	EM43:	.ASCIZ	/RH11 ILLEGAL REGISTER RESPONSE TEST/
3	061253	115	117	126	EM44:	.ASCIZ	/MOVE BYTE TO WORD COUNT TEST/
4	061310	102	111	123	EM45:	.ASCIZ	/BIS BYTE TO WORD COUNT/
5	061337	102	111	103	EM46:	.ASCIZ	/BIC BYTE TO WORD COUNT/
6	061366	122	105	107	EM47:	.ASCIZ	/REGISTER ADDRESS SELECT TEST/
7	061423	116	117	116	EM50:	.ASCIZ	/NON EXISTANT DRIVE (NED) TEST/
8	061461	101	123	040	EM51:	.ASCIZ	/AS REGISTER TEST/
9	061502	102	125	123	EM52:	.ASCIZ	/BUSS ADDRESS REGISTER DATA TEST/
10	061542	102	125	123	EM53:	.ASCIZ	/BUSS ADDRESS REGISTER MOVE BYTE/
11	061602	122	110	061	EM54:	.ASCIZ	/RH11 INTERRUPT FAILED TO OCCUR/
12	061641	122	105	123	EM55:	.ASCIZ	/RESET TEST/
13	061654	115	130	106	EM56:	.ASCIZ	/MXF BIT TEST/
14	061671	125	116	111	EM57:	.ASCIZ	/UNIBUS PARITY BIT TEST/
15	061720	104	117	105	EM60:	.ASCIZ	/DOES SELECTING THE RH11 CLEAR THE UNIT SELECT REGISTER/
16	062007	104	117	105	EM61:	.ASCIZ	/DOES TRE GET SET BY UNIBUS PARITY ERROR/
17	062057	116	117	116	EM62:	.ASCIZ	/NON EXISTANT DRIVE (NED) FAILED TO SET (TRE)/
18	062134	116	117	116	EM63:	.ASCIZ	/NON EXISTANT MEMORY (NEM) FAILED TO SET (TRE)/
19	062212	120	117	122	EM64:	.ASCIZ	/PORT SELECT FAILED TO CLEAR/
20	062246	103	117	116	EM65:	.ASCIZ	/CONTROLLER CLEAR FAILED TO CLEAR COMMAND REGISTER/
21	062330	122	105	123	EM66:	.ASCIZ	/RESELECT FAILED TO CLEAR COMMAND REGISTER/
22	062402	111	116	124	EM67:	.ASCIZ	/INTERRUPT CAUSED BY RDY!IE BITS SET FAILED TO OCCUR/
23	062467	111	105	040	EM70:	.ASCIZ	/IE FAILED TO CLEAR FOLLOWING AN INTERRUPT/
24	062541	101	123	040	EM71:	.ASCIZ	/AS REGISTER WRITE TEST/
25							
26	062570	120	103	040	DH42:	.ASCII	/PC TEST ILLEGAL/<CRLF>
27	062620	040	040	040		.ASCIZ	/ NO ADDRESS/
28	062650	120	103	040	DH44:	.ASCII	/PC TEST REGISTR CONT CONT/<CRLF>
29	062715	040	040	040		.ASCIZ	/ NO ADDRESS GOOD BAD/
30	062761	120	103	040	DH54:	.ASCII	/PC TEST REGISTR CONT/<CRLF>
31	063016	040	040	040		.ASCIZ	/ NO ADDRESS/
32							.EVEN
33							
34	063046	001116	003610	031620	DT42:	.WORD	\$ERRPC,TSTNM,REGADR,0
35	063056	001116	003610	031620	DT44:	.WORD	\$ERRPC,TSTNM,REGADR,\$GDDAT,\$BDDAT,0
36	063072	001116	003610	031620	DT54:	.WORD	\$ERRPC,TSTNM,REGADR,\$GDDAT,0
37							
38	063104	000	000	000	DF42:	.BYTE	0,0,0
39	063107	000	000	000	DF44:	.BYTE	0,0,0,0,0
40	063114	000	000	000	DF54:	.BYTE	0,0,0,0
41							
42		000200			.END		200

SYMBOL TABLE

ABS = 000200
 ACL = 000040
 ACU = 100000
 ADTIMO 036436
 AOE = 001000
 AS = 001332
 ATA = 100000
 ATABLE 003620
 ATTENT 001416
 ATO = 000001
 AT1 = 000002
 AT2 = 000004
 AT3 = 000010
 AT4 = 000020
 AT5 = 000040
 AT6 = 000100
 AT7 = 000200
 A16 = 000400
 A17 = 001000
 BA = 001310
 BADTMO 004540
 BAI = 000010
 BASECH 035506
 BEGIN 004620
 BEGIN2 004626
 BITST 031622
 BIT0 = 000001
 BIT00 = 000001
 BIT01 = 000002
 BIT02 = 000004
 BIT03 = 000010
 BIT04 = 000020
 BIT05 = 000040
 BIT06 = 000100
 BIT07 = 000200
 BIT08 = 000400
 BIT09 = 001000
 BIT1 = 000002
 BIT10 = 002000
 BIT11 = 004000
 BIT12 = 010000
 BIT13 = 020000
 BIT14 = 040000
 BIT15 = 100000
 BIT2 = 000004
 BIT3 = 000010
 BIT4 = 000020
 BIT5 = 000040
 BIT6 = 000100
 BIT7 = 000200
 BIT8 = 000400
 BIT9 = 001000
 BLT1 031650
 BLT2 031672
 BLT3 031702
 BPTVEC= 000014
 CA = 001326

CC = 001352
 CHECKD= 104414
 CHECKE 032306
 CHECKT 032116
 CKSWR = 104407
 CLAREA 031772
 CLDISK 032054
 CLR = 000040
 COMHD 036550
 COMPA 037622
 COMPAR 032756
 COMWHD 041672
 COUNTD 040270
 CPHALT 053407
 CPSAVE 046430
 CR = 000015
 CRC = 033270
 CRLF = 000200
 CSF = 000002
 CSU = 000010
 CS1 = 001314
 CS2 = 001312
 CYL 036710
 DAREAD 036764
 DATENV 034574
 DAWORD 036770
 DB = 001304
 DCK = 100000
 DCLEAR 001440
 DDISP = 177570
 DENVL = 000200
 DE1 = 000040
 DFF20 = 000002
 DF1 060676
 DF11 060714
 DF14 060723
 DF15 060733
 DF16 060736
 DF17 060743
 DF2 060703
 DF20 060752
 DF21 060761
 DF22 060766
 DF24 060772
 DF26 061001
 DF27 061011
 DF3 060707
 DF30 061017
 DF31 061025
 DF32 061035
 DF33 061046
 DF34 061056
 DF35 061065
 DF36 061075
 DF37 061103
 DF40 061114
 DF42 063104

DF44 = 063107
 DF5 = 000001
 DF54 = 063114
 DH1 = 053655
 DH11 = 054103
 DH14 = 054257
 DH15 = 054455
 DH16 = 054505
 DH17 = 054642
 DH2 = 053772
 DH20 = 055016
 DH21 = 055171
 DH22 = 055304
 DH24 = 055377
 DH26 = 055554
 DH27 = 055751
 DH30 = 056106
 DH31 = 056237
 DH32 = 056434
 DH33 = 056650
 DH34 = 057046
 DH35 = 057220
 DH36 = 057414
 DH37 = 057551
 DH40 = 057767
 DH42 = 062570
 DH44 = 062650
 DH54 = 062761
 DIGB = 000004
 DISK = 040626
 DISPLA 001142
 DISPRE 000174
 DLT = 100000
 DL64 = 000020
 DMD = 000001
 DPR = 000400
 DRY = 000200
 DST = 001320
 DSWR = 177570
 DS1 = 001336
 DT = 001340
 DTAGAP 041632
 DTE = 010000
 DTSY = 001000
 DT1 = 060144
 DT11 = 060206
 DT14 = 060226
 DT15 = 060250
 DT16 = 060260
 DT17 = 060276
 DT2 = 060160
 DT20 = 060316
 DT21 = 060336
 DT22 = 060352
 DT24 = 060364
 DT26 = 060404
 DT27 = 060426

DT3 = 060172
 DT30 = 060444
 DT31 = 060462
 DT32 = 060504
 DT33 = 060530
 DT34 = 060552
 DT35 = 060572
 DT36 = 060614
 DT37 = 060632
 DT40 = 060656
 DT42 = 063046
 DT44 = 063056
 DT54 = 063072
 DVA = 004000
 ECDATA = 034554
 ECH = 000100
 ECI = 004000
 ECORR 035200
 ECTEST 034612
 EC1 = 001344
 EC2 = 001346
 EMTVEC= 000030
 EM1 = 047474
 EM11 = 047645
 EM14 = 047700
 EM15 = 047720
 EM16 = 047770
 EM17 = 050011
 EM2 = 047557
 EM20 = 050035
 EM21 = 050056
 EM22 = 050100
 EM24 = 050473
 EM25 = 050566
 EM30 = 050626
 EM31 = 050747
 EM32 = 051071
 EM33 = 051355
 EM34 = 051427
 EM35 = 051522
 EM36 = 052016
 EM37 = 052203
 EM40 = 052271
 EM42 = 061124
 EM43 = 061207
 EM44 = 061253
 EM45 = 061310
 EM46 = 061337
 EM47 = 061366
 EM50 = 061423
 EM51 = 061461
 EM52 = 061502
 EM53 = 061542
 EM54 = 061602
 EM55 = 061641
 EM56 = 061654
 EM57 = 061671

EM6 = 047606
 EM60 = 061720
 EM61 = 062007
 EM62 = 062057
 EM63 = 062134
 EM64 = 062212
 EM65 = 062246
 EM66 = 062330
 EM67 = 062402
 EM70 = 062467
 EM71 = 062541
 ERCRC 042066
 ERFLGS 001406
 ERHDGP 042070
 ERHEAD 042064
 ERPOS 035176
 ERR = 040000
 ERROR = 104000
 ERRVEC= 000004
 ERSTAR 036522
 ERUNIT 036520
 ERWORD 037030
 ER1 = 001316
 ER2 = 001322
 ER3 = 001330
 EXT1 = 000001
 EXT10 = 000010
 EXT2 = 000002
 EXT20 = 000020
 EXT4 = 000004
 EXT40 = 000040
 E00 = 010272
 E11 = 011676
 E12 = 012024
 E44 = 014540
 FEN = 000200
 FER = 000020
 FILLEC 035352
 FIRST 003612
 FMT22 = 010000
 FNWORD 042106
 FORMAT 040272
 FOUT 042112
 FSYNER 042062
 FUTABL 001432
 GCRC 042056
 GECC1 034556
 GECC2 034560
 GO = 000001
 GRV = 000010
 GTSWR = 104406
 HADTMP 034600
 HARDER 034572
 HCCRCE 033672
 HCE = 000200
 HCI = 002000
 HCRC = 000400

B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I
J
K
L
M
N
B
C
D
E
F
G
H
I

HDESYN 042072
 HDWSYN 040624
 HEADER 040600
 HEDGAP 036724
 HEDSYN 036726
 HEGAP 040612
 HT = 000011
 IAE = 002000
 IBSAVE 046432
 IE = 000100
 ILF = 000001
 ILLEGL 001474
 ILR = 000002
 IOTVEC= 000020
 IR = 000100
 IXE = 004000
 IO0 010246
 IO2 010352
 IO3 010446
 IO4 010546
 IO5 010646
 IO6 010746
 IO7 011046
 I14 012216
 I15 012420
 I16 012562
 I24 013156
 JP1 016312
 JP2 016342
 JP3 016752
 JP4 016766
 KEY1 036714
 KEY2 036716
 KIPAR0= 172340
 KIPAR1= 172342
 KIPAR2= 172344
 KIPAR3= 172346
 KIPAR4= 172350
 KIPAR5= 172352
 KIPAR6= 172354
 KIPAR7= 172356
 KIPDR0= 172300
 KIPDR1= 172302
 KIPDR2= 172304
 KIPDR3= 172306
 KIPDR4= 172310
 KIPDR5= 172312
 KIPDR6= 172314
 KIPDR7= 172316
 LA 001350
 LAD 032034
 LERR 031616
 LF = 000012
 LST = 002000
 LST14A 012246
 LST15A 012450
 LST16A 012620

L00 010242
 L02 010350
 L03 010442
 L04 010540
 L05 010636
 L06 010740
 L07 011036
 L10 011172
 L11 011634
 L12 011762
 L13 012074
 L14 012212
 L15 012412
 L16 012552
 L21 012744
 L22 013040
 L30 013266
 L31 013366
 L34 013472
 L35 013606
 L36 013722
 L40 014036
 L41 014140
 L42 014250
 L43 014364
 L44 014474
 L45 014616
 MAKECY 034434
 MASK 031614
 MCLK = 000002
 MCPE = 020000
 MHS = 001000
 MID 034424
 MIDDLE 034322
 MINX = 000004
 MMVEC = 000250
 MOL = 010000
 MPE = 000400
 MR 001334
 MRD = 000020
 MSE = 000020
 MSTCK = 000010
 MWR = 000040
 MXF = 001000
 NCODE 034564
 NCOUNT 034566
 NDRVAS 053107
 NED = 010000
 NEM = 004000
 NHS = 002000
 NOPERA 001432
 NOSYNC 037024
 NOUNIT 001376
 NOWORD 036756
 NUNIT 001400
 N11 011664
 N12 012012

N14 012240
 N36 013762
 OCYL = 100000
 OF 001324
 OFREV = 000200
 OFSETC 001464
 OF100 = 000004
 OF200 = 000010
 OF25 = 000001
 OF400 = 000020
 OF50 = 000002
 OF800 = 000040
 OPDCL = 000100
 OPERSE 031004
 OPI = 020000
 OR = 000200
 OUT 036774
 000 010274
 010 011210
 021 012772
 043 014412
 044 014516
 PAR = 000010
 PAT = 000020
 PCJSR 001414
 PFECB 046632
 PFECB1 046642
 PFECB2 046724
 PFECB3 046756
 PFECB4 046766
 PFTSTN 046772
 PGE = 002000
 PIE1 = 100000
 PIE10 = 000100
 PIE11 = 000040
 PIE12 = 000020
 PIE13 = 000010
 PIE14 = 000004
 PIE15 = 000002
 PIE16 = 000001
 PIE17 = 100000
 PIE18 = 040000
 PIE19 = 020000
 PIE2 = 040000
 PIE20 = 010000
 PIE21 = 004000
 PIE22 = 002000
 PIE23 = 001000
 PIE24 = 000400
 PIE25 = 000200
 PIE26 = 000100
 PIE27 = 000040
 PIE28 = 000020
 PIE29 = 000010
 PIE3 = 020000
 PIE30 = 000004
 PIE31 = 000002

PIE32 = 000001
 PIE4 = 010000
 PIE5 = 004000
 PIE6 = 002000
 PIE7 = 001000
 PIE8 = 000400
 PIE9 = 000200
 PIP = 020000
 PIRQ = 177772
 PIRQVE= 000240
 PKACK 001470
 PLU = 020000
 POSITI 034570
 PRE = 000020
 PREAMB 052441
 PROG = 001000
 PRO = 000000
 PR1 = 000040
 PR2 = 000100
 PR3 = 000140
 PR4 = 000200
 PR5 = 000240
 PR6 = 000300
 PR7 = 000340
 PS = 177776
 PSEL = 002000
 PSU = 000001
 PSW = 177776
 PUTREG 031554
 PWRVEC= 000024
 P12 034604
 P22 034606
 P24 034610
 P3 034602
 RAW = 000020
 RCOM 043230
 RCYL 037014
 RDCHR = 104410
 RDHEAD 037032
 RDLIN = 104411
 RDOCT = 104412
 RDY = 000200
 READ 037624
 READAT 001456
 READIN 001472
 RECALI 001436
 REDATA 043232
 REFOR 001460
 REGADR 031620
 REGSA1 046106
 REINTO 002544
 RELEAS 001442
 RESVEC= 000010
 RETCL 001466
 RHAS 001256
 RHBA 001234
 RHBAE 001302

RHCA 001252
 RHCC 001276
 RHCS1 001240
 RHCS2 001236
 RHCS3 001300
 RHDB 001230
 RHDST 001244
 RHDS1 001262
 RHDT 001264
 RHEC1 001270
 RHEC2 001272
 RHER1 001242
 RHER2 001246
 RHER3 001254
 RHLA 001274
 RHMR 001260
 RHOF 001250
 RHSN 001266
 RHC 001232
 RH70 003614
 RH70CK 005246
 RKEY1 037020
 RKEY2 037022
 RMR = 000004
 RNCTR1 041670
 RNO 043226
 RNWAT1 041750
 RPTRP1 030316
 RPTRP2 030422
 RPVEC 001226
 RPVECT 043342
 RSETR 037016
 RSYNC 037012
 RUNCTR 036546
 RUNWAT 036626
 R11 011616
 R15 012442
 R16 012612
 R44 014542
 R6 = 0000006
 R7 = 0000007
 SAVDT 001410
 SAVER 032554
 SAVSN 001412
 SC = 100000
 SCOPE = 000004
 SCOP1 = 104413
 SCRC 042140
 SCYL 042130
 SC1 = 000100
 SC10 = 001000
 SC2 = 000200
 SC20 = 002000
 SC4 = 000400
 SEARCH 042756
 SECGAP 040530
 SECOTR 036712

SYMBOL TABLE

SECTR	042754	SW3	= 000010	TST34	013426	VAR3	016734	\$CNTLG	045703
SEECOM	001462	SW4	= 000020	TST35	013532	VAR4	016742	\$CNTLU	045676
SEGPFR	042060	SW5	= 000040	TST36	013646	VAR5	016774	\$CRLF	001223
SELECT	001402	SW6	= 000100	TST37	013772	VUF	= 000002	\$DBLK	044276
SELTST	006742	SW7	= 000200	TST4	006216	VU30	= 010000	\$DOAGN	030756
SERCH	001444	SW8	= 000400	TST40	014074	VV	= 000100	\$DTBL	044266
SETCK1	015436	SW9	= 001000	TST41	014176	WAIT.T	032434	\$ENDAD	030746
SETCK2	016132	S11	011612	TST42	014304	WAT	= 104415	\$ENDCT	030714
SETCK3	016656	S12	011746	TST43	014420	WC	001306	\$ENDMG	030765
SETDSK	033544	S36	013716	TST44	014550	WCE	= 040000	\$ENULL	030762
SILOSZ	003616	TAGDTE	001430	TST45	014670	WCF	= 000040	\$EOP	030660
SILOTB	040626	TBITVE=	000014	TST46	014724	WCRC	040610	\$EOPCT	030706
SKEY1	042134	TDF	= 000040	TST47	015256	WCU	= 000001	\$ERFLG	001103
SKEY2	042136	TESDTE	001426	TST5	006756	WCYL	042046	\$ERMAX	001115
SKI	= 040000	TESTAD	031002	TST50	015750	WECC1	041626	\$ERROR	046072
SN	001342	TIMCNT	032432	TST51	016474	WECC2	041630	\$ERRPC	001116
SRO	= 177572	TIMOT	= 000004	TST52	017134	WKEY1	042052	\$ERRTB	003630
SR1	= 177574	TKVEC	= 000060	TST53	017710	WKEY2	042054	\$ERTY	046434
SR2	= 177576	TMPILL	001422	TST54	020336	WLE	= 004000	\$ERTTL	001112
SR3	= 172516	TOLGAP	041634	TST55	021054	WORD	037620	\$ESCAP	001214
SSECTR	042132	TOTALA	001420	TST56	021600	WRCHDA	033120	\$FILLC	001156
SSYN	036722	TPVEC	= 000064	TST57	022342	WRCHDT	001450	\$FILLS	001155
SS1	017236	TRAPVE=	000034	TST6	007452	WRCHDK	001446	\$GDADR	001120
SS10	017404	TRE	= 040000	TST60	022766	WRCHHD	032606	\$GDDAT	001124
SS12	017434	TRK	036662	TST61	023504	WRDATA	040276	\$GET42	030736
SS13	017442	TRK1	= 004000	TST62	024230	WRFROM	001500	\$GTSWR	045210
SS14	017450	TRK10	= 040000	TST63	024772	WRHEAD	042142	\$HD	= 000000
SS15	017510	TRK2	= 010000	TST64	025416	WRIDAT	001452	\$HIOCT	046070
SS2	017674	TRK20	= 100000	TST65	026134	WRIFOR	001454	\$ICNT	001104
SS3	017264	TRK4	= 020000	TST66	026670	WRITE	040056	\$ILLUP	047456
SS4	017270	TRTVEC=	000014	TST67	027446	WRL	= 004000	\$INTAG	001135
SS5	017274	TSECC	001424	TST7	010000	WRU	= 000400	\$ITEMB	001114
SS7	017322	TSECCG	034562	TST70	030224	WSECTR	042050	\$LF	001224
STACK	= 001100	TSTNM	003610	TST71	030336	WSSYNC	040576	\$LPADR	001106
START	004634	TST1	005362	TST72	030432	WSU	= 000004	\$LPERR	001110
STKLMT=	177774	TST10	010142	TUF	= 000100	WTRK	042004	\$MNEW	045721
SWR	001140	TST11	010176	TY	037026	WWORD	040054	\$MSWR	045710
SWREG	000176	TST12	010302	TYPDS	= 104405	X	036720	\$MXCNT	044060
SW0	= 000001	TST13	010374	TYPE	= 104401	XE2	006414	\$NULL	001154
SW00	= 000001	TST14	010472	TYPOC	= 104402	X11	011672	\$NWTST=	000001
SW01	= 000002	TST15	010570	TYPON	= 104404	Y	036760	\$OCNT	047216
SW02	= 000004	TST16	010672	TYPOS	= 104403	Y11	011266	\$OMODE	047220
SW03	= 000010	TST17	010770	T.SCOP	032036	ZCODE	034576	\$OVER	044044
SW04	= 000020	TST2	006066	UN	006134	ZER	= 000400	\$PASS	001100
SW05	= 000040	TST20	011072	UNIT	001374	ZWORDS	040274	\$POWER	047464
SW06	= 000100	TST21	011216	UNITS	001354	\$AUTOB	001134	\$PWRAD	047452
SW07	= 000200	TST22	011700	UNITSL	001404	\$BDADR	001122	\$PWRDN	047312
SW08	= 000400	TST23	012026	UNLOAD	001434	\$BDDAT	001126	\$PWRMG	047446
SW09	= 001000	TST24	012146	UNS	= 040000	\$BELL	001216	\$PWRUP	047364
SW1	= 000002	TST25	012346	UPE	= 020000	\$CHARC	044574	\$QUES	001222
SW10	= 002000	TST26	012506	US1	= 000001	\$CKSWR	045120	\$RDCHR	045462
SW11	= 004000	TST27	012660	US2	= 000002	\$CMTAG	001100	\$RDLIN	045552
SW12	= 010000	TST3	006144	US4	= 000004	\$CM1	= 000006	\$RDOCT	045732
SW13	= 020000	TST30	013000	UWR	= 000010	\$CM2	= 000014	\$RDSZ	= 000011
SW14	= 040000	TST31	013076	U11	011570	\$CM3	= 000006	\$REGAD	001160
SW15	= 100000	TST32	013206	VAR1	016266	\$CM4	= 000006	\$REGO	001162
SW2	= 000004	TST33	013322	VAR2	016274	\$CNTLC	045671	\$REG1	001164

SYMBOL TABLE
\$REG2 001166
\$REG3 001170
\$REG4 001172
\$REG5 001174
\$RTNAD 030760
\$SAVR6 047462
\$SCOPE 043422
\$SETUP= 000117
\$SS1 = 000000
\$STUP = 177777
\$SVLAD 044016

\$SVPC = 000214
\$SWR = 167770
\$SWRMK= 000000
\$TIMES 001212
\$TKB 001146
\$TKCNT 044600
\$TKINT 044620
\$TKQEN= 044617
\$TKQIN 044602
\$TKQOU 044604
\$TKQSR 044606

\$TKS 001144
\$TKSRV 044670
\$TMP0 001176
\$TMP1 001200
\$TMP2 001202
\$TMP3 001204
\$TMP4 001206
\$TMP5 001210
\$TN = 000073
\$TPB 001152

\$TPFLG 001157
\$TPS 001150
\$TRAP 047222
\$TRAP2 047244
\$TRP = 000016
\$TRPAD 047256
\$TSTNM 001102
\$TTYIN 045660
\$TYPDS 044062
\$TYPE 044306

\$TYPEC 044456
\$TYPEX 044576
\$TYPOC 047020
\$TYPON 047034
\$TYPOS 046774
\$XOFF = 000023
\$XON = 000021
\$XTSTR 043460
\$\$GET4= 000000
\$OFILL 047217

. ABS. 063120 000
000000 001

ERRORS DETECTED: 0

VIRTUAL MEMORY USED: 56576 WORDS (221 PAGES)
DYNAMIC MEMORY AVAILABLE FOR 71 PAGES
CZRJHE.BIC,CZRJHE=CZRJHE.DOC,CZRJHE,[20,0]SYSMAC/M