

RK611
RK06, RK07

RK611 DSKLS CTRL5
CZR6ECO

AH-9114C-MC
FICHE 1 OF 1

MAR 1982
COPYRIGHT © 76-81
MADE IN USA



.REM 8 IDENTIFICATION

PRODUCT CODE: AC-9112C-MC
PRODUCT NAME: CZR6ECO RK611 DSKLS PRTS
DATE: AUGUST 10 1981
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: BRIAN LE BLANC

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 ERROR THAT MAY APPEAR IN THIS DOCUMENT.

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) 1976,1981 BY DIGITAL EQUIPMENT CORPORATION

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

32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87

THE TABLE OF CONTENTS

1.0 ABSTRACT

2.0 REQUIREMENTS

2.1 EQUIPMENT

2.2 PRELIMINARY PROGRAMS

3.0 OPERATING PROGRAMS

3.1 LOADING PROCEDURE

3.2 STARTING PROCEDURE

3.3 OPTIONAL SWITCH SETTING

3.4 RUN TIME

4.0 OPERATING PROCEDURES

4.1 'SOFTWARE' SWITCH REGISTER

4.2 CONTROL C (^C) OPERATION

4.3 CONTROL S (^S) OPERATION

4.4 CONTROL Q (^Q) OPERATION

5.0 PROGRAM DESCRIPTION

6.0 ERROR REPORTING

1.0 ABSTRACT

THE RK611 DISKLESS CONTROLLER DIAGNOSTIC: PART 3

A. TESTS MULTI-SECTOR DATA TRANSFERS.

B. TESTS MID-TRANSFER SEEKS.

C. TESTS CYLINDER OVERFLOW CHECKING.

D. TESTS NPR TRANSFERS TO MEMORY.

E. TESTS ECC ERROR DETECTION AND CORRECTION, BOTH

16 AND 18 BIT MODE.

F. TESTS WRITE CHECK BOTH 16 AND 18 BIT MODES AND FORCES
WRITE CHECK ERRORS.

NO RK06 DRIVE IS REQUIRED FOR PROGRAM EXECUTION.

2.0 REQUIREMENTS

2.1 EQUIPMENT

PDP-11 SYSTEM (16K CORE MEMORY)

CONSOLE TERMINAL

DECTAPE, PAPER TAPE READER, OR DECDISK

RK611 CONTROLLER

2.2 PRELIMINARY PROGRAMS

RK611 DISKLESS CONTROLLER DIAGNOSTIC: PART 1 (CZR6A)

RK611 DISKLESS CONTROLLER DIAGNOSTIC: PART 2 (CZR6B)

RK611 DISKLESS CONTROLLER DIAGNOSTIC: PART 3 (CZR6C)

RK611 DISKLESS CONTROLLER DIAGNOSTIC: PART 4 (CZR6D)

3.0 OPERATING PROCEDURES

3.1 LOADING PROCEDURE

THE PROGRAM CAN BE LOADED FROM PAPER TAPE USING ABSOLUTE LOADER OR FROM XXDP MEDIA SUPPORTED BY XXDP.

3.2 STARTING PROCEDURE

LOCATION 200 - START PROGRAM

LOCATION 204 - RESTART PROGRAM

LOCATION 214 - REQUEST BUS ADDRESS, VECTOR ADDRESS, AND PRIORITY MODIFICATION

3.3 OPTIONAL SWITCH SETTINGS

SW15 - HALT PROGRAM
SW14 - LOOP ON TEST
SW13 - INHIBIT ERROR TYPE OUT
SW12 - ABORT AFTER 20 ERRORS
SW11 - INHIBIT ITERATION COUNT
SW10 - BELL ON ERROR
SW9 - LOOP ON ERROR
SW8 - LOOP ON TEST IN SWITCHES 0-7

3.5 RUN TIME

FIRST PASS 1:40 MINUTES
SUBSEQUENT PASSES 3:40 MINUTES

4.0 OPERATING PROCEDURES

THE PROGRAM IS EXECUTED BY STARTING AT THE APPROPRIATE ADDRESS.

4.1 'SOFTWARE' SWITCH REGISTER

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E., AN 11/04 OR 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' AT ANY TIME EXCEPT WHEN THE PROGRAM IS AT A HIGHER PRIORITY PROCESSING AN RK06 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

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
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
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

REGISTER IMAGE MUST BE ENTERED. LEADING ZEROES ARE NOT REQUIRED.
'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING
ERRORS DURING SWITCH ENTRY.

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

4.2 CONTROL C (^C) OPERATION

IF ^C IS TYPED AT ANY TIME DURING THE PROGRAM EXECUTION THE
PROGRAM IS HALTED IMMEDIATELY. IF A MONITOR IS PRESENT (XXDP CHAIN,
ACT, APT) THE PROGRAM RETURNS CONTROL TO THE MONITOR. IF NO MONITOR
IS PRESENT, THE CPU IS HALTED. DEPRESSING THE CONTINUE KEY WILL DO
A PROGRAM RESTART.

4.3 CONTROL S (^S) OPERATION

IF ^S IS TYPED AT ANY TIME THE PROGRAM WILL GO INTO A STALL LOOP
UNTIL A CONTROL Q (^Q) IS TYPED.

4.4 CONTROL Q (^Q) OPERATION

IF A ^S HAS BEEN TYPED, TYPING THE ^Q CANCELS THE STALL
INITIATED BY THE ^S.

5.0 PROGRAM DESCRIPTION

**MULTI-SECTOR WRITE OPERATIONS

TEST 1 MULTI-SECTOR WRITE (PART 1)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND THE
TWO ECC WORDS. MAKE SURE THE NEXT HEADER RECOGNITION
CYCLE STARTS AND THE PROPER HEADER IS SEARCHED FOR.
PROVIDE A SECOND GOOD HEADER AND CHECK RECOGNITION
OF HEADER.

TEST 2 MULTI-SECTOR WRITE (PART 2)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 23. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER, CLOCK THROUGH 400 WORDS AND THE
TWO ECC WORDS. MAKE SURE THE NEXT HEADER RECOGNITION
CYCLE STARTS AND THE PROPER HEADER IS SEARCHED FOR.
PROVIDE A SECOND GOOD HEADER AND CHECK RECOGNITION
OF HEADER.

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
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

TEST 3 MID-TRANSFER SEEK (PART 1)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 25. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND THE
TWO WORD ECC. MAKE SURE THE CONTROLLER ISSUES
AN IMPLIED SEEK TO THE RIGHT HEAD AND CYLINDER.
MAKE SURE THE NEXT HEADER RECOGNITION PROCEEDS PROPERLY.

TEST 4 MID-TRANSFER SEEK (PART 2)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 2, SECTOR 25. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER, CLOCK THROUGH 400 WORDS AND THE
TWO WORD ECC. MAKE SURE THE CONTROLLER ISSUES

AN IMPLIED SEEK TO THE RIGHT HEAD AND CYLINDER.
MAKE SURE THE NEXT HEADER RECOGNITION PROCEEDS PROPERLY.

TEST 5 MID-TRANSFER SEEK AND 24 SECTOR FORMAT

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE DATA
DATA OF 401 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 23. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 18 BIT WORDS
AND THE 32 BIT ECC. CHECK THAT IMPLIED SEEK IS
ISSUE TO NEXT TRACK. CHECK FOR PROPER HEADER RECOGNITIO
OCCURS.

TEST 6 CYLINDER OVERFLOW (PART 1)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 632, HEAD 2, SECTOR 25. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND THE
TWO WORD ECC. MAKE SURE CYLINDER OVERFLOW ERROR DOES
NOT SET.

TEST 7 CYLINDER OVERFLOW (PART 2)

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
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

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 632, HEAD 2, SECTOR 25. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND
TWO ECC WORDS. MAKE SURE CYLINDER OVERFLOW ERROR SETS.

**NPR WRITING OF MEMORY

TEST 10 NPR WRITING OF MEMORY

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 40 DATA WORDS. VERIFY THAT THE WORDS
ARE WRITTEN PROPERLY IN MEMORY.

**ECC ERROR DETECTION/CORRECTION TESTS

TEST 11 READ DATA WITH NO ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
VERIFY THAT COMMAND COMPLETES WITHOUT ERROR.

TEST 12 READ DATA WITH ONE BIT BURST ERROR (PART 1)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
A ONE BIT BURST ERROR OF A 0 THAT SHOULD BE A 1.
VERIFY THE COUNTING OF THE POSITION
REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.

TEST 13 READ DATA WITH ONE BIT BURST ERROR (PART 2)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,

CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
A ONE BIT BURST ERROR OF A 1 THAT SHOULD BE A 0.
VERIFY THE COUNTING OF THE POSITION
REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.

TEST 14 READ DATA WITH 11 BIT BURST ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,

CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
AN 11 BIT BURST ERROR. VERIFY THE COUNING OF THE POSITI
REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.

TEST 15 READ DATA WITH 12 BIT BURST ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,

CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A
GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
A 12 BIT BURST ERROR. VERIFY THE COUNTING OF THE
POSITION REGISTER. MAKE SURE DATA CHECK, ECC HARD,
AND CONTROLLER ERROR SETS.

TEST 16 READ DATA WITH 23 BIT BURST ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,

CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A
GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
A 23 BIT BURST ERROR. VERIFY THE COUNTING OF THE
POSITION REGISTER. MAKE SURE DATA CHECK, ECC HARD,
AND CONTROLLER ERROR SETS.

TEST 17 READ DATA WITH 1 BIT ECC WORD ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A
GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
A 1 BIT ERROR IN ECC WORD 2, BIT 6. VERIFY THE
COUNTING OF THE POSITION REGISTER. MAKE SURE DATA

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
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367

CHECK AND CONTROLLER ERROR SETS.

368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423

TEST 20 18 BIT NPR WRITING OF MEMORY

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK

AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 40 DATA WORDS. VERIFY THAT THE WORDS
ARE WRITTEN PROPERLY IN MEMORY.

TEST 21 18 BIT READ DATA WITH NO ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
VERIFY THAT COMMAND COMPLETES WITHOUT ERROR.

TEST 22 18 BIT READ DATA WITH DCK

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
A ONE BIT BURST ERROR BEYOND THE 4128TH BIT OF THE SECTO
VERIFY THE COUNTING OF THE POSITION REGISTER AND THE
FINAL PATTERN AND POSITION. MAKE SURE DATA CHECK AND
CONTROLLER ERROR SETS.

TEST 23 18 BIT READ DATA WITH ECH

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS AND AND ECC INDICATING
A 12 BIT BURST ERROR. VERIFY THE COUNTING OF THE
POSITION REGISTER. MAKE SURE DATE CHECK, ECC HARD,
AND CONTROLLER ERROR SETS.

**SPECIAL READ TESTS

TEST 24 PARTIAL SECTOR READ

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 103 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
MAKE SURE ONLY 103 WORDS GOT TRANSFERRED TO MEMORY.

TEST 25 SILO UNLOAD BEFORE HEADER ERROR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
AFTER ECC SIMULATE A SECTOR PULSE AND A BAD SECTOR ERROR
MAKE SURE THE ENTIRE PREVIOUS SECTOR GETS LOADED INTO
MEMORY BEFORE BAD SECTOR ERROR AND CONTROLLER ERROR SETS

**WRITE CHECK TESTS

TEST 26 WRITE CHECK OF 400 WORDS

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
CHECK OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, THE 400 WORDS TO BE WRITE CHECKED,
AND A GOOD 32 BIT ECC. MAKE SURE NO ERRORS SET.

TEST 27 WRITE CHECK WITH DCK

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
CHECK OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, THE 400 WORDS TO BE WRITE CHECKED,
AND A 32 BIT ECC INDICATING A 1 BIT BURST ERROR.
VERIFY THAT THE POSITION REGISTER INCREMENTS PROPERLY
AND THAT THE FINAL PATTERN AND POSITION ARE CORRECT.

CHECK THAT DATA CHECK AND CONTROLLER ERROR ARE SET.

TEST 30 WRITE CHECK OF 18 BIT WORDS

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE

424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479

CHECK OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT, CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A GOOD HEADER, THE 400 WORDS TO BE WRITE CHECKED, AND A GOOD 32 BIT ECC. MAKE SURE NO ERRORS SET.

TEST 31 WRITE CHECK OF A PARTIAL SECTOR

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR. PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE CHECK OF 1 WORD TO AN RK06 IN 26 SECTOR FORMAT, CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A GOOD HEADER, TAKE ONE WORD TO BE WRITE-CHECKED, 377 WORDS NOT TO BE WRITE CHECKED, AND A GOOD 32 BIT ECC. MAKE SURE NO ERRORS SET.

TEST 32 WRITE CHECK ERROR (PART 1)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR. PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE CHECK OF 1 WORD TO AN RK06 IN 26 SECTOR FORMAT, CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A GOOD HEADER, AND ONE WORD OF DATA CONSISTING OF 000000. WRITE CHECKED DATA IS 000001. MAKE SURE WRITE CHECK ERROR REPEAT USING EACH OF THE FOLLOWING CONFIGURATIONS AS WRITE CHECKED DATA:

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

TEST 33 WRITE CHECK ERROR (PART 2)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR. PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE CHECK OF 1 WORD TO AN RK06 IN 26 SECTOR FORMAT, CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A GOOD HEADER, AND ONE WORD OF DATA CONSISTING OF 177777. WRITE CHECKED DATA IS 177776. MAKE SURE WRITE CHECK ERROR SETS. REPEAT USING EACH OF THE FOLLOWING

CONFIGURATION AS WRITE CHECKED DATA:

177775	177737	176777	157777
177773	177677	175777	137777
177767	177577	173777	077777
177757	177377	167777	

480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535

536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591

TEST 34 WRITE CHECK ERROR (PART 3)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
CHECK OF 1 WORD TO AN RK06 IN 24 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SFCTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, AND ONE WORD OF DATA CONSISTING
OF 200000. WRITE CHECK DATA IS 000000. MAKE SURE
WRITE CHECK ERROR SETS. REPEAT USING 400000 AS
SIMULATED DATA.

TEST 35 WRITE CHECK ERROR (PART 4)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
CHECK OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES SIMULATE A SECTOR PULSE,
A GOOD HEADER, 125 GOOD WORDS, AND A BAD WORD.
MAKE SURE WRITE CHECK ERROR SET. CHECK WORD COUNT
AND BUS ADDRESS.

DIAGNOSTIC MODE SECTOR PULSE

THE SECTOR PULSE IS PROVIDED BY THIS ROUTINE

DIAGNOSTIC MODE IMPLIED SEEK

THE CONTROLLER IS CLOCKED THROUGH THE SEEK MESSAGES,
GETTING STATUS, AND CLEARING THE DRIVE. SINCE THIS REQUI
DIFFERENT COUNT FOR WRITE DATA AND READ DATA/WRITE CHECK
THE ROUTINE LOOKS AT THE COMMAND THAT WAS
STARTED (L.CS1) AND SETS THE CLOCK COUNT ACCORDINGLY.

AT THE END OF THE IMPLIED SEEK THE CONTENTS OF CS1, MR1,
AND MR3 ARE CHECKED TO VERIFY THE OPERATION. ALL DISCREP
ARE REPORTED IN THE ORDER LISTED.

CALL:
JSR R4,DISEEK
ERROR 4 ;CS1 MISCOMPARE ERROR
ERROR 5 ;MR1 MISCOMPARE ERROR
ERROR 6 ;MR2 MISCOMPARE ERROR
ERROR 7 ;MR3 MISCOMPARE ERRON

DIAGNOSTIC MODE READ SYNC PATTERN (PREAMBLE)

592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647

THE 255 '0' BITS AND 1 '1' BIT ARE CLOCKED THROUGH THE R
THE CONTENTS OF MR1 ARE MONITORED TO WATCH FOR AN ERROR.
AT THE END OF THE SYNC READ, CS1 IS CHECKED TO INSURE IT
DID NOT GET CHANGED.

CALL:
JSR R4,DRSYNC
ERROR 10 ;MR1 CONTENTS IN ERROR

ERROR 11 ;CS1 CONTENTS IN ERROR

IF AN ERROR IS DETECTED, THE SUBROUTINE WILL SET UP TO A
THE PRESENT TEST UNLESS LOOP ON ERROR IS SET.

DIAGNOSTIC MODE HEADER READ AND COMPARE

THIS ROUTINE SIMULATES THE THREE HEADER WORDS READ AND
COMPARE THAT IS PART OF ALL DATA TRANSFER OPERATIONS.

THE EXPECTED HEADER IS CALCULATED FROM THE CYLINDER, TRA
AND SECTOR SPECIFIED IN THE 'L' REGISTERS. THE READING OF
3 WORDS IS DONE AND MR1 IS MONITORED TO INSURE IT DOES
NOT CHANGE. AT THE END OF THE READ/COMPARE, THE CONTENTS
OF RKCS1 IS CHECKED AND THE INCREMENT OF RKDCYL AND RKDA
VERIFIED. IF THE INCREMENT IS GOOD, THE 'L' REGISTERS AR
UPDATED TO THE NEW PACK ADDRESS IN CASE OF MULTI-SECTOR
OPERATIONS. THE BAD SECTOR ERROR DESIRED (BSEDES) SWITCH
IS TESTED AND IF IT IS NON-ZERO, THE 2ND AND 3RD HEADER
WORDS ARE CHANGED ACCORDING TO BSEDES CONTENTS. THIS
CAN BE USED TO FORCE HEADER TYPE ERROR.

CALL:
JSR R4,DHDCMP
ERROR 12 ;ERROR IN MR1
ERROR 13 ;ERROR IN CS1
ERROR 14 ;ERROR IN PACK ADDRESS

DIAGNOSTIC MODE GAP READ AND SYNC WRITE

THE READING OF THE GAP AND WRITING OF THE SYNC (OR PREAM
IS HANDLED IN THIS ROUTINE.

THE FIRST GAP BITS ARE PROCESSED IN THE HEADER COMPARE
ROUTINE AND THE REMAINDER ARE PROCESSED HERE. THEN A
CHECK IS MADE THAT WRITE GATE CAME ON AND 255 '0' BITS
AND A SINGLE '1' BIT IS WRITTEN.

CALL:
JSR R4,DWGPSN

648 ERROR 15 ;MR1 IN ERROR IN READ GAP
649 ERROR 16 ;CS1 IN ERROR AFTER READ GAP
650 ERROR 17 ;MR1 IN ERROR IN WRITE SYNC
651
652
653

654 ERROR 20 ;CS1 IN ERROR AFTER SYNC WRITE
655

656 ROUTINE CALLED:
657 JSR PC,WRTBIT
658 RETURN POINT IF ERROR IN WRTBIT
659 NO ERROR RETURN
660

661 *****
662 DIAGNOSTIC WRITE DATA AND ECC ROUTINE
663 *****

664 THE DATA IN THE OUPUT BUFFER IS WRITE SIMULATED, ECC IS
665 GENERATED AND WRITTEN, AND THE POSTAMBLE IS WRITTEN.
666

667 CALL:
668 JSR R4,DWRITE
669 ERROR 17 ;MR1 IN ERROR IN WRITING
670 ERROR 21 ;ECC ERROR IN WRITING
671 ERROR 22 ;CS1 ERROR AFTER WRITE
672 NO ERROR RETURN
673

674 ROUTINE CALLED:
675 JSR PC,WRTBIT
676 RETURN POINT IF ERROR IN WRTBIT
677 NO ERROR RETURN FROM WRTBIT
678

679 *****
680 DIAGNOSTIC MODE READ GAP AND DATA SYNC
681 *****

682 THE READING OF THE GAP AND READING OF THE SYNC (OR PREAM
683 IS HANDLED IN THIS ROUTINE.
684

685 THE FIRST GAP BITS ARE PROCESSED IN THE HEADER COMPARE
686 ROUTINE AND THE REMAINING GAP BITS ARE PROCESSED HERE.
687 THEN A CHECK IS MADE THAT READ GATE CAME ON
688 AT THE 129TH BIT OF SYNC.;

689 CALL:
690 JSR R4,DRGPSN
691 ERROR 15 ;MR1 IN ERROR IN READ GAP
692 ERROR 16 ;CS1 IN ERROR AFTER READ GAP
693
694 ERROR 32 ;MR1 IN ERROR IN READ SYNC
695 ERROR 33 ;CS1 IN ERROR AFTER SYNC READ
696

697 ROUTINE CALLED:
698 JSR PC,PDBIT
699

700 *****
701 DIAGNOSTIC MODE READ DATA OR DIAGNOSTIC WRITE CHECK ROUTINE
702 *****

704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759

.....

THIS ROUTINE IS USED TO SIMULATE THE READ OR WRITE CHECK OF DATA. THE ROUTINE WILL TRANSFER A FULL OR PARTIAL SEC

THE FIRST WORD AFTER THE CALL SPECIFIES HOW MANY WORDS ARE TO BE TRANSFERRED.

IF LESS THAN A FULL SECTOR, THE ECC AT THE END OF THE TRANSFER IS NOT CHECKED. THE ECC IS CHECKED AT THE EACH BIT IS TRANSFERRED.

EITHER 16 OR 18 BIT WORDS CAN BE SIMULATED DEPENDING ON THE STATE OF COUNT BIT IN L.CS1. IF 18 BITS ARE REQUIRED, BITS 16 AND 17 ARE ASSUMED TO BE ZERO.

THE CONTENTS OF LOCATION ECPOSX MUST BE LOADED WITH THE COUNT EXPECTED AT THE END OF THE DATA TRANSFER. THIS LOADING MUST OCCUR BEFORE THE ROUTINE IS CALLED.

ANOTHER LOCATION, ECPATX, IS SET TO 0 IF THE ECC IS EXPECTED TO BE CORRECT AND SET TO 1 IF AN ERROR IS BEING FORCED.

A 3RD LOCATION, ECCSRC, SPECIFIES THE SOURCE OF THE ECC WORDS TO BE USED IN CHECKING THE OPERATION. IF ECCSRC IS 1, THE ECC WORDS FED INTO THE DECODER ARE TAKEN FROM THE LAST TWO LOCATIONS OF OBUFF, IF ECCSRC IS 0, THE COMPUTE ECC WORDS FROM ECCHI AND ECCLO ARE USED.

CALL:

JSR R4,DREAD
LENGTH OF TRANSFER
ERROR 34 :MR1 IN ERROR READING DATA OR ECC
ERROR 35 :ECC ERROR READING DATA
ERROR 36 :CS1 ERROR AFTER READING DATA OR ECC
ERROR 41 :ECC REGISTER INCORRECT AFTER READ ECC
ERROR 42 :ERR IN ECC PAT CALCULATION
ERROR 43 :ERR IN ECC POS COUNTING

OR

JSR R4,DWRTCK
LENGTH OF TRANSFER
ERROR 53 :MR1 IN ERROR WRITE CHK OR ECC READ
ERROR 54 :ECC ERROR IN WRITE CHECK
ERROR 55 :CS1 ERROR AFTER IN WRT CHK OR ECC READ
ERROR 41 :ECC REGISTER INCORRECT AFTER READ ECC
ERROR 42 :ERR IN ECC PAT CALCULATION
ERROR 43 :ERR IN ECC POS COUNTING

ROUTINES CALLED:

RDBIT
ECCGEN

760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815

GENERATE ECC WORD

EACH TIME THIS ROUTINE IS CALLED THE ECC IS CALCULATED F
NEXT BIT TO BE READ OR WRITTEN AND THE BITS PREVIOUSLY
READ OR WRITTEN (STORED IN ECCHI AND ECCLO). THE RESULTS
THE CALCULATION ARE PLACED IN ECCHI AND ECCLO. THE LOW 0
11 BITS OF ECCHI ARE PLACED IN E.ECPT FOR EASY COMPARISSO
TO RKECPT.

CALL: JSR PC,ECCGEN
RETURN: RTS R4

SIMULATE ONE BIT OF WRITE DATA IN MAINTENANCE MODE

5835 ONE BIT OF DATA IS WRITE SIMULATED. THE CONTENTS OF
RKMRI IS COMPUTED (PRECOMP ADVANCE AND DELAY, MAINTENANC
ENCODED WRITE DATA) FOR THE PROPER TRANSITION OF THE
MAINTENANCE CLOCK AND CHECKED AT EACH TRANSITION. IF MR1
IS INCORRECT AT ANY TRANSITION, AN ERROR IS SET UP TO
CONTAIN A MESSAGE THAT IDENTIFIES THE TRANSITION WHEN TH
ERROR OCCURRED.

CALL: JSR PC,WRTBIT
RETURN: RTS PC+2 FOR NO ERROR RETURN
RTS PC FOR ERROR IN MR1

SIMULATE ONE BIT OR READ DATA IN MAINTENANCE MODE

ONE BIT OF DATA IS READ SIMULATED. NO CHECKING IS
DONE BY THIS ROUTINE.

CALL: JSR PC,RDBIT
RETURN: RTS PC

MEMORY CHECK ENABLE TRAP

IF THE PROCESSOR IN USE HAS MEMORY PARITY OPTION,
IT WILL BE ENABLED. THIS ROUTINE WILL PROCESS TRAPS CAUS
BY MEMORY PARITY ERRORS.

CLEAR INPUT BUFFER

BUILD DATA BUFFER

816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871

THE PATTERN SPECIFIED IN THE CALL IS LOADED INTO THE DAT
BUFFER. THE ENTIRE BUFFER IS ALWAYS LOADED (400(8) WORDS

LOAD 'L' REGISTERS

THE 'L' REGISTERS ARE LOADED FORM THE PARAMETERS FOLLOWI
THE SUBROUTINE CALL. DIAGNOSTIC MODE IS SET IN L.MR'
AND L.CS1 IS LOADED WITH THE COMMAND.

CALL:
JSR R4,LOADRK
.WORD CYLINDER
.BYTE ;SECTOR
.BYTE ;TRACK
.WORD ;BUFFER ADDRESS
.WORD ;WORD COUNT
.WORD ;COMMAND,FORMAT,DRIVE TYPE,&UPPER BUS AD

RETURN:
RTS R4

START THE OPERATION

THE INFORMATION IN THE 'L' REGISTERS ARE LOADED INTO THE
RK611 REGISTERS IN A STARAIGHT TRANSFER.

GET THE RK611 REGISTERS

ALL THE RK611 REGISTERS EXCEPT THE DATA BUFFER ARE
STORED IN THE 'T' REGISTERS.

'FIRST SHALL BE LAST, LAST SHALL BE FIRST' SUBROUTINE

THE CONTENTS OF R3 IS SWAPPED END FOR END, I.E. BIT 15
BECOMES BIT 0 AND VICE VERSA,BIT 14 BECOMES BIT
1 AND VICE VERSA, ETC.

CALL:
JSR R4,FSBLVV
WITH R3 LOADED WITH THE WORD TO BE SWAPPED

RETURN:
RTS R4
WITH R3 SWAPPED.

872
873
874

CHECK FOR MEMORY CHECK ENABLE

%

875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930

167400
000001

001100

000011
000012
000015
000200
177776

000000
000001
000002
000003
000004
000005
000006
000007

```
: *** REV 004 ***  
.NLIST CND,MD,MC  
.LIST ME  
.ENABL ABS,AMA  
$SWR= 167400  
$TN= 1  
.TITLE CZR6ECO RK611 DSKLS CTRL PRIS  
:*COPYRIGHT (C) 1976,1981  
:*DIGITAL EQUIPMENT CORP.  
:*MAYNARD, MASS. 01754  
:  
:*PROGRAM BY MARV TEGROTENHUIS  
:  
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
:*PACKAGE (MAINDEC-11-DZQAC-C5), JAN, 1981.  
:  
.SBTTL OPERATIONAL SWITCH SETTINGS  
:  
: SWITC H USE  
:-----  
: 15 HALT ON ERROR  
: 14 LOOP ON TEST  
: 13 INHIBIT ERROR TYPEOUTS  
: 12 ABORT PROGRAM AFTER 20 ERRORS  
: 11 INHIBIT ITERATIONS  
: 10 BELI ON ERROR  
: 9 LOOP ON ERROR  
: 8 LOOP ON TEST IN SWR<7:0>  
.SBTTL BASIC DEFINITIONS  
:  
:*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***  
STACK= 1100  
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL  
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL  
:  
:*MISCELLANEOUS DEFINITIONS  
HT= 11 ;;CODE FOR HORIZONTAL TAB  
LF= 12 ;;CODE FOR LINE FEED  
CR= 15 ;;CODE FOR CARRIAGE RETURN  
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED  
PS= 177776 ;;PROCESSOR STATUS WORD  
.EQUIV PS,PSW  
STKLMT= 177774 ;;STACK LIMIT REGISTER  
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER  
DSWR= 177570 ;;HARDWARE SWITCH REGISTER  
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER  
:  
:*GENERAL PURPOSE REGISTER DEFINITIONS  
R0= %0 ;;GENERAL REGISTER  
R1= %1 ;;GENERAL REGISTER  
R2= %2 ;;GENERAL REGISTER  
R3= %3 ;;GENERAL REGISTER  
R4= %4 ;;GENERAL REGISTER  
R5= %5 ;;GENERAL REGISTER  
R6= %6 ;;GENERAL REGISTER  
R7= %7 ;;GENERAL REGISTER
```

```
931      000006      SP=      %6          ;;STACK POINTER
932      000007      PC=      %7          ;;PROGRAM COUNTER
933
934      ;*PRIORITY LEVEL DEFINITIONS
935      000000      PR0=      0          ;;PRIORITY LEVEL 0
936      000040      PR1=      40         ;;PRIORITY LEVEL 1
937      000100      PR2=      100        ;;PRIORITY LEVEL 2
938      000140      PR3=      140        ;;PRIORITY LEVEL 3
939      000200      PR4=      200        ;;PRIORITY LEVEL 4
940      000240      PR5=      240        ;;PRIORITY LEVEL 5
941      000300      PR6=      300        ;;PRIORITY LEVEL 6
942      000340      PR7=      340        ;;PRIORITY LEVEL 7
943
944      ;*'SWITCH REGISTER' SWITCH DEFINITIONS
945      100000      SW15=     100000
946      040000      SW14=     40000
947      020000      SW13=     20000
948      010000      SW12=     10000
949      004000      SW11=     4000
950      002000      SW10=     2000
951      001000      SW09=     1000
952      000400      SW08=     400
953      000200      SW07=     200
954      000100      SW06=     100
955      000040      SW05=     40
956      000020      SW04=     20
957      000010      SW03=     10
958      000004      SW02=     4
959      000002      SW01=     2
960      000001      SW00=     1
961      .EQUIV      SW09,SW9
962      .EQUIV      SW08,SW8
963      .EQUIV      SW07,SW7
964      .EQUIV      SW06,SW6
965      .EQUIV      SW05,SW5
966      .EQUIV      SW04,SW4
967      .EQUIV      SW03,SW3
968      .EQUIV      SW02,SW2
969      .EQUIV      SW01,SW1
970      .EQUIV      SW00,SW0
971
972      ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
973      100000      BIT15=    100000
974      040000      BIT14=    40000
975      020000      BIT13=    20000
976      010000      BIT12=    10000
977      004000      BIT11=    4000
978      002000      BIT10=    2000
979      001000      BIT09=    1000
980      000400      BIT08=    400
981      000200      BIT07=    200
982      000100      BIT06=    100
983      000040      BIT05=    40
984      000020      BIT04=    20
985      000010      BIT03=    10
986      000004      BIT02=    4
```

```

987      000002      BIT01= 2
988      000001      BIT00= 1
989      .EQUIV BIT09,BIT9
990      .EQUIV BIT08,BIT8
991      .EQUIV BIT07,BIT7
992      .EQUIV BIT06,BIT6
993      .EQUIV BIT05,BIT5
994      .EQUIV BIT04,BIT4
995      .EQUIV BIT03,BIT3
996      .EQUIV BIT02,BIT2
997      .EQUIV BIT01,BIT1
998      .EQUIV BIT00,BIT0
999
1000     ;*BASIC "CPU" TRAP VECTOR ADDRESSES
1001     000004     ERRVEC= 4      ;;TIME OUT AND OTHER ERRORS
1002     000010     RESVEC= 10     ;;RESERVED AND ILLEGAL INSTRUCTIONS
1003     000014     TBITVEC=14    ;; "T" BIT
1004     000014     TRIVVEC= 14    ;;TRACE TRAP
1005     000014     BPTVEC= 14    ;;BREAKPOINT TRAP (BPT)
1006     000020     IOTVEC= 20    ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
1007     000024     PWRVEC= 24    ;;POWER FAIL
1008     000030     EMTVEC= 30    ;;EMULATOR TRAP (EMT) **ERROR**
1009     000034     TRAPVEC=34   ;; "TRAP" TRAP
1010     000060     TKVEC= 60    ;;TTY KEYBOARD VECTOR
1011     000064     TPVEC= 64    ;;TTY PRINTER VECTOR
1012     000240     PIRQVEC=240  ;;PROGRAM INTERRUPT REQUEST VECTOR
1013     .SBTTL MEMORY MANAGEMENT DEFINITIONS
1014
1015     ;*KT11 VECTOR ADDRESS
1016
1017     000250     MMVEC= 250
1018
1019     ;*KT11 STATUS REGISTER ADDRESSES
1020
1021     177572     SR0= 177572
1022     177574     SR1= 177574
1023     177576     SR2= 177576
1024     172516     SR3= 172516
1025
1026     ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
1027
1028     172300     KIPDR0= 172300
1029     172302     KIPDR1= 172302
1030     172304     KIPDR2= 172304
1031     172306     KIPDR3= 172306
1032     172310     KIPDR4= 172310
1033     172312     KIPDR5= 172312
1034     172314     KIPDR6= 172314
1035     172316     KIPDR7= 172316
1036
1037     ;*KERNEL "I" PAGE ADDRESS REGISTERS
1038
1039     172340     KIPAR0= 172340
1040     172342     KIPAR1= 172342
1041     172344     KIPAR2= 172344
1042     172346     KIPAR3= 172346
  
```

```
1043      172350      KIPAR4= 172350
1044      172352      KIPAR5= 172352
1045      172354      KIPAR6= 172354
1046      172356      KIPAR7= 172356
1047
1048      000114      MEMVEC= 114          ;MEMORY PARITY VECTOR
1049      172100      MEMBAS= 172100      ;MEM PARITY OPTION
1050      000004      WR.PAR= 4          ;WRITE BAD PARITY
1051      000001      PAR.EN= 1          ;ENABLE PARITY ENABLE
1052      120210      AVECT1= 120210     ;DEFINE RK611 VECTOR ADDRESS
1053      000005      APRIOR= >          ;DEFINE RK611 PRIORITY
1054      177440      ABASE= 177440     ;DEFINE BASE OF RK611 REGISTERS
1055
1056      .SBTTL  RK611 CONTROLLER REGISTER DEFINITION
1057
1058      000000      RKCS1= 0           ;CONTROL AND STATUS REGISTER 1
1059      000002      RKWC= 2           ;WORD COUNT REGISTER
1060      000004      RKBA= 4           ;BUS ADDRESS REGISTER
1061      000006      RKDA= 6           ;DESIRED TRACK SECTOR REGISTER
1062      000010      RKCS2= 10        ;CONTROL AND STATUS REGISTER 2
1063      000012      RKDS= 12          ;DRIVE STATUS REGISTER
1064      000014      RKER= 14          ;ERROR REGISTER
1065      000016      RKASOF= 16        ;ATTENTION SUMMARY AND OFFSET REGISTER
1066      000020      RKDCYL=20        ;DESIRED CYLINDER REGISTER
1067      000024      RKDB= 24          ;DATA BUFFER
1068      000026      RKMR1= 26        ;MAINTENANCE REGISTER 1
1069      000034      RKMR2= 34        ;MAINTENANCE REGISTER 2
1070      000036      RKMR3= 36        ;MAINTENANCE REGISTER 3
1071      000030      RKECPS= 30       ;ECC POSITION INFORMATION
1072      000032      RKECPT= 32       ;ECC PATTERN INFORMATION
1073      000022      R1SPAR= 22      ;SPARE REGISTER
1074
1075      .SBTTL  DRIVE COMMANDS
1076
1077      000001      SELDRV= 01        ;SELECT DRIVE
1078      000003      PACK= 03         ;PACK ACKNOWLEDGE
1079      000005      CLEAR= 05        ;DRIVE CLEAR
1080      000007      UNLOAD= 07       ;UNLOAD
1081      000011      SRTSPL= 11       ;START SPINDLE
1082      000013      RECAL= 13        ;RECALIBRATE
1083      000015      OFFSET= 15       ;OFFSET
1084      000017      SEEK= 17         ;SEEK
1085      000021      RDDATA= 21       ;READ DATA
1086      000023      WRDATA= 23       ;WRITE DATA
1087      000025      RDHEAD= 25      ;READ HEADER
1088      000027      WRHEAD= 27      ;WRITE HEADER AND DATA
1089      000031      WRTCHK= 31     ;WRITE CHECK
1090      000300      INTR= 300       ;GENERATE INTERRUPT TO CPU
1091
1092      .SBTTL  CONTROL AND STATUS REGISTER 1 BITS
1093
1094      000001      GO= BIT0          ;GO BIT
1095      000100      IE= BIT6          ;INTERRUPT ENABLE
1096      000200      RDY= BIT7         ;CONTROLLER READY
1097      000400      BA16= BIT8        ;BUS ADDRESS BIT 16
1098      001000      BA17= BIT9        ;BUS ADDRESS BIT 17
```

1099	002000	CDT=	BIT10	:CONTROLLER DRIVE TYPE (0-RK06)
1100	004000	CTO=	BIT11	:CONTROLLER TIMED OUT WAITING FOR
1101				: DRIVE RESPONSE
1102	010000	CFMT=	BIT12	:CONTROLLER DRIVE FOPMAT (0=26 SECTOR, 1=24 SECTOR)
1103	020000	SPAR=	BIT13	:DRIVE BUS PARITY ERROR DETECTED BY CONTROLLER
1104	040000	DI=	BIT14	:DRIVE INTERRUPT
1105	100000	CERR=	BIT15	:CONTROLLER ERROR
1106	100000	CCLR=	BIT15	:CONTROLLER CLFAR

.SBTTL CONTROL AND STATUS REGISTER 2 BITS

1110	000007	DRVMSK=	7	:MASK FOR DRIVE SELECTION CODE
1111	000010	RLS=	BIT3	:DESELECT OR RELEASE DRIVE IN BITS 0-2
1112	000020	BAI=	BIT4	:BUS ADDRESS INCREMENT INHIBIT
1113	000040	SCLR=	BIT5	:CLEAR CONTROLLER AND ALL DRIVES
1114	000100	IR=	BIT6	:INPUT READY
1115	000200	OR=	BIT7	:OUTPUT READY
1116	000400	UFE=	BIT8	:UNIT FIELD ERROR
1117	001000	MDS=	BIT9	:MULTIPLE DRIVE SELECT
1118	002000	PGE=	BIT10	:PROGRAMMING ERROR
1119	004000	NEM=	BIT11	:NON-EXISTENT MEMORY
1120	010000	NED=	BIT12	:NON-EXISTENT DRIVE
1121	020000	UPE=	BIT13	:UNIBUS PARITY ERROR
1122	040000	WCE=	BIT14	:WRITE CHECK ERROR
1123	100000	DLT=	BIT15	:DATA LATE ERROR

.SBTTL ERROR REGISTER BIT DEFINITION

1127	000001	ILF=	BIT0	:ILLEGAL FUNCTION CODE
1128	000002	SKI=	BIT1	:SEEK INCOMPLETE
1129	000004	NXF=	BIT2	:NON-EXECUTABLE DRIVE FUNCTION
1130	000010	DRPAR=	BIT3	:DRIVE DETECTED DRIVE BUS PARITY ERROR
1131	000020	FMTE=	BIT4	:FORMAT ERROR
1132	000040	DTYE=	BIT5	:DRIVE TYPE ERROR
1133	000100	ECH=	BIT6	:ECC HARD
1134	000200	BSE=	BIT7	:BAD SECTOR ERROR
1135	000400	HVRC=	BIT8	:HEADER VRC ERROR
1136	001000	COE=	BIT9	:CYLINDER ADDRESS OVERFLOW ERROR
1137	002000	IDAE=	BIT10	:INVALID DISK ADDRESS ERROR
1138	004000	WLE=	BIT11	:WRITE LOCK ERROR
1139	010000	DTE=	BIT12	:DRIVE TIMING ERROR
1140	020000	OPI=	BIT13	:OPERATION (SEARCH) INCOMPLETE
1141	040000	UNS=	BIT14	:DRIVE UNSAFE
1142	100000	DCK=	BIT15	:DATA CHECK

.SBTTL STATUS REGISTER BIT DEFINITION

1146	000001	DRA=	BIT0	:DRIVE AVAILABLE (CONTROLLER IS SET IF
1147				: THIS BIT IS RESET)
1148	000004	OFST=	BIT2	:DRIVE OFFSET
1149	000010	ACLO=	BIT3	:AC LOW
1150	000020	SPDLSS=	BIT4	:SPEED LOSS
1151	000040	DROT=	BIT5	:DRIVE OFF TRACK
1152	000100	VV=	BIT6	:VOLUME VALID
1153	000200	DRDY=	BIT7	:DRIVE READY
1154	000400	DDT=	BIT8	:DRIVE TYPE (0=RK06)

1155	004000	WRL=	BIT11	:WRITE LOCK
1156	020000	PIP=	BIT13	:POSITIONING IN PROGRESS
1157	040000	DSC=	BIT14	:DRIVE STATUS CHANGE
1158	100000	SVAL=	BIT15	:STATUS VALID


```

1159
1160      .SBTTL MAINTENANCE REGISTER 1 BIT DEFINITION
1161
1162      000017      MESMSK= 17      ;MESSAGE MASK
1163
1164      000020      PAT= BIT4      ;FORCE EVEN PARITY ON DRIVE MESSAGE LINES
1165      000040      DMD= BIT5      ;DIAGNOSTIC MODE
1166      000100      MSP= BIT6      ;MAINTENANCE SFCTOR PULSE
1167      000200      MIND= BIT7     ;MAINTENANCE INDEX
1168      000400      MCLK= BIT8     ;MAINTENANCE CLOCK
1169      001000      MERD= BIT9     ;MAINTENANCE ENCODED READ DATA
1170      002000      MEWD= BIT10    ;MAINTENACNE ENCODED WRITE DATA
1171      004000      PCA= BIT11    ;PRECOMPENSATION ADVANCE
1172      010000      PCD= BIT12    ;PRECOMPENSATION DELAY
1173      020000      ECCW= BIT13   ;ECC WORD IS BEING READ OR WRITTEN
1174      040000      WRTGAT= BIT14 ;WRITE GATE
1175      100000      RDGATE= BIT15 ;READ GATE
1176
1177      .SBTTL TRANSMITTED MESSAGE A
1178
1179      000020      S.SEEK= BIT4    ;SEEK COMMAND
1180      000040      S.RECL= BIT5   ;RECALIBRATE COMMAND
1181      000100      S.STSP= BIT6   ;START SPINDLE COMMAND
1182      000200      S.RTC= BIT7    ;DRIVE RETURN TO CENTERLINE COMMAND
1183      000400      S.CLR= BIT8    ;CLEAR ERROR AND DSC
1184      001000      S.FMT= BIT9    ;FORMAT
1185      002000      S.UNLD= BIT10   ;UNLOAD
1186      004000      S.PACK= BIT11  ;SET VOLUME VALID (PACK ACKNOWLEDGE)
1187
1188      .SBTTL TRAP CATCHER
1189
1190      000000      .=0
1191      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
1192      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
1193      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
1194      000174      000000      .=174
1195      000176      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
1196      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
1197      000200      000137      002400      .SBTTL STARTING ADDRESS(ES)
1198      000204      000137      002370      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
1199      000214      000137      002360      JMP RESTRT ;;UMP TO RESTART ROUTINE
1200      000214      000137      002360      .=214
1201      000214      000137      002360      JMP PARM ;;JUMP TO OPERATOR ASSIGNED PARMETERS
1202
1203      .SBTTL ACT11 HOOKS
1204
1205      000220      000046      ;*****
1206      000046      023276      ;HOOKS REQUIRED BY ACT11
1207      000046      000052      $SVPC=      ;SAVE PC
1208      000052      000000      .=46
1209      000052      000000      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
1210      000114      031726      .=52
1211      000114      000340      .WORD 0      ;;2)SET LOC.52 TO ZERO
1212      000114      001000      .=$SVPC      ;; RESTORE PC
1213      000116      001000      .=MEMVEC
1214      001000      .-1000      MEMERR
                                PR7
                                .-1000

```

1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236

00C024
000044

001000
000024
000200
000044
001000
001000

001000
000000
001250
000132
001604
000144
000030

.SBTTL APT PARAMETER BLOCK

```
*****  
:SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT  
*****  
.$X=      ;;SAVE CURRENT LOCATION  
-24      ;;SET POWER FAIL TO POINT TO START OF PROGRAM  
200      ;;FOR APT START UP  
=44      ;;POINT TO APT INDIRECT ADDRESS PNTR.  
$APTHDR  ;;POINT TO APT HEADER BLOCK  
=.$X     ;;RESET LOCATION COUNTER  
*****  
:SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC  
:INTERFACE SPEC.
```

```
$APTHD:  
$HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.  
$MADR:  .WORD $MAIL  ;;ADDRESS OF APT MAILBOX (BITS 0-15)  
$TSTM:  .WORD 90.    ;;RUN TIM OF LONGEST TEST  
$PASTM: .WORD 900.   ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)  
$UNITM: .WORD 100.   ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT  
        .WORD $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
```

```
1237 .SBTTL COMMON TAGS
1238
1239 ::*****
1240 :*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
1241 :*USED IN THE PROGRAM.
1242
1243 001100 .=1100
1244 001100 $CMTAG: ::START OF COMMON TAGS
1245 001100 000000 .WORD 0
1246 001102 000 .STSTNM: .BYTE 0 ::CONTAINS THE TEST NUMBER
1247 001103 000 $ERFLG: .BYTE 0 ::CONTAINS ERROR FLAG
1248 001104 000000 $ICNT: .WORD 0 ::CONTAINS SUBTEST ITERATION COUNT
1249 001106 000000 $LPADR: .WORD 0 ::CONTAINS SCOPE LOOP ADDRESS
1250 001110 000000 $LPERR: .WORD 0 ::CONTAINS SCOPE RETURN FOR ERRORS
1251 001112 000000 $ERTTL: .WORD 0 ::CONTAINS TOTAL ERRORS DETECTED
1252 001114 000 $ITEMB: .BYTE 0 ::CONTAINS ITEM CONTROL BYTE
1253 001115 001 $ERMAX: .BYTE 1 ::CONTAINS MAX. ERRORS PER TEST
1254 001116 000000 $ERRPC: .WORD 0 ::CONTAINS PC OF LAST ERROR INSTRUCTION
1255 001120 000000 $GDADR: .WORD 0 ::CONTAINS ADDRESS OF 'GOOD' DATA
1256 001122 000000 $BDADR: .WORD 0 ::CONTAINS ADDRESS OF 'BAD' DATA
1257 001124 000000 $GDDAT: .WORD 0 ::CONTAINS 'GOOD' DATA
1258 001126 000000 $BDDAT: .WORD 0 ::CONTAINS 'BAD' DATA
1259 001130 000000 .WORD 0 ::RESERVED--NOT TO BE USED
1260 001132 000000 .WORD 0
1261 001134 000 $AUTOB: .BYTE 0 ::AUTOMATIC MODE INDICATOR
1262 001135 000 $INTAG: .BYTE 0 ::INTERRUPT MODE INDICATOR
1263 001136 000000 .WORD 0
1264 001140 177570 $WR: .WORD DSWR ::ADDRESS OF SWITCH REGISTER
1265 001142 177570 DISP_AY: .WORD DDISP ::ADDRESS OF DISPLAY REGISTER
1266 001144 177560 $TKS: 177560 ::TTY KBD STATUS
1267 001146 177562 $TKB: 177562 ::TTY KBD BUFFER
1268 001150 177564 $TPS: 177564 ::TTY PRINTER STATUS REG. ADDRESS
1269 001152 177566 $TPB: 177566 ::TTY PRINTER BUFFER REG. ADDRESS
1270 001154 000 $NULL: .BYTE 0 ::CONTAINS NULL CHARACTER FOR FILLS
1271 001155 002 $FILLS: .BYTE 2 ::CONTAINS # OF FILLER CHARACTERS REQUIRED
1272 001156 012 $FILLC: .BYTE 12 ::INSERT FILL CHARS. AFTER A 'LINE FEED'
1273 001157 000 $TPFLG: .BYTE 0 ::'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
1274 001160 000000 $REGAD: .WORD 0 ::CONTAINS THE ADDRESS FROM
1275 ::WHICH ($REGO) WAS OBTAINED
1276 001162 000000 $REG0: .WORD 0 ::CONTAINS (($REGAD)+0)
1277 001164 000000 $REG1: .WORD 0 ::CONTAINS (($REGAD)+2)
1278 001166 000000 $REG2: .WORD 0 ::CONTAINS (($REGAD)+4)
1279 001170 000000 $REG3: .WORD 0 ::CONTAINS (($REGAD)+6)
1280 001172 000000 $REG4: .WORD 0 ::CONTAINS (($REGAD)+10)
1281 001174 000000 $REG5: .WORD 0 ::CONTAINS (($REGAD)+12)
1282 001176 000000 $REG6: .WORD 0 ::CONTAINS (($REGAD)+14)
1283 001200 000000 $REG7: .WORD 0 ::CONTAINS (($REGAD)+16)
1284 001202 000000 $TMP0: .WORD 0 ::USER DEFINED
1285 001204 000000 $TMP1: .WORD 0 ::USER DEFINED
1286 001206 000000 $TMP2: .WORD 0 ::USER DEFINED
1287 001210 000000 $TMP3: .WORD 0 ::USER DEFINED
1288 001212 000000 $TMP4: .WORD 0 ::USER DEFINED
1289 001214 000000 $TMP5: .WORD 0 ::USER DEFINED
1290 001216 000000 $TMP6: .WORD 0 ::USER DEFINED
1291 001220 000000 $TMP7: .WORD 0 ::USER DEFINED
1292 001222 000000 $TMP8: .WORD 0 ::USER DEFINED
```

1293	001224	000000	\$TMP11: .WORD 0	::USER DEFINED
1294	001226	000000	\$TMP12: .WORD 0	::USER DEFINED
1295	001230	000000	\$TMP13: .WORD 0	::USER DEFINED
1296	001232	000000	\$TMP14: .WORD 0	::USER DEFINED
1297	001234	000000	\$TIMES: 0	::MAX. NUMBER OF ITERATIONS
1298	001236	000000	\$ESCAPE: 0	::ESCAPE ON ERROR ADDRESS
1299	001240	177607 000377	\$BELL: .ASCIZ <207><377><377>	::CODE FOR BELL
1300	001244	077	\$QUES: .ASCII /?/	::QUESTION MARK
1301	001245	015	\$CRLF: .ASCII <15>	::CARRIAGE RETURN
1302	001246	000012	\$LF: .ASCIZ <12>	::LINE FEED
1303			:*****	
1304			.SBTTL APT MAILBOX-ETABLE	
1305			:*****	
1306			.EVEN	
1308	001250		\$MAIL: .WORD	::APT MAILBOX
1309	001250	000000	\$MSGTY: .WORD AMSGTY	::MESSAGE TYPE CODE
1310	001252	000000	\$FATAL: .WORD AFATAL	::FATAL ERROR NUMBER
1311	001254	000000	\$TESTN: .WORD ATESTN	::TEST NUMBER
1312	001256	000000	\$PASS: .WORD APASS	::PASS COUNT
1313	001260	000000	\$DEVCT: .WORD ADEVCT	::DEVICE COUNT
1314	001262	000000	\$UNIT: .WORD AUNIT	::I/O UNIT NUMBER
1315	001264	000000	\$MSGAD: .WORD AMSGAD	::MESSAGE ADDRESS
1316	001266	000000	\$MSGLG: .WORD AMSGLG	::MESSAGE LENGTH
1317	001270		\$ETABLE: .WORD	::APT ENVIRONMENT TABLE
1318	001270	000	\$ENV: .BYTE AENV	::ENVIRONMENT BYTE
1319	001271	000	\$ENVM: .BYTE AENVM	::ENVIRONMENT MODE BITS
1320	001272	000000	\$SWREG: .WORD ASWREG	::APT SWITCH REGISTER
1321	001274	000000	\$USWR: .WORD AUSWR	::USER SWITCHES
1322	001276	000000	\$CPUOP: .WORD ACPUOP	::CPU TYPE, OPTIONS
1323			BITS 15-11=CPU TYPE	
1324			11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05	
1325			11/70=06, PDQ=07, Q=10	
1326			BIT 10-REAL TIME CLOCK	
1327			BIT 9-FLOATING POINT PROCESSOR	
1328			BIT 8-MEMORY MANAGEMENT	
1329	001300	000	\$MAMS1: .BYTE AMAMS1	::HIGH ADDRESS, M.S. BYTE
1330	001301	000	\$MTYP1: .BYTE AMTYP1	::MEM. TYPE, BLK#1
1331			MEM. TYPE BYTE -- (HIGH BYTE)	
1332			900 NSEC CORE=001	
1333			300 NSEC BIPOLAR=002	
1334			500 NSEC MOS=003	
1335	001302	000000	\$MADR1: .WORD AMADR1	::HIGH ADDRESS, BLK#1
1336			MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE	
1337	001304	000	\$MAMS2: .BYTE AMAMS2	::HIGH ADDRESS, M.S. BYTE
1338	001305	000	\$MTYP2: .BYTE AMTYP2	::MEM. TYPE, BLK#2
1339	001306	000000	\$MADR2: .WORD AMADR2	::MEM. LAST ADDRESS, BLK#2
1340	001310	000	\$MAMS3: .BYTE AMAMS3	::HIGH ADDRESS, M.S. BYTE
1341	001311	000	\$MTYP3: .BYTE AMTYP3	::MEM. TYPE, BLK#3
1342	001312	000000	\$MADR3: .WORD AMADR3	::MEM. LAST ADDRESS, BLK#3
1343	001314	000	\$MAMS4: .BYTE AMAMS4	::HIGH ADDRESS, M.S. BYTE
1344	001315	000	\$MTYP4: .BYTE AMTYP4	::MEM. TYPE, BLK#4
1345	001316	000000	\$MADR4: .WORD AMADR4	::MEM. LAST ADDRESS, BLK#4
1346	001320	120210	\$VECT1: .WORD AVECT1	::INTERRUPT VECTOR#1, BUS PRIORITY#1
1347	001322	000000	\$VECT2: .WORD AVECT2	::INTERRUPT VECTOR#2, BUS PRIORITY#2
1348	001324	177440	\$BASE: .WORD ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST

CZR6ECO RK611 DSKLS CTRL PRT5
CZR6EC.P11 14-SEP-81 13:57

MACY11 30(1046) 14-SEP-81 15:14 PAGE 29
APT MAILBOX-ETABLE

C 3

SEQ 0028

C1
C2

1349 001326 000000
1350 001330
1351

\$DEV: .WORD ADEV: ;;DEVICE MAP
\$ETEND:
.MEXIT

1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366 001330
1367
1368 001330 043176
1369 001332 041555
1370 001334 040002
1371 001336 040320
1372
1373 001340 000000
1374 001342 000000
1375 001344 000000
1376 001346 000000
1377
1378 001350 000000
1379 001352 000000
1380 001354 000000
1381 001356 000000
1382
1383 001360 043243
1384 001362 043307
1385 001364 040006
1386 001366 040324
1387
1388
1389 001370 043243
1390 001372 043326
1391 001374 040024
1392 001376 040354
1393
1394
1395 001400 043243
1396 001402 043345
1397 001404 040034
1398 001406 040400
1399
1400
1401 001410 043243
1402 001412 043364
1403 001414 040044
1404 001416 040424
1405
1406
1407 001420 043455

.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

\$ERRTB:

ERROR 1 ;UNEXPECTED PARITY ERROR
EM000
DH000C
DT000
DF000
ERROR 2 ;UNUSED
0
0
0
0
ERROR 3 ;UNUSED
0
0
0
0
ERROR 4 ;ERROR ATTEMPTING IMPLIED SEEK
;CS1 IN ERROR
EM5004
E5004A
DT0004
DF0004
ERROR 5 ;MR1 IN ERROR ATTEMPTING IMPLIED SEEK
EM5004
E5004B
DT0005
DF0005
ERROR 6 ;MR2 IN ERROR ATTEMPTING IMPLIED SEEK
EM5004
E5004C
DT0006
DF0006
ERROR 7 ;MR3 IN ERROR ATTEMPTING IMPLIED SEEK
EM5004
E5004D
DT0007
DF0007
ERROR 10 ;MR1 ERROR READING PREAMBLE
EM5010

1408	001422	043326	E5004B	
1409	001424	040054	DT0010	
1410	001426	040450	DF0010	
1411				
1412			ERROR 11	;CS1 ERROR READING PREAMBLE
1413	001430	043455	EM5010	
1414	001432	043307	E5004A	
1415	001434	040072	DT0011	
1416	001436	040474	DF0011	
1417				
1418			ERROR 12	;MR1 IN ERROR HEADER READ/COMPARE
1419	001440	043535	EM5011	
1420	001442	043326	E5004B	
1421	001444	040054	DT0010	
1422	001446	040450	DF0010	
1423				
1424			ERROR 13	;CS1 IN ERROR HEADER READ/COMPARE
1425	001450	043535	EM5011	
1426	001452	043307	E5004A	
1427	001454	040072	DT0011	
1428	001456	040474	DF0011	
1429				
1430			ERROR 14	;PACK ADDRESS ERROR HDR READ/COMPARE
1431	001460	043535	EM5011	
1432	001462	043610	E5011A	
1433	001464	040116	DT0014	
1434	001466	040524	DF0014	
1435				
1436			ERROR 15	;MR1 ERROR READING GAP
1437	001470	043640	EM5015	
1438	001472	043326	E5004B	
1439	001474	040054	DT0010	
1440	001476	040450	DF0010	
1441				
1442			ERROR 16	;CS1 ERROR AFTER READING GAP
1443	001500	043455	EM5010	
1444	001502	043307	E5004A	
1445	001504	040072	DT0011	
1446	001506	040474	DF0011	
1447				
1448			ERROR 17	;ERROR IN WRITING
1449	001510	000000	EMW: 0	;REGISTER IN ERROR AND CLOCK
1450	001512	000000	0	;TRANSITION IS IDENTIFIED
1451	001514	040132	DT0017	
1452	001516	040550	DF0017	
1453				
1454			ERROR 20	;ERROR IN CS1 AFTER WRITING SYNC
1455	001520	043675	EM5020	
1456	001522	043307	E5004A	
1457	001524	040072	DT0011	
1458	001526	040474	DF0011	
1459				
1460			ERROR 21	;ERROR IN ECC WORDS
1461	001530	043754	EM5021	
1462	001532	044022	E5021A	
1463	001534	040154	DT0021	

1464	001536	040574		DF0021	
1465					
1466			:	ERROR 22	:ERR IN CS1 AFTER WRITE DATA
1467	001540	043754		EM5021	
1468	001542	043307		E5004A	
1469	001544	040072		DT0011	
1470	001546	040474		DF0011	
1471					
1472			:	ERROR 23	:ERROR IN MR2
1473	001550	044052		EM5023	:DOING MULTI-SECTOR WRITE
1474	001552	043345		E5004C	
1475	001554	040034		DT0006	
1476	001556	040400		DF0006	
1477					
1478			:	ERROR 24	:MR3 IN ERROR
1479	001560	044052		EM5023	:DOING MULTI-SECTOR WRITE
1480	001562	043364		E5004D	
1481	001564	040044		DT0007	
1482	001566	040424		DF0007	
1483					
1484			:	ERROR 25	:CS1 IN ERROR
1485	001570	044052		EM5023	:AFTER MULTI-SECTOR WRITE
1486	001572	043307		E5004A	
1487	001574	040006		DT0004	
1488	001576	040324		DF0004	
1489					
1490			:	ERROR 26	
1491	001600	044472		EM5026	:NO COE ERROR
1492	001602	000000	EH026:	.WORD 0	
1493	001604	040166		DT0026	
1494	001606	040620		DF0026	
1495					
1496			:	ERROR 27	
1497	001610	044632		EM5027	:OTHER ERROR WHILE FORCING COE
1498	001612	041524		DH000A	
1499	001614	040166		DT0026	
1500	001616	040620		DF0026	
1501					
1502			:	ERROR 30	
1503	001620	045004		EM5030	:UNEXPECTED CYL OVERFLOW
1504	001622	000000	EH030:	.WORD 0	
1505	001624	040166		DT0026	
1506	001626	040620		DF0026	
1507					
1508			:	ERROR 31	
1509	001630	045077		EM5031	:UNEXPECTED ERROR TESTING COE
1510	001632	000000	EH031:	.WORD 0	
1511	001634	040166		DT0026	
1512	001636	040620		DF0026	
1513					
1514			:	ERROR 32	:MR1 ERROR READING DATA SYNC
1515	001640	045164		EM5032	
1516	001642	043326		E5004B	
1517	001644	040054		DT0010	
1518	001646	040450		DF0010	
1519					

1520			:	ERROR 33	;CS1 ERROR AFTER READING SYNC
1521	001650	045164		EM5032	
1522	001652	043307		E5004A	
1523	001654	040072		DT0011	
1524	001656	040474		DF0011	
1525					
1526			:	ERROR 34	;MR1 IN ERROR READING DATA OR ECC
1527	001660	045242		EM5034	
1528	001662	043326		E5004B	
1529	001664	040054		DT0010	
1530	001666	040450		DF0010	
1531					
1532			:	ERROR 35	;ECC ERROR READING DATA
1533	001670	045307		EM5035	
1534	001672	044022		E5021A	
1535	001674	040154		DT0021	
1536	001676	040574		DF0021	
1537					
1538			:	ERROR 36	;CS1 ERROR AFTER READING DATA OR ECC
1539	001700	045242		EM5034	
1540	001702	043307		E5004A	
1541	001704	040072		DT0011	
1542	001706	040474		DF0011	
1543					
1544			:	ERROR 37	;DATA COMPARE ERROR (1ST MESSAGE)
1545	001710	045350		EM5037	
1546	001712	041524		DH000A	
1547	001714	040210		DT0037	
1548	001716	040654		DF0037	
1549					
1550			:	ERROR 40	;DATA COMPARE ERROR (2ND THRU 10TH ERR)
1551	001720	000000		0	
1552	001722	000000		0	
1553	001724	040214		DT0040	
1554	001726	040674		DF0040	
1555					
1556			:	ERROR 41	;ECC REGISTER INCORRECT AFTER ECC READ
1557	001730	045424		EM5041	
1558	001732	041524		DH000A	
1559	001734	040222		DT0041	
1560	001736	040700		DF0041	
1561					
1562			:	ERROR 42	;ERROR IN ECC PAT CALCULATION AFTER ECC ERROR
1563	001740	045511		EM5042	
1564	001742	041524		DH000A	
1565	001744	040154		DT0021	
1566	001746	040720		DF0042	
1567					
1568			:	ERROR 43	;ERROR IN ECC POSITION COUNTING AFTER ECC ERROR
1569	001750	045572		EM5043	
1570	001752	041524		DH000A	
1571	001754	040240		DT0043	
1572	001756	040740		DF0043	
1573					
1574			:	ERROR 44	;ERROR AFTER PROCESSING DATA CHECK ERR
1575	001760	045651		EM5044	

1576	001762	043307	E5004A	
1577	001764	040006	DT0004	
1578	001766	040324	DF0004	
1579				
1580			ERROR 45	
1581	001770	045651	EM5044	;ERROR AFTER PROCESSING DATA CHECK ERR
1582	001772	043403	E5004E	
1583	001774	040252	DT0045	
1584	001776	040760	DF0045	
1585				
1586			ERROR 46	
1587	002000	043243	EM50046	;TRANSFER LENGTH ERROR PARTIAL SEC READ
1588	002002	041524	DH000A	
1589	002004	040262	DT0046	
1590	002006	041004	DF0046	
1591				
1592			ERROR 47	
1593	002010	046013	EM5047	;BSE DID NOT PREVENT DA OR DC INCREMENT
1594	002012	043610	E5011A	
1595	002014	040116	DT0014	
1596	002016	040524	DF0014	
1597				
1598			ERROR 50	
1599	002020	046106	EM5050	;BSE DID NOT CAUSE CERR
1600	002022	043307	E5004A	
1601	002024	040006	DT0004	
1602	002026	041024	DF0050	
1603				
1604			ERROR 51	
1605	002030	046161	EM5051	;BSE FAILED TO SET WHEN EXPECTED
1606	002032	043403	E5004E	
1607	002034	040252	DT0045	
1608	002036	040760	DF0045	
1609				
1610			ERROR 52	
1611	002040	046236	EM5052	;DATA XFER ABORT TO SOON
1612	002042	041524	DH000A	
1613	002044	040262	DT0046	
1614	002046	041004	DF0046	
1615				
1616			ERROR 53	
1617	002050	046356	EM5053	;MRI ERROR IN WRT CHK OR ECC READ AFTER WRT CHK
1618	002052	043326	E5004B	
1619	002054	040054	DT0010	
1620	002056	040450	DF0010	
1621				
1622			ERROR 54	
1623	002060	046445	EM5054	;ECC ERROR IN WRITE CHECK OP
1624	002062	044022	E5021A	
1625	002064	040154	DT0021	
1626	002066	040574	DF0021	
1627				
1628			ERROR 55	
1629	002070	046356	EM5053	;CSI ERROR AFTER WRT CHK OR ECC READ IN WRT CHK
1630	002072	043307	E5004A	
1631	002074	040072	DT0011	

1632	002076	040474	DF0011	
1633				
1634			ERROR 56	
1635	002100	045242	EM5034	;CS1 ERROR AFTER READ DATA OR ECC
1636	002102	043307	E5004A	
1637	002104	040006	DT0004	
1638	002106	040324	DF0004	
1639				
1640			ERROR 57	
1641	002110	046356	EM5053	;CS1 ERROR AFTER WRITE CHECK OR ECC REAL
1642	002112	043307	E5004A	
1643	002114	040006	DT0004	
1644	002116	040324	DF0004	
1645				
1646			ERROR 60	
1647	002120	046510	EM5060	;WCE FAILED TO SET IN 16 BIT MODE
1648	002122	041524	DH000A	
1649	002124	040210	DT0037	
1650	002126	041050	DF0060	
1651				
1652			ERROR 61	
1653	002130	046567	EM5061	;TRANSFER LENGTH PROBLEM - RKBA
1654	002132	043421	E5004F	
1655	002134	040262	DT0046	
1656	002136	041070	DF0061	
1657				
1658			ERROR 62	
1659	002140	046567	EM5061	;TRANSFER LENGTH PROBLEM - RKWC
1660	002142	043437	E5004G	
1661	002144	040272	DT0062	
1662	002146	041114	DF0062	
1663				
1664			ERROR 63	
1665	002150	046644	EM5063	;WCE FAILED TO SET IN 18 BIT MODE
1666	002152	043437	E5004G	
1667	002154	040302	DT0063	
1668	002156	041140	DF0063	
1669				
1670			.SBT*L	TEMPORARY STORAGE FOR RK611 CONTROLLER REGISTER
1671	002160	000000	T.CS1:	.WORD 0 ;CONTROL AND STATUS REGISTER 1
1672	002162	000000	T.WC:	.WORD 0 ;WORD COUNT REGISTER
1673	002164	000000	T.BA:	.WORD 0 ;BUS ADDRESS REGISTER
1674	002166	000000	T.DA:	.WORD 0 ;DESIRED TRACK SECTOR REGISTER
1675	002170	000000	T.CS2:	.WORD 0 ;CONTROL AND STATUS REGISTER 2
1676	002172	000000	T.DS:	.WORD 0 ;DRIVE STATUS REGISTER
1677	002174	000000	T.ER:	.WORD 0 ;ERROR REGISTER
1678	002176	000000	T.ASOF:	.WORD 0 ;ATTENTION SUMMARY AND OFFSET REGISTER
1679	002200	000000	T.DCYL:	.WORD 0 ;DESIRED CYLINDER REGISTER
1680	002202	000000	T.DB:	.WORD 0 ;DATA BUFFER
1681	002204	000000	T.MR1:	.WORD 0 ;MAINTENANCE REGISTER 1
1682	002206	000000	T.MR2:	.WORD 0 ;MAINTENANCE REGISTER 2
1683	002210	000000	T.MR3:	.WORD 0 ;MAINTENANCE REGISTER 3
1684	002212	000000	T.ECPS:	.WORD 0 ;ECC POSITION INFORMATION
1685	002214	000000	T.ECPT:	.WORD 0 ;ECC PATTERN INFORMATION
1686	002216	000000	T.SPARE:	.WORD 0 ;SPARE REGISTER
1687				

```
1688 .SBTTL EXPECTED RK611 CONTROLLER REGISTERS
1689
1690 002220 000000 E.CS1: .WORD 0 ;CONTROL AND STATUS REGISTER 1
1691 002222 000000 E.WC: .WORD 0 ;WORD COUNT REGISTER
1692 002224 000000 E.BA: .WORD 0 ;BUS ADDRESS REGISTER
1693 002226 000000 E.DA: .WORD 0 ;DESIRED TRACK SECTOR REGISTER
1694 002230 000000 E.CS2: .WORD 0 ;CONTROL AND STATUS REGISTER 2
1695 002232 000000 E.DS: .WORD 0 ;DRIVE STATUS REGISTER
1696 002234 000000 E.ER: .WORD 0 ;ERROR REGISTER
1697 002236 000000 E.ASOF: .WORD 0 ;ATTENTION SUMMARY AND OFFSET REGISTER
1698 002240 000000 E.DCYL: .WORD 0 ;DESIRED CYLINDER REGISTER
1699 002242 000000 E.DB: .WORD 0 ;DATA BUFFER
1700 002244 000000 E.MR1: .WORD 0 ;MAINTENANCE REGISTER 1
1701 002246 000000 E.MR2: .WORD 0 ;MAINTENANCE REGISTER 2
1702 002250 000000 E.MR3: .WORD 0 ;MAINTENANCE REGISTER 3
1703 002252 000000 E.ECPS: .WORD 0 ;ECC POSITION INFORMATION
1704 002254 000000 E.ECPT: .WORD 0 ;ECC PATTERN INFORMATION
1705 002256 000000 E.SPAR: .WORD 0 ;SPARE REGISTER
1706
1707 .SBTTL 'L' REGISTERS FOR COMMAND START
1708 002260 000000 L.CS1: .WORD 0 ;CONTROL AND STATUS 1
1709 002262 000000 L.WC: .WORD 0 ;WORD COUNT
1710 002264 000000 L.BA: .WORD 0 ;BUFER ADDRESS
1711 002266 L.DA: ;DESIRED
1712 002266 000 L.DS: .BYTE 0 ;SECTOR
1713 002267 000 L.DT: .BYTE 0 ;TRACK
1714 002270 000000 L.CS2: .WORD 0 ;CONTROL AND STATUS
1715 002272 000000 L.ASOF: .WORD 0 ;ATTENTION AND OFFSET
1716 002274 000000 L.DCYL: .WORD 0 ;DESIRED CYLINDER
1717 002276 000000 L.MR1: .WORD 0 ;MAINT. REG 1
1718 002300 000000 L.MR2: .WORD 0 ;MAINT. REG 2
1719 002302 000000 L.MR3: .WORD 0 ;MAINT. REG 3
1720
1721
1722
1723 .SBTTL PROGRAM DEFINED VARIABLES
1724
1725 002304 000210 RKVEC: .WORD 210 ;RK611 VECTOR
1726 002306 000240 RKPRI: .WORD PR5 ;RK611 PRIORITY
1727 002310 000000 TRAPPC: .WORD 0 ;PC FOR MEMORY CHECK ENABLE TRAP
1728 002312 000000 SRTFLG: .WORD 0 ;START FLAG
1729 : 0 = 200
1730 : 1 = 214
1731 : -1 = 204
1732 002314 000000 ERRCNT: .WORD 0 ;ERROR COUNT FOR SWITCH 12 ABORT
1733 002316 000000 P1.BIT: .WORD 0 ;NEXT BIT IN DATA SIMULATION
1734 002320 000000 PR.BIT: .WORD 0 ;PRESENT BIT IN DATA SIMULATION
1735 002322 000000 M1.BIT: .WORD 0 ;PREVIOUS BIT IN DATA SIMULATION
1736 002324 000000 M2.BIT: .WORD 0 ;BIT BEFORE PREVIOUS BIT
1737 002326 000000 BITCNT: .WORD 0 ;BIT POSITION
1738 002330 000000 WRDCNT: .WORD 0 ;WORD COUNT FOR NPR TRANSFER
1739 002332 000000 MEMPAR: .WORD 0 ;MEMORY EMABLE ON FIRST 24K
1740 002334 000000 BADPAR: .WORD 0 ;BAD PARITY FLAG
1741 002336 000000 ECCXOR: .WORD 0 ;ECC XOR SWITCH
1742 002340 000000 ECCHI: .WORD 0 ;ECC HIGH STORAGE
1743 002342 000000 ECCLO: .WORD 0 ;ECC LOW STORAGE
```

CZR6ECO RK611 DSKLS CTRL PRT5
CZR6EC.P11 14-SEP-81 13:57

MACY11 30(1046) 14-SEP-81 15:14
PROGRAM DEFINED VARIABLES

K 3
PAGE 37

SEQ 0036

CZ
CZ

1744	002344	000000	ECCSRC: .WORD	0	:ECC SOURCE FLAG
1745	002346	000000	ECPATX: .WORD	0	:EXPECTED PATTERN ON ERROR
1746	002350	000000	ECPOSX: .WORD	0	:EXPECTED POSITION ON ERROR
1747	002352	000000	BSEDES: .WORD	0	:BAD SECTOR DESIRED
1748	002354	000000	HIBITS: .WORD	0	:HI ORDER BITS FOR 18 BIT WORD XFER
1749	002356	000000	SAVSWR: .WORD	0	:STORAGE FOR SWITCH REGISTER

```

1750      .SBTTL PROGRAM SETUP
1751
1752 002360 012737 000001 002312 PARM:  MOV #1,SRTFLG ;LOAD START FLAG FOR PARMETER START
1753 002366 000406                                BR      START1
1754
1755 002370 012737 177777 002312 RESTRT: MOV #-1,SRTFLG ;LOAD START FLAG FOR RESTART
1756 002376 000402                                BR      START1
1757
1758 002400 005037 002312      START:  CLR  SRTFLG ;CLEAR START FLAG
1759 002404 000005      START1: RESET ;RESET THE WHOLE SYSTEM
1760 002406 105037 001103      CLRB  $ERFLG ;CLEAR ERROR FLAG
1761 002412 012706 001100      MOV   #STACK,SP ;INITIALIZE STACK POINTER
1762 002416 012746 000340      MOV   #PR7,-(SP) ;LOAD STACK TO LOCK OUT ALL INTERRUPTS
1763 002422 012746 002430      MOV   #1$,-(SP) ;LOAD START OF PROGRAM
1764 002426 000002      RTI   ;LOAD PSW
1765
1766 002430 004737 036062 1$:    JSR   PC,@#STKINT ;INIT KEYBOARD
1767      .SBTTL INITIALIZE THE COMMON TAGS
1768      ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
1769 002434 012706 001100      MOV   #$CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
1770 002440 005026      CLR   (R6)+ ;:CLEAR MEMORY LOCATION
1771 002442 022706 001140      CMP   #SWR,R6 ;:DONE?
1772 002446 001374      BNE   -6 ;:LOOP BACK IF NO
1773 002450 012706 001100      MOV   #STACK,SP ;:SETUP THE STACK POINTER
1774      ;;INITIALIZE A FEW VECTORS
1775 002454 012737 033366 000020      MOV   #$$SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
1776 002462 012737 000340 000022      MOV   #340,@#IOTVEC+2 ;:LEVEL 7
1777 002470 012737 034266 000030      MOV   #$error,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
1778 002476 012737 000340 000032      MOV   #340,@#EMTVEC+2 ;:LEVEL 7
1779 002504 012737 037712 000034      MOV   #STRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
1780 002512 012737 090340 000036      MOV   #340,@#TRAPVEC+2 ;:LEVEL 7
1781 002520 012737 037556 000024      MOV   #SPWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
1782 002526 012737 000340 000026      MOV   #340,@#PWRVEC+2 ;:LEVEL 7
1783 002534 013737 023142 023134      MOV   $ENDCT,$EOPCT ;:SETUP END-OF-PROGRAM COUNTER
1784 002542 005037 001234      CLR   $TIMES ;:INITIALIZE NUMBER OF ITERATIONS
1785 002546 005037 001236      CLR   $ESCAPE ;:CLEAR THE ESCAPE ON ERROR ADDRESS
1786 002552 112737 000001 001115      MOV   #1,$ERMAX ;:ALLOW ONE ERROR PER TEST
1787 002560 012737 002560 001106      MOV   #,$LPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
1788 002566 012737 002566 001110      MOV   #,$LPERR ;:SETUP THE ERROR LOOP ADDRESS
1789      ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1790      ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1791 002574 013746 000004      MOV   @#ERRVEC,-(SP) ;:SAVE ERROR VECTOR
1792 002600 012737 002634 000004      MOV   #64$,@#ERRVEC ;:SET UP ERROR VECTOR
1793 002606 012737 177570 001140      MOV   #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
1794 002614 012737 177570 001142      MOV   #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
1795 002622 022777 177777 176310      CMP   #-1,@SWR ;:TRY TO REFERENCE HARDWARE SWR
1796 002630 001012      BNE   66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
1797      ;:AND THE HARDWARE SWR IS NOT = -1
1798 002632 000403      BR    65$ ;:BRANCH I. NO TIMEOUT
1799 002634 012716 002642      64$:  MOV   #65$,(SP) ;:SET UP FOR TRAP RETURN
1800 002640 000002      RTI
1801 002642 012737 000176 001140      65$:  MOV   #SWREG,SWR ;:POINT TO SOFTWARE SWR
1802 002650 012737 000174 001142      MOV   #DISPREG,DISPLAY
1803 002656 012637 000004      66$:  MOV   (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
1804
1805 002662 005037 001256      CLR   $PASS ;:CLEAR PASS COUNT
  
```

```

1806 002666 132737 000200 001271 BITB #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
1807 002674 001403 BEQ 67$ ;;YES,USE NON-APT SWITCH
1808 002676 012737 001272 001140 MOV #$$SWREG,$SWR ;;NO,USE APT SWITCH REGISTER
1809 002704 67$:
1810 002704 005037 002314 CLR ERRCNT ;CLEAR ERROR COUNT FOR SWITCH 1? ABORT
1811 .SBTTL TYPE PROGRAM NAME
1812 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1813 002710 005227 177777 INC #-1 ;;FIRST TIME?
1814 002714 001056 BNE 68$ ;;BRANCH IF NO
1815 002716 022737 023276 000042 CMP #$$ENDAD,@#42 ;;ACT-11?
1816 002724 001452 BEQ 68$ ;;BRANCH IF YES
1817 002726 104401 002774 TYPE ,69$ ;;TYPE ASCIZ STRING
1818 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
1819 002732 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
1820 002736 001012 BNE 70$ ;;BRANCH IF YES
1821 002740 123727 001270 000001 CMPB $ENV,#1 ;;ARE WE RUNNING UNDER APT?
1822 002746 001406 BEQ 70$ ;;BRANCH IF YES
1823 002750 023727 001140 000176 CMP $SWR,#$SWREG ;;SOFTWARE SWITCH REG SFLECTED?
1824 002756 001005 BNE 71$ ;;BRANCH IF NO
1825 002760 104406 GTSWR ;;GET SOFT-SWR SETTINGS
1826 002762 000403 BR 71$
1827 002764 112737 000001 001134 70$: MOVB #1,$AUTOB ;;SET AUTO-MODE INDICATOR
1828 002772 71$:
1829 002772 000427 BR 68$ ;;GET OVER THE ASCIZ
1830 ;;69$: .ASCIZ <CRLF>/RK611 DISKLESS DIAGNOSTIC: PART 5 CZR6ECO/<CRLF>
1831 003052 68$:
1832 003052 005227 177777 INC #-1 ;TEST IF PASS 0
1833 003056 001002 BNE 4$ ;NO - SKIP
1834 003060 104401 041276 TYPE ,OPR006 ;TYPE PASS TIME MESSAGE
1835 003064 022737 000001 002312 4$: CMP #1,$RTFLG ;CHECK IF PARAMETER START
1836 003072 001121 BNE 15$ ;NO, CONTINUE SETUP
1837 003074 104401 041164 5$: TYPE ,OPR001 ;TYPE 'RK611 BUS ADDRESS ( ) -'
1838 003100 013746 001324 MOV $BASE,-(SP) ;;SAVE $BASE FOR TYPEOUT
1839 003104 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1840 003106 104401 041216 TYPE ,OPR002
1841 003112 104412 RDOCT ;GET VALUE
1842 003114 012637 001202 MOV (SP)+,$TMP0
1843 003120 001407 BEQ 7$ ;CHECK IF <CR>
1844 003122 022737 160000 001202 CMP #160000,$TMP0 ;CHECK IF IN I/O PAGE
1845 003130 101361 BHI 5$
1846 003132 013737 001202 001324 7$: MOV $TMP0,$BASE ;LOAD NEW BUS ADDRESS
1847 003140 104401 041224 TYPE ,OPR003 ;TYPE 'RK611 VECTOR ADDRESS ( ) ='
1848 003144 013746 001320 MOV $VECT1,-(SP) ;TYPE OUT VECTOR ADDRESS
1849 003150 042716 160000 BIC #160000,(SP)
1850 003154 104401 041216 TYPE ,OPR002
1851 003160 104412 RDOCT ;GET VALUE
1852 003162 012637 001202 MOV (SP)+,$TMP0
1853 003166 001412 BEQ 10$ ;CHECK IF <CR>
1854 003170 022737 001000 001202 CMP #1000,$TMP0 ;CHECK IF LEGAL
1855 003176 101760 BLOS 7$
1856 003200 042737 017777 001320 BIC #1777,$VECT1 ;LOAD NEW VECTOR ADDRESS
1857 003206 053737 001202 001320 BIS $TMP0,$VECT1
1858 003214 104401 041254 10$: TYPE ,OPR004 ;TYPE 'RK611 PRIORITY ( ) ='
1859 003220 005046 CLR -(SP)
1860 003222 113716 001321 MOVB $VECT1+1,(SP)
1861 003226 006216 ASR (SP) ;SHIFT 5 BITS RIGHT

```

1862	003230	006216			ASR	(SP)	
1863	003232	006216			ASR	(SP)	
1864	003234	006216			ASR	(SP)	
1865	003236	006216			ASR	(SP)	
1866	003240	104402			TYPOC		
1867	003242	104401	041216		TYPE	,OPR002	
1868	003246	104412			RDOCT		;GET VALUE
1869	003250	012637	001202		MOV	(SP)+,\$TMP0	
1870	003254	001430			BEQ	15\$;CHECK FOR DEFAULT
1871	003256	022737	000007	001202	CMP	#7,\$TMP0	;CHECK IF LEGAL
1872	003264	103753			BLO	10\$	
1873	003266	022737	000004	001202	CMP	#4,\$TMP0	
1874	003274	101347			BHI	10\$	
1875	003276	006337	001202		ASL	\$TMP0	;SHIFT 5 BITS LEFT
1876	003302	006337	001202		ASL	\$TMP0	
1877	003306	006337	001202		ASL	\$TMP0	
1878	003312	006337	001202		ASL	\$TMP0	
1879	003316	006337	001202		ASL	\$TMP0	
1880	003322	042737	160000	001320	BIC	#160000,\$VECT1	;STORE NEW PRIORITY
1881	003330	153737	001202	001321	BISB	\$TMP0,\$VECT1+1	
1882	003336	013737	001320	002304	15\$: MOV	\$VECT1,RKVEC	;STORE RK611 VECTOR
1883	003344	042737	160000	002304	BIC	#160000,RKVEC	
1884	003352	113737	001321	002306	MOVB	\$VECT1+1,RKPRI	;STORE PRIORITY
1885	003360	004737	032652		JSR	PC,\$SIZE	;SIZE MEMORY
1886							
1887	003364	013702	001324		NEWPAS: MOV	\$BASE,R2	;SET RK POINTER
1888	003370	005037	001236		CLR	\$ESCAPE	;CLEAR ESCAPE
1889	003374	004737	032516		JSR	PC,PARCHK	;CHECK OF MEMORY CHECK ENABLE
1890	003400	012746	000000		MOV	#PRO,-(SP)	;LOCK OUT INTERRUPTS
1891	003404	012746	003412		MOV	#TST1,-(SP)	
1892	003410	000002			RTI		

.SBTTL **MULTI-SECTOR WRITE OPERATIONS

*TEST 1 MULTI-SECTOR WRITE (PART 1)

CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND THE
TWO ECC WORDS. MAKE SURE THE NEXT HEADER RECOGNITION
CYCLE STARTS AND THE PROPER HEADER IS SEARCHED FOR.
PROVIDE A SECOND GOOD HEADER AND CHECK RECOGNITION
OF HEADER.

TST1: SCOPE
MOV #10.,\$TIMES ;;DO 10. ITERATIONS
MOV #CCLR,RKCS1(R2) ;CLR CONTROLLER
CLR BSEDES ;CLEAR BAD SEC DESIRED
JSR R4,DSECTR ;SIMULATE SECTOR PULSE TO SET
;ECCW IN RKMRI FROM INIT POWER UP
JSR R4,LOADRK ;LOAD 'L' REGISTERS
0 ;CYLINDER 0
.BYTE 0 ;SECTOR 0
.BYTE 0 ;TRACK 0
OBUFF ;BUFFER ADDRESS OBUFF
-401 ;WORD COUNT -401
WRDATA ;COMMAND WRDATA
JSR R4,BLDDAT ;GO BUILD DATA
1 ;PATTERN 1
JSR R4,OPSTR1 ;START THE OPERATION
JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
ERROR 4 ;CS1 MISCOMPARE
ERROR 5 ;MR1 ""
ERROR 6 ;MR2 ""
ERROR 7 ;MR3 ""
JSR R4,DSECTR ;SIMULATE SECTOR PULSE
JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
ERROR 10 ;MR1 CONTENTS IN ERROR
ERROR 11 ;CS1 CONTENTS IN ERROR
JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
ERROR 12 ;MR1 CONTENTS IN ERROR
ERROR 13 ;CS1 IN ERROR AFTER SEARCH
ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH

1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910 003412 000004
1911 003414 012737 000012 001234
1912
1913 003422 012762 100000 000000
1914
1915 003430 005037 002352
1916 003434 004437 023316
1917
1918
1919 003440 004437 032230
1920 003444 000000
1921 003446 000
1922 003447 000
1923 003450 050130
1924 003452 177377
1925 003454 000023
1926
1927 003456 004437 032014
1928 003462 000001
1929
1930 003464 004437 032274
1931
1932 003470 004437 023334
1933 003474 104004
1934 003476 104005
1935 003500 104006
1936 003502 104007
1937
1938 003504 004437 023316
1939
1940 003510 004437 023740
1941 003514 104010
1942 003516 104011
1943
1944 003520 004437 024276
1945 003524 104012
1946 003526 104013
1947 003530 104014
1948

1949	003532	012737	043675	001510		MOV	#EM5020,EMW	;SET MESSAGE FOR WRITE FAILURE
1950	003540	004437	025204			JSR	R4,DWCPSN	;SIMULATE GAP AND SYNC FOR WRITE
1951	003544	104015				ERROR	15	;MR1 IN ERROR IN READ GAP
1952	003546	104016				ERROR	16	;CS1 IN ERROR AFTER READ GAP
1953	003550	104017				ERROR	17	;MR1 IN ERROR IN WRITING SYNC
1954	003552	104020				ERROR	20	;CS1 IN ERROR AFTER WRITING SYNC
1955								
1956	003554	012737	043754	001510		MOV	#EM5021,EMW	;SET MESSAGE FOR WRITE FAILURE
1957	003562	004437	025604			JSR	R4,DWRITE	;SIMULATE WRITE DATA AND ECC
1958	003566	104017				ERROR	17	;MR1 IN ERROR IN WRITING DATA OR ECC
1959	003570	104021				ERROR	21	;ECC IN ERROR WHILE WRITING DATA
1960	003572	104022				ERROR	22	;CS1 IN ERROR AFTER WRITING DATA AND ECC
1961								
1962	003574	012700	000034			MOV	#7*4,R0	;ISSUE CLOCKS TO NEXT HDR COMPARE
1963	003600	012762	000440	000026	1\$:	MOV	#DMD,MCLK,RKMR1(R2)	;CLOCK LOOP
1964	003606	012762	000040	000026		MOV	#DMD,RKMR1(R2)	
1965	003614	005300				DEC	R0	
1966	003616	001370				BNE	1\$	
1967								
1968	003620	013703	002274			MOV	L.DCYL,R3	;GET DESIRED CYLINDER
1969	003624	004437	032444			JSR	R4,FSBLVV	;INVERT WORD
1970	003630	010337	002246			MOV	R3,E.MR2	;STORE EXPECTED MR2
1971	003634	113703	002267			MOVB	L.DT,R3	;GET DESIRED TRACK/SECTOR
1972	003640	042703	177400			BIC	#177400,R3	;CLEAR UNUSED BITS
1973	003644	012700	000005			MOV	#5,R0	;SET COUNT FOR SHIFTING
1974	003650	006303			5\$:	ASL	R3	;SHIFT TRACK FOR HEADER ALIGNMENT
1975	003652	005300				DEC	R0	;DEC COUNT
1976	003654	001375				BNE	5\$;LOOP UNTIL ZERO
1977	003656	153703	002266			BISB	L.DS,R3	;INSERT SECTOR NUMBER
1978								
1979	003662	032737	010000	002260		BIT	#CFMT,L.CS1	;TEST IF 18 BIT MODE
1980	003670	001402				BEQ	6\$;NO - SKIP
1981	003672	052703	001000			BIS	#BIT9,R3	;SET FORMAT BIT FOR WORD TWO
1982	003676				6\$:			
1983	003676	004437	032444			JSR	R4,FSBLVV	;INVERT WORD
1984	003702	010337	002250			MOV	R3,E.MR3	;STORE EXPECTED MR3
1985	003706	016237	000034	002206		MOV	RKMR2(R2),T.MR2	;GET MR2
1986	003714	016237	000036	002210		MOV	RKMR3(R2),T.MR3	;GET MR3
1987	003722	016237	000000	002160		MOV	RKCS1(R2),T.CS1	;GET CS1
1988	003730	013737	002260	002220		MOV	L.CS1,E.CS1	;GET EXPECTED CS1
1989	003736	023737	002160	002220		CMP	T.CS1,E.CS1	;CHECK IF OK
1990	003744	001402				BEQ	3\$;YES - SKIP
1991	003746	104025				ERROR	25	;CS1 IN ERROR AT MULTI-SECTOR TIME
1992	003750	000427				BR	TST2	;GO TO NEXT TEST
1993								
1994	003752	023737	002206	002246	3\$:	CMP	T.MR2,E.MR2	;CHECK IF MR2 OK
1995	003760	001402				BEQ	2\$;YES - SKIP
1996	003762	104023				ERROR	23	;MR2 IN ERROR AT MULTI-SECTOR TIME
1997	003764	000421				BR	TST2	;GO TO NEXT TEST
1998								
1999	003766	023737	002210	002250	2\$:	CMP	T.MR3,E.MR3	;CHECK IF MR3 OK
2000	003774	001402				BEQ	4\$;YES - SKIP
2001	003776	104024				ERROR	24	;MR3 IN ERROR AT MULTI-SECTOR TIME
2002	004000	000413				BR	TST2	;GO TO NEXT TEST
2003								
2004	004002	004437	023316		4\$:	JSR	R4,DSECTR	;SIMULATE 2ND SECTOR PULSE

2005
 2006 004006 004437 023740
 2007 004012 104010
 2008 004014 104011
 2009
 2010 004016 004437 024276
 2011 004022 104012
 2012 004024 104013
 2013 004026 104014

```

JSR R4,DRSYNC ;SIMULATE PREAMBLE
ERROR 10 ;MR1 ERROR
ERROR 11 ;CS1 ERROR

JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
ERROR 12 ;MR1 ERROR
ERROR 13 ;CS1 ERROR
ERROR 14 ;RKDCYL OR RKDA IN ERROR
  
```

```

*****
*TEST 2 MULTI-SECTOR WRITE (PART 2)
*
* CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
* PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
* DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
* CYLINDER 0, HEAD 0, SECTOR 23. CLOCK THROUGH SEEK
* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
* AND A GOOD HEADER, CLOCK THROUGH 400 WORDS AND THE
* TWO ECC WORDS. MAKE SURE THE NEXT HEADER RECOGNITION
* CYCLE STARTS AND THE PROPER HEADER IS SEARCHED FOR.
* PROVIDE A SECOND GOOD HEADER AND CHECK RECOGNITION
* OF HEADER.
  
```

2029
 2030 004030 000004
 2031 004032 012737 000012 001234
 2032
 2033 004040 012762 100000 000000
 2034
 2035 004046 005037 002352
 2036
 2037 004052 004437 032230
 2038 004056 000000
 2039 004060 023
 2040 004061 000
 2041 004062 050130
 2042 004064 177377
 2043 004066 000023
 2044
 2045 004070 004437 032014
 2046 004074 000002
 2047
 2048 004076 004437 032274
 2049
 2050 004102 004437 023334
 2051 004106 104004
 2052 004110 104005
 2053 004112 104006
 2054 004114 104007
 2055
 2056 004116 004437 023316
 2057
 2058 004122 004437 023740
 2059 004126 104010
 2060 004130 104011

```

*****
TEST2: SCOPE
MOV #10, $TIMES ;DO 10. ITERATIONS
MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
CLR BSEDES ;CLEAR BAD SEC DESIRED
JSR R4,LCADRK ;LOAD 'L' REGISTERS
0 ;CYLINDER 0
.BYTE 23 ;SECTOR 23
.BYTE 0 ;TRACK 0
OBUFF ;BUFFER ADDRESS OBUFF
-401 ;WORD COUNT -401
WRDATA ;COMMAND WRDATA

JSR R4,BLDDAT ;GO BUILD DATA
2 ;PATTERN 2

JSR R4,OPSTRT ;START THE OPERATION

JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
ERROR 4 ;CS1 MISCOMPARE
ERROR 5 ;MR1
ERROR 6 ;MR2
ERROR 7 ;MR3

JSR R4,DSECTR ;SIMULATE SECTOR PULSE

JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
ERROR 10 ;MR1 CONTENTS IN ERROR
ERROR 11 ;CS1 CONTENTS IN ERROR
  
```

2061							
2062	004132	004437	024276			JSR	R4,DHDCMP ;SIMULATE HEADER SEARCH
2063	004136	104012				ERROR	12 ;MR1 CONTENTS IN ERROR
2064	004140	104013				ERROR	13 ;CS1 IN ERROR AFTER SEARCH
2065	004142	104014				ERROR	14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2066							
2067	004144	012737	043675	001510		MOV	#EM5020,EMW ;SET MESSAGE FOR WRITE FAILURE
2068	004152	004437	025204			JSR	R4,DWGPSN ;SIMULATE GAP AND SYNC FOR WRITE
2069	004156	104015				ERROR	15 ;MR1 IN ERROR IN READ GAP
2070	004160	104016				ERROR	16 ;CS1 IN ERROR AFTER READ GAP
2071	004162	104017				ERROR	17 ;MR1 IN ERROR IN WRITING SYNC
2072	004164	104020				ERROR	20 ;CS1 IN ERROR AFTER WRITING SYNC
2073							
2074	004166	012737	043754	001510		MOV	#EM5021,EMW ;SET MESSAGE FOR WRITE FAILURE
2075	004174	004437	025604			JSR	R4,DWRITE ;SIMULATE WRITE DATA AND ECC
2076	004200	104017				ERROR	17 ;MR1 IN ERROR IN WRITING DATA OR ECC
2077	004202	104021				ERROR	21 ;ECC IN ERROR WHILE WRITING DATA
2078	004204	104022				ERROR	22 ;CS1 IN ERROR AFTER WRITING DATA AND ECC
2079							
2080	004206	012700	000034			MOV	#7*4,R0 ;ISSUE CLOCKS TO NEXT HDR COMPARE
2081	004212	012762	000440	000026	1\$:	MOV	#DMD,MCLK,RKMR1(R2) ;CLOCK LOOP
2082	004220	012762	000040	000026		MOV	#DMD,RKMR1(R2)
2083	004226	005300				DEC	R0
2084	004230	001370				BNF	1\$
2085							
2086	004232	013703	002274			MOV	L.DCYL,R3 ;GET DESIRED CYLINDER
2087	004236	004437	032444			JSR	R4,FSBLVV ;INVERT WORD
2088	004242	010337	002246			MOV	R3,E.MR2 ;STORE EXPECTED MR2
2089	004246	113703	002267			MOVB	L.DT,R3 ;GET DESIRED TRACK/SECTOR
2090	004252	042703	177400			BIC	#177400,R3 ;CLEAR UNUSED BITS
2091	004256	012700	000005			MOV	#5,R0 ;SET COUNT FOR SHIFTING
2092	004262	006303			5\$:	ASL	R3 ;SHIFT TRACK FOR HEADER ALIGNMENT
2093	004264	005300				DEC	R0 ;DEC COUNT
2094	004266	001375				BNE	5\$;LOOP UNTIL ZERO
2095	004270	153703	002266			BISB	L.DS,R3 ;INSERT SECTOR NUMBER
2096							
2097	004274	032737	010000	002260		BIT	#CFMT,L.CS1 ;TEST IF 18 BIT MODE
2098	004302	001402				BEQ	6\$;NO - SKIP
2099	004304	052703	001000			BIS	#BIT9,R3 ;SET FORMAT BIT FOR WORD TWO
2100	004310				6\$:		
2101	004310	004437	032444			JSR	R4,FSBLVV ;INVERT WORD
2102	004314	010337	002250			MOV	R3,E.MR3 ;STORE EXPECTED MR3
2103	004320	016237	000034	002206		MOV	RKMR2(R2),T.MR2 ;GET MR2
2104	004326	016237	000036	002210		MOV	RKMR3(R2),T.MR3 ;GET MR3
2105	004334	016237	000000	002160		MOV	RKCS1(R2),T.CS1 ;GET CS1
2106	004342	013737	002260	002220		MOV	L.CS1,E.CS1 ;GET EXPECTED CS1
2107	004350	023737	002160	002220		CMP	T.CS1,E.CS1 ;CHECK IF OK
2108	004356	001402				BEQ	3\$;YES - SKIP
2109	004360	104025				ERROR	25 ;CS1 IN ERROR AT MULTI-SECTOR TIME
2110	004362	000427				BR	TST3 ;GO TO NEXT TEST
2111							
2112	004364	023737	002206	002246	3\$:	CMP	T.MR2,E.MR2 ;CHECK IF MR2 OK
2113	004372	001402				BEQ	2\$;YES - SKIP
2114	004374	104023				ERROR	23 ;MR2 IN ERROR AT MULTI-SECTOR TIME
2115	004376	000421				BR	TST5 ;GO TO NEXT TEST
2116							

```

2117 004400 023737 002210 002250 28:  CMP      T.MR3,E.MR3      :CHECK IF MR3 OK
2118 004406 001402                BEQ      4$              :YES - SKIP
2119 004410 104024                ERROR   24              :MR3 IN ERROR AT MULTI-SECTOR TIME
2120 004412 000413                BR      TST3            :;GO TO NEXT TEST
2121
2122 004414 004437 023316          48:  JSR      R4,DSECTR      :SIMULATE 2ND SECTOR PULSE
2123
2124 004420 004437 023740          JSR      R4,DRSYNC      :SIMULATE PREAMBLE
2125 004424 104010                ERROR   10              :MR1 ERROR
2126 004426 104011                ERROR   11              :CS1 ERROR
2127
2128 004430 004437 024276          JSR      R4,DHDCMP      :SIMULATE HEADER SEARCH
2129 004434 104012                ERROR   12              :MR1 ERROR
2130 004436 104013                ERROR   13              :CS1 ERROR
2131 004440 104014                ERROR   14              :RKDCYL OR RKDA IN ERROR
2132
2133
2134 :*****
2135 :TEST 3          MID-TRANSFER SEEK (PART 1)
2136 :
2137 :CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
2138 :PUT THE CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A WRITE
2139 :DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
2140 :CYLINDER 0, HEAD 0, SECTOR 25.  CLOCK THROUGH SEEK
2141 :AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE
2142 :AND A GOOD HEADER.  CLOCK THROUGH 400 WORDS AND THE
2143 :TWO WORD ECC.  MAKE SURE THE CONTROLLER ISSUES
2144 :AN IMPLIED SEEK TO THE RIGHT HEAD AND CYLINDER.
2145 :MAKE SURE THE NEXT HEADER RECOGNITION PROCEEDS PROPERLY.
2146 :*****
2147 004442 000004                TST3:  SCOPE
2148 004444 012737 000012 001234    MOV      #10.,$TIMES    ;;DO 10. ITERATIONS
2149
2150 004452 012762 100000 000000    MOV      #CCLR,RK(CS1(R2)) :CLEAR CONTROLLER
2151
2152 004460 005037 002352          CLR      BSEDES        :CLEAR BAD SEC DESIRED
2153
2154 004464 004437 032230          JSR      R4,LOADRK      :LOAD 'L' REGISTERS
2155 004470 000000                0                      :CYLINDER 0
2156 004472 025                    .BYTE   25              :SECTOR 25
2157 004473 000                    .BYTE   0               :TRACK 0
2158 004474 050130                OBUF   -401            :BUFFER ADDRESS OBUF
2159 004476 177377                -401                    :WORD COUNT -401
2160 004500 000023                WRDATA                   :COMMAND WRDATA
2161
2162 004502 004437 032014          JSR      R4,BLDDAT      :GO BUILD DATA
2163 004506 000003                3                      :PATTERN 3
2164
2165 004510 004437 032274          JSR      R4,OPSTRT      :START THE OPERATION
2166
2167 004514 004437 023334          JSR      R4,DISEEK      :SIMULATE IMPLIED SEEK
2168 004520 104004                ERROR   4               :CS1 MISCOMPARE
2169 004522 104005                ERROR   5               :MR1 ..
2170 004524 104006                ERROR   6               :MR2 ..
2171 004526 104007                ERROR   7               :MR3 ..
2172

```

```

2173 004530 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
2174
2175 004534 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
2176 004540 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
2177 004542 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
2178
2179 004544 004437 024276 JSR R4,DHDCMF ;SIMULATE HEADER SEARCH
2180 004550 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
2181 004552 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
2182 004554 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2183
2184 004556 012737 043675 001510 MOV #EM5020,EMW ;SET MESSAGE FOR WRITE FAILURE
2185 004564 004437 025204 JSR R4,DWGPSN ;SIMULATE GAP AND SYNC FOR WRITE
2186 004570 104015 ERROR 15 ;MR1 IN ERROR IN READ GAP
2187 004572 104016 ERROR 16 ;CS1 IN ERROR AFTER READ GAP
2188 004574 104017 ERROR 17 ;MR1 IN ERROR IN WRITING SYNC
2189 004576 104020 ERROR 20 ;CS1 IN ERROR AFTER WRITING SYNC
2190
2191 004600 012737 043754 001510 MOV #EM5021,EMW ;SET MESSAGE FOR WRITE FAILURE
2192 004606 004437 025604 JSR R4,DWRITE ;SIMULATE WRITE DATA AND ECC
2193 004612 104017 ERROR 17 ;MR1 IN ERROR IN WRITING DATA OR ECC
2194 004614 104021 ERROR 21 ;ECC IN ERROR WHILE WRITING DATA
2195 004616 104022 ERROR 22 ;CS1 IN ERROR AFTER WRITING DATA AND ECC
2196
2197 004620 012700 000324 MOV #53.*4,R0 ;ISSUE CLOCKS TO NEXT HDR COMPARE
2198 004624 012762 000440 000026 1$: MOV #DMD!MCLK,RKMR1(R2) ;CLOCK LCOP
2199 004632 012762 000040 000026 MOV #DMD,RKMR1(R2)
2200 004640 005300 DEC R0
2201 004642 001370 BNE 1$
2202
2203 004644 013703 002274 MOV L.DCYL,R3 ;GET DESIRED CYLINDER
2204 004650 004437 032444 JSR R4,FSBLVV ;INVERT WORD
2205 004654 010337 002246 MOV R3,E.MR2 ;STORE EXPECTED MR2
2206 004660 113703 002267 MOVVB L.DT,R3 ;GET DESIRED TRACK/SECTOR
2207 004664 042703 177400 BIC #177400,R3 ;CLEAR UNUSED BITS
2208 004670 012700 000005 MOV #5,R0 ;SET COUNT FOR SHIFTING
2209 004674 006303 5$: ASL R3 ;SHIFT TRACK FOR HEADER ALIGNMENT
2210 004676 005300 DEC R0 ;DEC COUNT
2211 004700 001375 BNE 5$ ;LOOP UNTIL ZERO
2212 004702 153703 002266 BISB L.DS,R3 ;INSERT SECTOR NUMBER
2213
2214 004706 032737 010000 002260 BIT #CFMT,L.CS1 ;TEST IF 18 BIT MODE
2215 004714 001402 BEQ 6$ ;NO - SKIP
2216 004716 052703 001000 BIS #BIT9,R3 ;SET FORMAT BIT FOR WORD TWO
2217 004722 6$:
2218 004722 004437 032444 JSR R4,FSBLVV ;INVERT WORD
2219 004726 010337 002250 MOV R3,E.MR3 ;STORE EXPECTED MR3
2220 004732 016237 000034 002206 MOV RKMR2(R2),T.MR2 ;GET MR2
2221 004740 016237 000036 002210 MOV RKMR3(R2),T.MR3 ;GET MR3
2222 004746 016237 000000 002160 MOV RKCS1(R2),T.CS1 ;GET CS1
2223 004754 013737 002260 002220 MOV L.CS1,E.CS1 ;GET EXPECTED CS1
2224 004762 023737 002160 002220 CMP T.CS1,E.CS1 ;CHECK IF OK
2225 004770 001402 BEQ 3$ ;YES - SKIP
2226 004772 104025 ERROR 25 ;CS1 IN ERROR AT MULTI-SECTOR TIME
2227 004774 000427 BR TST4 ;GO TO NEXT TEST
2228

```

```

2229 004776 023737 002206 002246 3$: CMP T,MR2,E.MR2 ;CHECK IF MR2 OK
2230 005004 001402 BEQ 2$ ;YES - SKIP
2231 005006 104023 ERROR 23 ;MR2 IN ERROR AT MULTI-SECTOR TIME
2232 005010 000421 BR TST4 ;GO TO NEXT TEST
2233
2234 005012 023737 002210 002250 2$: CMP T,MR3,E.MR3 ;CHECK IF MR3 OK
2235 005020 001402 BEQ 4$ ;YES - SKIP
2236 005022 104024 ERROR 24 ;MR3 IN ERROR AT MULT.-SECTOR TIME
2237 005024 000413 BR TST4 ;GO TO NEXT TEST
2238
2239 005026 004437 023316 4$: JSR R4,DSECTR ;SIMULATE 2ND SECTOR PULSE
2240
2241 005032 004437 023740 JSR R4,DRSYNC ;SIMULATE PREAMBLE
2242 005036 104010 ERROR 10 ;MR1 ERROR
2243 005040 104011 ERROR 11 ;CS1 ERROR
2244
2245 005042 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
2246 005046 104012 ERROR 12 ;MR1 ERROR
2247 005050 104013 ERROR 13 ;CS1 ERROR
2248 005052 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR
2249

```

 *TEST 4 MID-TRANSFER SEEK (PART 2)

*
 * CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
 * PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
 * DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
 * CYLINDER 0, HEAD 2, SECTOR 25. CLOCK THROUGH SEEK
 * AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
 * AND A GOOD HEADER, CLOCK THROUGH 400 WORDS AND THE
 * TWO WORD ECC. MAKE SURE THE CONTROLLER ISSUES
 * AN IMPLIED SEEK TO THE RIGHT HEAD AND CYLINDER.
 * MAKE SURE THE NEXT HEADER RECOGNITION PROCEEDS PROPERLY.
 *

```

2264 005054 000004 TST4: SCOPE
2265 005056 012737 000012 001234 MOV #10, $TIMES ;DO 10. ITERATIONS
2266
2267 005064 012762 100000 000000 MOV #CLR,RKCS1(R2) ;CLEAR CONTROLLER
2268
2269 005072 005037 002352 CLR BSEDES ;CLEAR BAD SEC DESIRED
2270
2271 005076 004437 032230 JSR R4,LOADRK ;LOAD 'L' REGISTERS
2272 005102 000000 0 ;CYLINDER 0
2273 005104 025 ;SECTOR 25
2274 005105 002 ;TRACK 2
2275 005106 050130 OBUFF ;BUFFER ADDRESS OBUFF
2276 005110 177377 -401 ;WORD COUNT -401
2277 005112 000023 WRDATA ;COMMAND WRDATA
2278
2279 005114 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA
2280 005120 000004 4 ;PATTERN 4
2281
2282 005122 004437 032274 JSR R4,OPSTRT ;START THE OPERATION
2283
2284 005126 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK

```

2285	005132	104004						ERROR	4	:CS1 MISCOMPARE
2286	005134	104005						ERROR	5	:MR1
2287	005136	104006						ERROR	6	:MR2
2288	005140	104007						ERROR	7	:MR3
2289										
2290	005142	004437	023316					JSR	R4,DSECTR	:SIMULATE SECTOR PULSE
2291										
2292	005146	004437	023740					JSR	R4,DRSYNC	:SIMULATE HEADER PREAMBLE
2293	005152	104010						ERROR	10	:MR1 CONTENTS IN ERROR
2294	005154	104011						ERROR	11	:CS1 CONTENTS IN ERROR
2295										
2296	005156	004437	024276					JSR	R4,DHDCMP	:SIMULATE HEADER SEARCH
2297	005162	104012						ERROR	12	:MR1 CONTENTS IN ERROR
2298	005164	104013						ERROR	13	:CS1 IN ERROR AFTER SEARCH
2299	005166	104014						ERROR	14	:RKDCYL OR RKDA IN ERROR AFTER SEARCH
2300										
2301	005170	012737	043675	001510				MOV	#EM5020,EMW	:SET MESSAGE FOR WRITE FAILURE
2302	005176	004437	025204					JSR	R4,DWGPSN	:SIMULATE GAP AND SYNC FOR WRITE
2303	005202	104015						ERROR	15	:MR1 IN ERROR IN READ GAP
2304	005204	104016						ERROR	16	:CS1 IN ERROR AFTER READ GAP
2305	005206	104017						ERROR	17	:MR1 IN ERROR IN WRITING SYNC
2306	005210	104020						ERROR	20	:CS1 IN ERROR AFTER WRITING SYNC
2307										
2308	005212	012737	043754	001510				MOV	#EM5021,EMW	:SET MESSAGE FOR WRITE FAILURE
2309	005220	004437	025604					JSR	R4,DWRITE	:SIMULATE WRITE DATA AND ECC
2310	005224	104017						ERROR	17	:MR1 IN ERROR IN WRITING DATA OR ECC
2311	005226	104021						ERROR	21	:ECC IN ERROR WHILE WRITING DATA
2312	005230	104022						ERROR	22	:CS1 IN ERROR AFTER WRITING DATA AND ECC
2313										
2314	005232	012700	000324					MOV	#53.*4,R0	:ISSUE CLOCKS TO NFXST HDR COMPARE
2315	005236	012762	000440	000026	1\$:			MOV	#DMD!MCLK,RKMR1(R2)	:CLOCK LOOP
2316	005244	012762	000040	000026				MOV	#DMD,RKMR1(R2)	
2317	005252	005300						DEC	R0	
2318	005254	001370						BNE	1\$	
2319										
2320	005256	013703	002274					MOV	L.DCYL,R3	:GET DESIRED CYLINDER
2321	005262	004437	032444					JSR	R4,FSBLVV	:INVERT WORD
2322	005266	010337	002246					MOV	R3,E.MR2	:STORE EXPECTED MR2
2323	005272	113703	002267					MOV	L.DT,R3	:GET DESIRED TRACK/SECTOR
2324	005276	042703	177400					BIC	#177400,R3	:CLEAR UNUSED BITS
2325	005302	012700	000005					MOV	#5,R0	:SET COUNT FOR SHIFTING
2326	005306	006303			5\$:			ASL	R3	:SHIFT TRACK FOR HEADER ALIGNMENT
2327	005310	005300						DEC	R0	:DEC COUNT
2328	005312	001375						BNE	5\$:LOOP UNTIL _ERO
2329	005314	153703	002266					BISB	L.DS,R3	:INSERT SECTOR NUMBER
2330										
2331	005320	032737	010000	002260				BIT	#CFMT,L.CS1	:TEST IF 18 BIT MODE
2332	005326	001402						BEQ	6\$:NO - SKIP
2333	005330	052703	001000					BIS	#BIT9,R3	:SET FORMAT BIT FOR WORD TWO
2334	005334				6\$:					
2335	005334	004437	032444					JSR	R4,FSBLVV	:INVERT WORD
2336	005340	010337	002250					MOV	R3,E.MR3	:STORE EXPECTED MR3
2337	005344	016237	000034	002206				MOV	RKMR2(R2),T.MR2	:GET MR2
2338	005352	016237	000036	002210				MOV	RKMR3(R2),T.MR3	:GET MR3
2339	005360	016237	000000	002160				MOV	RKCS1(R2),T.CS1	:GET CS1
2340	005366	013737	002260	002220				MOV	L.CS1,E.CS1	:GET EXPECTED CS1


```

2341 005374 023737 002160 002220      CMP      T,CS1,E.CS1      ;CHECK IF OK
2342 005402 001402                      BEQ      3$              ;YES - SKIP
2343 005404 104025                      ERROR   25              ;CS1 IN ERROR AT MULTI-SECTOR TIME
2344 005406 000427                      BR      TST5            ;GO TO NEXT TEST
2345
2346 005410 023737 002206 002246 3$:    CMP      T,MR2,E.MR2      ;CHECK IF MR2 OK
2347 005416 001402                      BEQ      2$              ;YES - SKIP
2348 005420 104023                      ERROR   23              ;MR2 IN ERROR AT MULT.-SECTOR TIME
2349 005422 000421                      BR      TST5            ;GO TO NEXT TEST
2350
2351 005424 023737 002210 002250 2$:    CMP      T,MR3,E.MR3      ;CHECK IF MR3 OK
2352 005432 001402                      BEQ      4$              ;YES - SKIP
2353 005434 104024                      ERROR   24              ;MR3 IN ERROR AT MULTI-SECTOR TIME
2354 005436 000423                      BR      TST5            ;GO TO NEXT TEST
2355
2356 005440 004437 023316          4$:    JSR      R4,DSECTR        ;SIMULATE 2ND SECTOR PULSE
2357
2358 005444 004437 023740          JSR      R4,DRSYNC        ;SIMULATE PREAMBLE
2359 005450 104010                      ERROR   10              ;MR1 ERROR
2360 005452 104011                      ERROR   11              ;CS1 ERROR
2361
2362 005454 004437 024276          JSR      R4,DHDCMP        ;SIMULATE HEADER SEARCH
2363 005460 104012                      ERROR   12              ;MR1 ERROR
2364 005462 104013                      ERROR   13              ;CS1 ERROR
2365 005464 104014                      ERROR   14              ;RKDCYL OR RKDA IN ERROR
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382 005466 000004          TST5:  SCOPE
2383 005470 012737 000012 001234      MOV      #10.,$TIMES      ;;DO 10. ITERATIONS
2384
2385 005476 012762 100000 000000      MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
2386
2387 005504 005037 002352          CLR      BSEDES          ;CLEAR BAD SEC DESIRED
2388 005510 012700 002000          MOV      #2000,R0        ;SET A COUNT TO DELAY FOR CLEAR
2389 005514 005300          20$:    DEC      R0
2390 005516 001376          BNE     20$              ;LOOP UNTIL ZERO
2391
2392 005520 004437 032230          JSR      R4,LOADRK        ;LOAD 'L' REGISTERS
2393 005524 000000          0              ;CYLINDER 0
2394 005526 023              .BYTE   23              ;SECTOR 23
2395 005527 000              .BYTE   0              ;TRACK 0
2396 005530 050130          OBUFF          ;BUFFER ADDRESS OBUFF
  
```

```

*****
*TEST 5          MID-TRANSFER SEEK AND 24 SECTOR FORMAT
*
* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
* PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A WRITE DATA
* DATA OF 401 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
* CYLINDER 0, HEAD 0, SECTOR 23.  CLOCK THROUGH SEEK
* AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE
* AND A GOOD HEADER.  CLOCK THROUGH 400 18 BIT WORDS
* AND THE 32 BIT ECC.  CHECK THAT IMPLIED SEEK IS
* ISSUE TO NEXT TRACK.  CHECK FOR PROPER HEADER RECOGNITION
* OCCURS.
*****
  
```

```
2397 005532 177377 -401 ;WORD COUNT -401
2398 005534 010023 WRDATA!CFMT ;COMMAND WRDATA.CFMT
2399
2400 005536 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA
2401 005542 000007 7 ;PATTERN 7
2402
2403 005544 004437 032274 JSR R4,OPSTR ;START THE OPERATION
2404
2405 005550 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
2406 005554 104004 ERROR 4 ;CS1 MISCOMPARE
2407 005556 104005 ERROR 5 ;MR1
2408 005560 104006 ERROR 6 ;MR2
2409 005562 104007 ERROR 7 ;MR3
2410
2411 005564 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
2412
2413 005570 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
2414 005574 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
2415 005576 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
2416
2417 005600 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
2418 005604 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
2419 005606 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
2420 005610 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2421
2422 005612 012737 043675 001510 MOV #EM5020,EMW ;SET MESSAGE FOR WRITE FAILURE
2423 005620 004437 025204 JSR R4,DWGPSN ;SIMULATE GAP AND SYNC FOR WRITE
2424 005624 104015 ERROR 15 ;MR1 IN ERROR IN READ GAP
2425 005626 104016 ERROR 16 ;CS1 IN ERROR AFTER READ GAP
2426 005630 104017 ERROR 17 ;MR1 IN ERROR IN WRITING SYNC
2427 005632 104020 ERROR 20 ;CS1 IN ERROR AFTER WRITING SYNC
2428
2429 005634 012737 043754 001510 MOV #EM5021,EMW ;SET MESSAGE FOR WRITE FAILURE
2430 005642 004437 025604 JSR R4,DWRITE ;SIMULATE WRITE DATA AND ECC
2431 005646 104017 ERROR 17 ;MR1 IN ERROR IN WRITING DATA OR ECC
2432 005650 104021 ERROR 21 ;ECC IN ERROR WHILE WRITING DATA
2433 005652 104022 ERROR 22 ;CS1 IN ERROR AFTER WRITING DATA AND ECC
2434
2435 005654 012700 000324 MOV #53.*4,R0 ;ISSUE CLOCKS TO NEXT HDR COMPARE
2436 005660 012762 000440 000026 1$: MOV #DMD!MCLK,RKMR1(R2) ;CLOCK LOOP
2437 005666 012762 000040 000026 MOV #DMD,RKMR1(R2)
2438 005674 005300 DEC R0
2439 005676 001370 BNE 1$
2440
2441 005700 013703 002274 MOV L.DCYL,R3 ;GET DESIRED CYLINDER
2442 005704 004437 032444 JSR R4,FSBLVV ;INVERT WORD
2443 005710 010337 002246 MOV R3,E.MR2 ;STORE EXPECTED MR2
2444 005714 113703 002267 MOV L.DT,R3 ;GET DESIRED TRACK/SECTOR
2445 005720 042703 177400 BIC #177400,R3 ;CLEAR UNUSED BITS
2446 005724 012700 000005 MOV #5,R0 ;SET COUNT FOR SHIFTING
2447 005730 006303 5$: ASL R3 ;SHIFT TRACK FOR HEADER ALIGNMENT
2448 005732 005300 DEC R0 ;DEC COUNT
2449 005734 001375 BNE 5$ ;LOOP UNTIL ZERO
2450 005736 153703 002266 BISB L.DS,R3 ;INSERT SECTOR NUMBER
2451
2452 005742 032737 010000 002260 BIT #CFMT,L.CS1 ;TEST IF 18 BIT MODE
```

```

2453 005750 001402          BEQ      6$          ;NO - SKIP
2454 005752 052703 001000    BIS      #BIT9,R3    ;SET FORMAT BIT FOR WORD TWO
2455 005756          6$:
2456 005756 004437 032444    JSR      R4,FSBLVV   ;INVERT WORD
2457 005762 010337 002250    MOV      R3,E.MR3    ;STORE EXPECTED MR3
2458 005766 016237 000C34 002206    MOV      RKMR2(R2),T.MR2 ;GET MR2
2459 005774 016237 000036 002210    MOV      RKMR3(R2),T.MR3 ;GET MR3
2460 006002 016237 000000 002160    MOV      RKCS1(R2),T.CS1 ;GET CS1
2461 006010 013737 002260 002220    MOV      L.CS1,E.CS1   ;GET EXPECTED CS1
2462 006016 023737 002160 002220    CMP      T.CS1,E.CS1   ;CHECK IF OK
2463 006024 001402          BEQ      3$          ;YES - SKIP
2464 006026 104025          ERROR    25          ;CS1 IN ERROR AT MULTI-SECTOR TIME
2465 006030 000427          BR       TST6        ;GO TO NEXT TEST
2466
2467 006032 023737 002206 002246 3$:    CMP      T.MR2,E.MR2   ;CHECK IF MR2 OK
2468 006040 001402          BEQ      2$          ;YES - SKIP
2469 006042 104023          ERROR    23          ;MR2 IN ERROR AT MULTI-SECTOR TIME
2470 006044 000421          BR       TST6        ;GO TO NEXT TEST
2471
2472 006046 023737 002210 002250 2$:    CMP      T.MR3,E.MR3   ;CHECK IF MR3 OK
2473 006054 001402          BFQ      4$          ;YES - SKIP
2474 006056 104024          ERROR    24          ;MR3 IN ERROR AT MULTI-SECTOR TIME
2475 006060 000413          BR       TST6        ;GO TO NEXT TEST
2476
2477 006062 004437 023316          4$:    JSR      R4,DSECTR    ;SIMULATE 2ND SECTOR PULSE
2478
2479 006066 004437 023740          JSR      R4,DRSYNC    ;SIMULATE PREAMBLE
2480 006072 104010          ERROR    10          ;MR1 ERROR
2481 006074 104011          ERROR    11          ;CS1 ERROR
2482
2483 006076 004437 024276          JSR      R4,DHDCMP    ;SIMULATE HEADER SEARCH
2484 006102 104012          ERROR    12          ;MR1 ERROR
2485 006104 104013          ERROR    13          ;CS1 ERROR
2486 006106 104014          ERROR    14          ;RKDCYL OR RKDA IN ERROR
2487

```

```

*****
*TEST 6          CYLINDER OVERFLOW (PART 1)
*
*          CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
*          PUT THE CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A WRITE
*          DATA OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
*          CYLINDER 632, HEAD 2, SECTOR 25.  CLOCK THROUGH SEEK
*          AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE
*          AND A GOOD HEADER.  CLOCK THROUGH 400 WORDS AND THE
*          TWO WORD ECC.  MAKE SURE CYLINDER OVERFLOW ERROR DOES
*          NOT SET.
*****

```

```

2500
2501 006110 000004          TST6:  SCOPE
2502 006112 012737 000012 001234    MOV      #10.,$TIMES  ;DO 10. ITERATIONS
2503
2504 006120 005037 002352          CLR      BSEDES      ;CLEAR BAD SEC DESIRED
2505 006124 012762 100000 000000    MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
2506
2507 006132 004437 032230          JSR      R4,LOADRK    ;LOAD 'L' REGISTERS
2508 006136 000632          632          ;CYLINDER 632

```

```

2509 006140 025 .BYTE 25 ;SECTOR 25
2510 006141 002 .BYTE 2 ;TRACK 2
2511 006142 050130 OBUFF ;BUFFER ADDRESS OBUFF
2512 006144 177400 -400 ;WORD COUNT -400
2513 006146 000023 WRDATA ;COMMAND WRDATA
2514
2515 006150 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA
2516 006154 000001 1 ;PATTERN 1
2517
2518 006156 004437 032274 JSR R4,OPSTRT ;START THE OPERATION
2519
2520 006162 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
2521 006166 104004 ERROR 4 ;CS1 MISCOMPARE
2522 006170 104005 ERROR 5 ;MR1 ""
2523 006172 104006 ERROR 6 ;MR2 ""
2524 006174 104007 ERROR 7 ;MR3 ""
2525
2526 006176 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
2527
2528 006202 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
2529 006206 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
2530 006210 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
2531
2532 006212 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
2533 006216 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
2534 006220 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
2535 006222 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2536
2537 006224 012737 043675 001510 MOV #EM5020,EMW ;SET MESSAGE FOR WRITE FAILURE
2538 006232 004437 025204 JSR R4,DWGPSN ;SIMULATE GAP AND SYNC FOR WRITE
2539 006236 104015 ERROR 15 ;MR1 IN ERROR IN READ GAP
2540 006240 104016 ERROR 16 ;CS1 IN ERROR AFTER READ GAP
2541 006242 104017 ERROR 17 ;MR1 IN ERROR IN WRITING SYNC
2542 006244 104020 ERROR 20 ;CS1 IN ERROR AFTER WRITING SYNC
2543
2544 006246 012737 043754 001510 MOV #EM5021,EMW ;SET MESSAGE FOR WRITE FAILURE
2545 006254 004437 025604 JSR R4,DWRITE ;SIMULATE WRITE DATA AND ECC
2546 006260 104017 ERROR 17 ;MR1 IN ERROR IN WRITING DATA OR ECC
2547 006262 104021 ERROR 21 ;ECC IN ERROR WHILE WRITING DATA
2548 006264 104022 ERROR 22 ;CS1 IN ERROR AFTER WRITING DATA AND ECC
2549
2550
2551 006266 012700 000016 MOV #3*4+2,R0 ;SET COUNT TO END OPERATION
2552 006272 012762 000440 000026 1$: MOV #DMD!MCLK,RKMR1(R2) ;SET CLOCK
2553 006300 012762 000040 000026 MOV #DMD,RKMR1(R2) ;CLEAR CLOCK
2554 006306 005300 DEC R0 ;DEC COUNT
2555 006310 001370 BNE 1$ ;LOOP UNTIL 0
2556
2557 006312 004437 032356 JSR R4,GETREG ;GET RK611 REGISTERS
2558 006316 032737 100000 002160 BIT #CERR,T.CS1 ;TEST IF ANY ERROR SET
2559 006324 001415 BEQ 3$ ;NO - SKIP TO EXIT
2560
2561 006326 032737 001000 002174 BIT #COE,T.ER ;TEST IF IT IS COE
2562 006334 001405 BEQ 2$ ;NO - SKIP
2563 006336 012737 044545 001622 MOV #E5026A,E4030 ;SET MESSAGE HEADER
2564 006344 104030 ERROR 30 ;'UNEXPECTED COE ERROR'

```

CZR6ECO RK611 DSKLS CTRL PRT5 MACY11 30(1046) 14-SEP-81 15:14 N 4 PAGE 53
CZR6EC.P11 14-SEP-81 13:57 T6 CYLINDER OVERFLOW (PART 1)

SEQ 0052

CZR
CZR

2565 006346 000404 BR 3\$;EXIT
2566
2567 006350 012737 044545 001632 2\$: MOV #E5026A,EH031 ;SET HEADER

البيانات الواردة في هذا الملف هي ملكية خاصة لشركة جنرال إلكتريك ولا ينبغي استخدامها أو توزيعها خارج نطاق العمل المصرح به.

2568 006356 104031 ERROR 31 ;'UNEXPECTED ERROR TESTING COE''

2569
 2570 006360 38:

2571
 2572 :*****
 2573 :TEST 7 CYLINDER OVERFLOW (PART 2)
 2574 :*

2575 : CLEAR THE RK611 CONTROLLER WITH A CONTROLLER CLEAR.
 2576 : PUT THE CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
 2577 : DATA OF 401 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
 2578 : CYLINDER 632, HEAD 2, SECTOR 25. CLOCK THROUGH SEEK
 2579 : AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE
 2580 : AND A GOOD HEADER. CLOCK THROUGH 400 WORDS AND
 2581 : TWO ECC WORDS. MAKE SURE CYLINDER OVERFLOW ERROR SETS.
 2582 :*

2583 :*****

2584 006360 000004 TST7: SCOPE
 2585 006362 012737 000012 001234 MOV #10.,\$TIMES ;;DO 10. ITERATIONS
 2586
 2587 006370 005037 002352 CLR BSEDES ;CLEAR BAD SEC DESIRED
 2588 006374 012762 100000 000000 MOV #CLR,RKCS1(R2) ;CLEAR CONTROLLER
 2589
 2590 006402 004437 032230 JSR R4,LOADRK ;LOAD 'L' REGISTERS
 2591 006406 000632 532 ;CYLINDER 632
 2592 006410 025 .BYTE 25 ;SECTOR 25
 2593 006411 002 .BYTE 2 ;TRACK 2
 2594 006412 050130 OBUFF ;BUFFER ADDRESS OBUFF
 2595 006414 177377 -401 ;WORD COUNT -401
 2596 006416 000023 WRDATA ;COMMAND WRDATA
 2597
 2598 006420 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA
 2599 006424 000002 2 ;PATTERN 2
 2600
 2601 006426 004437 032274 JSR R4,OPSTRT ;START THE OPERATION
 2602
 2603 006432 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
 2604 006436 104004 ERROR 4 ;CS1 MISCOMPARE
 2605 006440 104005 ERROR 5 ;MR1 ''
 2606 006442 104006 ERROR 6 ;MR2 ''
 2607 006444 104007 ERROR 7 ;MR3 ''
 2608
 2609 006446 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
 2610
 2611 006452 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
 2612 006456 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
 2613 006460 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
 2614
 2615 006462 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
 2616 006466 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
 2617 006470 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
 2618 006472 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
 2619
 2620 006474 012737 043675 001510 MOV #EM5020,EMW ;SET MESSAGE FOR WRITE FAILURE
 2621 006502 004437 025204 JSR R4,DWGPSN ;SIMULATE GAP AND SYNC FOR WRITE
 2622 006506 104015 ERROR 15 ;MR1 IN ERROR IN READ GAP
 2623 006510 104016 ERROR 16 ;CS1 IN ERROR AFTER READ GAP

```

2624 006512 104017          ERROR 17          ;MR1 IN ERROR IN WRITING SYNC
2625 006514 104020          ERROR 20          ;CS1 IN ERROR AFTER WRITING SYNC
2626
2627 006516 012737 043754 001510  MOV    #EM5021,EMW    ;SET MESSAGE FOR WRITE FAILURE
2628 006524 004437 025604          JSR    R4,DWRITE     ;SIMULATE WRITE DATA AND ECC
2629 006530 104017          ERROR 17          ;MR1 IN ERROR IN WRITING DATA OR ECC
2630 006532 104021          ERROR 21          ;ECC IN ERROR WHILE WRITING DATA
2631 006534 104022          ERROR 22          ;CS1 IN ERROR AFTER WPITING DATA AND ECC
2632
2633
2634 006536 012700 000016          MOV    #3*4+2,R0     ;SET COUNT TO COMPLETE OPERATION
2635 006542 012762 000440 000026 1$:    MOV    #DMD.MCLK,RKMR1(R2) ;RUN THE CLOCK
2636 006550 012762 000040 000026          MOV    #DMD,RKMR1(R2)
2637 006556 005300          DEC    R0            ;DEC COUNT
2638 006560 001370          BNE    1$           ;LOOP UNTIL 0
2639
2640 006562 004437 032356          JSR    R4,GETREG     ;GET RK611 REGS
2641 006566 032737 001000 002174          BIT    #COE,T.ER     ;TEST IF COE SET
2642 006574 00101?          BNE    3$           ;YES - EXIT TEST
2643
2644 006576 032737 100000 002160          BIT    #CERR,T.CS1  ;TEST IF ANY OTHER ERROR SET
2645 006604 001402          BEQ    2$           ;NO - SKIP
2646 006606 104027          ERROR 27          ;'NO COE WHEN EXPECTED'
2647 006610 000404          BR     3$           ;GO TO EXIT
2648
2649 006612 012737 044545 001602 2$:    MOV    #E5026A,EH026 ;SET HEADER
2650 006620 104026          ERROR 26          ;'UNEXPECTED ERROR WHEN FORCING COE'
2651
2652 006622          3$:
2653
2654          .SBTTL **NPR WRITING OF MEMORY
2655
2656          ;*****
2657          ;*TEST 10      NPR WRITING OF MEMORY
2658          ;*
2659          ;*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
2660          ;*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
2661          ;*      OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
2662          ;*      CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
2663          ;*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE,
2664          ;*      A GOOD HEADER, 40 DATA WORDS.  VERIFY THAT THE WORDS
2665          ;*      ARE WRITTEN PROPERLY IN MEMORY.
2666          ;*
2667          ;*****
2668 006622 000004          TST10: SCOPE
2669 006624 012737 000012 001234          MOV    #10.,$TIMES  ;;DO 10. ITERATIONS
2670
2671 006632 005037 002352          CLR    BSEDES       ;CLEAR BAD SEC DESIRED
2672 006636 012762 100000 000000          MOV    #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
2673
2674 006644 005037 002344          CLR    ECCSRC       ;CLEAR ECC SOURCE
2675 006650 005037 002354          CLR    HIBITS       ;CLEAR HI ORDER BITS
2676 006654 004437 032230          JSR    R4,LOADRK    ;LOAD 'L' REGISTERS
2677 006660 000000          0                ;CYLINDER 0
2678 006662 000          .BYTE 0            ;SECTOR 0
2679 006663 000          .BYTE 0            ;TRACK 0
  
```

```

2680 006664 047124      ;IBUFF      ;BUFFER ADDRESS IBUFF
2681 006666 177400      -400      ;WORD COUNT -400
2682 006670 000021      RDDATA    ;COMMAND RDDATA
2683
2684 006672 004437 031770 JSR      R4,CLRIBF    ;GO CLEAR INPUT BUFFER
2685
2686 006676 004437 032014 JSR      R4,BLDDAI    ;GO BUILD DATA
2687 006702 000003      3          ;PATTERN 3
2688
2689
2690 006704 004437 032274 JSR      R4,OPSTRT    ;START THE OPERATION
2691 006710 004437 023334 JSR      R4,DISEEK    ;SIMULATE IMPLIED SEEK
2692 006714 104004      ERROR 4      ;CSI MISCMPARE
2693 006716 104005      ERROR 5      ;MR1
2694 006720 104006      ERROR 6      ;MR2
2695 006722 104007      ERROR 7      ;MR3
2696
2697 006724 004437 023316 JSR      R4,DSECTR    ;SIMULATE SECTOR PULSE
2698
2699 006730 004437 023740 JSR      R4,DRSYNC    ;SIMULATE HEADER PREAMBLE
2700 006734 104010      ERROR 10     ;MR1 CONTENTS IN ERROR
2701 006736 104011      ERROR 11     ;CSI CONTENTS IN ERROR
2702
2703 006740 004437 024276 JSR      R4,DHDCMP    ;SIMULATE HEADER SEARCH
2704 006744 104012      ERROR 12     ;MR1 CONTENTS IN ERROR
2705 006746 104013      ERROR 13     ;CSI IN ERROR AFTER SEARCH
2706 006750 104014      ERROR 14     ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2707
2708 006752 004437 026500 JSR      R4,DRGPSN    ;GO READ GAP AND SYNC
2709 006756 104015      ERROR 15     ;MR1 IN ERROR READING GAP
2710 006760 104016      ERROR 16     ;CSI IN ERROR AFTER READING GAP
2711 006762 104032      ERROR 32     ;MR1 IN ERROR READING SYNC
2712 006764 104033      ERROR 33     ;CSI IN ERROR AFTER READING SYNC
2713
2714 006766 012737 000000 002346 MOV      #0,ECPATX    ;LOAD EXPECTED PATTERN
2715 006774 012737 004066 002350 MOV      #4066,ECPOSX ;LOAD EXPECTED POSITION
2716 007002 004437 027176 JSR      R4,DREAD     ;GO SIMULATE DATA READ
2717 007006 000060      60          ;NUMBER OF WORDS TO SIMULATE
2718 007010 104034      ERROR 34     ;MR1 IN ERROR READING DATA OR ECC
2719 007012 104035      ERROR 35     ;ECC ERROR READING DATA
2720 007014 104036      ERROR 36     ;CSI ERROR AFTER READING DATA OR ECC
2721 007016 104041      ERROR 41     ;ECC REG INCORRECT AFTER ECC READ
2722 007020 104042      ERROR 42     ;ERR IN ECC PAT CALC.
2723 007022 104043      ERROR 43     ;ERR IN ECC POS COUNT
2724
2725
2726 007024 012700 050130 MOV      #0BUFF,R0    ;SET POINTER TO GOOD DATA
2727 007030 012701 047124 MOV      #IBUFF,R1    ;SET POINTER TO INPUT DATA
2728
2729 007034 012704 000012 MOV      #10,R4       ;SET ERROR LIMIT COUNT
2730 007040 005037 001236 CLR      $ESCAPE     ;CLEAR ESCAPE
2731 007044 005037 001220 CLR      $TMP7       ;CLEAR ERROR REPORT SWITCH
2732 007050 012703 000040 MOV      #40,R3       ;SET COMPARE LIMIT
2733 007054 005005      CLR      R5         ;CLEAR WORD COUNTER
2734
2735 007056 021011      18: (MP (R0),(R1) ;COMPARE DATA

```



```

2736 007060 001005          BNE      2$          ;SKIP IF NOT EQUAL
2737 007062 005303      4$: DEC      R3          ;ELSE DEC WORD COUNT LIMIT
2738 007064 001424      BEQ      5$          ;IF 0 - SKIP TO EXIT
2739 007066 005205      INC      R5          ;BUMP WORD COUNT
2740 007070 022021      CMP      (R0)+,(R1)+ ;BUMP DATA POINTERS
2741 007072 000771      BR       1$          ;LOOP
2742
2743 007074 005304      2$: DEC      R4          ;DEC ERROR LIMIT
2744 007076 001417      REG      5$          ;IF 0 - SKIP
2745
2746 007100 011037 001162    MOV      (R0),$REG0    ;GET GOOD WORD FOR REPORT
2747 007104 011137 001164    MOV      (R1),$REG1    ;GET BAD WORD
2748 007110 010537 001166    MOV      R5,$REG2     ;GET WORD NUMBER
2749
2750 007114 005737 001220    TST      $TMP7         ;DECIDE WHICH ERROR REPORT TO USE
2751 007120 001004          BNE      3$          ;SKIP IF NOT 0
2752 007122 005237 001220    INC      $TMP7         ;ELSE BUMP $TMP7
2753 007126 104037          ERROR    37          ;REPORT FIRST ERROR LINE
2754 007130 000754          BR       4$          ;SKIP
2755
2756 007132 104040      3$: ERROR 40          ;REPORT ALL OTHER LINES
2757 007134 000752      BR       4$          ;SKIP
2758
2759 007136      5$:
2760
2761      .SBT'L  **ECC ERROR DETECTION/CORRECTION TESTS
2762
2763      ;*****
2764      ;*TEST 11      READ DATA WITH NO ERROR
2765      ;*
2766      ;*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
2767      ;*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
2768      ;*      OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
2769      ;*      CLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
2770      ;*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE,
2771      ;*      A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
2772      ;*      VERIFY THAT COMMAND COMPLETES WITHOUT ERROR.
2773      ;*
2774      ;*****
2775 007136 000004      TST11: SCOPE
2776 007140 012737 000001 001234    MOV      #1,$TIMES    ;;DO 1 ITERATION
2777
2778 007146 005037 002352    CLR      BSEDES       ;CLEAR BAD SEC DESIRED
2779 007152 012762 100000 000000    MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLIER
2780 007160 005037 002344    CLR      ECCSRC       ;CLEAR ECC SOURCE FLAG
2781 007164 005037 002344    CLR      ECCSRC       ;CLEAR ECC SOURCE
2782 007170 005037 002354    CLR      HIBITS       ;CLEAR HI ORDER BITS
2783 007174 004437 032230    JSR      R4,LOADRK    ;LOAD "L" REGISTERS
2784 007200 000000          0                    ;CYLINDER 0
2785 007202 000          .BYTE    0            ;SECTOR 0
2786 007203 000          .BYTE    0            ;TRACK 0
2787 007204 047124          Ibuff          ;BUFFER ADDRESS Ibuff
2788 007206 177400          -400          ;WORD COUNT -400
2789 007210 000021          RDDATA        ;COMMAND RDDATA
2790
2791 007212 004437 031770    JSR      R4,CLRIBF    ;GO CLEAR INPUT BUFFER
  
```

```

2792
2793 007216 004437 032014      SR      R4,BLDDAT      ;GO BUILD DATA
2794 007222 000004              4              ;PATTERN 4
2795
2796
2797 007224 004437 032274      JSR      R4,OPSTRT    ;START THE OPERATION
2798 007230 004437 023334      JSR      R4,DISEEK    ;SIMULATE IMPLIED SEEK
2799 007234 104004              ERROR     4           ;CS1 MISCOMPARE
2800 007236 104005              ERROR     5           ;MR1
2801 007240 104006              ERROR     6           ;MR2
2802 007242 104007              ERROR     7           ;MR3
2803
2804 007244 004437 023316      JSR      R4,DSECTR    ;SIMULATE SECTOR PULSE
2805
2806 007250 004437 023740      JSR      R4,DRSYNC    ;SIMULATE HEADER PREAMBLE
2807 007254 104010              ERROR     10          ;MR1 CONTENTS IN ERROR
2808 007256 104011              ERROR     11          ;CS1 CONTENTS IN ERROR
2809
2810 007260 004437 024276      JSR      R4,DHDCMP    ;SIMULATE HEADER SEARCH
2811 007264 104012              ERROR     12          ;MR1 CONTENTS IN ERROR
2812 007266 104013              ERROR     13          ;CS1 IN ERROR AFTER SEARCH
2813 007270 104014              ERROR     14          ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
2814
2815 007272 004437 026500      JSR      R4,DRGPSN    ;GO READ GAP AND SYNC
2816 007276 104015              ERROR     15          ;MR1 IN ERROR READING GAP
2817 007300 104016              ERROR     16          ;CS1 IN ERROR AFTER READING GAP
2818 007302 104032              ERROR     32          ;MR1 IN ERROR READING SYNC
2819 007304 104033              ERROR     33          ;CS1 IN ERROR AFTER READING SYNC
2820
2821 007306 012737 000000 002346  MOV      #0,ECPATX    ;LOAD EXPECTED PATTERN
2822 007314 012737 004066 002350  MOV      #4066,ECPOSX ;LOAD EXPECTED POSITION
2823 007322 004437 027176      JSR      R4,DRREAD    ;GO SIMULATE DATA READ
2824 007326 000400              400            ;NUMBER OF WORDS TO SIMULATE
2825 007330 104034              ERROR     34          ;MR1 IN ERROR READING DATA OR ECC
2826 007332 104035              ERROR     35          ;ECC ERROR READING DATA
2827 007334 104036              ERROR     36          ;CS1 ERROR AFTER READING DATA OR ECC
2828 007336 104041              ERROR     41          ;ECC REG INCORRECT AFTER ECC READ
2829 007340 104042              ERROR     42          ;ERR IN ECC PAT CALC.
2830 007342 104043              ERROR     43          ;ERR IN ECC POS COUNT
2831
2832
2833 007344 012700 002710      MOV      #40.*37.,R0  ;SET COUNT TO EMPTY SILO
2834
2835 007350 012762 000440 000026 1$:  MOV      #DMD!MCLK,RKMR1(R2) ;CLOCK CONTROLLER
2836 007356 012762 000040 000026  MOV      #DMD,RKMR1(R2)
2837 007364 005300              DEC      R0           ;DEC COUNT
2838 007366 001370              BNE     1$           ;LOOP UNTIL 0
2839
2840 007370 016237 000000 002160  MOV      RKCS1(R2),T.CS1 ;GET CS1
2841 007376 004437 032356      JSR      R4,GETREG    ;GET 611 REGS
2842 007402 042737 000001 002220  BIC      #GO,E.CS1    ;CLEAR GO BIT
2843 007410 052737 000200 002220  BIS      #RDY,E.CS1   ;SET READY BIT
2844 007416 023737 002160 002220  CMP      T.CS1,E.CS1  ;CHECK IF CS1 CORRECT
2845 007424 001401              BEQ     2$           ;YES - SKIP
2846 007426 104056              ERROR     56          ;'CS1 IN ERROR AFTER READ DATA'
2847 007430
  
```

2\$:

```

2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863 007430 000004
2864 007432 012737 000001 001234
2865
2866 007440 005037 002352
2867 007444 012762 100000 000000
2868
2869 007452 005037 002354
2870 007456 004437 032230
2871 007462 000000
2872 007464 000
2873 007465 000
2874 007466 047124
2875 007470 177400
2876 007472 000021
2877
2878 007474 004437 031770
2879
2880 007500 004437 032014
2881 007504 000007
2882
2883 007506 012700 000530
2884
2885 007512 012737 177446 051130
2886 007520 012737 015457 051132
2887 007526 012737 000001 002344
2888 007534 032760 000100 050130
2889 007542 001004
2890 007544 052760 000100 050130
2891 007552 000403
2892 007554 042760 000100 050130 100$:
2893 007562 101$:
2894
2895 007562 004437 032274
2896 007566 004437 023334
2897 007572 104004
2898 007574 104005
2899 007576 104006
2900 007600 104007
2901
2902 007602 004437 023316
2903

```

TEST 12 READ DATA WITH ONE BIT BURST ERROR (PART 1)

CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
A ONE BIT BURST ERROR OF A 0 THAT SHOULD BE A 1.
VERIFY THE COUNTING OF THE POSITION
REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.

```

TST12: SCOPE
MOV #1,$TIMES ;DO 1 ITERATION

CLR BSEDES ;CLEAR BAD SEC DESIRED
MOV #CLR,RK(CS1(R2) ;CLEAR CONTROLLER

CLR HIBITS ;CLEAR HI ORDER BITS
JSR R4,LOADRK ;LOAD 'L' REGISTERS
0 ;CYLINDER 0
.BYTE 0 ;SECTOR 0
.BYTE 0 ;TRACK 0
IBUFF ;BUFFER ADDRESS IBUFF
-400 ;WORD COUNT -400
RDDATA ;COMMAND RDDATA

JSR R4,CLRIBF ;GO CLEAR INPUT BUFFER

JSR R4,BLDDAT ;GO BUILD DATA
7 ;PATTERN 7

MOV #530,R0 ;SET INDEX INTO BUFFER

MOV #177446,OBUFF+1000 ;STORE ECC WORDS
MOV #15457,OBUFF+1002
MOV #1,ECCSRC ;SET SOURCE TO INDICATE ECC IN BUFFER
BIT #BIT6,OBUFF(R0) ;TEST IF BIT SET
BNE 100$ ;YES - SKIP
BIS #BIT6,OBUFF(R0) ;ELSE SET BIT
BR 101$
BIC #BIT6,OBUFF(R0) ;CLEAR BIT

JSR R4,OPSTRT ;START THE OPERATION
JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
ERROR 4 ;CS1 MISCOMPARE
ERROR 5 ;MR1
ERROR 6 ;MR2
ERROR 7 ;MR3

JSR R4,DSECTR ;SIMULATE SECTOR PULSE

```

2904	007606	004437	02374C		JSR	R4,DRSYNC		;SIMULATE HEADER PREAMBLE
2905	007612	104010			ERROR	10		;MR1 CONTENTS IN ERROR
2906	007614	104011			ERROR	11		;CS1 CONTENTS IN ERROR
2907								
2908	007616	004437	024276		JSR	R4,DHDCMP		;SIMULATE HEADER SEARCH
2909	007622	104012			ERROR	12		;MR1 CONTENTS IN ERROR
2910	007624	104013			ERROR	13		;CS1 IN ERROR AFTER SEARCH
2911	007626	104014			ERROR	14		;RKDCYL OR RKDA IN ERPOP AFTER SEARCH
2912								
2913	007630	004437	026500		JSR	R4,DRGPSN		;GO READ GAP AND SYNC
2914	007634	104015			ERROR	15		;MR1 IN ERROR READING GAP
2915	007636	104016			ERROR	16		;CS1 IN ERROR AFTER READING GAP
2916	007640	104032			ERROR	32		;MR1 IN ERROR READING SYNC
2917	007642	104033			ERROR	33		;CS1 IN ERROR AFTER READING SYNC
2918								
2919	007644	012737	000001	002346	MOV	#1,ECPTX		;LOAD EXPECTED PATTERN
2920	007652	012737	004066	002350	MOV	#4066,ECPOSX		;LOAD EXPECTED POSITION
2921	007660	004437	027176		JSR	R4,DREAD		;GO SIMULATE DATA READ
2922	007664	000400			400			;NUMBER OF WORDS TO SIMULATE
2923	007666	104034			ERROR	34		;MR1 IN ERROR READING DATA OR ECC
2924	007670	104035			ERROR	35		;ECC ERROR READING DATA
2925	007672	104036			ERROR	36		;CS1 ERROR AFTER READING DATA OR ECC
2926	007674	104041			ERROR	41		;ECC REG INCORRECT AFTER ECC READ
2927	007676	104042			ERROR	42		;ERR IN ECC PAT CALC.
2928	007700	104043			ERROR	43		;ERR IN ECC POS COUNT
2929								
2930								
2931	007702	012737	000020	002326	MOV	#20,BITCNT		;SET BIT COUNT FOR CORRECTION COUNT
2932	007710	012700	000005		MOV	#5,R0		;SET COUNT FOR ECCPOS PASS THRU 0
2933	007714	012701	013672		MOV	#13672,R1		;SET COUNT FOR ECCPOS TO 0 FIRST PASS
2934	007720	005037	001214		CLR	\$TMP5		;CLEAR SWITCH
2935								
2936	007724	004737	031616		3\$: JSR	PC,RDBIT		;GO RFAD BIT
2937	007730	004737	030462		JSR	PC,ECCGEN		;GENERATE ECC
2938	007734	016237	000032	002214	MOV	RKECPT(R2),T.ECPT		;GET PATTERN
2939	007742	016237	000030	002212	MOV	RKECPS(R2),T.ECPS		;GET POSITION
2940	007750	005237	002350		INC	ECPOSX		;BUMP POSITION EXPECTED
2941	007754	013737	002350	002252	MOV	ECPOSX,E.ECPS		;SET FOR COMPARE AND REPORT
2942	007762	023737	002254	002214	CMP	E.ECPT,T.ECPT		;CHECK IF PATTERN CORRECT
2943	007770	001401			BEQ	1\$;YES - SKIP
2944	007772	104042			ERROR	42		
2945	007774	023737	002252	002212	1\$: CMP	E.ECPS,T.ECPS		;CHECK IF POSITION CORRECT
2946	010002	001401			BEQ	2\$;YES - SKIP
2947	010004	104043			ERROR	43		
2948	010006	005237	002326		2\$: INC	BITCNT		;BUMP BIT COUNT
2949	010012	005301			DEC	R1		;DEC COUNT TO ZERO IN ECCPOS
2950	010014	001343			BNE	3\$;NOT YET ZERO - LOOP
2951	010016	005300			DEC	R0		;DEC PASS THRU 0 COUNT
2952	010020	001406			BEQ	4\$;IF 0 - SKIP
2953	010022	012737	177777	002350	MOV	#-1,ECPOSX		;PRESET EXPECTED POS TO GO TO ZERO
2954	010030	012701	020000		MOV	#20000,R1		;SET COUNT TO NEXT 0 IN ECC POS
2955	010034	000733			BR	3\$;GO DO LOOP AGAIN
2956								
2957	010036	005737	001214		4\$: TST	\$TMP5		;TEST SWITCH
2958	010042	001012			BNE	5\$;ALREADY SET - SKIP
2959	010044	012701	005276		MOV	#5276,R1		;SET COUNT TO FINAL POSITION

```

2960 010050 012700 000001      MOV    #1,R0          ;SET R0 TO RETURN AFTER ONE PASS
2961 010054 012737 177777 002350  MOV    #-1,ECPOSX    ;SET EXP POS TO GO TO 0 ON NEXT BIT
2962 010062 005237 001214      INC    $TMP5         ;BUMP SWITCH
2963 010066 000716              BR     3$           ;GO DO LOOP AGAIN
2964
2965 010070 004737 031616      5$:   JSR    PC,RDBIT    ;ONE MORE BIT TO SET READY
2966 010074 004437 032356      JSR    R4,GETREG    ;GET '611 REGISTERS
2967 010100 013737 002260 002220  MOV    L,CS1,E,CS1  ;GET EXPECTED CS1
2968 010106 052737 100200 002220  BIS    #RDY!CERR,E,CS1 ;SET READY AND ERROR
2969 010114 042737 000001 002220  BIC    #GO,E,CS1    ;CLEAR GO
2970 010122 023737 002160 002220  CMP    T,CS1,E,CS1  ;CHECK IF CS1 CORRECT
2971 010130 001402              BEQ    6$           ;YES - SKIP
2972 010132 104044              ERROR  44
2973 010134 000410              BR     7$
2974 010136 032737 000100 002174  6$:   BIT    #ECH,T,ER    ;TEST IF ECC HARD ERROR SET
2975 010144 001404              BEQ    7$           ;NO - SKIP
2976 010146 012737 100000 002234  MOV    #DCK,E,ER    ;SET EXPECTED ERROR REG
2977 010154 104045              ERROR  45
2978 010156      7$:
2979

```

```

2980
2981      :*****
2982      :*TEST 13      READ DATA WITH 11 BIT BURST ERROR
2983      :*
2984      :*          CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
2985      :*          PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
2986      :*          OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
2987      :*          CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
2988      :*          AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
2989      :*          A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
2990      :*          AN 11 BIT BURST ERROR. VERIFY THE COUNTING OF THE POSITION
2991      :*          REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
2992      :*          MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.
2993      :*****

```

```

2994 010156 000004      TST13: SCOPE
2995 010160 012737 000001 001234  MOV    #1,$TIMES    ;;DO 1 ITERATION
2996 010166 005037 002352      CLR    BSEDES       ;CLEAR BAD SEC DESIRED
2997 010172 012762 100000 000000  MOV    #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
2998
2999 010200 005037 002354      CLR    HIBITS       ;CLEAR HI ORDER BITS
3000 010204 004437 032230      JSR    R4,LOADRK    ;LOAD 'L' REGISTERS
3001 010210 000000              0                   ;CYLINDER 0
3002 010212              .BYTE 0             ;SECTOR 0
3003 010213              .BYTE 0             ;TRACK 0
3004 010214 047124      Ibuff          ;BUFFER ADDRESS Ibuff
3005 010216 177400      -400            ;WORD COUNT -400
3006 010220 000021      RDDATA          ;COMMAND RDDATA
3007
3008 010222 004437 031770      JSR    R4,CLRIBF    ;GO CLEAR INPUT BUFFER
3009
3010 010226 004437 032014      JSR    R4,BLDDAT    ;GO BUILD DATA
3011 010232 000007              7                   ;PATTERN 7
3012
3013 010234 012700 000530      MOV    #530,R0      ;SET INDEX INTO BUFFER
3014
3015 010240 012737 177446 051130  MOV    #177446,Obuff+1000 ;STORE ECC WORDS

```

```

3016 010246 012737 015457 051132 MOV #15457,OBUFF+1002
3017 010254 012737 000001 002344 MOV #1,ECCSRC ;SET SCURCE TO INDICATE ECC IN BUFFER
3018 010262 032760 000100 050130 BIT #BIT6,OBUFF(R0) ;TEST IF BIT SET
3019 010270 001004 BNE 100$ ;YES - SKIP
3020 010272 052760 000100 050130 BIS #BIT6,OBUFF(R0) ;ELSE SET BIT
3021 010300 000403 BR 101$
3022 010302 042760 000100 050130 100$: BIC #BIT6,OBUFF(R0) ;CLEAR BIT
3023 010310 101$:
3024
3025 010310 004437 032274 JSR R4,OPSTRT ;START THE OPERATION
3026 010314 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
3027 010320 104004 ERROR 4 ;CS1 MISCOMPARE
3028 010322 104005 ERROR 5 ;MR1 ..
3029 010324 104006 ERROR 6 ;MR2 ..
3030 010326 104007 ERROR 7 ;MR3 ..
3031
3032 010330 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
3033
3034 010334 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
3035 010340 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
3036 010342 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
3037
3038 010344 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
3039 010350 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
3040 010352 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
3041 010354 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
3042
3043 010356 004437 026500 JSR R4,DRGPSN ;GO READ GAP AND SYNC
3044 010362 104015 ERROR 15 ;MR1 IN ERROR READING GAP
3045 010364 104016 ERROR 16 ;CS1 IN ERROR AFTER READING GAP
3046 010366 104032 ERROR 32 ;MR1 IN ERROR READING SYNC
3047 010370 104033 ERROR 33 ;CS1 IN ERROR AFTER READING SYNC
3048
3049 010372 012737 000001 002346 MOV #1,ECPATX ;LOAD EXPECTED PATTERN
3050 010400 012737 004066 002350 MOV #4066,ECPOSX ;LOAD EXPECTED POSITION
3051 010406 004437 027176 JSR R4,DREAD ;GO SIMULATE DATA READ
3052 010412 000400 400 ;NUMBER OF WORDS TO SIMULATE
3053 010414 104034 ERROR 34 ;MR1 IN ERROR READING DATA OR ECC
3054 010416 104035 ERROR 35 ;ECC ERROR READING DATA
3055 010420 104036 ERROR 36 ;CS1 ERROR AFTER READING DATA OR ECC
3056 010422 104041 ERROR 41 ;ECC REG INCORRECT AFTER ECC READ
3057 010424 104042 ERROR 42 ;ERR IN ECC PAT CALC.
3058 010426 104043 ERROR 43 ;ERR IN ECC POS COUNT
3059
3060
3061 010430 012737 000020 002326 MOV #20,BITCNT ;SET BIT COUNT FOR CORRECTION COUNT
3062 010436 012700 000005 MOV #5,R0 ;SET COUNT FOR ECCPOS PASS THRU 0
3063 010442 012701 013672 MOV #13672,R1 ;SET COUNT FOR ECCPOS TO 0 FIRST PASS
3064 010446 005037 001214 CLR $TMP5 ;CLEAR SWITCH
3065
3066 010452 004737 031616 3$: JSR PC,RDBIT ;GO READ BIT
3067 010456 004737 030462 JSR PC,ECCGEN ;GENERATE ECC
3068 010462 016237 000032 002214 MOV RKECPT(R2),T.ECPT ;GET PATTERN
3069 010470 016237 000030 002212 MOV RKECPS(R2),T.ECPS ;GET POSITION
3070 010476 005237 002350 INC ECPOSX ;BUMP POSITION EXPECTED
3071 010502 013737 002350 002252 MOV ECPOSX,E.ECPS ;SET FOR COMPARE AND REPORT
  
```

```

3072 010510 023737 002254 002214      CMP      E.ECPT,T.ECPT      ;CHECK IF PATTERN CORRECT
3073 010516 001401                      BEQ      1$                ;YES - SKIP
3074 010520 104042                      ERROR    42
3075 010522 023737 002252 002212 1$:   CMP      E.ECPS,T.ECPS      ;CHECK IF POSITION CORRECT
3076 010530 001401                      BEQ      2$                ;YES - SKIP
3077 010532 104043                      ERROR    43
3078 010534 005237 002326              2$:   INC      BITCNT          ;BUMP BIT COUNT
3079 010540 005301                      DEC      R1                ;DEC COUNT TO ZERO IN ECCPOS
3080 010542 001343                      BNE      3$                ;NOT YET ZERO - LOOP
3081 010544 005300                      DEC      R0                ;DEC PASS THRU 0 COUNT
3082 010546 001406                      BEQ      4$                ;IF 0 - SKIP
3083 010550 012737 177777 002350      MOV      #-1,ECP0SX        ;PRESET EXPECTED POS TO GO TO ZERO
3084 010556 012701 020000              MOV      #20000,R1         ;SET COUNT TO NEXT 0 IN ECC POS
3085 010562 000733                      BR       3$                ;GO DO LOOP AGAIN
3086
3087 010564 005737 001214              4$:   TST      $TMP5          ;TEST SWITCH
3088 010570 001012                      BNE      5$                ;ALREADY SET - SKIP
3089 010572 012701 005276              MOV      #5276,R1         ;SET COUNT TO FINAL POSITION
3090 010576 012700 000001              MOV      #1,R0            ;SET R0 TO RETURN AFTER ONE PASS
3091 010602 012737 177777 002350      MOV      #-1,ECP0SX        ;SET EXP POS TO GO TO 0 ON NEXT BIT
3092 010610 005237 001214              INC      $TMP5          ;BUMP SWITCH
3093 010614 000716                      BR       3$                ;GO DO LOOP AGAIN
3094
3095 010616 004737 031616              5$:   JSR      PC,RDBIT        ;ONE MORE BIT TO SET READY
3096 010622 004437 032356              JSR      R4,GETREG        ;GET '611 REGISTERS
3097 010626 013737 002260 002220      MOV      L.CS1,E.CS1      ;GET EXPECTED CS1
3098 010634 052737 100200 002220      BIS      #RDY!CERR,E.CS1  ;SET READY AND ERROR
3099 010642 042737 000001 002220      BIC      #GO,E.CS1        ;CLEAR GO
3100 010650 023737 002160 002220      CMP      1.CS1,E.CS1      ;CHECK IF CS1 CORRECT
3101 010656 001402                      BEQ      6$                ;YES - SKIP
3102 010660 104044                      ERROR    44
3103 010662 000410                      BR       7$
3104 010664 032737 000100 002174 6$:   BIT      #ECH,T.ER        ;TEST IF ECC HARD ERROR SET
3105 010672 001404                      BEQ      7$                ;NO SKIP
3106 010674 012737 100000 002234      MOV      #DCK,E.ER        ;SET EXPECTED ERROR REG
3107 010702 104045                      ERROR    45
3108
3109 010704              7$:
3110
3111
3112
3113
3114
3115
3116
3117
3118
3119
3120
3121
3122
3123
3124
3125
3126 010704 000004
3127 010706 012737 000001 001234
  
```

```

*****
*TEST 14      READ DATA WITH ONE BIT BURST ERROR (PART 2)
*
*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
*      OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
*      CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE,
*      A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
*      A ONE BIT BURST ERROR OF A 1 THAT SHOULD BE A 0.
*      VERIFY THE COUNTING OF THE POSITION
*      REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.
*      MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.
*****
TST14:  SCOPE
        MOV      #1,$TIMES      ;:DO 1 ITERATION
  
```

```
3128  
3129 010714 005037 002352 CLR BSEDES ;CLEAR BAD SEC DESIRED  
3130 010720 012762 100000 000000 MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER  
3131  
3132 010726 005037 002354 CLR HIBITS ;CLEAR HI ORDER BITS  
3133 010732 004437 032230 JSR R4,LOADRK ;LOAD 'L' REGISTERS  
3134 010736 000000 0 ;CYLINDER 0  
3135 010740 000 .BYTE 0 ;SECTOR 0  
3136 010741 000 .BYTE 0 ;TRACK 0  
3137 010742 047124 IBUFF ;BUFFER ADDRESS IBUFF  
3138 010744 177400 -400 ;WORD COUNT -400  
3139 010746 000021 RDDATA ;COMMAND RDDATA  
3140  
3141 010750 004437 031770 JSR R4,CLRIBF ;GO CLEAR INPUT BUFFER  
3142  
3143 010754 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA  
3144 010760 000005 5 ;PATTERN 5  
3145  
3146 010762 012700 000530 MOV #530,R0 ;SET INDEX INTO BUFFER  
3147  
3148 010766 012737 126073 051130 MOV #126073,OBUFF+1000 ;STORE ECC WORDS  
3149 010774 012737 151052 051132 MOV #151052,OBUFF+1002  
3150 011002 012737 000001 002344 MOV #1,ECCSRC ;SET SOURCE TO INDICATE ECC IN BUFFER  
3151 011010 032760 000100 050130 BIT #BIT6,OBUFF(R0) ;TEST IF BIT SET  
3152 011016 001004 BNE 100$ ;YES - SKIP  
3153 011020 052760 000100 050130 BIS #BIT6,OBUFF(R0) ;ELSE SET BIT  
3154 011026 000403 BR 101$  
3155 011030 042760 000100 050130 100$: BIC #BIT6,OBUFF(R0) ;CLEAR BIT  
3156 011036 101$:  
3157  
3158 011036 004437 032274 JSR R4,OPSTRT ;START THE OPERATION  
3159 011042 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK  
3160 011046 104004 ERROR 4 ;CS1 MISCOMPARE  
3161 011050 104005 ERROR 5 ;MR1  
3162 011052 104006 ERROR 6 ;MR2  
3163 011054 104007 ERROR 7 ;MR3  
3164  
3165 011056 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE  
3166  
3167 011062 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE  
3168 011066 104010 ERROR 10 ;MR1 CONTENTS IN ERROR  
3169 011070 104011 ERROR 11 ;CS1 CONTENTS IN ERROR  
3170  
3171 011072 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH  
3172 011076 104012 ERROR 12 ;MR1 CONTENTS IN ERROR  
3173 011100 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH  
3174 011102 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH  
3175  
3176 011104 004437 026500 JSR R4,DRGPSN ;GO READ GAP AND SYNC  
3177 011110 104015 ERROR 15 ;MR1 IN ERROR READING GAP  
3178 011112 104016 ERROR 16 ;CS1 IN ERROR AFTER READING GAP  
3179 011114 104032 ERROR 32 ;MR1 IN ERROR READING SYNC  
3180 011116 104033 ERROR 33 ;CS1 IN ERROR AFTER READING SYNC  
3181  
3182 011120 012737 000001 002346 MOV #1,ECPATX ;LOAD EXPECTED PATTERN  
3183 011126 012737 004066 002350 MOV #4066,ECPOSX ;LOAD EXPECTED POSITION
```


3184	011134	004437	027176		JSR	R4,DREAD	;GO SIMULATE DATA READ
3185	011140	000400			400		;NUMBER OF WORDS TO SIMULATE
3186	011142	104034			ERROR	34	;MR1 IN ERROR READING DATA OR ECC
3187	011144	104035			ERROR	35	;ECC ERROR READING DATA
3188	011146	104036			ERROR	36	;CS1 ERROR AFTER READING DATA OR ECC
3189	011150	104041			ERROR	41	;ECC REG INCORRECT AFTER ECC READ
3190	011152	104042			ERROR	42	;ERR IN ECC PAT CALC.
3191	011154	104043			ERROR	43	;ERR IN ECC POS COUNT
3192							
3193							
3194	011156	012737	000020	002326	MOV	#20,BITCNT	;SET BIT COUNT FOR CORRECTION COUNT
3195	011164	012700	000005		MOV	#5,R0	;SET COUNT FOR ECCPOS PASS THRU 0
3196	011170	012701	013672		MOV	#13672,R1	;SET COUNT FOR ECCPOS TO 0 FIRST PASS
3197	011174	005037	001214		CLR	\$TMP5	;CLEAR SWITCH
3198							
3199	011200	004737	031616		3\$: JSR	PC,RDBIT	;GO READ BIT
3200	011204	004737	030462		JSR	PC,ECCGEN	;GENERATE ECC
3201	011210	016237	000032	002214	MOV	RKECPT(R2),T.ECPT	;GET PATTERN
3202	011216	016237	000030	002212	MOV	RKECPS(R2),T.ECPS	;GET POSITION
3203	011224	005237	002350		INC	ECPOSX	;BUMP POSITION EXPECTED
3204	011230	013737	002350	002252	MOV	ECPOSX,E.ECPS	;SET FOR COMPARE AND REPORT
3205	011236	023737	002254	002214	CMP	E.ECPT,T.ECPT	;CHECK IF PATTERN CORRECT
3206	011244	001401			BEQ	1\$;YES - SKIP
3207	011246	104042			ERROR	42	
3208	011250	023737	002252	002212	1\$: CMP	E.ECPS,T.ECPS	;CHECK IF POSITION CORRECT
3209	011256	001401			BEQ	2\$;YES - SKIP
3210	011260	104043			ERROR	43	
3211	011262	005237	002326		2\$: INC	BITCNT	;BUMP BIT COUNT
3212	011266	005301			DEC	R1	;DEC COUNT TO ZERO IN ECCPOS
3213	011270	001343			BNE	3\$;NOT YET ZERO - LOOP
3214	011272	005300			DEC	R0	;DEC PASS THRU 0 COUNT
3215	011274	001406			BEQ	4\$;IF 0 - SKIP
3216	011276	012737	177777	002350	MOV	#-1,ECPOSX	;PRESET EXPECTED POS TO GO TO ZERO
3217	011304	012701	020000		MOV	#20000,R1	;SET COUNT TO NEXT 0 IN ECC POS
3218	011310	000733			BR	3\$;GO DO LOOP AGAIN
3219							
3220	011312	005737	001214		4\$: TST	\$TMP5	;TEST SWITCH
3221	011316	001012			BNE	5\$;ALREADY SET - SKIP
3222	011320	012701	005276		MOV	#5276,R1	;SET COUNT TO FINAL POSITION
3223	011324	012700	000001		MOV	#1,R0	;SET R0 TO RETURN AFTER ONE PASS
3224	011330	012737	177777	002350	MOV	#-1,ECPOSX	;SET EXP POS TO GO TO 0 ON NEXT BIT
3225	011336	005237	001214		INC	\$TMP5	;BUMP SWITCH
3226	011342	000716			BR	3\$;GO DO LOOP AGAIN
3227							
3228	011344	004737	031616		5\$: JSR	PC,RDBIT	;ONE MORE BIT TO SET READY
3229	011350	004437	032356		JSR	R4,GETREG	;GET '611 REGISTERS
3230	011354	013737	002260	002220	MOV	L.CS1,E.CS1	;GET EXPECTED CS1
3231	011362	052737	100200	002220	BIS	#RDY.CERR,E.CS1	;SET READY AND ERROR
3232	011370	042737	000001	002220	BIC	#GO,E.CS1	;CLEAR GO
3233	011376	023737	002160	002220	CMP	T.CS1,E.CS1	;CHECK IF CS1 CORRECT
3234	011404	001402			BEQ	6\$;YES - SKIP
3235	011406	104044			ERROR	44	
3236	011410	000410			BR	7\$	
3237	011412	032737	000100	002174	6\$: BIT	#ECH,T.ER	;TEST IF ECC HARD ERROR SET
3238	011420	001404			BEQ	7\$;NO SKIP
3239	011422	012737	100000	002234	MOV	#DCK,E.ER	;SET EXPECTED ERROR REG

3240 011430 104045

ERROR 45

3241
3242 011432

7\$:

3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256

```
*****  
*TEST 15 READ DATA WITH 11 BIT BURST ERROR  
*  
* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAP.  
* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA  
* OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,  
* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK  
* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,  
* A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING  
* AN 11 BIT BURST ERROR. VERIFY THE COUNTING OF THE POSITION  
* REGISTER AND THE FINAL PATTERN AND POSITION INFORMATION.  
* MAKE SURE DATA CHECK AND CONTROLLER ERROR SETS.  
*****
```

3257
3258 011432 000004
3259 011434 012737 000001 001234
3260
3261 011442 005037 002352
3262
3263 011446 012762 100000 000000
3264
3265 011454 005037 002354
3266 011460 004437 032230
3267 011464 000000
3268 011466 000
3269 011467 000
3270 011470 047124
3271 011472 177400
3272 011474 000021
3273
3274 011476 004437 031770
3275
3276 011502 004437 032014
3277 011506 000005
3278
3279 011510 012700 000530
3280
3281 011514 012737 126073 051130
3282 011522 012737 151052 051132
3283 011530 012737 000001 002344
3284 011536 042760 005000 050130
3285 011544 052760 050100 050130
3286 011552 062700 000002
3287 011556 052760 000001 050130
3288
3289 011564 004437 032274
3290 011570 004437 023334
3291 011574 104004
3292 011576 104005
3293 011600 104006
3294 011602 104007
3295

```
TST15: SCOPE  
MOV #1,$TIMES ;DO 1 ITERATION  
CLR BSEDES ;CLEAR BAD SEC DESIRED  
MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER  
CLR HIBITS ;CLEAR HI ORDER BITS  
JSR R4,LOADRK ;LOAD 'L' REGISTERS  
0 ;CYLINDER 0  
.BYTE 0 ;SECTOR 0  
.BYTE 0 ;TRACK 0  
IBUFF ;BUFFER ADDRESS IBUFF  
-400 ;WORD COUNT -400  
RDDATA ;COMMAND RDDATA  
JSR R4,CLRIBF ;GO CLEAR INPUT BUFFER  
JSR R4,BLDDAT ;GO BUILD DATA  
5 ;PATTERN 5  
MOV #530,R0 ;SET INDEX INTO BUFFER  
MOV #126073,OBUFF+1000 ;STORE ECC WORDS  
MOV #151052,OBUFF+1002  
MOV #1,ECCSRC ;SET SOURCE TO INDICATE ECC IN BUFFER  
BIC #BIT9!BIT11,OBUFF(R0) ;CREATE 11 BIT BURST ERROR  
BIS #BIT6!BIT12!BIT14,OBUFF(R0)  
ADD #2,R0 ;BUMP TO NEXT WORD  
BIS #BIT0,OBUFF(R0)  
JSR R4,OPSTRT ;START THE OPERATION  
JSR R4,DISEEK ;SIMULATE IMPLIED SEEK  
ERROR 4 ;CS1 MISCOMPARE  
ERROR 5 ;MR1 ..  
ERROR 6 ;MR2 ..  
ERROR 7 ;MR3 ..
```

3296	011604	004437	02306		JSR	R4,DSECTR	:SIMULATE SECTOR PULSE
3297							
3298	011610	004437	023740		JSR	R4,DRSYN	:SIMULATE HEADER PREAMBLE
3299	011614	104010			ERROR	10	:MR1 CONTENTS IN ERROR
3300	011616	104011			ERROR	11	:CS1 CONTENTS IN ERROR
3301							
3302	011620	004437	024276		JSR	R4,DHDCMF	:SIMULATE HEADER SEARCH
3303	011624	104012			ERROR	12	:MR1 CONTENTS IN ERROR
3304	011626	104013			ERROR	13	:CS1 IN ERROR AFTER SEARCH
3305	011630	104014			ERROR	14	:RKDCYL OR RKDA IN ERROR AFTER SEARCH
3306							
3307	011632	004437	026500		JSR	R4,DRGPSN	:GO READ GAP AND SYNC
3308	011636	104015			ERROR	15	:MR1 IN ERROR READING GAP
3309	011640	104016			ERROR	16	:CS1 IN ERROR AFTER READING GAP
3310	011642	104032			ERROR	32	:MR1 IN ERROR READING SYNC
3311	011644	104033			ERROR	33	:CS1 IN ERROR AFTER READING SYNC
3312							
3313	011646	012737	000001	002346	MOV	#1,ECPTX	:LOAD EXPECTED PATTERN
3314	011654	012737	004066	002350	MOV	#4066,ECPOSX	:LOAD EXPECTED POSITION
3315	011662	004437	027176		JSR	R4,DREAD	:GO SIMULATE DATA READ
3316	011666	000400			400		:NUMBER OF WORDS TO SIMULATE
3317	011670	104034			ERROR	34	:MR1 IN ERROR READING DATA OR ECC
3318	011672	104035			ERROR	35	:ECC ERROR READING DATA
3319	011674	104036			ERROR	36	:CS1 ERROR AFTER READING DATA OR ECC
3320	011676	104041			ERROR	41	:ECC REG INCORRECT AFTER ECC READ
3321	011700	104042			ERROR	42	:ERR IN ECC PAT CALC.
3322	011702	104043			ERROR	43	:ERR IN ECC POS COUNT
3323							
3324							
3325	011704	012737	000020	002326	MOV	#20,BITCNT	:SET BIT COUNT FOR CORRECTION COUNT
3326	011712	012700	000005		MOV	#5,R0	:SET COUNT FOR ECCPOS PASS THRU 0
3327	011716	012701	013672		MOV	#13672,R1	:SET COUNT FOR ECCPOS TO 0 FIRST PASS
3328	011722	005037	001214		CLR	\$TMP5	:CLEAR SWITCH
3329							
3330	011726	004737	031616		3\$: JSR	PC,RDBIT	:GO READ BIT
3331	011732	004737	030462		JSR	PC,ECCGEN	:GENERATE ECC
3332	011736	016237	000032	002214	MOV	RKECPT(R2),T.ECPT	:GET PATTERN
3333	011744	016237	000030	002212	MOV	RKECPS(R2),T.ECPS	:GET POSITION
3334	011752	005237	002350		INC	ECPOSX	:BUMP POSITION EXPECTED
3335	011756	013737	002350	002252	MOV	ECPOSX,E.ECPS	:SET FOR COMPARE AND REPORT
3336	011764	023737	002254	002214	CMP	E.ECPT,T.ECPT	:CHECK IF PATTERN CORRECT
3337	011772	001401			BEQ	1\$:YES - SKIP
3338	011774	104042			ERROR	42	
3339	011776	023737	002252	002212	1\$: CMP	E.ECPS,T.ECPS	:CHECK IF POSITION CORRECT
3340	012004	001401			BEQ	2\$:YES - SKIP
3341	012006	104043			ERROR	43	
3342	012010	005237	002326		2\$: INC	BITCNT	:BUMP BIT COUNT
3343	012014	005301			DEC	R1	:DEC COUNT TO ZERO IN ECCPOS
3344	012016	001343			BNE	3\$:NOT YET ZERO - LOOP
3345	012020	005300			DEC	R0	:DEC PASS THRU 0 COUNT
3346	012022	001406			BEQ	4\$:IF 0 - SKIP
3347	012024	012737	177777	002350	MOV	#-1,ECPOSX	:PRESET EXPECTED POS TO GO TO ZERO
3348	012032	012701	020000		MOV	#20000,R1	:SET COUNT TO NEXT 0 IN ECC POS
3349	012036	000733			BR	3\$:GO DO LOOP AGAIN
3350							
3351	012040	005737	001214		4\$: TST	\$TMP5	:TEST SWITCH

```

3352 012044 001012          BNE      5$          ;ALREADY SET - SKIP
3353 012046 012701 005310   MOV      #5310,R1    ;SET COUNT TO FINAL POSITION
3354 012052 012700 000001   MOV      #1,R0      ;SET R0 TO RETURN AFTER ONE PASS
3355 012056 012737 177777 002350   MOV      #-1,ECPOSX ;SET EXP POS TO GO TO 0 ON NEXT BIT
3356 012064 005237 001214   INC      $TMP5      ;BUMP SWITCH
3357 012070 000716          BR       3$          ;GO DO LOOP AGAIN
3358
3359 012072 004737 031616   5$:      JSR      PC,RT-IT ;ONE MORE BIT TO SET PEADY
3360 012076 004437 032356   JSR      R4,GETREG  ;GET '611 REGISTERS
3361 012102 013737 002260 002220   MOV      L,CS1,E.CS1 ;GET EXPECTED CS1
3362 012110 052737 100200 002220   BIS      #RDY!CFRR,E.CS1 ;SET READY AND ERROR
3363 012116 042737 000001 002220   BIC      #GO,E.CS1   ;CLEAR GO
3364 012124 023737 002160 002220   CMP      T,CS1,E.CS1 ;CHECK IF CS1 CORRECT
3365 012132 001402          BEQ      6$          ;YES - SKIP
3366 012134 104044          ERROR    44
3367 012136 000410          BR       7$
3368 012140 032737 000100 002174 6$:      BIT      #ECH,T.ER   ;TEST IF ECC HARD ERROR SET
3369 012146 001404          BEQ      7$          ;NO SKIP
3370 012150 012737 100000 002234   MOV      #DCK,E.ER  ;SET EXPECTED ERROR REG
3371 012156 104045          ERROR    45
3372
3373 012160          7$:
3374
3375
3376
3377
3378
3379
3380
3381
3382
3383
3384
3385
3386
3387
3388
3389 012160 000004          TST16: SCOPE
3390 012162 012737 000001 001234   MOV      #1,$TIMES  ;;DO 1 ITERATION
3391
3392 012170 005037 002352          CLR      BSEDES     ;CLEAR BAD SEC DESIRED
3393 012174 012762 100000 000000   MOV      #CLR,RKCS1(R2) ;CLEAR CONTROLLER
3394
3395 012202 005037 002354          CLR      HIBITS     ;CLEAR HI ORDER BITS
3396 012206 004437 032230   JSR      R4,LOADRK  ;LOAD 'L' REGISTERS
3397 012212 000000          0              ;CYLINDER 0
3398 012214 000          .BYTE    0          ;SECTOR 0
3399 012215 000          .BYTE    0          ;TRACK 0
3400 012216 047124          Ibuff      ;BUFFER ADDRESS Ibuff
3401 012220 177400          -400         ;WORD COUNT -400
3402 012222 000021          RDDATA      ;COMMAND RDDATA
3403
3404 012224 004437 031770          JSR      R4,CLRIBF  ;GO CLEAR INPUT BUFFER
3405
3406 012230 004437 032014          JSR      R4,BLDDAT  ;GO BUILD DATA
3407 012234 000005          5              ;PATTERN 5

```

```

*****
*TEST 16      READ DATA WITH 12 BIT BURST ERROR
*
*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
*      OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
*      CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE, A
*      GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
*      A 12 BIT BURST ERROR.  VERIFY THE COUNTING OF THE
*      POSITION REGISTER.  MAKE SURE DATA CHECK, ECC HARD,
*      AND CONTROLLER ERROR SETS.
*****

```

```

3408
3409 012236 012700 000530      MOV      #530,R0          ;SET INDEX INTO BUFFER
3410
3411 012242 012737 126073 051130  MOV      #126073,OBUFF+1000 ;STORE ECC WORDS
3412 012250 012737 151052 051132  MOV      #151052,OBUFF+1002
3413 012256 012737 000001 002344  MOV      #1,ECCSRC        ;SET SOURCE TO INDICATE ECC IN BUFFER
3414 012264 042760 005000 050130  BIC      #BIT9!BIT11,OBUFF(R0) ;CREATE 11 BIT BURST ERROR
3415 012272 052760 050100 050130  BIS      #BIT6!BIT12!BIT14,OBUFF(R0)
3416 012300 062700 000002      ADD      #2,R0           ;BUMP TO NEXT WORD
3417 012304 052760 000001 050130  BIS      #BIT0,OBUFF(R0)
3418 012312 042760 000002 050130  BIC      #BIT1,OBUFF(R0) ;MAKE IT A 12 BIT BURST ERROR
3419
3420 012320 004437 032274      JSR      R4,OPSTRT       ;START THE OPERATION
3421 012324 004437 023334      JSR      R4,DISEEK       ;SIMULATE IMPLIED SEEK
3422 012330 104004      ERROR   4               ;CS1 MISCOMPARE
3423 012332 104005      ERROR   5               ;MR1
3424 012334 104006      ERROR   6               ;MR2
3425 012336 104007      ERROR   7               ;MR3
3426
3427 012340 004437 023316      JSR      R4,DSECTR       ;SIMULATE SECTOR PULSE
3428
3429 012344 004437 023740      JSR      R4,DRSYNC       ;SIMULATE HEADER PREAMBLE
3430 012350 104010      ERROR   10              ;MR1 CONTENTS IN ERROR
3431 012352 104011      ERROR   11              ;CS1 CONTENTS IN ERROR
3432
3433 012354 004437 024276      JSR      R4,DHDCMP       ;SIMULATE HEADER SEARCH
3434 012360 104012      ERROR   12              ;MR1 CONTENTS IN ERROR
3435 012362 104013      ERROR   13              ;CS1 IN ERROR AFTER SEARCH
3436 012364 104014      ERROR   14              ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
3437
3438 012366 004437 026500      JSR      R4,DRGPSN       ;GO READ GAP AND SYNC
3439 012372 104015      ERROR   15              ;MR1 IN ERROR READING GAP
3440 012374 104016      ERROR   16              ;CS1 IN ERROR AFTER READING GAP
3441 012376 104032      ERROR   32              ;MR1 IN ERROR READING SYNC
3442 012400 104033      ERROR   33              ;CS1 IN ERROR AFTER READING SYNC
3443
3444 012402 012737 000001 002346  MOV      #1,ECPATX       ;LOAD EXPECTED PATTERN
3445 012410 012737 004066 002350  MOV      #4066,ECPOSX    ;LOAD EXPECTED POSITION
3446 012416 004437 027176      JSR      R4,DREAD        ;GO SIMULATE DATA READ
3447 012422 000400      400                    ;NUMBER OF WORDS TO SIMULATE
3448 012424 104034      ERROR   34              ;MR1 IN ERROR READING DATA OR ECC
3449 012426 104035      ERROR   35              ;ECC ERROR READING DATA
3450 012430 104036      ERROR   36              ;CS1 ERROR AFTER READING DATA OR ECC
3451 012432 104041      ERROR   41              ;ECC REG INCORRECT AFTER ECC READ
3452 012434 104042      ERROR   42              ;ERR IN ECC PAT CALC.
3453 012436 104043      ERROR   43              ;ERR IN ECC POS COUNT
3454
3455
3456 012440 012737 000020 002326  MOV      #20,BITCNT      ;SET BIT COUNT FOR CORRECTION COUNT
3457 012446 012700 000005      MOV      #5,R0           ;SET COUNT FOR ECCPOS PASS THRU 0
3458 012452 012701 013672      MOV      #13672,R1       ;SET COUNT FOR ECCPOS TO 0 FIRST PASS
3459 012456 005037 001214      CLR      $TMP5           ;CLEAR SWITCH
3460
3461 012462 004737 031616      3$: JSR      PC,RDBIT       ;GO READ BIT
3462 012466 004737 030462      JSR      PC,ECCGEN       ;GENERATE ECC
3463 012472 016237 000032 002214  MOV      RKECPT(R2),T.ECPT ;GET PATTERN

```

```

3464 012500 016237 000030 002212 MOV RKECPS(R2),T.ECPS ;GET POSITION
3465 012506 005237 002350 INC ECPOSX ;BUMP POSITION EXPECTED
3466 012512 013737 002350 002252 MOV ECPOSX,E.ECPS ;SET FOR COMPARE AND REPORT
3467 012520 023737 002254 002214 CMP E.ECPT,T.ECPT ;CHECK IF PATTERN CORRECT
3468 012526 001401 BEQ 1$ ;YES - SKIP
3469 012530 104042 ERROR 42
3470 012532 023737 002252 002212 1$: CMP E.ECPS,T.ECPS ;CHECK IF POSITION CORRECT
3471 012540 001401 BEQ 2$ ;YES - SKIP
3472 012542 104043 ERROR 43
3473 012544 005237 002326 2$: INC BITCNT ;BUMP BIT COUNT
3474 012550 005301 DEC R1 ;DEC COUNT TO ZERO IN ECCPOS
3475 012552 001343 BNE 3$ ;NOT YET ZERO - LOOP
3476 012554 005300 DEC R0 ;DEC PASS THRU 0 COUNT
3477 012556 001406 BEQ 4$ ;IF 0 - SKIP
3478 012560 012737 177777 002350 MOV #-1,ECPOSX ;PRESET EXPECTED POS TO GO TO ZERO
3479 012566 012701 020000 MOV #20000,R1 ;SET COUNT TO NEXT 0 IN ECC POS
3480 012572 000733 BR 3$ ;GO DO LOOP AGAIN
3481
3482 012574 005737 001214 4$: TST $TMP5 ;TEST SWITCH
3483 012600 001012 BNE 5$ ;ALREADY SET - SKIP
3484 012602 012701 010041 MOV #10041,R1 ;SET COUNT TO FINAL POSITION
3485 012606 012700 000001 MOV #1,R0 ;SET R0 TO RETURN AFTER ONE PASS
3486 012612 012737 177777 002350 MOV #-1,ECPOSX ;SET EXP POS TO GO TO 0 ON NEXT BIT
3487 012620 005237 001214 INC $TMP5 ;BUMP SWITCH
3488 012624 000716 BR 3$ ;GO DO LOOP AGAIN
3489
3490 012626 004737 031616 5$: JSR PC,RDBIT ;ONE MORE BIT TO SET READY
3491 012632 004437 032356 JSR R4,GETREG ;GET '611 REGISTERS
3492 012636 013737 002260 002220 MOV L.CS1,E.CS1 ;GET EXPECTED CS1
3493 012644 052737 100200 002220 BIS #RDY!CERR,E.CS1 ;SET READY AND ERROR
3494 012652 042737 000001 002220 BIC #GO,E.CS1 ;CLEAR GO
3495 012660 023737 002160 002220 CMP T.CS1,E.CS1 ;CHECK IF CS1 CORRECT
3496 012666 001402 BEQ 6$ ;YES - SKIP
3497 012670 104044 ERROR 44
3498 012672 000410 BR 7$
3499 012674 032737 000100 002174 6$: BIT #ECM,T.ER ;TEST IF ECC HARD ERROR SET
3500 012702 001004 BNE 7$ ;YES - SKIP
3501 012704 012737 100100 002234 MOV #DCK.ECH,E.ER ;SET EXPECTED ERROR REG
3502 012712 104045 ERROR 45
3503
3504 012714 7$:

```

```

3505
3506
3507 *****
3508 *TEST 17 READ DATA WITH 23 BIT BURST ERROR
3509 *
3510 * CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
3511 * PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
3512 * OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
3513 * CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
3514 * AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE, A
3515 * GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
3516 * A 23 BIT BURST ERROR. VERIFY THE COUNTING OF THE
3517 * POSITION REGISTER. MAKE SURE DATA CHECK, ECC HARD,
3518 * AND CONTROLLER ERROR SETS.
3519 *****

```

```

3520 012714 000004          TST17: SCOPE
3521 012716 012737 000001 001234 MOV      #1,STIMES      ;;DO 1 ITERATION
3522
3523 012724 005037 002352          CLR      BSEDES      ;CLEAR BAD SEC DESIRED
3524 012730 012762 100000 000000 MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
3525
3526 012736 005037 002354          CLR      HIBITS      ;CLEAR HI ORDER BITS
3527 012742 004437 032230          JSR      R4,LOADRK   ;LOAD 'L' REGISTERS
3528 012746 000000          0          ;CYLINDER 0
3529 012750 000          .BYTE 0          ;SECTOR 0
3530 012751 000          .BYTE 0          ;TRACK 0
3531 012752 047124          Ibuff     ;BUFFER ADDRESS Ibuff
3532 012754 177400          -400      ;WORD COUNT -400
3533 012756 000021          RDDATA    ;COMMAND RDDATA
3534
3535 012760 004437 031770          JSR      R4,CLRIBF   ;GO CLEAR INPUT BUFFER
3536
3537 012764 004437 032014          JSR      R4,BLDDAT   ;GO BUILD DATA
3538 012770 000005          5          ;PATTERN 5
3539
3540 012772 012700 000530          MOV      #530,R0     ;SET INDEX INTO BUFFER
3541
3542 012776 012737 126073 051130 MOV      #126073,OBUFF+1000 ;STORE ECC WORDS
3543 013004 012737 151052 051132 MOV      #151052,OBUFF+1002
3544 013012 012737 000001 002344 MOV      #1,ECCSRC    ;SET SOURCE TO INDICATE ECC IN BUFFER
3545 013020 042760 005000 050130 BIR      #BIT9!BIT11,OBUFF(R0) ;CREATE 11 BIT BURST ERROR
3546 013026 052760 050100 050130 BIS      #BIT6!BIT12!BIT14,OBUFF(R0)
3547 013034 062700 000002          ADD      #2,R0       ;BUMP TO NEXT WORD
3548 013040 052760 000001 050130 BIS      #BIT0,OBUFF(R0)
3549 013046 052760 010000 050130 BIS      #BIT12,OBUFF(R0) ;MAKE IT A 23 BIT BURST ERROR
3550
3551 013054 004437 032274          JSR      R4,OPSTRT   ;START THE OPERATION
3552 013060 004437 023334          JSR      R4,DISEEK   ;SIMULATE IMPLIED SEEK
3553 013064 104004          ERROR 4          ;CS1 MISCOMPARE
3554 013066 104005          ERROR 5          ;MR1 ..
3555 013070 104006          ERROR 6          ;MR2 ..
3556 013072 104007          ERROR 7          ;MR3 ..
3557
3558 013074 004437 023316          JSR      R4,DSECTR   ;SIMULATE SECTOR PULSE
3559
3560 013100 004437 023740          JSR      R4,DRSYNC   ;SIMULATE HEADER PREAMBLE
3561 013104 104010          ERROR 10         ;MR1 CONTENTS IN ERROR
3562 013106 104011          ERROR 11         ;CS1 CONTENTS IN ERROR
3563
3564 013110 004437 024276          JSR      R4,DHDCMP   ;SIMULATE HEADER SEARCH
3565 013114 104012          ERROR 12         ;MR1 CONTENTS IN ERROR
3566 013116 104013          ERROR 13         ;CS1 IN ERROR AFTER SEARCH
3567 013120 104014          ERROR 14         ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
3568
3569 013122 004437 026500          JSR      R4,DRGPSN   ;GO READ GAP AND SYNC
3570 013126 104015          ERROR 15         ;MR1 IN ERROR READING GAP
3571 013130 104016          ERROR 16         ;CS1 IN ERROR AFTER READING GAP
3572 013132 104032          ERROR 32         ;MR1 IN ERROR READING SYNC
3573 013134 104033          ERROR 33         ;CS1 IN ERROR AFTER READING SYNC
3574
3575 013136 012737 000001 002346 MOV      #1,ECPATX    ;LOAD EXPECTED PATTERN
  
```

3576	013144	012737	004066	002350	MOV	#4066,ECPOSX	:LOAD EXPECTED POSITION
3577	013152	004437	027176		JSR	R4,DREAD	:GO SIMULATE DATA READ
3578	013156	000400			400		:NUMBER OF WORDS TO SIMULATE
3579	013160	104034			ERROR	34	:MR1 IN ERROR READING DATA OR ECC
3580	013162	104035			ERROR	35	:ECC ERROR READING DATA
3581	013164	104036			ERROR	36	:CS1 ERROR AFTER READING DATA OR ECC
3582	013166	104041			ERROR	41	:ECC REG INCORRECT AFTER ECC READ
3583	013170	104042			ERROR	42	:ERR IN ECC PAT CALC.
3584	013172	104043			ERROR	43	:ERR IN ECC POS COUNT
3585							
3586							
3587	013174	012737	000020	002326	MOV	#20,BITCNT	:SFT BIT COUNT FOR CORRECTION COUNT
3588	013202	012700	000005		MOV	#5,R0	:SET COUNT FOR ECCPOS PASS THRU 0
3589	013206	012701	013672		MOV	#13672,R1	:SET COUNT FOR ECCPOS TO 0 FIRST PASS
3590	013212	005037	001214		CLR	\$TMP5	:CLEAR SWITCH
3591							
3592	013216	004737	031616		3\$: JSR	PC,RDBIT	:GO READ BIT
3593	013222	004737	030462		JSR	PC,ECCGEN	:GENERATE ECC
3594	013226	016237	000032	002214	MOV	RKECPT(R2),T.ECPT	:GET PATTERN
3595	013234	016237	000030	002212	MOV	RKECPS(R2),T.ECPS	:GET POSITION
3596	013242	005237	002350		INC	ECPOSX	:BUMP POSITION EXPECTED
3597	013246	013737	002350	002252	MOV	ECPOSX,E.ECPS	:SET FOR COMPARE AND REPORT
3598	013254	023737	002254	002214	CMP	E.ECPT,T.ECPT	:CHECK IF PATTERN CORRECT
3599	013262	001401			BEQ	1\$:YES - SKIP
3600	013264	104042			ERROR	42	
3601	013266	023737	002252	002212	1\$: CMP	E.ECPS,T.ECPS	:CHECK IF POSITION CORRECT
3602	013274	001401			BEQ	2\$:YES - SKIP
3603	013276	104043			ERROR	43	
3604	013300	005237	002326		2\$: INC	BITCNT	:BUMP BIT COUNT
3605	013304	005301			DEC	R1	:DEC COUNT TO ZERO IN ECCPOS
3606	013306	001343			BNE	3\$:NOT YET ZERO - LOOP
3607	013310	005300			DEC	R0	:DEC PASS THRU 0 COUNT
3608	013312	001406			BEQ	4\$:IF 0 - SKIP
3609	013314	012737	177777	002350	MOV	#-1,ECPOSX	:PRESET EXPECTED POS TO GO TO ZERO
3610	013322	012701	020000		MOV	#20000,R1	:SET COUNT TO NEXT 0 IN ECC POS
3611	013326	000733			BR	3\$:GO DO LOOP AGAIN
3612							
3613	013330	005737	001214		4\$: TST	\$TMP5	:TEST SWITCH
3614	013334	001012			BNE	5\$:ALREADY SET - SKIP
3615	013336	012701	010041		MOV	#10041,R1	:SET COUNT TO FINAL POSITION
3616	013342	012700	000001		MOV	#1,R0	:SET R0 TO RETURN AFTER ONE PASS
3617	013346	012737	177777	002350	MOV	#-1,ECPOSX	:SET EXP POS TO GO TO 0 ON NEXT BIT
3618	013354	005237	001214		INC	\$TMP5	:BUMP SWITCH
3619	013360	000716			BR	3\$:GO DO LOOP AGAIN
3620							
3621	013362	004737	031616		5\$: JSR	PC,RDBIT	:ONE MORE BIT TO SET READY
3622	013366	004437	032356		JSR	R4,GETREG	:GET '611 REGISTERS
3623	013372	013737	002260	002220	MOV	L.CS1,E.CS1	:GET EXPECTED CS1
3624	013400	052737	100200	002220	BIS	#RDY!CERR,E.CS1	:SET READY AND ERROR
3625	013406	042737	000001	002220	BIC	#GO,E.CS1	:CLEAR GO
3626	013414	023737	002160	002220	CMP	T.CS1,E.CS1	:CHECK IF CS1 CORRECT
3627	013422	001402			BEQ	6\$:YES - SKIP
3628	013424	104044			ERROR	44	
3629	013426	000410			BR	7\$	
3630	013430	032737	000100	002174	6\$: BIT	#ECH,T.ER	:TEST IF ECC HARD ERROR SET
3631	013436	001004			BNE	7\$:YES - SKIP


```

3632 013440 012737 100100 002234      MOV    #DCK!ECH,E.ER  ;SET EXPECTED ERROR REG
3633 013446 104045      ERROR  45
3634
3635 013450      7$:
3636
3637      :*****
3638      :*TEST 20      READ DATA WITH 1 BIT ECC WORD ERROR
3639      :*
3640      :*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
3641      :*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
3642      :*      OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
3643      :*      CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
3644      :*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE, A
3645      :*      GOOD HEADER, 400 DATA WORDS AND AN ECC INDICATING
3646      :*      A 1 BIT ERROR IN ECC WORD 2, BIT 6.  VERIFY THE
3647      :*      COUNTING OF THE POSITION REGISTER.  MAKE SURE DATA
3648      :*      CHECK AND CONTROLLER ERROR SETS.
3649      :*
3650      :*****
3651 013450 000004      TST20: SCOPE
3652 013452 012737 000001 001234      MOV    #1,$TIMES      ;;DO 1 ITERATION
3653
3654 013460 005037 002352      CLR    BSEDES          ;CLEAR BAD SEC DESIRED
3655 013464 012762 100000 000000      MOV    #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
3656
3657 013472 005037 002354      CLR    HIBITS          ;CLEAR HI ORDER BITS
3658 013476 004437 032230      JSR    R4,LOADRK       ;LOAD 'L' REGISTERS
3659 013502 000000      0                      ;CYLINDER 0
3660 013504      000                  ;SECTOR 0
3661 013505      000                  ;TRACK 0
3662 013506 047124      Ibuff                ;BUFFER ADDRESS Ibuff
3663 013510 177400      -400                 ;WORD COUNT -400
3664 013512 000021      RDDATA              ;COMMAND RDDATA
3665
3666 013514 004437 031770      JSR    R4,CLRIBF       ;GO CLEAR INPUT BUFFER
3667
3668 013520 004437 032014      JSR    R4,BLDDAT       ;GO BUILD DATA
3669 013524 000005      5                     ;PATTERN 5
3670
3671
3672 013526 012700 001002      MOV    #1002,R0        ;SET INDEX INTO BUFFER
3673 013532 012737 126073 051130      MOV    #126073,OBUFF+1000 ;STORE ECC WORDS
3674 013540 012737 151052 051132      MOV    #151052,OBUFF+1002
3675 013546 012737 000001 002344      MOV    #1,ECCSRC       ;SET SOURCE TO INDICATE ECC IN BUFFER
3676 013554 032760 000100 050130      BIT    #BIT6,OBUFF(R0) ;TEST IF BIT SET
3677 013562 001004      BNE    100$           ;YES - SKIP
3678 013564 052760 000100 050130      BIS    #BIT6,OBUFF(R0) ;ELSE SET BIT
3679 013572 000403      BR     101$
3680 013574 042760 000100 050130 100$: BIC    #BIT6,OBUFF(R0) ;CLEAR BIT
3681 013602      101$:
3682
3683 013602 004437 032274      JSR    R4,OPSTRT       ;START THE OPERATION
3684 013606 004437 023334      JSR    R4,DISEEK       ;SIMULATE IMPLIED SEEK
3685 013612 104004      ERROR  4              ;CS1 MISCOMPARE
3686 013614 104005      ERROR  5              ;MR1
3687 013616 104006      ERROR  6              ;MR2
  
```

```
3688 013620 104007          ERROR 7          :MR3 ''
3689
3690 013622 004437 023316    JSR    R4,DSECTR  :SIMULATE SECTOR PULSE
3691
3692 013626 004437 023740    JSR    R4,DRSYNC  :SIMULATE HEADER PREAMBLE
3693 013632 104010          ERROR 10         :MR1 CONTENTS IN ERROR
3694 013634 104011          ERROR 11         :CS1 CONTENTS IN ERROR
3695
3696 013636 004437 024276    JSR    R4,DHDCMP  :SIMULATE HEADER SEARCH
3697 013642 104012          ERROR 12         :MR1 CONTENTS IN ERROR
3698 013644 104013          ERROR 13         :CS1 IN ERROR AFTER SEARCH
3699 013646 104014          ERROR 14         :RKDCYL OR RKDA IN ERROR AFTER SEARCH
3700
3701 013650 004437 026500    JSR    R4,DRGPSN  :GO READ GAP AND SYNC
3702 013654 104015          ERROR 15         :MR1 IN ERROR READING GAP
3703 013656 104016          ERROR 16         :CS1 IN ERROR AFTER READING GAP
3704 013660 104032          ERROR 32         :MR1 IN ERROR READING SYNC
3705 013662 104033          ERROR 33         :CS1 IN ERROR AFTER READING SYNC
3706
3707 013664 012737 000001 002346    MOV    #1,ECPATX  :LOAD EXPECTED PATTERN
3708 013672 012737 004066 002350    MOV    #4066,ECPOSX :LOAD EXPECTED POSITION
3709 013700 004437 027176    JSR    R4,DREAD   :GO SIMULATE DATA READ
3710 013704 000400          400             :NUMBER OF WORDS TO SIMULATE
3711 013706 104034          ERROR 34         :MR1 IN ERROR READING DATA OR ECC
3712 013710 104035          ERROR 35         :ECC ERROR READING DATA
3713 013712 104036          ERROR 36         :CS1 ERROR AFTER READING DATA OR ECC
3714 013714 104041          ERROR 41         :ECC REG INCORRECT AFTER ECC READ
3715 013716 104042          ERROR 42         :ERR IN ECC PAT CALC.
3716 013720 104043          ERROR 43         :ERR IN ECC POS COUNT
3717
3718
3719 013722 012737 000020 002326    MOV    #20,BITCNT  :SET BIT COUNT FOR CORRECTION COUNT
3720 013730 012700 000005          MOV    #5,RC       :SET COUNT FOR ECCPOS PASS THRU 0
3721 013734 012701 013672          MOV    #13672,R1   :SET COUNT FOR ECCPOS TO 0 FIRST PASS
3722 013740 005037 001214          CLR    $TMP5       :CLEAR SWITCH
3723
3724 013744 004737 031616          3$: JSR    PC,RDBIT   :GO READ BIT
3725 013750 004737 030462          JSR    PC,ECCGEN   :GENERATE ECC
3726 013754 016237 000032 002214    MOV    RKECPT(R2),T.ECPT :GET PATTERN
3727 013762 016237 000030 002212    MOV    RKECPS(R2),T.ECPS :GET POSITION
3728 013770 005237 002350          INC    ECPOSX      :BUMP POSITION EXPECTED
3729 013774 013737 002350 002252    MOV    ECPOSX,E.ECPS :SET FOR COMPARE AND REPORT
3730 014002 023737 002254 002214    CMP    E.ECPT,T.ECPT :CHECK IF PATTERN CORRECT
3731 014010 001401          BEQ    1$         :YES - SKIP
3732 014012 104042          ERROR 42
3733 014014 023737 002252 002212 1$: CMP    E.ECPS,T.ECPS :CHECK IF POSITION CORRECT
3734 014022 001401          BEQ    2$         :YES - SKIP
3735 014024 104043          ERROR 43
3736 014026 005237 002326          2$: INC    BITCNT     :BUMP BIT COUNT
3737 014032 005301          DEC    R1         :DEC COUNT TO ZERO IN ECCPOS
3738 014034 001343          BNE    3$        :NOT YET ZERO - LOOP
3739 014036 005300          DEC    R0         :DEC PASS THRU 0 COUNT
3740 014040 001406          BEQ    4$        :IF 0 - SKIP
3741 014042 012737 177777 002350    MOV    #-1,ELPOSX  :PRESET EXPECTED POS TO GO TO ZERO
3742 014050 012701 020000          MOV    #20000,R1  :SET COUNT TO NEXT 0 IN ECC POS
3743 014054 000733          BR     3$        :GO DO LOOP AGAIN
```

```

3744
3745 014056 005737 001214      4$:  TST      $TMP5      ;TEST SWITCH
3746 014062 001012              BNE      5$          ;ALREADY SET - SKIP
3747 014064 012701 010016      MOV      #10016,R1   ;SET COUNT TO FINAL POSITION
3748 014070 012700 000001      MOV      #1,R0       ;SET R0 TO RETURN AFTER ONE PASS
3749 014074 012737 177777      MOV      #-1,ECSX    ;SET EXP POS TO GO TO 0 ON NEXT BIT
3750 014102 005237 001214      INC      $TMP5      ;BUMP SWITCH
3751 014106 000716              BR       3$          ;GO DO LOOP AGAIN
3752
3753 014110 004737 031616      5$:  JSR      PC,RDBIT   ;ONE MORE BIT TO SET READY
3754 014114 004437 032356      JSR      R4,GETREG   ;GET '611 REGISTERS
3755 014120 013737 002260 002220  MOV      L,CS1,E.CS1 ;GET EXPECTED CS1
3756 014126 052737 100200 002220  BIS      #RDY!CERR,E.CS1 ;SET READY AND ERROR
3757 014134 042737 000001 002220  BIC      #GO,E.CS1   ;CLEAR GO
3758 014142 023737 002160 002220  CMP      T,CS1,E.CS1 ;CHECK IF CS1 CORRECT
3759 014150 001402              BEQ      6$          ;YES - SKIP
3760 014152 104044              ERROR    44
3761 014154 000410              BR       7$
3762 014156 032737 000100 002174  6$:  BII      #ECH,T.ER   ;TEST IF ECC HARD ERROR SET
3763 014164 001404              BEQ      7$          ;YES - SKIP
3764 014166 012737 100000 002234  MOV      #DCK,E.ER   ;SET EXPECTED ERROR REG
3765 014174 104045              ERROR    45
3766
3767 014176      7$:
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777
3778
3779
3780
3781 014176 000004      TST21: SCOPE
3782 014200 012737 000012 001234  MOV      #10.,$TIMES ;:DO 10. ITERATIONS
3783
3784 014206 005037 002352      CLR      BSEDES     ;CLEAR BAD SEC DESIRED
3785 014212 012762 100000 000000  MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
3786
3787 014220 005037 002344      CLR      ECCSRC     ;CLEAR ECC SOURCE
3788 014224 005037 002354      CLR      HIBITS     ;CLEAR HI ORDER BITS
3789 014230 004437 032230      JSR      R4,LOADRK  ;LOAD 'L' REGISTERS
3790 014234 000000      0              ;CYLINDER 0
3791 014236      000          .BYTE 0          ;SECTOR 0
3792 014237      000          .BYTE 0          ;TRACK 0
3793 014240 047124      Ibuff        ;BUFFER ADDRESS Ibuff
3794 014242 177400      -400         ;WORD COUNT -400
3795 014244 010021      RDDATA!CFMT    ;COMMAND RDDATA!CFMT
3796
3797 014246 004437 031770      JSR      R4,CLRIBF  ;GO CLEAR INPUT BUFFER
3798
3799 014252 004437 032014      JSR      R4,BLDDAT  ;CO BUILD DATA
  
```

```

*****
*TEST 21      18 BIT NPR WRITING OF MEMORY
*
*      CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
*      PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A READ DATA
*      OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
*      CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
*      AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE,
*      A GOOD HEADER, 40 DATA WORDS.  VERIFY THAT THE WORDS
*      ARE WRITTEN PROPERLY IN MEMORY.
*****
  
```

```
3800 014256 000003          3          ;PATTERN 3
3801
3802
3803 014260 004437 032274    JSR      R4,OPSTRT      ;START THE OPERATION
3804 014264 004437 023334    JSR      R4,DISEEK     ;SIMULATE IMPLIED SEEK
3805 014270 104004          ERROR    4             ;CS1 MISCOMPARE
3806 014272 104005          ERROR    5             ;MR1
3807 014274 104006          ERROR    6             ;MR2
3808 014276 104007          ERROR    7             ;MR3
3809
3810 014300 004437 023316    JSR      R4,DSECTR     ;SIMULATE SECTOR PULSE
3811
3812 014304 004437 023740    JSR      R4,DRSYNC     ;SIMULATE HEADER PREAMBLE
3813 014310 104010          ERROR    10            ;MR1 CONTENTS IN ERROR
3814 014312 104011          ERROR    11            ;CS1 CONTENTS IN ERROR
3815
3816 014314 004437 024276    JSR      R4,DHDCMP     ;SIMULATE HEADER SEARCH
3817 014320 104012          ERROR    12            ;MR1 CONTENTS IN ERROR
3818 014322 104013          ERROR    13            ;CS1 IN ERROR AFTER SEARCH
3819 014324 104014          ERROR    14            ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
3820
3821 014326 004437 026500    JSR      R4,DRGPSN     ;GO READ GAP AND SYNC
3822 014332 104015          ERROR    15            ;MR1 IN ERROR READING GAP
3823 014334 104016          ERROR    16            ;CS1 IN ERROR AFTER READING GAP
3824 014336 104032          ERROR    32            ;MR1 IN ERROR READING SYNC
3825 014340 104033          ERROR    33            ;CS1 IN ERROR AFTER READING SYNC
3826
3827 014342 012737 000000 002346  MOV      #0,ECPATX     ;LOAD EXPECTED PATTERN
3828 014350 012737 005066 002350  MOV      #5066,ECPOSX  ;LOAD EXPECTED POSITION
3829 014356 004437 027176    JSR      R4,DREAD      ;GO SIMULATE DATA READ
3830 014362 000060          60             ;NUMBER OF WORDS TO SIMULATE
3831 014364 104034          ERROR    34            ;MR1 IN ERROR READING DATA OR ECC
3832 014366 104035          ERROR    35            ;ECC ERROR READING DATA
3833 014370 104036          ERROR    36            ;CS1 ERROR AFTER READING DATA OR ECC
3834 014372 104041          ERROR    41            ;ECC REG INCORRECT AFTER ECC READ
3835 014374 104042          ERROR    42            ;ERR IN ECC PAT CALC.
3836 014376 104043          ERROR    43            ;ERR IN ECC POS COUNT
3837
3838
3839 014400 012700 050130    MOV      #0BUFF,R0    ;SET POINTER TO GOOD DATA
3840 014404 012701 047124    MOV      #1BUFF,R1    ;SET POINTER TO INPUT DATA
3841
3842 014410 012704 000012    MOV      #10,R4       ;SET ERROR LIMIT COUNT
3843 014414 005037 001236    CLR      $ESCAPE      ;CLEAR ESCAPE
3844 014420 005037 001220    CLR      $TMP7        ;CLEAR ERROR REPORT SWITCH
3845 014424 012703 000040    MOV      #40,R3       ;SET COMPARE LIMIT
3846 014430 005005          CLR      R5           ;CLEAR WORD COUNTER
3847
3848 014432 021011          1$: CMP      (R0),(R1)  ;COMPARE DATA
3849 014434 001005          BNE     2$           ;SKIP IF NOT EQUAL
3850 014436 005303          4$: DEC     R3       ;ELSE DEC WORD COUNT LIMIT
3851 014440 001424          BEQ     5$           ;IF 0 - SKIP TO EXIT
3852 014442 005205          INC     R5           ;BUMP WORD COUNT
3853 014444 022021          CMP     (R0)+,(R1)+  ;BUMP DATA POINTERS
3854 014446 000771          BR      1$           ;LOOP
3855
```

```
3856 014450 005304      2$: DEC R4 ;DEC ERROR LIMIT
3857 014452 001417      BEQ 5$ ;IF 0 - SKIP
3858
3859 014454 011037 001162      MOV (R0), $REG0 ;GET GOOD WORD FOR REPORT
3860 014460 011137 001164      MOV (R1), $REG1 ;GET BAD WORD
3861 014464 010537 001166      MOV R5, $REG2 ;GET WORD NUMBER
3862
3863 014470 005737 001220      TST $TMP7 ;DECIDE WHICH ERROR REPORT TO USE
3864 014474 001004 ;SKIP IF NOT 0
3865 014476 005237 001220      INC $TMP7 ;ELSE BUMP $TMP7
3866 014502 104037 ;REPORT FIRST ERROR LINE
3867 014504 000754      BR 4$ ;SKIP
3868
3869 014506 104040      3$: ERROR 40 ;REPORT ALL OTHER LINES
3870 014510 000752      BR 4$ ;SKIP
3871
3872 014512      5$:
3873
3874 ;*****
3875 ;*TEST 22 18 BIT READ DATA WITH NO ERROR
3876 ;*
3877 ;* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
3878 ;* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
3879 ;* OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
3880 ;* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
3881 ;* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
3882 ;* A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
3883 ;* VERIFY THAT COMMAND COMPLETES WITHOUT ERROR.
3884 ;*
3885 ;*****
3886 014512 000004      TST22: SCOPE
3887 014514 012737 000001 001234      MOV #1, $TIMES ;:DO 1 ITERATION
3888
3889 014522 005037 002352      CLR BSEDES ;CLEAR BAD SEC DESIRED
3890 014526 012762 100000 000000      MOV #CCLR, RKCS1(R2) ;CLEAR CONTROLLER
3891 014534 005037 002344      CLR ECCSRC ;CLEAR ECC SOURCE FLAG
3892 014540 005037 002344      CLR ECCSRC ;CLEAR ECC SOURCE
3893 014544 005037 002354      CLR HIBITS ;CLEAR HI ORDER BITS
3894 014550 004437 032230      JSR R4, LOADRK ;LOAD 'L' REGISTERS
3895 014554 000000      0 ;CYLINDER 0
3896 014556 000 ;SECTOR 0
3897 014557 000 ;TRACK 0
3898 014560 047124      Ibuff ;BUFFER ADDRESS Ibuff
3899 014562 177400      -400 ;WORD COUNT -400
3900 014564 010021      RDDATA!CFMT ;COMMAND RDDATA!CFMT
3901
3902 014566 004437 031770      JSR R4, CLRIBF ;GO CLEAR INPUT BUFFER
3903
3904 014572 004437 032014      JSR R4, BLDDAT ;GO BUILD DATA
3905 014576 000004      4 ;PATTERN 4
3906
3907
3908 014600 004437 032274      JSR R4, OPSTRT ;START THE OPERATION
3909 014604 004437 023334      JSR R4, DISEEK ;SIMULATE IMPLIED SEEK
3910 014610 104004      ERROR 4 ;CS1 MISCOMPARE
3911 014612 104005      ERROR 5 ;MR1
```

```
3912 014614 104006 ERROR 6 :MR2 ''
3913 014616 104007 ERROR 7 :MR3 ''
3914
3915 014620 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
3916
3917 014624 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
3918 014630 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
3919 014632 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
3920
3921 014634 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
3922 014640 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
3923 014642 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
3924 014644 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
3925
3926 014646 004437 026500 JSR R4,DRGPSN ;GO READ GAP AND SYNC
3927 014652 104015 ERROR 15 ;MR1 IN ERROR READING GAP
3928 014654 104016 ERROR 16 ;CS1 IN ERROR AFTER READING GAP
3929 014656 104032 ERROR 32 ;MR1 IN ERROR READING SYNC
3930 014660 104033 ERROR 33 ;CS1 IN ERROR AFTER READING SYNC
3931
3932 014662 012737 000000 002346 MOV #0,ECPATX ;LOAD EXPECTED PATTERN
3933 014670 012737 005066 002350 MOV #5066,FCPOSX ;LOAD EXPECTED POSITION
3934 014676 004437 027176 JSR R4,DREAD ;GO SIMULATE DATA READ
3935 014702 000400 400 ;NUMBER OF WORDS TO SIMULATE
3936 014704 104034 ERROR 34 ;MR1 IN ERROR READING DATA OR ECC
3937 014706 104035 ERROR 35 ;ECC ERROR READING DATA
3938 014710 104036 ERROR 36 ;CS1 ERROR AFTER READING DATA OR ECC
3939 014712 104041 ERROR 41 ;ECC REG INCORRECT AFTER ECC READ
3940 014714 104042 ERROR 42 ;ERR IN ECC PAT CALC.
3941 014716 104043 ERROR 43 ;ERR IN ECC POS COUNT
3942
3943
3944 014720 012700 003100 MOV #40.*40.,R0 ;SET COUNT TO EMPTY SILO
3945
3946 014724 012762 000440 000026 1$: MOV #DMD!MCLK,RKMR1(R2) ;CLOCK CONTROLLER
3947 014732 012762 000040 000026 MOV #DMD,RKMR1(R2)
3948 014740 005300 DEC R0 ;DEC COUNT
3949 014742 001370 BNE 1$ ;LOOP UNTIL 0
3950
3951 014744 004437 032356 JSR R4,GETREG ;GET 611 REGS
3952 014750 013737 002260 002220 MOV L.CS1,E.CS1 ;GET EXPECTED CS1
3953 014756 042737 000001 002220 BIC #GO,E.CS1 ;CLEAR GO BIT
3954 014764 052737 000200 002220 BIS #RDY,E.CS1 ;SET READY BIT
3955 014772 023737 002160 002220 CMP T.CS1,E.CS1 ;CHECK IF CS1 CORRECT
3956 015000 001401 BEQ 2$ ;YES - SKIP
3957 015002 104056 ERROR 56 ;'CS1 IN ERROR AFTER READ DATA'
3958 015004
3959
3960 :*****
3961 :*TEST 23 18 BIT READ DATA WITH DCK
3962 :*
3963 :* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
3964 :* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
3965 :* OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
3966 :* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
3967 :* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
3968 :* A GOOD HEADER, 400 DATA WORDS, AND AN ECC INDICATING
```

```
3968      ;*      A ONE BIT BURST ERROR BEYOND THE 4128TH BIT OF THE SECTOR.  
3969      ;*      VERIFY THE COUNTING OF THE POSITION REGISTER AND THE  
3970      ;*      FINAL PATTERN AND POSITION.  MAKE SURE DATA CHECK AND  
3971      ;*      CONTROLLER ERROR SETS.  
3972      ;*  
3973      ;*****  
3974 015004 000004 TST23: SCOPE  
3975 015006 012737 000001 001234 MOV #1,$TIMES ;DO 1 ITERATION  
3976  
3977 015014 005037 002352 CLR BSEDES ;CLEAR BAD SEC DESIRED  
3978 015020 012762 100000 000000 MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER  
3979  
3980 015026 005037 002354 CLR HIBITS ;CLEAR HI ORDER BITS  
3981 015032 004437 032230 JSR R4,LOADRK ;LOAD 'L' REGISTERS  
3982 015036 000000 0 ;CYLINDER 0  
3983 015040 000 .BYTE 0 ;SECTOR 0  
3984 015041 000 .BYTE 0 ;TRACK 0  
3985 015042 047124 IBUFF ;BUFFER ADDRESS IBUFF  
3986 015044 177400 -400 ;WORD COUNT -400  
3987 015046 010021 RDDATA!CFMT ;COMMAND RDDATA.CFMT  
3988  
3989 015050 004437 031770 JSR R4,CLRIBF ;GO CLEAR INPUT BUFFER  
3990  
3991 015054 004437 032014 JSR R4,BLDDAT ;GO BUILD DATA  
3992 015060 000007 7 ;PATTERN 7  
3993  
3994  
3995 015062 012700 000734 MOV #734,R0 ;SET INDEX INTO BUFFER  
3996 015066 012737 006030 051130 MOV #6030,OBUFF+1000 ;STORE ECC WORDS  
3997 015074 012737 045070 051132 MOV #45070,OBUFF+1002  
3998 015102 012737 090001 002344 MOV #1,ECCSRC ;SET SOURCE TO INDICATE ECC IN BUFFER  
3999 015110 032760 000100 050130 BIT #BIT6,OBUFF(R0) ;TEST IF BIT SET  
4000 015116 001004 BNE 100$ ;YES - SKIP  
4001 015120 052760 000100 050130 BIS #BIT6,OBUFF(R0) ;ELSE SET BIT  
4002 015126 000403 BR 101$  
4003 015130 042760 000100 050130 100$: BIC #BIT6,OBUFF(R0) ;CLEAR BIT  
4004 015136 101$:  
4005  
4006 015136 004437 032274 JSR R4,OPSTRT ;START THE OPERATION  
4007 015142 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK  
4008 015146 104004 ERROR 4 ;CS1 MISCOMPARE  
4009 015150 104005 ERROR 5 ;MR1 ..  
4010 015152 104006 ERROR 6 ;MR2 ..  
4011 015154 104007 ERROR 7 ;MR3 ..  
4012  
4013 015156 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE  
4014  
4015 015162 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE  
4016 015166 104010 ERROR 10 ;MR1 CONTENTS IN ERROR  
4017 015170 104011 ERROR 11 ;CS1 CONTENTS IN ERROR  
4018  
4019 015172 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH  
4020 015176 104012 ERROR 12 ;MR1 CONTENTS IN ERROR  
4021 015200 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH  
4022 015202 104014 ERROR 14 ;RKDCVL OR RKDA IN ERROR AFTER SEARCH  
4023
```

```

4024 015204 004437 026500      JSR      R4,DRGPSN      ;GO READ GAP AND SYNC
4025 015210 104015      ERROR   15              ;MR1 IN ERPOR READING GAP
4026 015212 104016      ERROR   16              ;CS1 IN ERROR AFTER READING GAP
4027 015214 104032      ERROR   32              ;MR1 IN ERROR READING SYNC
4028 015216 104033      ERROR   33              ;CS1 IN ERROR AFTER READING SYNC
4029
4030 015220 012737 000001 002346      MOV      #1,ECPATX      ;LOAD EXPECTED PATTERN
4031 015226 012737 005066 002350      MOV      #5066,ECPOSX   ;LOAD EXPECTED POSITION
4032 015234 004437 027176      JSR      R4,DRDREAD     ;GO SIMULATE DATA READ
4033 015240 000400      400                    ;NUMBER OF WORDS TO SIMULATE
4034 015242 104034      ERROR   34              ;MR1 IN ERROR READING DATA OR ECC
4035 015244 104035      ERROR   35              ;ECC ERROR READING DATA
4036 015246 104036      ERROR   36              ;CS1 ERROR AFTER READING DATA OR ECC
4037 015250 104041      ERROR   41              ;ECC REG INCORRECT AFTER ECC READ
4038 015252 104042      ERROR   42              ;ERR IN ECC PAT CALC.
4039 015254 104043      ERROR   43              ;ERR IN ECC POS COUNT
4040
4041
4042 015256 012737 000022 002326      MOV      #22,BITCNT     ;SET BIT COUNT FOR CORRECTION COUNT
4043 015264 012700 000005      MOV      #5,R0          ;SET COUNT FOR ECCPOS PASS THRU 0
4044 015270 012701 012672      MOV      #12672,R1     ;SET COUNT FOR ECCPOS TO 0 FIRST PASS
4045 015274 005037 001214      CLR      $TMP5          ;CLEAR SWITCH
4046
4047 015300 004737 031616      3$:      JSR      PC,RDBIT       ;GO READ BIT
4048 015304 004737 030462      JSR      PC,ECCGEN      ;GENERATE ECC
4049 015310 016237 000032 002214      MOV      RKECPT(R2),T.ECPT ;GET PATTERN
4050 015316 016237 000030 002212      MOV      RKECPS(R2),T.ECPS ;GET POSITION
4051 015324 005237 002350      INC      ECPOSX         ;BUMP POSITION EXPECTED
4052 015330 013737 002350 002252      MOV      ECPOSX,E.ECPS  ;SET FOR COMPARE AND REPORT
4053 015336 023737 002254 002214      CMP      E.ECPT,T.ECPT  ;CHECK IF PATTERN CORRECT
4054 015344 001401      BEQ     1$              ;YES - SKIP
4055 015346 104042      ERROR   42
4056 015350 023737 002252 002214 1$:      CMP      E.ECPS,T.ECPS  ;CHECK IF POSITION CORRECT
4057 015356 001401      BEQ     2$              ;YES - SKIP
4058 015360 104043      ERROR   43
4059 015362 005237 002326      2$:      INC      BITCNT         ;BUMP BIT COUNT
4060 015366 005301      DEC     R1              ;DEC COUNT TO ZERO IN ECCPOS
4061 015370 001343      BNE     3$              ;NOT YET ZERO - LOOP
4062 015372 005300      DEC     R0              ;DEC PASS THRU 0 COUNT
4063 015374 001406      BEQ     4$              ;IF 0 - SKIP
4064 015376 012737 177777 002350      MOV      #-1,ECPOSX     ;PRESET EXPECTED POS TO GO TO ZERO
4065 015404 012701 020000      MOV      #20000,R1     ;SET COUNT TO NEXT 0 IN ECC POS
4066 015410 000733      BR      3$              ;GO DO LOOP AGAIN
4067
4068 015412 005737 001214      4$:      TST     $TMP5          ;TEST SWITCH
4069 015416 001012      BNE     5$              ;ALREADY SET - SKIP
4070 015420 012701 010272      MOV      #10272,R1     ;SET COUNT TO FINAL POSITION
4071 015424 012700 000001      MOV      #1,R0         ;SET R0 TO RETURN AFTER ONE PASS
4072 015430 012737 177777 002350      MOV      #-1,ECPOSX     ;SET EXP POS TO GO TO 0 ON NEXT BIT
4073 015436 005237 001214      INC     $TMP5          ;BUMP SWITCH
4074 015442 000716      BR      3$              ;GO DO LOOP AGAIN
4075
4076 015444 004737 031616      5$:      JSR      PC,RDBIT       ;ONE MORE BIT TO SET READY
4077 015450 004437 032356      JSR      R4,GETREG      ;GET '611 REGISTERS
4078 015454 013737 002260 002220      MOV      L.CS1,E.CS1   ;GET EXPECTED CS1
4079 015462 052737 100200 002220      BIS     #RDY!CERR,E.CS1 ;SET READY AND ERROR

```


4080	015470	042737	000001	002220		BIC	#GO,E.CS1	:CLEAR GO
4081	015476	023737	002160	002220		CMP	T.CS1,E.CS1	:CHECK IF CS1 CORRECT
4082	015504	001402				BEQ	6\$:YES - SKIP
4083	015506	104044				ERROR	44	
4084	015510	000410				BR	7\$	
4085	015512	032737	000100	002174	6\$:	BIT	#ECH,T.ER	:TEST IF ECC HARD ERROR SET
4086	015520	001404				BEQ	7\$:NO SKIP
4087	015522	012737	100000	002234		MOV	#DCK,E.ER	:SET EXPECTED FRROR REG
4088	015530	104045				ERROR	45	
4089								
4090	015532							7\$:
4091								
4092								
4093								
4094								
4095								
4096								
4097								
4098								
4099								
4100								
4101								
4102								
4103								
4104								
4105								
4106	015532	000004						
4107	015534	012737	000001	001234				
4108								
4109	015542	005037	002352					
4110	015546	012762	100000	000000				
4111								
4112	015554	005037	002354					
4113	015560	004437	032230					
4114	015564	000000						
4115	015566	000						
4116	015567	000						
4117	015570	047124						
4118	015572	177400						
4119	015574	010021						
4120								
4121	015576	004437	031770					
4122								
4123	015602	004437	032014					
4124	015606	000005						
4125								
4126								
4127	015610	012700	000734					
4128	015614	012737	004020	051130				
4129	015622	012737	071720	051132				
4130	015630	012737	000001	002344				
4131	015636	042760	005000	050130				
4132	015644	052760	050100	050130				
4133	015652	062700	000002					
4134	015656	052760	000001	050130				
4135	015664	042760	000002	050130				

 :TEST 24 18 BIT READ DATA WITH ECH
 :*
 :* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
 :* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
 :* OF 400 WORDS TO AN RK06 IN 24 SECTOR FORMAT,
 :* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
 :* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
 :* A GOOD HEADER, 400 DATA WORDS AND AND ECC INDICATING
 :* A 12 BIT BURST ERROR. VERIFY THE COUNTING OF THE
 :* POSITION REGISTER. MAKE SURE DATE CHECK, ECC HARD,
 :* AND CONTROLLER ERROR SETS.
 :*
 :*****
 TST24: SCOPE
 MOV #1,\$TIMES ;;DO 1 ITERATION
 CLR BSEDES ;CLEAR BAD SEC DESIRED
 MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
 CLR HIBITS ;CLEAR HI ORDER BITS
 JSR R4,LOADRK ;LOAD 'L' REGISTERS
 0 ;CYLINDER 0
 .BYTE 0 ;SECTOR 0
 .BYTE 0 ;TRACK 0
 Ibuff ;BUFFER ADDRESS Ibuff
 -400 ;WORD COUNT -400
 RDDATA!CFMT ;COMMAND RDDATA!CFMT
 JSR R4,CLRIBF ;GO CLEAR INPUT BUFFER
 JSR R4,BLDDAT ;GO BUILD DATA
 5 ;PATTERN 5
 MOV #734,R0 ;SET INDEX INTO BUFFER
 MOV #4020,OBUFF+1000 ;STORE ECC WORDS
 MOV #71720,OBUFF+1002
 MOV #1,ECCSRC ;SET SOURCE TO INDICATE ECC IN BUFFER
 BIC #BIT9!BIT11,OBUFF(R0) ;CREATE 11 BIT BURST ERROR
 BIS #BIT6!BIT12!BIT14,OBUFF(R0)
 ADD #2,R0 ;BUMP TO NEXT WORD
 BIS #BIT0,OBUFF(R0)
 BIC #BIT1,OBUFF(R0) ;MAKE IT A 12 BIT BURST ERROR

4136										
4137	015672	004437	032274			JSR	R4,OPSTRT		:START THE OPERATION	
4138	015676	004437	023334			JSR	R4,DISEEK		:SIMULATE IMPLIED SEEK	
4139	015702	104004				ERROR	4		:CS1 MISCOMPARE	
4140	015704	104005				ERROR	5		:MR1 ..	
4141	015706	104006				ERROR	6		:MR2 ..	
4142	015710	104007				ERROR	7		:MR3 ..	
4143										
4144	015712	004437	023316			JSR	R4,DSECTR		:SIMULATE SECTOR PULSE	
4145										
4146	015716	004437	023740			JSR	R4,DRSYNC		:SIMULATE HEADER PREAMBLE	
4147	015722	104010				ERROR	10		:MR1 CONTENTS IN ERROR	
4148	015724	104011				ERROR	11		:CS1 CONTENTS IN ERROR	
4149										
4150	015726	004437	024276			JSR	R4,DHDCMP		:SIMULATE HEADER SEARCH	
4151	015732	104012				ERROR	12		:MR1 CONTENTS IN ERROR	
4152	015734	104013				ERROR	13		:CS1 IN ERROR AFTER SEARCH	
4153	015736	104014				ERROR	14		:RKDCYL OR RKDA IN ERROR AFTER SEARCH	
4154										
4155	015740	004437	026500			JSR	R4,DRGPSN		:GO READ GAP AND SYNC	
4156	015744	104015				ERROR	15		:MR1 IN ERROR READING GAP	
4157	015746	104016				ERROR	16		:CS1 IN ERROR AFTER READING GAP	
4158	015750	104032				ERROR	32		:MR1 IN ERROR READING SYNC	
4159	015752	104033				ERROR	33		:CS1 IN ERROR AFTER READING SYNC	
4160										
4161	015754	012737	000001	002346		MOV	#1,ECPTX		:LOAD EXPECTED PATTERN	
4162	015762	012737	005066	002350		MOV	#5066,ECPOSX		:LOAD EXPECTED POSITION	
4163	015770	004437	027176			JSR	R4,DREAD		:GO SIMULATE DATA READ	
4164	015774	000400					400		:NUMBER OF WORDS TO SIMULATE	
4165	015776	104034				ERROR	34		:MR1 IN ERROR READING DATA OR ECC	
4166	016000	104035				ERROR	35		:ECC ERROR READING DATA	
4167	016002	104036				ERROR	36		:CS1 ERROR AFTER READING DATA OR ECC	
4168	016004	104041				ERROR	41		:ECC REG INCORRECT AFTER ECC READ	
4169	016006	104042				ERROR	42		:ERR IN ECC PAT CALC.	
4170	016010	104043				ERROR	43		:ERR IN ECC POS COUNT	
4171										
4172										
4173	016012	012737	000022	002326		MOV	#22,BITCNT		:SET BIT COUNT FOR CORRECTION COUNT	
4174	016020	012700	000005			MOV	#5,R0		:SET COUNT FOR ECCPOS PASS THRU 0	
4175	016024	012701	012672			MOV	#12672,R1		:SET COUNT FOR ECCPOS TO 0 FIRST PASS	
4176	016030	005037	001214			CLR	\$TMP5		:CLEAR SWITCH	
4177										
4178	016034	004737	031616		3\$:	JSR	PC,RDBIT		:GO READ BIT	
4179	016040	004737	030462			JSR	PC,ECCGEN		:GENERATE ECC	
4180	016044	016237	000032	002214		MOV	RKECPT(R2),T.ECPT		:GET PATTERN	
4181	016052	016237	000030	002212		MOV	RKECPS(R2),T.ECPS		:GET POSITION	
4182	016060	005237	002350			INC	ECPOSX		:BUMP POSITION EXPECTED	
4183	016064	013737	002350	002252		MOV	ECPOSX,E.ECPS		:SET FOR COMPARE AND REPORT	
4184	016072	023737	002254	002214		CMP	E.ECPT,T.ECPT		:CHECK IF PATTERN CORRECT	
4185	016100	001401				BEQ	1\$:YES - SKIP	
4186	016102	104042				ERROR	42			
4187	016104	023737	002252	002212	1\$:	CMP	E.ECPS,T.ECPS		:CHECK IF POSITION CORRECT	
4188	016112	001401				BEQ	2\$:YES - SKIP	
4189	016114	104043				ERROR	43			
4190	016116	005237	002326		2\$:	INC	BITCNT		:BUMP BIT COUNT	
4191	016122	005301				DEC	R1		:DEC COUNT TO ZERO IN ECCPOS	

```

4192 016124 001343      BNE 3$      ;NOT YET ZERO - LOOP
4193 016126 005300      DEC R0      ;DEC PASS THRU 0 COUNT
4194 016130 001406      BEQ 4$      ;IF 0 - SKIP
4195 016132 012737 177777 002350  MOV #-1,ECP0SX ;PRESET EXPECTED POS TO GO TO ZERO
4196 016140 012701 020000  MOV #20000,R1 ;SET COUNT TO NEXT 0 IN ECC POS
4197 016144 000733      BR 3$      ;GO DO LOOP AGAIN
4198
4199 016146 005737 001214      4$: TST $TMP5 ;TEST SWITCH
4200 016152 001012      BNE 5$      ;ALREADY SET - SKIP
4201 016154 012701 011041  MOV #11041,R1 ;SET COUNT TO FINAL POSITION
4202 016160 012700 000001  MOV #1,R0 ;SET R0 TO RETURN AFTER ONE PASS
4203 016164 012737 177777 002350  MOV #-1,ECP0SX ;SET EXP POS TO GO TO 0 ON NEXT BIT
4204 016172 005237 001214  INC $TMP5 ;BUMP SWITCH
4205 016176 000716      BR 3$      ;GO DO LOOP AGAIN
4206
4207 016200 004737 031616      5$: JSR PC,RDBIT ;ONE MORE BIT TO SET READY
4208 016204 004437 032356  JSR R4,GETREG ;GET '611 REGISTERS
4209 016210 013737 002260 002220  MOV L.CS1,E.CS1 ;GET EXPECTED CS1
4210 016216 052737 100200 002220  BIS #RDY.CERR,E.CS1 ;SET READY AND ERROR
4211 016224 042737 000001 002220  BIC #GO,E.CS1 ;CLEAR GO
4212 016232 023737 002160 002220  CMP T.CS1,E.CS1 ;CHECK IF CS1 CORRECT
4213 016240 001402      BEQ 6$      ;YES - SKIP
4214 016242 104044      ERROR 44
4215 016244 000410      BR 7$
4216 016246 032737 000100 002174  6$: BIT #ECH,T.ER ;TEST IF ECC HARD ERROR SET
4217 016254 001004      BNE 7$      ;YES - SKIP
4218 016256 012737 100100 002234  MOV #DCK.ECH,E.ER ;SET EXPECTED ERROR REG
4219 016264 104045      ERROR 45
4220
4221 016266      /$:
4222
4223      .SBTTL **SPECIAL READ TESTS
4224
4225      ;*****
4226      ;*TEST 25 PARTIAL SECTOR READ
4227      ;*
4228      ;* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
4229      ;* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A READ DATA
4230      ;* OF 103 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
4231      ;* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
4232      ;* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
4233      ;* A GOOD HEADER, 400 DATA WORDS, AND A GOOD 32 BIT ECC.
4234      ;* MAKE SURE ONLY 103 WORDS GOT TRANSFERRED TO MEMORY.
4235      ;*
4236      ;*****
4237 016266 000004      TST25: SCOPE
4238 016270 012737 000001 001234  MOV #1,$TIMES ;;DO 1 ITERATION
4239
4240 016276 005037 002352      CLR BSEDES ;CLEAR BAD SEC DESIRED
4241 016302 012762 100000 000000  MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
4242
4243 016310 005037 002344      CLR ECCSRC ;CLEAR ECC SOURCE
4244 016314 005037 002354      CLR HIBITS ;CLEAR HI ORDER BITS
4245 016320 004437 032230      JSR R4,LOADRK ;LOAD 'L' REGISTERS
4246 016324 000000      0 ;CYLINDER 0
4247 016326 000      .BYTE 0 ;SECTOR 0

```

4248	016327	000			.BYTE 0	:TRACK 0
4249	016330	047124			IBUFF	:BUFFER ADDRESS IBUFF
4250	016332	177675			-103	:WORD COUNT -103
4251	016334	000021			RDDATA	:COMMAND RDDATA
4252						
4253	016336	004437	031770		JSR R4,CLRIBF	:GO CLEAR INPUT BUFFER
4254						
4255	016342	004437	032014		JSR R4,BLDDAT	:GO BUILD DATA
4256	016346	000006			6	:PATTERN 6
4257						
4258						
4259	016350	004437	032274		JSR R4,OPSTRT	:START THE OPERATION
4260	016354	004437	023334		JSR R4,DISEEK	:SIMULATE IMPLIED SEEK
4261	016360	104004			ERROR 4	:CS1 MISCOMPARE
4262	016362	104005			ERROR 5	:MR1 ..
4263	016364	104006			ERROR 6	:MR2 ..
4264	016366	104007			ERROR 7	:MR3 ..
4265						
4266	016370	004437	023316		JSR R4,DSECTR	:SIMULATE SECTOR PULSE
4267						
4268	016374	004437	023740		JSR R4,DRSYNC	:SIMULATE HEADER PREAMBLE
4269	016400	104010			ERROR 10	:MR1 CONTENTS IN ERROR
4270	016402	104011			ERROR 11	:CS1 CONTENTS IN ERROR
4271						
4272	016404	004437	024276		JSR R4,DHDCMP	:SIMULATE HEADER SEARCH
4273	016410	104012			ERROR 12	:MR1 CONTENTS IN ERROR
4274	016412	104013			ERROR 13	:CS1 IN ERROR AFTER SEARCH
4275	016414	104014			ERROR 14	:RKDCYL OR RKDA IN ERROR AFTER SEARCH
4276						
4277	016416	004437	026500		JSR R4,DRGPSN	:GO READ GAP AND SYNC
4278	016422	104015			ERROR 15	:MR1 IN ERROR READING GAP
4279	016424	104016			ERROR 16	:CS1 IN ERROR AFTER READING GAP
4280	016426	104032			ERROR 32	:MR1 IN ERROR READING SYNC
4281	016430	104033			ERROR 33	:CS1 IN ERROR AFTER READING SYNC
4282						
4283	016432	012737	000000	002346	MOV #0,ECPATX	:LOAD EXPECTED PATTERN
4284	016440	012737	004066	002350	MOV #4066,ECPOSX	:LOAD EXPECTED POSITION
4285	016446	004437	027176		JSR R4,DREAD	:GO SIMULATE DATA READ
4286	016452	000400			400	:NUMBER OF WORDS TO SIMULATE
4287	016454	104034			ERROR 34	:MR1 IN ERROR READING DATA OR ECC
4288	016456	104035			ERROR 35	:ECC ERROR READING DATA
4289	016460	104036			ERROR 36	:CS1 ERROR AFTER READING DATA OR ECC
4290	016462	104041			ERROR 41	:ECC REG INCORRECT AFTER ECC READ
4291	016464	104042			ERROR 42	:ERR IN ECC PAT CALC.
4292	016466	104043			ERROR 43	:ERR IN ECC POS COUNT
4293						
4294						
4295	016470	013700	002164		MOV T.BA,R0	:GET ACTUAL BA
4296	016474	162700	047124		SUB #IBUFF,R0	:SUB START VALUE OF BA
4297	016500	022700	000206		CMP #206,R0	:TEST IF CORRECT NUM OF WORDS XFER
4298	016504	001404			BEQ 4\$:YES - SKIP
4299	016506	012737	047332	002224	MOV #IBUFF+206,E.BA	:SET EXPECTED BA
4300	016514	104046			ERROR 46	:REPORT XFER LENGTH ERROR
4301						
4302	016516	012701	000014	4\$:	MOV #3.*4.,R1	:SET COUNT TO END OF OPERATION
4303	016522	012762	000440	000026 3\$:	MOV #DMD!MCLK,RKMR1(R2)	:RUN THE CLOCK


```

4360
4361 016704 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
4362
4363 016710 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
4364 016714 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
4365 016716 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
4366
4367 016720 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
4368 016724 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
4369 016726 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
4370 016730 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
4371
4372 016732 004437 026500 JSR R4,DRGPSN ;GO READ GAP AND SYNC
4373 016736 104015 ERROR 15 ;MR1 IN ERROR READING GAP
4374 016740 104016 ERROR 16 ;CS1 IN ERROR AFTER READING GAP
4375 016742 104032 ERROR 32 ;MR1 IN ERROR READING SYNC
4376 016744 104033 ERROR 33 ;CS1 IN ERROR AFTER READING SYNC
4377
4378 016746 012737 000000 002346 MOV #0,ECPATX ;LOAD EXPECTED PATTERN
4379 016754 012737 004066 002350 MOV #4066,ECPOSX ;LOAD EXPECTED POSITION
4380 016762 004437 027176 JSR R4,DREAD ;GO SIMULATE DATA READ
4381 016766 000400 400 ;NUMBER OF WORDS TO SIMULATE
4382 016770 104034 ERROR 34 ;MR1 IN ERROR READING DATA OR ECC
4383 016772 104035 ERROR 35 ;ECC ERROR READING DATA
4384 016774 104036 ERROR 36 ;CS1 ERROR AFTER READING DATA OR ECC
4385 016776 104041 ERROR 41 ;ECC REG INCORRECT AFTER ECC READ
4386 017000 104042 ERROR 42 ;ERR IN ECC PAT CALC.
4387 017002 104043 ERROR 43 ;ERR IN ECC POS COUNT
4388
4389
4390 017004 012701 000040 MOV #8*4,R1 ;SET COUNT TO END OF SECTOR
4391 017010 012762 000440 000026 6$: MOV #DMD!MCLK,RKMR1(R2) ;RUN THE CLOCK
4392 017016 012762 000040 000026 MOV #DMD,RKMR1(R2)
4393 017024 005301 DEC R1 ;DEC THE COUNT
4394 017026 001370 BNE 6$ ;LOOP UNTIL ZERO
4395
4396 017030 052737 100000 002352 BIS #BIT15,BSEDES ;SET FOR BAD SECTOR ERROR
4397
4398 017036 004437 023316 JSR R4,DSECTR ;ISSUE SECTOR PULSE
4399
4400 017042 004437 023740 JSR R4,DRSYNC ;READ SYNC
4401 017046 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
4402 017050 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
4403
4404 017052 004437 024276 JSR R4,DHDCMP ;DO HEADER COMPARE WITH BSE ERROR
4405 017056 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
4406 017060 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
4407 017062 000411 BR 1$ ;EXPECTED RETURN (NO DA INCREMENT)
4408
4409 ;
4410 ; IF THIS CODE IS EXECUTED, THE CONTROLLER DID NOT INHIBIT
4411 ; THE ADDRESS INCREMENT BECAUSE OF THE BSE ERROR.
4412 017064 005037 002352 CLR BSEDES ;CLEAR BSE DESIRED SWITCH
4413 017070 013737 002274 002240 MOV L.DCYL,E.DCYL ;SET EXPECTED CYL
4414 017076 013737 002266 002226 MOV L.DA,E.DA ;SET EXPECTED SEC/TRACK
4415 017104 104047 ERROR 47 ;UNEXPECTED ADDRESS INCREMENT

```

```

4416
4417 017106 016237 000014 002174 1$: MOV RKER(R2),T.ER ;GET ERROR REG
4418 017114 032737 000200 002174 BIT #BSE,T.ER ;TEST IF BSE SET
4419 017122 001005 BNE 2$ ;YES - SKIP
4420 017124 012737 000200 002234 MOV #BSE,E.ER ;SET EXPECTED ERROR REG
4421 017132 104051 ERROR 51 ;'BSE DID NOT SET'
4422 017134 000453 BR TST27 ;GO TO NEXT TEST
4423
4424 017136 004437 032356 2$: JSR R4,GETREG ;GET 611 REGS
4425 017142 013737 002260 002220 MOV L.CS1,E.CS1 ;SET EXPECTED CS1
4426 017150 023737 002220 002160 CMP E.CS1,T.CS1 ;CHECK IF CS1 OK
4427 017156 001402 BEQ 3$ ;YES - SKIP
4428 017160 104056 ERROR 56 ;'CS1 ERROR AFTER WRITE CHECK'
4429 017162 000440 BR TST27 ;GO TO NEXT TEST
4430
4431 017164 012701 010540 3$: MOV #278.*16.,R1 ;SET COUNT TO NEXT ECC FIELD
4432 017170 005037 002320 CLR PR.BIT ;CLEAR PRESENT AND PREVIOUS BITS
4433 017174 005037 002322 CLR M1.BIT
4434 017200 004737 031616 4$: JSR PC,RDBIT ;SIMULATE READ BIT
4435 017204 005301 DEC R1 ;DEC COUNT
4436 017206 001374 BNE 4$ ;LOOP UNTIL ZERO
4437
4438 017210 004437 032356 JSR R4,GETREG ;GET 611 REGS
4439 017214 042737 000001 002220 BIC #GO,E.CS1 ;SET UP EXPECTED CS1
4440 017222 052737 100200 002220 BIS #RDY!CERR,E.CS1
4441 017230 023737 002220 002160 CMP E.CS1,T.CS1 ;CHECK IF CS1 OK
4442 017236 001402 BEQ 7$ ;YES - SKIP
4443 017240 104056 ERROR 56 ;'CS1 ERROR AFTER READ DATA'
4444 017242 000410 BR TST27 ;GO TO NEXT TEST
4445
4446 017244 012737 050124 002224 7$: MOV #IBUFF+1000,E.BA ;SET EXPECTED BA
4447 017252 023737 002224 002164 CMP E.BA,T.BA ;TEST IF BA CORRECT
4448 017260 001401 BEQ 5$ ;YES - SKIP
4449 017262 104052 ERROR 52 ;BUS ADDRESS INCORRECT
4450
4451 017264 5$:
4452
4453 .SBTTL **WRITE CHECK TESTS
4454
4455 :*****
4456 :*TEST 27 WRITE CHECK OF 400 WORDS
4457 :*
4458 :* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
4459 :* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
4460 :* CHECK OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
4461 :* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
4462 :* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
4463 :* A GOOD HEADER, THE 400 WORDS TO BE WRITE CHECKED,
4464 :* AND A GOOD 32 BIT ECC. MAKE SURE NO ERRORS SET.
4465 :*
4466 :*****
4467 017264 000004 TST27: SCOPE
4468 017266 012737 000012 001234 MOV #10.,$TIMES ;DO 10. ITERATIONS
4469 017274 005037 002354 CLR HIBITS ;CLEAR HI ORDER BITS
4470 017300 005037 002352 CLR BSEDES ;CLEAR BSE DESIRED
4471

```

4472	017304	005037	002344		CLR	ECCSRC	:CLEAR ECC SOURCE FLAG
4473	017310	005037	002344		CLR	ECCSRC	:CLEAR ECC SOURCE
4474	017314	004437	032230		JSR	R4,LOADRK	:LOAD 'L' REGISTERS
4475	017320	000000			0		:CYLINDER 0
4476	017322	000			.BYTE	0	:SECTOR 0
4477	017323	000			.BYTE	0	:TRACK 0
4478	017324	04724			IBUFF		:BUFFER ADDRESS IBUFF
4479	017326	177400			-400		:WORD COUNT -400
4480	017330	000031			WRTCHK		:COMMAND WRTCHK
4481							
4482	017332	004437	031770		JSR	R4,CLRIBF	:GO CLEAR INPUT BUFFER
4483							
4484	017336	004437	032014		JSR	R4,BLDDAT	:GO BUILD DATA
4485	017342	000004			4		:PATTERN 4
4486							
4487	017344	012700	000400		MOV	#400,R0	:SET A COUNT FOR BUFFER SIZE
4488	017350	012701	047124		MOV	#IBUFF,R1	:SET ADDRESS OF IBUFF
4489	017354	012703	050130		MOV	#OBUFF,R3	:SET ADDRESS OF OBUFF
4490	017360	012321		75\$:	MOV	(R3)+,(R1)+	:MOV PATTERN TO IBUFF
4491	017362	005300			DEC	R0	:DEC COUNT
4492	017364	001375			BNE	75\$:LOOP UNTIL PATTERN IS MOVED
4493							
4494	017366	012762	100000	000000	MOV	#CCLR,RKCS1(R2)	:CLEAR CONTROLLER
4495	017374	004437	032274		JSR	R4,OPSTRT	:START THE OPERATION
4496	017400	004437	023334		JSR	R4,DISEEK	:SIMULATE IMPLIED SEEK
4497	017404	104004			ERROR	4	:CS1 MISCOMPARE
4498	017406	104005			ERROR	5	:MR1
4499	017410	104006			ERROR	6	:MR2
4500	017412	104007			ERROR	7	:MR3
4501							
4502	017414	004437	023316		JSR	R4,DSECTR	:SIMULATE SECTOR PULSE
4503							
4504	017420	004437	023740		JSR	R4,DRSYNC	:SIMULATE HEADER PREAMBLE
4505	017424	104010			ERROR	10	:MR1 CONTENTS IN ERROR
4506	017426	104011			ERROR	11	:CS1 CONTENTS IN ERROR
4507							
4508	017430	004437	024276		JSR	R4,DHDCMP	:SIMULATE HEADER SEARCH
4509	017434	104012			ERROR	12	:MR1 CONTENTS IN ERROR
4510	017436	104013			ERROR	13	:CS1 IN ERROR AFTER SEARCH
4511	017440	104014			ERROR	14	:RKDCYL OR RKDA IN ERROR AFTER SEARCH
4512							
4513	017442	004437	026500		JSR	R4,DRGPSN	:GO READ GAP AND SYNC
4514	017446	104015			ERROR	15	:MR1 IN ERROR READING GAP
4515	017450	104016			ERROR	16	:CS1 IN ERROR AFTER READING GAP
4516	017452	104032			ERROR	32	:MR1 IN ERROR READING SYNC
4517	017454	104033			ERROR	33	:CS1 IN ERROR AFTER READING SYNC
4518							
4519	017456	012737	000000	002346	MOV	#0,ECPATX	:LOAD EXPECTED PATTERN
4520	017464	012737	004066	002350	MOV	#4066,ECPOSX	:LOAD EXPECTED POSITION
4521	017472	004437	027176		JSR	R4,DWRTCK	:GO SIMULATE WRITE CHECK
4522	017476	000400			400		:NUMBER OF WORDS TO SIMULATE
4523	017500	104053			ERROR	53	:MR1 IN ERROR ON WRT CK OR ECC READ
4524	017502	104054			ERROR	54	:ECC ERROR IN WRITE CHECK OP
4525	017504	104055			ERROR	55	:CS1 ERROR AFTER WRT CHK OR ECC READ
4526	017506	104041			ERROR	41	:ECC REG INCORRECT AFTER ECC READ
4527	017510	104042			ERROR	42	:ERR IN ECC PAT CALC.

CZR6ECO RK611 DSKLS CTRL PRT5
CZR6EC.P11 14-SEP-81 13:57

MACY11 30(1046) 14-SEP-81 15:14 K 7 PAGE 89
T27 WRITE CHECK OF 400 WORDS

SEQ 0088

CZ
CZI

4528 017512 104043 ERROR 43 ;ERR IN ECC POS COUNT
4529
4530
4531 017514 012700 002710 MOV #40.*37.,R0 ;SET COUNT TO EMPTY SILO
4532
4533 017520 012762 000440 000026 1\$: MOV #DMD.MCLK,RKMR1(R2) ;CLOCK CONTROLLER

```

4534 017526 012762 000040 000026      MOV    #DMD,RKMR1(R2)
4535 017534 005300                DEC    R0                ;DEC COUNT
4536 017536 001370                BNE    1$                ;LOOP UNTIL 0
4537
4538 017540 004437 032356                JSR    R4,GETREG        ;GET 611 REGS
4539 017544 013737 002260 002220        MOV    L.CS1,E.CS1     ;GET EXPECTED CS1
4540 017552 042737 000001 002220        BIC    #GO,E.CS1       ;CLEAR GO BIT
4541 017560 052737 000200 002220        BIS    #RDY,E.CS1     ;SET READY BIT
4542 017566 023737 002160 002220        CMP    T.CS1,E.CS1    ;CHECK IF CS1 CORRECT
4543 017574 001401                BEQ    2$                ;YES - SKIP
4544 017576 104057                ERROR  57                ;'CS1 IN ERROR AFTER WRITE CHECK''
4545
2$:
4546
4547
4548
4549
4550
4551
4552
4553
4554
4555
4556
4557
4558
4559
4560
4561
4562 017600 000004
4563 017602 012737 000001 001234      TST30: SCOPE
4564
4565 017610 005037 002352                MOV    #1,$TIMES        ;;DO 1 ITERATION
4566 017614 005037 002354                CLR    BSEDES           ;CLEAR BAD SECTOR DESIRED
4567 017620 004437 032230                CLR    HIBITS          ;CLEAR HI ORDER BITS
4568 017624 000000                JSR    R4,LOADRK       ;LOAD 'L' REGISTERS
4569 017626 000000                0                        ;CYLINDER 0
4570 017627 000000                .BYTE 0                ;SECTOR 0
4571 017630 047124                .BYTE 0                ;TRACK 0
4572 017632 177400                Ibuff                ;BUFFER ADDRESS Ibuff
4573 017634 000031                -400                   ;WORD COUNT -400
4574
4575 017636 004437 031770                WRTCHK                ;COMMAND WRTCHK
4576
4577 017642 004437 032014                JSR    R4,CLRIBF       ;GO CLEAR INPUT BUFFER
4578 017646 000005                JSR    R4,BLDDAT       ;GO BUILD DATA
4579
4580 017650 012700 000400                5                        ;PATTERN 5
4581 017654 012701 047124                MOV    #400,R0         ;SET A COUNT FOR BUFFER SIZE
4582 017660 012703 050130                MOV    #IBUFF,R1       ;SET ADDRESS OF Ibuff
4583 017664 012321                MOV    #OBUFF,R3       ;SET ADDRESS OF OBUFF
4584 017666 005300                75$: MOV    (R3)+,(R1)+  ;MOV PATTERN TO Ibuff
4585 017670 001375                DEC    R0                ;DEC COUNT
4586
4587 017672 012700 001002                BNE    75$             ;LOOP UNTIL PATTERN IS MOVED
4588 017676 012737 126073 051130        MOV    #1002,R0        ;SET INDEX INTO BUFFER
4589 017704 012737 151052 051132        MOV    #126073,OBUFF+1000 ;STORE ECC WORDS
        MOV    #151052,OBUFF+1002

```

```

4590 017712 012737 000001 002344      MOV      #1,ECCSRC      ;SET SOURCE TO INDICATE ECC IN BUFFER
4591 017720 032760 000100 050130      BIT      #BIT6,OBUFF(R0) ;TEST IF BIT SET
4592 017726 001004          BNE      100$          ;YES - SKIP
4593 017730 052760 000100 050130      BIS      #BIT6,OBUFF(R0) ;ELSE SET BIT
4594 017736 000403          BR       101$
4595 017740 042760 000100 050130 100$:   BIC      #BIT6,OBUFF(R0) ;CLEAR BIT
4596 017746          101$:
4597
4598 017746 012762 100000 000000 110$:   MOV      #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
4599 017754 004437 032274          JSR      R4,OPSTRT      ;START THE OPERATION
4600 017760 004437 023334          JSR      R4,DISEEK      ;SIMULATE IMPLIED SEEK
4601 017764 104004          ERROR   4             ;CS1 MISCOMPARE
4602 017766 104005          ERROR   5             ;MR1
4603 017770 104006          ERROR   6             ;MR2
4604 017772 104007          ERROR   7             ;MR3
4605
4606 017774 004437 023316          JSR      R4,DSECTR      ;SIMULATE SECTOR PULSE
4607
4608 020000 004437 023740          JSR      R4,DRSYNC      ;SIMULATE HEADER PREAMBLE
4609 020004 104010          ERROR   10            ;MR1 CONTENTS IN ERROR
4610 020006 104011          ERROR   11            ;CS1 CONTENTS IN ERROR
4611
4612 020010 004437 024276          JSR      R4,DHDCMP      ;SIMULATE HEADER SEARCH
4613 020014 104012          ERROR   12            ;MR1 CONTENTS IN ERROR
4614 020016 104013          ERROR   13            ;CS1 IN ERROR AFTER SEARCH
4615 020020 104014          ERROR   14            ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
4616
4617 020022 004437 026500          JSR      R4,DRGPSN      ;GO READ GAP AND SYNC
4618 020026 104015          ERROR   15            ;MR1 IN ERROR READING GAP
4619 020030 104016          ERROR   16            ;CS1 IN ERROR AFTER READING GAP
4620 020032 104032          ERROR   32            ;MR1 IN ERROR READING SYNC
4621 020034 104033          ERROR   33            ;CS1 IN ERROR AFTER READING SYNC
4622
4623 020036 012737 000001 002346      MOV      #1,ECPATX      ;LOAD EXPECTED PATTERN
4624 020044 012737 004066 002350      MOV      #4066,ECPOSX   ;LOAD EXPECTED POSITION
4625 020052 004437 027176          JSR      R4,DWRTCK      ;GO SIMULATE WRITE CHECK
4626 020056 000400          400                  ;NUMBER OF WORDS TO SIMULATE
4627 020060 104053          ERROR   53            ;MR1 IN ERROR ON WRT CK OR ECC READ
4628 020062 104054          ERROR   54            ;ECC ERROR IN WRITE CHECK OP
4629 020064 104055          ERROR   55            ;CS1 ERROR AFTER WRT CHK OR ECC READ
4630 020066 104041          ERROR   41            ;ECC REG INCORRECT AFTER ECC READ
4631 020070 104042          ERROR   42            ;ERR IN ECC PAT CALC.
4632 020072 104043          ERROR   43            ;ERR IN ECC POS COUNT
4633
4634
4635 020074 012737 000020 002326      MOV      #20,BITCNT     ;SET BIT COUNT FOR CORRECTION COUNT
4636 020102 012700 000005          MOV      #5,R0          ;SET COUNT FOR ECCPOS PASS THRU 0
4637 020106 012701 013672          MOV      #13672,R1      ;SET COUNT FOR ECCPOS TO 0 FIRST PASS
4638 020112 005037 001214          CLR      $TMP5          ;CLEAR SWITCH
4639
4640 020116 004737 031616          3$:   JSR      PC,RDBIT       ;GO READ BIT
4641 020122 004737 030462          JSR      PC,ECCGEN      ;GENERATE ECC
4642 020126 016237 000032 002214      MOV      RKECPT(R2),T.ECPT ;GET PATTERN
4643 020134 016237 000030 002212      MOV      RKECPS(R2),T.ECPS ;GET POSITION
4644 020142 005237 002350          INC      ECPOSX         ;BUMP POSITION EXPECTED
4645 020146 013737 002350 002252      MOV      ECPOSX,E.ECPS  ;SET FOR COMPARE AND REPORT

```

```

4646 020154 023737 002254 002214      CMP      E.ECPT,T.ECPT      ;CHECK IF PATTERN CORRECT
4647 020162 001401                BEQ      1$                ;YES - SKIP
4648 020164 104042                ERROR    42
4649 020166 023737 002252 002212 1$:    CMP      E.ECPS,T.ECPS      ;CHECK IF POSITION CORRECT
4650 020174 001401                BEQ      2$                ;YES - SKIP
4651 020176 104043                ERROR    43
4652 020200 005237 002326          2$:    INC      BITCNT            ;BUMP BIT COUNT
4653 020204 005301                DEC      R1                ;DEC COUNT TO ZERO IN ECCPOS
4654 020206 001343                BNE     3$                ;NOT YET ZERO - LOOP
4655 020210 005300                DEC      R0                ;DEC PASS THRU 0 COUNT
4656 020212 001406                BEQ      4$                ;IF 0 - SKIP
4657 020214 012737 177777 002350      MOV      #-1,ECP0SX        ;PRESET EXPECTED POS TO GO TO ZERO
4658 020222 012701 020000          MOV      #20000,R1        ;SET COUNT TO NEXT 0 IN ECC POS
4659 020226 000733                BR       3$                ;GO DO LOOP AGAIN
4660
4661 020230 005737 001214          4$:    TST      $TMP5            ;TEST SWITCH
4662 020234 001012                BNE     5$                ;ALREADY SET - SKIP
4663 020236 012701 010016          MOV      #10016,R1        ;SET COUNT TO FINAL POSITION
4664 020242 012700 000001          MOV      #1,R0            ;SET R0 TO RETURN AFTER ONE PASS
4665 020246 012737 177777 002350      MOV      #-1,ECP0SX        ;SET EXP POS TO GO TO 0 ON NEXT BIT
4666 020254 005237 001214          INC      $TMP5            ;BUMP SWITCH
4667 020260 000716                BR       3$                ;GO DO LOOP AGAIN
4668
4669 020262 004737 031616          5$:    JSR      PC,RDBIT          ;ONE MORE BIT TO SET READY
4670 020266 004437 032356          JSR      R4,GETREG        ;GET '611 REGISTERS
4671 020272 013737 002260 002220      MOV      L.CS1,E.CS1      ;GET EXPECTED CS1
4672 020300 052737 100200 002220      BIS      #RDY!CERR,E.CS1  ;SET READY AND ERROR
4673 020306 042737 000001 002220      BIC      #GO,E.CS1        ;CLEAR GO
4674 020314 023737 002160 002220      CMP      T.CS1,E.CS1      ;CHECK IF CS1 CORRECT
4675 020322 001402                BEQ      6$                ;YES - SKIP
4676 020324 104044                ERROR    44
4677 020326 000410                BR       7$
4678 020330 032737 000100 002174 6$:    BIT      #ECH,T.ER        ;TEST IF ECC HARD ERROR SET
4679 020336 001404                BEQ      7$                ;YES - SKIP
4680 020340 012737 100000 002234      MOV      #DCK,E.ER        ;SET EXPECTED ERROR REG
4681 020346 104045                ERROR    45
4682
4683 020350          7$:
4684
4685
4686
4687
4688
4689
4690
4691
4692
4693
4694
4695
4696
4697 020350 000004          TEST3: SCOPE
4698 020352 012737 000001 001234      MOV      #1,$TIMES        ;DO 1 ITERATION
4699
4700 020360 005037 002352          CLR      BSEDES          ;CLEAR BAD SECTOR DESIRED
4701 020364 005037 002354          CLR      #IBITS          ;CLEAR HI ORDER BITS

```

4702	020370	005037	002344		CLR	ECCSRC		:CLEAR ECC SOURCE
4703	020374	004437	032230		JSR	R4,LCADRK		:LOAD 'L' REGISTERS
4704	020400	000000			0			:CYLINDER 0
4705	020402	000			.BYTE	0		:SECTOR 0
4706	020403	000			.BYTE	0		:TRACK 0
4707	020404	047124			IBUFF			:BUFFER ADDRESS IBUFF
4708	020406	177400			-400			:WORD COUNT -400
4709	020410	010031			WRTCHK!CFM*			:COMMAND WRTCHK CFMT
4710								
4711	020412	004437	031770		JSR	R4,CLRIBF		:GO CLEAR INPUT BUFFER
4712								
4713	020416	004437	032014		JSR	R4,BLDDAT		:GO BUILD DATA
4714	020422	000004			4			:PATTERN 4
4715								
4716	020424	012700	000400		MOV	#400,R0		:SET A COUNT FOR BUFFER SIZE
4717	020430	012701	047124		MOV	#IBUFF,R1		:SET ADDRESS OF IBUFF
4718	020434	012703	050130		MOV	#OBUFF,R3		:SET ADDRESS OF OBUFF
4719	020440	012321		75\$:	MOV	(R3)+,(R1)+		:MOV PATTERN TO IBUFF
4720	020442	005300			DEC	R0		:DEC COUNT
4721	020444	001375			BNE	75\$:LOOP UNTIL PATTERN IS MOVED
4722								
4723	020446	012762	100000	000000 110\$:	MOV	#CLR,RKCS1(R2)		:CLEAR CONTROLLER
4724	020454	004437	032274		JSR	R4,OPSTRT		:START THE OPERATION
4725	020460	004437	023334		JSR	R4,DISEEK		:SIMULATE IMPLIED SEEK
4726	020464	104004			ERROR	4		:CS1 MISCOMPARE
4727	020466	104005			ERROR	5		:MR1 ..
4728	020470	104006			ERROR	6		:MR2 ..
4729	020472	104007			ERROR	7		:MR3 ..
4730								
4731	020474	004437	023316		JSR	R4,DSECTR		:SIMULATE SECTOR PULSE
4732								
4733	020500	004437	023740		JSR	R4,DRSYNC		:SIMULATE HEADER PREAMBLE
4734	020504	104010			ERROR	10		:MR1 CONTENTS IN ERROR
4735	020506	104011			ERROR	11		:CS1 CONTENTS IN ERROR
4736								
4737	020510	004437	024276		JSR	R4,DHDCMP		:SIMULATE HEADER SEARCH
4738	020514	104012			ERROR	12		:MR1 CONTENTS IN ERROR
4739	020516	104013			ERROR	13		:CS1 IN ERROR AFTER SEARCH
4740	020520	104014			ERROR	14		:RKDCYL OR RKDA IN ERROR AFTER SEARCH
4741								
4742	020522	004437	026500		JSR	R4,DRGPSN		:GO READ GAP AND SYNC
4743	020526	104015			ERROR	15		:MR1 IN ERROR READING GAP
4744	020530	104016			ERROR	16		:CS1 IN ERROR AFTER READING GAP
4745	020532	104032			ERROR	32		:MR1 IN ERROR READING SYNC
4746	020534	104033			ERROR	33		:CS1 IN ERROR AFTER READING SYNC
4747								
4748	020536	012737	000000	002346	MOV	#0,ECPATX		:LOAD EXPECTED PATTERN
4749	020544	012737	005066	002350	MOV	#5066,ECPOSX		:LOAD EXPECTED POSITION
4750	020552	004437	027176		JSR	R4,DWRTCK		:GO SIMULATE WRITE CHECK
4751	020556	000400			400			:NUMBER OF WORDS TO SIMULATE
4752	020560	104053			ERROR	53		:MR1 IN ERROR ON WRT CK OR ECC READ
4753	020562	104054			ERROR	54		:ECC ERROR IN WRITE CHECK OP
4754	020564	104055			ERROR	55		:CS1 ERROR AFTER WRT CHK OR ECC READ
4755	020566	104041			ERROR	41		:ECC REG INCORRECT AFTER ECC READ
4756	020570	104042			ERROR	42		:ERR IN ECC PAT CALC.
4757	020572	104043			ERROR	43		:ERR IN ECC POS COUNT

```

4758
4759
4760 020574 012700 003100      MOV      #40.*40.,R0      ;SET COUNT TO EMPTY SILO
4761
4762 020600 012762 000440 000026 1$:  MOV      #DMD!MCLK,RKMR1(R2) ;CLOCK CONTROLLER
4763 020606 012762 000640 000026      MOV      #DMD,RKMR1(R2)
4764 020614 005300      DEC      R0              ;DEC COUNT
4765 020616 001370      BNE     1$              ;LOOP UNTIL 0
4766
4767 020620 004437 032356      JSR     R4,GETREG        ;GET 611 REGS
4768 020624 013737 002260 002220      MOV     L.CS1,E.CS1     ;GET EXPECTED CS1
4769 020632 042737 000001 002220      BIC     #GO,E.CS1       ;CLEAR GO BIT
4770 020640 052737 000200 002220      BIS     #RDY,E.CS1      ;SET READY BIT
4771 020646 023737 002160 002220      CMP     T.CS1,E.CS1     ;CHECK IF CS1 CORRECT
4772 020654 001401      BEQ     2$              ;YES - SKIP
4773 020656 104057      ERROR   57              ;'CS1 IN ERROR AFTER WRITE CHECK'
4774 020660
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784
4785
4786
4787
4788 020660 000004      TST32: SCOPE
4789 020662 012737 000001 001234      MOV     #1,$TIMES       ;:DO 1 ITERATION
4790
4791 020670 005037 002352      CLR     BSEDES          ;:CLEAR BAD SECTOR DESIRED
4792 020674 005037 002354      CLR     HIBITS          ;:CLEAR HI ORDER BITS
4793 020700 005037 002344      CLR     ECCSRC          ;:CLEAR ECC SOURCE
4794 020704 004437 032230      JSR     R4,LOADRK       ;:LOAD 'L' REGISTERS
4795 020710 000000      0              ;:CYLINDER 0
4796 020712 000      .BYTE 0              ;:SECTOR 0
4797 020713 000      .BYTE 0              ;:TRACK 0
4798 020714 047124      IBUFF          ;:BUFFER ADDRESS IBUFF
4799 020716 177777      -1           ;:WORD COUNT -1
4800 020720 000031      WRTCHK        ;:COMMAND WRTCHK
4801
4802 020722 004437 031770      JSR     R4,CLRIBF       ;:GO CLEAR INPUT BUFFER
4803
4804 020726 004437 032014      JSR     R4,BLDDAT       ;:GO BUILD DATA
4805 020732 000006      6              ;:PATTERN 6
4806
4807 020734 012700 000400      MOV     #400,R0         ;:SET A COUNT FOR BUFFER SIZE
4808 020740 012701 047124      MOV     #IBUFF,R1       ;:SET ADDRESS OF IBUFF
4809 020744 012703 050130      MOV     #OBUF,R3        ;:SET ADDRESS OF OBUF
4810 020750 012321      75$:  MOV     (R3)+,(R1)+     ;:MOV PATTERN TO IBUFF
4811 020752 005300      DEC     R0              ;:DEC COUNT
4812 020754 001375      BNE     75$            ;:LOOP UNTIL PATTERN IS MOVED
4813

```

```

4814 020756 012762 100000 000000 110$: MOV #CCLR,RKCS1(R2) ;CLEAR CONTROLLER
4815 020764 004437 032274 JSR R4,OPSTRT ;START THE OPERATION
4816 020770 004437 023334 JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
4817 020774 104004 ERROR 4 ;CS1 MISCOMPARE
4818 020776 104005 ERROR 5 ;MR1 ..
4819 021000 104006 ERROR 6 ;MR2 ..
4820 021002 104007 ERROR 7 ;MR3 ..
4821
4822 021004 004437 023316 JSR R4,DSECTR ;SIMULATE SECTOR PULSE
4823
4824 021010 004437 023740 JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
4825 021014 104010 ERROR 10 ;MR1 CONTENTS IN ERROR
4826 021016 104011 ERROR 11 ;CS1 CONTENTS IN ERROR
4827
4828 021020 004437 024276 JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
4829 021024 104012 ERROR 12 ;MR1 CONTENTS IN ERROR
4830 021026 104013 ERROR 13 ;CS1 IN ERROR AFTER SEARCH
4831 021030 104014 ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
4832
4833 021032 004437 026500 JSR R4,DRGPSN ;GO READ GAP AND SYNC
4834 021036 104015 ERROR 15 ;MR1 IN ERROR READING GAP
4835 021040 104016 ERROR 16 ;CS1 IN ERROR AFTER READING GAP
4836 021042 104032 ERROR 32 ;MR1 IN ERROR READING SYNC
4837 021044 104033 ERROR 33 ;CS1 IN ERROR AFTER READING SYNC
4838
4839 021046 012737 000000 002346 MOV #0,ECPATX ;LOAD EXPECTED PATTERN
4840 021054 012737 004066 002350 MOV #4066,ECPO SX ;LOAD EXPECTED POSITION
4841 021062 004437 027176 JSR R4,DWRTCK ;GO SIMULATE WRITE CHECK
4842 021066 000400 400 ;NUMBER OF WORDS TO SIMULATE
4843 021070 104053 ERROR 53 ;MR1 IN ERROR ON WRT CK OR ECC READ
4844 021072 104054 ERROR 54 ;ECC ERROR IN WRITE CHECK OP
4845 021074 104055 ERROR 55 ;CS1 ERROR AFTER WRT CHK OR ECC READ
4846 021076 104041 ERROR 41 ;ECC REG INCORRECT AFTER ECC READ
4847 021100 104042 ERROR 42 ;ERR IN ECC PAT CALC.
4848 021102 104043 ERROR 43 ;ERR IN ECC POS COUNT
4849
4850
4851 021104 013700 002164 MOV T,BA,R0 ;GET ACTUAL BA
4852 021110 162700 047124 SUB #IBUFF,R0 ;SUB START VALUE OF BA
4853 021114 022700 000002 CMP #2,R0 ;TEST IF CORRECT NUM OF WORDS XFER
4854 021120 001404 BEQ 4$ ;YES - SKIP
4855 021122 012737 047126 002224 MOV #IBUFF+2,E.BA ;SET EXPECTED BA
4856 021130 104046 ERROR 46 ;REPORT XFER LENGTH ERROR
4857
4858 021132 012701 000040 4$: MOV #8,*4,R1 ;SET COUNT TO END OF OPERATION
4859 021136 012762 000440 000026 3$: MOV #DMD.MCLK,RKMR1(R2) ;RUN CLOCK
4860 021144 012762 000040 000026 MOV #DMD,RKMR1(R2)
4861 021152 005301 DEC R1 ;DEC COUNT
4862 021154 001370 BNE 3$ ;LOOP UNTIL ZERO
4863
4864 021156 004437 032356 1$: JSR R4,GETREG ;GET 611 REGS
4865 021162 013737 002260 002220 MOV L.CS1,E.CS1 ;GET EXPECTED CS1
4866 021170 042737 000001 002220 BIC #GO,E.CS1 ;CLEAR GO BIT
4867 021176 052737 000200 002220 BIS #RDY,E.CS1 ;SET READY BIT
4868 021204 023737 002160 002220 CMP T.CS1,E.CS1 ;CHECK IF CS1 CORRECT
4869 021212 001401 BEQ 2$ ;YES - SKIP

```

4870 021214 104057
4871 021216

ERROR 57 ;'CS1 IN ERROR AFTER WRITE CHECK'

2\$:

4872
4873
4874
4875
4876
4877
4878
4879
4880
4881
4882
4883
4884
4885
4886
4887
4888
4889
4890
4891

```

*****
*TEST 33 WRITE CHECK ERROR (PART 1)
*
* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
* CHECK OF 1 WORD TO AN RK06 IN 26 SECTOR FORMAT,
* CLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
* A GOOD HEADER, AND ONE WORD OF DATA CONSISTING OF 000000.
* WRITE CHECKED DATA IS 000001. MAKE SURE WRITE CHECK ERROR SETS.
* REPEAT USING EACH OF THE FOLLOWING CONFIGURATIONS
* AS WRITE CHECKED DATA:
*
* 000002 000040 001000 020000
* 000004 000100 002000 040000
* 000010 000200 004000 100000
* 000020 000400 010000
*****

```

4892
4893 021216 000004
4894 021220 012737 000001 001234
4895
4896 021226 005037 002352
4897 021232 005037 002354
4898 021236 005037 002344
4899 021242 004437 032230
4900 021246 000000
4901 021250 000
4902 021251 000
4903 021252 047124
4904 021254 177777
4905 021256 000031
4906
4907 021260 012705 000020
4908 021264 005037 050130
4909 021270 012737 000001 047124
4910 021276 032737 000001 047124
4911 021304 001003
4912 021306 012737 177777 050130
4913 021314 012737 021322 001110 112\$:
4914
4915 021322 012762 100000 000000 110\$:
4916 021330 004437 032274
4917 021334 004437 023334
4918 021340 104004
4919 021342 104005
4920 021344 104006
4921 021346 104007
4922
4923 021350 004437 023316
4924
4925 021354 004437 023740

```

*****
TST33: SCOPE
MOV #1,$TIMES ;:DO 1 ITERATION
CLR BSEDES ;:CLEAR BAD SECTOR DESIRED
CLR HIBITS ;:CLEAR HI ORDER BITS
CLR ECCSRC ;:CLEAR ECC SOURCE
JSR R4,LOADRK ;:LOAD 'L' REGISTERS
0 ;:CYLINDER 0
.BYTE 0 ;:SECTOR 0
.BYTE 0 ;:TRACK 0
IBUFF ;:BUFFER ADDRESS IBUFF
-1 ;:WORD COUNT -1
WRTCHK ;:COMMAND WRTCHK

MOV #16,$R5 ;:SET PATTERN COUNT
CLR OBUFF ;:CLEAR FIRST WORD OF OBUFF
MOV #1,IBUFF ;:SET IN FIRST PATTERN
BIT #BIT0,IBUFF ;:IS BIT 0 SET
BNE 112$ ;:YES - PATTERN SHOULD BE 0
MOV #-1,OBUFF ;:ELSE PATTERN SHOULD BE ALL ONES
MOV #110$,$LPERR ;:SET LOCAL LOOP ON ERROR

MOV #CCLR,RKCS1(R2) ;:CLEAR CONTROLLER
JSR R4,OPSTRT ;:START THE OPERATION
JSR R4,DISEEK ;:SIMULATE IMPLIED SEEK
ERROR 4 ;:CS1 MISCOMPARE
ERROR 5 ;:MR1 ..
ERROR 6 ;:MR2 ..
ERROR 7 ;:MR3 ..

JSR R4,DSECTR ;:SIMULATE SECTOR PULSE
JSR R4,DRSYNC ;:SIMULATE HEADER PREAMBLE

```



```

4926 021360 104010          ERROR 10          ;MR1 CONTENTS IN ERROR
4927 021362 104010          ERROR 11          ;CS1 CONTENTS IN ERROR
4928
4929 021364 004437 024276    JSR    R4,DHDCMP    ;SIMULATE HEADER SEARCH
4930 021370 104012          ERROR 12          ;MR1 CONTENTS IN ERROR
4931 021372 104013          ERROR 13          ;CS1 IN ERROR AFTER SEARCH
4932 021374 104014          ERROR 14          ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
4933
4934 021376 004437 026500    JSR    R4,DRGPSN    ;GO READ GAP AND SYNC
4935 021402 104015          ERROR 15          ;MR1 IN ERROR READING GAP
4936 021404 104016          ERROR 16          ;CS1 IN ERROR AFTER READING GAP
4937 021406 104032          ERROR 32          ;MR1 IN ERROR READING SYNC
4938 021410 104033          ERROR 33          ;CS1 IN ERROR AFTER READING SYNC
4939
4940 021412 012737 000000 002346  MOV    #0,ECPATX    ;LOAD EXPECTED PATTERN
4941 021420 012737 004066 002350  MOV    #4066,ECPOSX ;LOAD EXPECTED POSITION
4942 021426 004437 027176    JSR    R4,DWRTCK    ;GO SIMULATE WRITE CHECK
4943 021432 000001          1          ;NUMBER OF WORDS TO SIMULATE
4944 021434 104053          ERROR 53          ;MR1 IN ERROR ON WRT CK OR ECC READ
4945 021436 104054          ERROR 54          ;ECC ERROR IN WRITE CHECK OP
4946 021440 104055          ERROR 55          ;CS1 ERROR AFTER WRT CHK OR ECC READ
4947 021442 104041          ERROR 41          ;ECC REG INCORRECT AFTER ECC READ
4948 021444 104042          ERROR 42          ;ERR IN ECC PAT CALC.
4949 021446 104043          ERROR 43          ;ERR IN ECC POS COUNT
4950
4951 021450 012701 000040          MOV    #8,*4.,R1    ;SET COUNT
4952 021454 012762 000440 000026 1$:  MOV    #DMD!MCLK,RKMR1(R2) ;RUN CLOCK
4953 021462 012762 000040 000026  MOV    #DMD,RKMR1(R2)
4954 021470 005301          DEC    R1          ;DEC COUNT
4955 021472 001370          BNE    1$          ;LOOP UNTIL ZERO
4956
4957 021474 004437 032356    JSR    R4,GETREG    ;GET REGS
4958 021500 032737 040000 002170  BIT    #WCE,T.CS2    ;TEST IF WCE SET
4959 021506 001012          BNE    2$          ;YES - SKIP
4960 021510 013737 002170 001162  MOV    T.CS2,$REG0  ;SET CS2 FOR REPORT
4961 021516 013737 047124 001164  MOV    IBUFF,$REG1  ;SET WORD READ
4962 021524 013737 050130 001166  MOV    OBUFF,$REG2  ;SET WORD FROM BUFFER
4963 021532 104060          ERROR 60          ;'WCE FAILED TO SET'
4964
4965 021534 104415          2$:  SCOP1          ;LOCAL LOOP ON ERROR
4966
4967 021536 006337 047124          ASL    IBUFF        ;SHIFT FOR NEXT TEST PATTERN
4968 021542 005305          DEC    R5          ;DEC PATTERN COUNT
4969 021544 001266          BNE    110$        ;LOOP UNTIL ALL PATTERNS TESTED
4970
4971
4972
4973
4974
4975
4976
4977
4978
4979
4980
4981

```

```

*****
*TEST 34          WRITE CHECK ERROR (PART 2)
*
*
* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
* PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A WRITE
* CHECK OF 1 WORD TO AN RK06 IN 26 SECTOR FORMAT,
* CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
* AND DRIVE CLEAR MESSAGES.  SIMULATE A SECTOR PULSE
* A GOOD HEADER, AND ONE WORD OF DATA CONSISTING OF 177777.
* WRITE CHECKED DATA IS 177776.  MAKE SURE WRITE
* CHECK ERROR SETS.  REPEAT USING EACH OF THE FOLLOWING

```

4982
4983
4984
4985
4986
4987
4988
4989
4990
4991
4992
4993
4994
4995
4996
4997
4998
4999
5000
5001
5002
5003
5004
5005
5006
5007
5008
5009
5010
5011
5012
5013
5014
5015
5016
5017
5018
5019
5020
5021
5022
5023
5024
5025
5026
5027
5028
5029
5030
5031
5032
5033
5034
5035
5036
5037

021546 000004
021550 012737 000001 001234
021556 C05037 002352
021562 005037 002354
021566 005037 002344
021572 004437 032230
021576 000000
021600 000
021601 000
021602 047124
021604 177777
021606 000031
021610 012705 000020
021614 005037 050130
021620 012737 177776 047124
021626 032737 000001 047124
021634 001003
021636 012737 177777 050130
021644 012737 021652 001110 112\$
021652 012762 100000 000000 110\$
021660 004437 032274
021664 004437 023334
021670 104004
021672 104005
021674 104006
021676 104007
021700 004437 023316
021704 004437 023740
021710 104010
021712 104011
021714 004437 024276
021720 104012
021722 104013
021724 104014
021726 004437 026500
021732 104015
021734 104016
021736 104032
021740 104033
021742 012737 000000 002346

OCNFIGURATION AS WRITE CHECKED DATA:
177775 177737 176777 157777
177773 177677 175777 137777
177767 177577 173777 077777
177757 177377 167777

TST34: SCOPE
MOV #1,\$TIMES ;;DO 1 ITERATION
CLR BSEDES ;CLEAR BAD SECTOR DESIRED
CLR HIBITS ;CLEAR HI ORDER BITS
CLR ECCSRC ;CLEAR ECC SOURCE
JSR R4,LOADRK ;LOAD 'L' REGISTERS
0 ;CYLINDER 0
.BYTE 0 ;SECTOR 0
.BYTE 0 ;TRACK 0
IBUFF ;BUFFER ADDRESS IBUFF
-1 ;WORD COUNT -1
WRTCHK ;COMMAND WRTCHK
MOV #16,R5 ;SET PATTERN COUNT
CLR OBUFF ;CLEAR FIRST WORD OF OBUFF
MOV #177776,IBUFF ;SET IN FIRST PATTERN
BIT #BIT0,IBUFF ;IS BIT 0 SET
BNE 112\$;YES - PATTERN SHOULD BE 0
MOV #-1,OBUFF ;ELSE PATTERN SHOULD BE ALL ONES
MOV #110\$,\$LPERR ;SET LOCAL LOOP ON ERROR
MOV #CLR,RKCS1(R2) ;CLEAR CONTROLLER
JSR R4,OPSTRT ;START THE OPERATION
JSR R4,DISEEK ;SIMULATE IMPLIED SEEK
ERROR 4 ;CS1 MISCOMPARE
ERROR 5 ;MR1 ..
ERROR 6 ;MR2 ..
ERROR 7 ;MR3 ..
JSR R4,DSECTR ;SIMULATE SECTOR PULSE
JSR R4,DRSYNC ;SIMULATE HEADER PREAMBLE
ERROR 10 ;MR1 CONTENTS IN ERROR
ERROR 1 ;CS1 CONTENTS IN ERROR
JSR R4,DHDCMP ;SIMULATE HEADER SEARCH
ERROR 2 ;MR1 CONTENTS IN ERROR
ERROR 3 ;CS1 IN ERROR AFTER SEARCH
ERROR 14 ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
JSR R4,DRGPSN ;GO READ GAP AND SYNC
ERROR 15 ;MR1 IN ERROR READING GAP
ERROR 16 ;CS1 IN ERROR AFTER READING GAP
ERROR 32 ;MR1 IN ERROR READING SYNC
ERROR 33 ;CS1 IN ERROR AFTER READING SYNC
MOV #0,ECPATX ;LOAD EXPECTED PATTERN

```

5038 021750 012737 004066 002350 MOV #4066,ECPOSX ;LOAD EXPECTED POSITION
5039 021756 004437 027176 JSR R4,DWR'CK ;GO SIMULATE WRITE CHECK
5040 021762 000001 1 ;NUMBER OF WORDS TO SIMULATE
5041 021764 104053 ERROR 53 ;MR1 IN ERROR ON WRT CK OR ECC READ
5042 021766 104054 ERROR 54 ;ECC ERROR IN WRITE CHECK OP
5043 021770 104055 ERROR 55 ;CS1 ERROR AFTER WRT CHK OR ECC READ
5044 021772 104041 ERROR 41 ;ECC REG INCORRECT AFTEP ECC READ
5045 021774 104042 ERROR 42 ;ERR IN ECC PAT CALC.
5046 021776 104043 ERROR 43 ;ERR IN ECC POS COUNT
5047
5048- 022000 012701 000040 MOV #8,*4,R1 ;SET COUNT
5049 022004 012762 000440 000026 1$: MOV #DMD:MCLK,RKMR1(R2) ;RUN CLOCK
5050 022012 012762 000040 000026 MOV #DMD,RKMR1(R2)
5051 022020 005301 DEC R1 ;DEC COUNT
5052 022022 001370 BNF 1$ ;LOOP UNTIL ZERO
5053
5054 022024 004437 032356 JSR R4,GETREG ;GET REGS
5055 022030 032737 040000 002170 BIT #WCE,T.CS2 ;TEST IF WCE SET
5056 022036 001012 BNE 2$ ;YES - SKIP
5057 022040 013737 002170 001162 MOV T.CS2,$REGO ;SET CS2 FOR REPORT
5058 022046 013737 047124 001164 MOV Ibuff,$REG' ;SET WORD READ
5059 022054 013737 050130 001166 MOV Obuff,$REG2 ;SET WORD FROM BUFFER
5060 022062 104060 ERROR 60 ;'WCE FAILED TO SET'
5061
5062 022064 104415 2$: SCOPI ;LOCAL LOOP ON ERROR
5063
5064 022066 006337 047124 ASL Ibuff ;SHIFT FOR NEXT TEST PATTERN
5065 022072 005537 047124 ADC Ibuff ;PUT CARRY BACK IN
5066 022076 005305 DEC R5 ;DEC PATTERN COUNT
5067 022100 001264 BNE 110$ ;LOOP UNTIL ALL PATTERNS TESTED
5068
5069
5070 :*****
5071 :*TEST 35 WRITE CHECK ERROR (PART 3)
5072 :*
5073 :* CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
5074 :* PUT CONTROLLER IN DIAGNOSTIC MODE. ISSUE A WRITE
5075 :* CHECK OF 1 WORD TO AN RK06 IN 24 SECTOR FORMAT,
5076 :* CYLINDER 0, HEAD 0, SECTOR 0. CLOCK THROUGH SEEK
5077 :* AND DRIVE CLEAR MESSAGES. SIMULATE A SECTOR PULSE,
5078 :* A GOOD HEADER, AND ONE WORD OF DATA CONSISTING
5079 :* OF 200000. WRITE CHECK DATA IS 000000. MAKE SURE
5080 :* WRITE CHECK ERROR SETS. REPEAT USING 400000 AS
5081 :* SIMULATED DATA.
5082 :*****
5083 022102 000004 TST35: SCOPE
5084 022104 012737 000001 001234 MOV #1,$TIMES ;:DO 1 ITERATION
5085
5086 022112 005037 002352 CLR BSEDES ;CLEAR BAD SECTOR DESIRED
5087 022116 005037 002344 CLR ECCSRC ;CLEAR ECC SOURCE
5088 022122 004437 032230 JSR R4,LOADRK ;LOAD 'L' REGISTERS
5089 022126 000000 0 ;CYLINDER 0
5090 022130 000 .BYTE 0 ;SECTOR 0
5091 022131 000 .BYTE 0 ;TRACK 0
5092 022132 047124 Ibuff ;BUFFER ADDRESS Ibuff
5093 022134 177777 -1 ;WORD COUNT -1

```

```

5094 022136 010031          WRTCHK.CFMT          :COMMAND WRTCHK.CFMT
5095
5096 022140 012737 040000 002354  MOV    #40000,HIBITS  ;SET HIBITS FOR PATTERN OF 200000
5097 022146 005037 047124          CLR    IBUFF        ;SET LOW BITS FOR READ
5098 022152 005037 050130          CLR    OBUFF        ;SET 0 IN BUFFER
5099 022156 012737 022164 001110 112$:  MOV    #110$, $LPERR ;SET LOCAL LOOP ON ERROR
5100 022164 012762 100000 000000 110$:  MOV    #CCLR,RKLS1(R2) ;CLEAR CONTROLLER
5101
5102 022172 012700 004000          MOV    #4000,R0      ;SET STALL COUNT
5103 022176 005300          DEC    R0            ;DEC COUNT
5104 022200 001376          BNE    4$           ;LOOP UNTIL ZERO
5105
5106 022202 004437 032274          JSR    R4,OPSTRT    ;START THE OPERATION
5107 022206 004437 023334          JSR    R4,DISEEK    ;SIMULATE IMPLIED SEEK
5108 022212 104004          ERROR  4           ;CS1 MISCOMPARE
5109 022214 104005          ERROR  5           ;MR1
5110 022216 104006          ERROR  6           ;MR2
5111 022220 104007          ERROR  7           ;MR3
5112
5113 022222 004437 023316          JSR    R4,DSECTR    ;SIMULATE SECTOR PULSE
5114
5115 022226 004437 023740          JSR    R4,DRSYNC    ;SIMULATE HEADER PREAMBLE
5116 022232 104010          ERROR  10          ;MR1 CONTENTS IN ERROR
5117 022234 104011          ERROR  11          ;CS1 CONTENTS IN ERROR
5118
5119 022236 004437 024276          JSR    R4,DHDCMP    ;SIMULATE HEADER SEARCH
5120 022242 104012          ERROR  12          ;MR1 CONTENTS IN ERROR
5121 022244 104013          ERROR  13          ;CS1 IN ERROR AFTER SEARCH
5122 022246 104014          ERROR  14          ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
5123
5124 022250 004437 026500          JSR    R4,DRGPSN    ;GO READ GAP AND SYNC
5125 022254 104015          ERROR  15          ;MR1 IN ERROR READING GAP
5126 022256 104016          ERROR  16          ;CS1 IN ERROR AFTER READING GAP
5127 022260 104032          ERROR  32          ;MR1 IN ERROR READING SYNC
5128 022262 104033          ERROR  33          ;CS1 IN ERROR AFTER READING SYNC
5129
5130 022264 012737 000000 002346  MOV    #0,ECPATX    ;LOAD EXPECTED PATTERN
5131 022272 012737 005066 002350  MOV    #5066,ECPOSX ;LOAD EXPECTED POSITION
5132 022300 004437 027176          JSR    R4,DWRTCK    ;GO SIMULATE WRITE CHECK
5133 022304 000001          1           ;NUMBER OF WORDS TO SIMULATE
5134 022306 104053          ERROR  53          ;MR1 IN ERROR ON WRT CK OR ECC READ
5135 022310 104054          ERROR  54          ;ECC ERROR IN WRITE CHECK OP
5136 022312 104055          ERROR  55          ;CS1 ERROR AFTER WRT CHK OR ECC READ
5137 022314 104041          ERROR  41          ;ECC REG INCORRECT AFTER ECC READ
5138 022316 104042          ERROR  42          ;ERR IN ECC PAT CALC.
5139 022320 104043          ERROR  43          ;ERR IN ECC POS COUNT
5140
5141 022322 012701 000040          MOV    #8,*4,R1     ;SET COUNT
5142 022326 012762 000440 000026 1$:  MOV    #DMD!MCLK,RKMR1(R2) ;RUN CLOCK
5143 022334 012762 000040 000026  MOV    #DMD,RKMR1(R2)
5144 022342 005301          DEC    R1           ;DEC COUNT
5145 022344 001370          BNE    1$           ;LOOP UNTIL ZERO
5146
5147 022346 004437 032356          JSR    R4,GETREG    ;GET REGS
5148 022352 032737 040000 002170  BIT    #WCE,T.CS2   ;TEST IF WCE SET
5149 022360 001025          BNE    2$           ;YES - SKIP
  
```

```

5150 022362 013737 002170 001162      MOV      T,CS2,$REG0      ;SET CS2 FOR REPORT
5151 022370 013737 047124 001164      MOV      IBUFF,$REG1     ;SET WORD READ
5152 022376 013737 050130 001166      MOV      OBUFF,$REG2     ;SET WORD FROM BUFFER
5153 022404 005037 001170          CLR      $REG3           ;SET UP FOR REPORT
5154 022410 013737 002354 001172      MOV      HIBITS,$REG4
5155 022416 012700 000016          MOV      #14,R0
5156 022422 006037 001172      3$:     ROR      $REG4
5157 022426 005300          DEC      R0
5158 022430 001374          RNE     3$
5159 022432 104063          ERROR   63              ;'WCE FAILED TO SET'
5160
5161 022434 104415          2$:     SCOP1           ;LOCAL LOOP ON ERROR
5162
5163 022436 023727 002354 100000      CMP      HIBITS,#100000  ;TEST IF SECOND PATTERN DONE
5164 022444 001404          BEQ     5$              ;YES - EXIT
5165 022446 012737 100000 002354      MOV      #100000,HIBITS ;SET FOR PATTERN 400000
5166 022454 000643          BR     110$            ;LOOP UNTIL ALL PATTERNS TESTED
5167 022456          5$:
5168
5169
5170
5171
5172
5173
5174
5175
5176
5177
5178
5179
5180
5181
5182 022456 000004          *****
5183 022460 012737 000001 001234      TST36:  SCOPE          ;TEST 36      WRITE CHECK ERROR (PART 4)
5184
5185 022466 005037 002354          ;
5186 022472 005037 002352          ; CLEAR RK611 CONTROLLER WITH A CONTROLLER CLEAR.
5187 022476 004437 032230          ; PUT CONTROLLER IN DIAGNOSTIC MODE.  ISSUE A WRITE
5188 022502 000000          ; CHECK OF 400 WORDS TO AN RK06 IN 26 SECTOR FORMAT,
5189 022504 000          ; CYLINDER 0, HEAD 0, SECTOR 0.  CLOCK THROUGH SEEK
5190 022505 000          ; AND DRIVE CLEAR MESSAGES SIMULATE A SECTOR PULSE,
5191 022506 047124          ; A GOOD HEADER, 125 GOOD WORDS, AND A BAD WORD.
5192 022510 177400          ; MAKE SURE WRITE CHECK ERROR SET.  CHECK WORD COUNT
5193 022512 000031          ; AND BUS ADDRESS.
5194
5195 022514 004437 031770          *****
5196
5197 022520 004437 032014          TST36:  MOV      #1,$TIMES      ;;DO 1 ITERATION
5198 022524 000005          CLR      HIBITS          ;CLEAR HI BITS
5199
5200 022526 012700 000400          CLR      BSEDES         ;CLEAR BAD SECTOR DESIRED
5201 022532 012701 047124          JSR     R4,LOADRK       ;LOAD 'L' REGISTERS
5202 022536 012703 050130          0          ;CYLINDER 0
5203 022542 012321          .BYTE   0              ;SECTOR 0
5204 022544 005300          .BYTE   0              ;TRACK 0
5205 022546 001375          IBUFF          ;BUFFER ADDRESS IBUFF
5206
5207
5208
5209
5210
5211
5212
5213
5214
5215
5216
5217
5218
5219
5220
5221
5222
5223
5224
5225
5226
5227
5228
5229
5230
5231
5232
5233
5234
5235
5236
5237
5238
5239
5240
5241
5242
5243
5244
5245
5246
5247
5248
5249
5250
5251
5252
5253
5254
5255
5256
5257
5258
5259
5260
5261
5262
5263
5264
5265
5266
5267
5268
5269
5270
5271
5272
5273
5274
5275
5276
5277
5278
5279
5280
5281
5282
5283
5284
5285
5286
5287
5288
5289
5290
5291
5292
5293
5294
5295
5296
5297
5298
5299
5300
5301
5302
5303
5304
5305
5306
5307
5308
5309
5310
5311
5312
5313
5314
5315
5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358
5359
5360
5361
5362
5363
5364
5365
5366
5367
5368
5369
5370
5371
5372
5373
5374
5375
5376
5377
5378
5379
5380
5381
5382
5383
5384
5385
5386
5387
5388
5389
5390
5391
5392
5393
5394
5395
5396
5397
5398
5399
5400
5401
5402
5403
5404
5405
5406
5407
5408
5409
5410
5411
5412
5413
5414
5415
5416
5417
5418
5419
5420
5421
5422
5423
5424
5425
5426
5427
5428
5429
5430
5431
5432
5433
5434
5435
5436
5437
5438
5439
5440
5441
5442
5443
5444
5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464
5465
5466
5467
5468
5469
5470
5471
5472
5473
5474
5475
5476
5477
5478
5479
5480
5481
5482
5483
5484
5485
5486
5487
5488
5489
5490
5491
5492
5493
5494
5495
5496
5497
5498
5499
5500
5501
5502
5503
5504
5505
5506
5507
5508
5509
5510
5511
5512
5513
5514
5515
5516
5517
5518
5519
5520
5521
5522
5523
5524
5525
5526
5527
5528
5529
5530
5531
5532
5533
5534
5535
5536
5537
5538
5539
5540
5541
5542
5543
5544
5545
5546
5547
5548
5549
5550
5551
5552
5553
5554
5555
5556
5557
5558
5559
5560
5561
5562
5563
5564
5565
5566
5567
5568
5569
5570
5571
5572
5573
5574
5575
5576
5577
5578
5579
5580
5581
5582
5583
5584
5585
5586
5587
5588
5589
5590
5591
5592
5593
5594
5595
5596
5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617
5618
5619
5620
5621
5622
5623
5624
5625
5626
5627
5628
5629
5630
5631
5632
5633
5634
5635
5636
5637
5638
5639
5640
5641
5642
5643
5644
5645
5646
5647
5648
5649
5650
5651
5652
5653
5654
5655
5656
5657
5658
5659
5660
5661
5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677
5678
5679
5680
5681
5682
5683
5684
5685
5686
5687
5688
5689
5690
5691
5692
5693
5694
5695
5696
5697
5698
5699
5700
5701
5702
5703
5704
5705
5706
5707
5708
5709
5710
5711
5712
5713
5714
5715
5716
5717
5718
5719
5720
5721
5722
5723
5724
5725
5726
5727
5728
5729
5730
5731
5732
5733
5734
5735
5736
5737
5738
5739
5740
5741
5742
5743
5744
5745
5746
5747
5748
5749
5750
5751
5752
5753
5754
5755
5756
5757
5758
5759
5760
5761
5762
5763
5764
5765
5766
5767
5768
5769
5770
5771
5772
5773
5774
5775
5776
5777
5778
5779
5780
5781
5782
5783
5784
5785
5786
5787
5788
5789
5790
5791
5792
5793
5794
5795
5796
5797
5798
5799
5800
5801
5802
5803
5804
5805
5806
5807
5808
5809
5810
5811
5812
5813
5814
5815
5816
5817
5818
5819
5820
5821
5822
5823
5824
5825
5826
5827
5828
5829
5830
5831
5832
5833
5834
5835
5836
5837
5838
5839
5840
5841
5842
5843
5844
5845
5846
5847
5848
5849
5850
5851
5852
5853
5854
5855
5856
5857
5858
5859
5860
5861
5862
5863
5864
5865
5866
5867
5868
5869
5870
5871
5872
5873
5874
5875
5876
5877
5878
5879
5880
5881
5882
5883
5884
5885
5886
5887
5888
5889
5890
5891
5892
5893
5894
5895
5896
5897
5898
5899
5900
5901
5902
5903
5904
5905
5906
5907
5908
5909
5910
5911
5912
5913
5914
5915
5916
5917
5918
5919
5920
5921
5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937
5938
5939
5940
5941
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990
5991
5992
5993
5994
5995
5996
5997
5998
5999
6000

```

```

5206 022550 012700 000530      MOV      #530,R0          ;SET INDEX INTO BUFFER
5207
5208 022554 012737 126073 051130  MOV      #126073,OBUFF+1000 ;STORE ECC WORDS
5209 022562 012737 151052 051132  MOV      #151052,OBUFF+1002
5210 022570 012737 000001 002344  MOV      #1,ECCSRC        ;SET SOURCE TO INDICATE ECC IN BUFFER
5211 022576 032760 000100 050130  BIT      #BIT6,OBUFF(R0)   ;TEST IF BIT SET
5212 022604 001004          BNE      100$             ;YES - SKIP
5213 022606 052760 000100 050130  BIS      #BIT6,OBUFF(R0)   ;ELSE SET BIT
5214 022614 000403          BR       101$
5215 022616 042760 000100 050130 100$:  BIC      #BIT6,OBUFF(R0)   ;CLEAR BIT
5216 022624          101$:
5217
5218 022624 012762 100000 000000 110$:  MOV      #CCLR,RKCS1(R2)  ;CLEAR CONTROLLER
5219 022632 004437 032274          JSR      R4,OPSTRT        ;START THE OPERATION
5220 022636 004437 023334          JSR      R4,DISEEK        ;SIMULATE IMPLIED SEEK
5221 022642 104004          ERROR   4                ;CS1 MISCOMPARE
5222 022644 104005          ERROR   5                ;MR1
5223 022646 104006          ERROR   6                ;MR2
5224 022650 104007          ERROR   7                ;MR3
5225
5226 022652 004437 023316          JSR      R4,DSECTR        ;SIMULATE SECTOR PULSE
5227
5228 022656 004437 023740          JSR      R4,DRSYNC        ;SIMULATE HEADER PREAMBLE
5229 022662 104010          ERROR   10               ;MR1 CONTENTS IN ERROR
5230 022664 104011          ERROR   11               ;CS1 CONTENTS IN ERROR
5231
5232 022666 004437 024276          JSR      R4,DHDCMP        ;SIMULATE HEADER SEARCH
5233 022672 104012          ERROR   12               ;MR1 CONTENTS IN ERROR
5234 022674 104013          ERROR   13               ;CS1 IN ERROR AFTER SEARCH
5235 022676 104014          ERROR   14               ;RKDCYL OR RKDA IN ERROR AFTER SEARCH
5236
5237 022700 004437 026500          JSR      R4,DRGPSN        ;GO READ GAP AND SYNC
5238 022704 104015          ERROR   15               ;MR1 IN ERROR READING GAP
5239 022706 104016          ERROR   16               ;CS1 IN ERROR AFTER READING GAP
5240 022710 104032          ERROR   32               ;MR1 IN ERROR READING SYNC
5241 022712 104033          ERROR   33               ;CS1 IN ERROR AFTER READING SYNC
5242
5243 022714 012737 000000 002346  MOV      #0,ECPATX        ;LOAD EXPECTED PATTERN
5244 022722 012737 004066 002350  MOV      #4066,ECPOSX     ;LOAD EXPECTED POSITION
5245 022730 004437 027176          JSR      R4,DWRTCK        ;GO SIMULATE WRITE CHECK
5246 022734 000310          310                      ;NUMBER OF WORDS TO SIMULATE
5247 022736 104053          ERROR   53               ;MR1 IN ERROR ON WRT CK OR ECC READ
5248 022740 104054          ERROR   54               ;ECC ERROR IN WRITE CHECK OP
5249 022742 104055          ERROR   55               ;CS1 ERROR AFTER WRT CHK OR ECC READ
5250 022744 104041          ERROR   41               ;ECC REG INCORRECT AFTER ECC READ
5251 022746 104042          ERROR   42               ;ERR IN ECC PAT CALC.
5252 022750 104043          ERROR   43               ;ERR IN ECC POS COUNT
5253
5254
5255 022752 012701 000020          MOV      #4,*4,R1         ;SET COUNT TO END OF OPERATION
5256 022756 012762 000440 000026 1$:    MOV      #DMD.MCLK,RKMR1(R2) ;RUN CLOCK
5257 022764 012762 000040 000026  MOV      #DMD,RKMR1(R2)
5258 022772 005301          DEC      R1                ;DEC COUNT
5259 022774 001370          BNE     1$                 ;LOOP UNTIL ZERO
5260
5261 022776 004437 032356          JSR      R4,GETREG        ;GET '611 REGS

```

```

5262 023002 032737 040000 002170 BIT #WCE,T.CS2 ;TEST IF WRITE CHECK SET
5263 023010 001013 BNE 2$ ;YES - SKIP
5264 023012 013737 002170 001162 MOV T.CS2,$REG0 ;SET UP FOR REPORT
5265 023020 016037 047124 001164 MOV Ibuff(R0),$REG1 ;BUFFER WORD
5266 023026 016037 050130 001166 MOV Obuff(R0),$REG2 ;SIMULATED WORD
5267 023034 104060 ERROR 60 ;WCE NOT SET ERROR
5268 023036 000423 BR 4$ ;EXIT
5269
5270 023040 012737 047126 002224 2$: MOV #Ibuff+2,E.BA ;SET UP EXPECTED BA
5271 023046 060037 002224 ADD R0,E.BA ;ADD IN THE NUMBER OF WORDS TO THE ERROR
5272 023052 023737 002224 002164 CMP E.BA,T.BA ;CHECK IF CORRECT
5273 023060 001402 BEQ 3$ ;YES - SKIP
5274 023062 104061 ERROR 61 ;BUS ADDRESS ERROR
5275 023064 000410 BR 4$ ;EXIT
5276
5277 023066 012737 177655 002222 3$: MOV #-123,E.WC ;SET EXPECTED WORD COUNT
5278 023074 023737 002162 002222 CMP T.WC,E.WC ;CHECK IF CORRECT
5279 023102 001401 BEQ 4$ ;YES - SKIP
5280 023104 104062 ERROR 62 ;WORD COUNT ERROR
5281
5282 023106 4$:
5283
  
```

5284
5285
5286
5287
5288
5289
5290
5291
5292
5293 023106
5294 023106 000004
5295 023110 005037 001102
5296 023114 005037 001234
5297 023120 005237 001256
5298 023124 042737 100000 001256
5299 023132 005327
5300 023134 000001
5301 023136 003063
5302 023140 012737
5303 023142 000001
5304 023144 023134
5305 023146 104401 023154
5306 023152 000407
5307
5308 023172
5309 023172 013746 001256
5310
5311 023176 104405
5312 023200 104401 023206
5313 023204 000421
5314
5315 023250
5316 023250 013746 001112
5317
5318 023254 104405
5319 023256 104401 001245
5320 023262 005037 001112
5321 023266 013700 000042
5322 023272 001405
5323 023274 000005
5324 023276 004710
5325 023300 000240
5326 023302 000240
5327 023304 000240
5328 023306
5329 023306 000137
5330 023310 003364
5331 023312 377 377 000
5332 023316
5333
5334
5335
5336 023316 012762 000140 000026
5337 023324 012762 000040 000026
5338 023332 000204
5339

```

.SBTTL END OF PASS ROUTINE

*****
*INCREMENT THE PASS NUMBER ($PASS)
*TYPE 'END PASS #XXXXX TOTAL NUMBER OF ERRORS SINCE LAST REPORT YYYYY'
*WHERE XXXXX AND YYYYY ARE DECIMAL NUMBERS
*IF THERES A MONITOR GO TO IT
*IF THERE ISN'T JUMP TO NEWPAS

$EOP:
      SCOPE
      CLR $STNM          ;;ZERO THE TEST NUMBER
      CLR $TIMES        ;;ZERO THE NUMBER OF ITERATIONS
      INC $PASS         ;;INCREMENT THE PASS NUMBER
      BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
      DEC (PC)+         ;;LOOP?
$EOPCT: .WORD 1
      BGT $DOAGN        ;;YES
      MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
$ENDCT: .WORD 1
      $FOPCT
      TYPE ,65$         ;;TYPE ASCIZ STRING
      BR 64$           ;;GET OVER THE ASCIZ
;;65$: .ASCIZ <12><15>/END PASS #/
64$:
      MOV $PASS,-(SP)  ;;SAVE $PASS FOR TYPEOUT
                        ;;TYPE PASS NUMBER
                        ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPDS
      TYPE ,67$        ;;TYPE ASCIZ STRING
      BR 66$          ;;GET OVER THE ASCIZ
;;67$: .ASCIZ / TOTAL ERRORS SINCE LAST REPORT /
66$:
      MOV $ERTTL,-(SP) ;;SAVE $ERTTL FOR TYPEOUT
                        ;;TOTAL NUMBER OF ERRORS
                        ;;GO TYPE--DECIMAL ASCII WITH SIGN
      TYPDS
      TYPE , $CRLF     ;;TYPE CARRIAGE RETURN, LINE FEED
$GET42: MOV @42,R0    ;;CLEAR ERROR TOTAL
        BEQ $DOAGN    ;;GET MONITOR ADDRESS
        RESET        ;;BRANCH IF NO MONITOR
                        ;;CLEAR THE WORLD
$ENDAD: JSR PC,(R0)   ;;GO TO MONITOR
        NOP          ;;SAVE ROOM
        NOP          ;;FOR
        NOP          ;;ACT11
$DOAGN: JMP @ (PC)+   ;;RETURN
$RTNAD: .WORD NEWPAS
$ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
        .EVEN

.SBTTL DIAGNOSTIC MODE SECTOR PULSE
* THE SECTOR PULSE IS PROVIDED BY THIS ROUTINE
DSECTR: MOV #DMD!MSP,RKMR1(R2) ;;SET SECTOR PULE
        MOV #DMD,RKMR1(R2) ;;RESET SECTOR PULSE
        RTS R4

```



```

5340 .SBTTL DIAGNOSTIC MODE IMPLIED SEEK
5341 :* THE CONTROLLER IS CLOCKED THROUGH THE SEEK MESSAGES,
5342 :* GETTING STATUS, AND CLEARING THE DRIVE. SINCE THIS REQUIRES A
5343 :* DIFFERENT COUNT FOR WRITE DATA AND READ DATA/WRITE CHECK,
5344 :* THE ROUTINE LOOKS AT THE COMMAND THAT WAS
5345 :* STARTED (L.CS1) AND SETS THE CLOCK COUNT ACCORDINGLY.
5346 :*
5347 :* AT THE END OF THE IMPLIED SEEK THE CONTENTS OF CS1, MR1, MR2
5348 :* AND MR3 ARE CHECKED TO VERIFY THE OPERATION. ALL DISCREPENCIES
5349 :* ARE REPORTED IN THE ORDER LISTED.
5350 :*
5351 :* CALL:
5352 :* JSR R4,DISEEK
5353 :* ERROR 4 ;CS1 MISCOMPARE ERROR
5354 :* ERROR 5 ;MR1 MISCOMPARE ERROR
5355 :* ERROR 6 ;MR2 MISCOMPARE ERROR
5356 :* ERROR 7 ;MR3 MISCOMPARE ERROR
5357
5358 023334 DISEEK:
5359 023334 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
5360 023336 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
5361 023340 032737 000002 002260 BIT #BIT1,L.CS1 ;:CHECK IF WRITE DATA
5362 023346 001003 BNE 10$ ;:YES - SKIP
5363 023350 012700 000316 MOV #51.*4+2,R0 ;:SET COUNT FOR READ OR WRITE CHECK
5364 023354 000412 BR 1$
5365
5366 023356 012700 004612 10$: MOV #66.*37.,R0 ;:COUNT FOR CLOCK THROUGH SILO LOAD
5367 023362 032737 010000 002260 BIT #CFMT,L.CS1 ;:TEST IF 22 SECTOR MODE
5368 023370 001402 BEQ 11$ ;:NO - SKIP
5369 023372 012700 005222 MOV #66.*41.,R0 ;:ELSE CHANGE FOR 18 BIT WORDS
5370 023376 062700 090016 11$: ADD #4*3+2,R0 ;:AND SHIFT REGISTER LOAD
5371
5372 023402 012762 000440 000026 1$: MOV #DMD!MCLK,RKMR1(R2) ;:SET CLOCK
5373 023410 012762 000040 000026 MOV #DMD,RKMR1(R2) ;:CLEAR CLOCK
5374 023416 005300 DEC R0 ;:LOOP UNTIL R0 IS ZERO
5375 023420 001370 BNE 1$
5376
5377 023422 013703 002274 MOV L.DCYL,R3 ;:GET DESIRED CYLINDER
5378 023426 004437 032444 JSR R4,FSBLVV ;:REVERSE ORDER OF BITS
5379 023432 010337 002246 MOV R3,E.MR2 ;:STORE AS EXPECTED MR2
5380 023436 113703 002267 MOV#B L.DT,R3 ;:GET DESIRED TRACK/SECTOR
5381 023442 042703 177400 BIC #177400,R3 ;:CLEAR UNUSED BITS
5382 023446 012700 000005 MOV #5,R0 ;:SET SHIFT COUNT
5383 023452 006303 15$: ACL R3 ;:SHIFT FOR TRACK ALIGNMENT
5384 023454 005300 DEC R0 ;:DEC COUNT
5385 023456 001375 BNE 15$ ;:LOOP UNTIL ZERO
5386 023460 153703 002266 B!SB L.DS,R3 ;:INSERT SECTOR NUMBER
5387 023464 032737 010000 002260 BIT #CFMT,L.CS1 ;:TEST IF 18 BIT MODE
5388 023472 001402 BEQ 16$ ;:NO - SKIP
5389 023474 052703 001000 BIS #BIT9,R3 ;:ELSE SET FORMAT BIT IN MDR WORD
5390 023500 004437 032444 16$: JSR R4,FSBLVV ;:REVERSE BIT ORDER
5391 023504 010337 002250 MOV R3,E.MR3 ;:STORE AS EXPECTED MR3
5392
5393 023510 013737 002260 002220 MOV L.CS1,E.CS1 ;:GET EXPECTED CS1
5394 023516 012737 022040 002244 MOV #DMD!ECCW MEWD,E.MR1 ;:SET UP EXPECTED MR1
5395 023524 004437 032356 JSR R4,GETREG ;:GET RK REGS

```

```
5396
5397 023530 023737 002160 002220      CMP      T.CS1,F.CS1      ;CHECK CS1
5398 023536 001411                      BEQ      2$              ;OK - SKIP
5399 023540 012737 023562 001236      MOV      #2$, $ESCAPE   ;SET ESCAPE TO RETURN FOR
5400                                ;REMAINING TESTS
5401 023546 032777 001000 155364      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5402 023554 001056                      BNE      9$              ;YES - SKIP
5403
5404 023556 010446                      4$:      MOV      R4,-(SP) ;SET STACK FOR RETURN TO SUB-ROUTINE
5405 023560 000204                      RTS      R4
5406
5407 023562 005724                      2$:      TST      (R4)+    ;BUMP TO NEXT ERROR CALL
5408 023564 023737 002244 002204      CMP      E.MR1,T.MR1    ;CHECK MR1
5409 023572 001410                      BEQ      3$              ;OK - SKIP
5410 023574 032777 001000 155336      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5411 023602 001043                      BNE      9$              ;YES - SKIP
5412 023604 012737 023614 001236      MOV      #3$, $ESCAPE   ;SET ESCAPE FOR RETURN TO SUB-ROUTINE
5413 023612 000761                      BR       4$
5414
5415 023614 005724                      3$:      TST      (R4)+    ;SAME OPERATION AS ABOVE TO
5416 023616 023737 002246 002206      CMP      E.MR2,T.MR2    ;CHECK MR2
5417 023624 001410                      BEQ      5$              ;OK - SKIP
5418 023626 032777 001000 155304      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5419 023634 001026                      BNF      9$              ;YES - SKIP
5420 023636 012737 023646 001236      MOV      #5$, $ESCAPE   ;SET ESCAPE FOR RETURN TO SUB-ROUTINE
5421 023644 000744                      BR       4$
5422
5423 023646 005724                      5$:      TST      (R4)+    ;SAME OPERATION AS ABOVE TO
5424 023650 023737 002250 002210      CMP      E.MR3,T.MR3    ;CHECK MR3
5425 023656 001414                      BEQ      8$              ;OK - SKIP
5426 023660 032777 001000 155252      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5427 023666 001011                      BNE      9$              ;YES - SKIP
5428 023670 012737 023700 001236      MOV      #6$, $ESCAPE   ;SET ESCAPE FOR RETURN TO SUB-ROUTINE
5429 023676 000727                      BR       4$
5430
5431 023700 032777 001000 155232      6$:      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5432 023706 001001                      BNE      9$              ;YES - SKIP
5433 023710 005724                      8$:      TST      (R4)+    ;BUMP TO NO ERROR RETURN
5434 023712 005037 001236      9$:      CLR      $ESCAPE       ;CLEAR ESCAPE
5435 023716 012603                      MOV      (SP)+,R3       ;;POP STACK INTO R3
5436 023720 012600                      MOV      (SP)+,R0       ;;POP STACK INTO R0
5437 023722 000204                      RTS      R4
5438
5439 023724 005037 001236      7$:      CLR      $ESCAPE       ;CLEAR ESCAPE
5440 023730 012706 001100      MOV      #STACK,SP     ;CLEAN OFF STACK
5441 023734 000177 155150      JMP      @ $LPERR      ;GO TO LOOP START
5442
5443      .SBTTL  DIAGNOSTIC MODE READ SYNC PATTERN (PREAMBLE)
5444      ;*      THE 255 '0' BITS AND 1 '1' BIT ARE CLOCKED THROUGH THE READ.
5445      ;*      THE CONTENTS OF MR1 ARE MONITORED TO WATCH FOR AN ERROR.
5446      ;*      AT THE END OF THE SYNC READ, CS1 IS CHECKED TO INSURE IT
5447      ;*      DID NOT GET CHANGED.
5448      ;*
5449      ;*      CALL:
5450      ;*      JSR      R4,DRSYNC
5451      ;*      ERROR   10      ;MR1 CONTENTS IN ERROR
```

```
5452      :*      ERROR 11      ;CS1 CONTENTS IN ERROR
5453      :*
5454      :*      IF AN ERROR IS DETECTED, THE SUBROUTINE WILL SET UP TO ABORT
5455      :*      THE PRESENT TEST UNLESS LOOP ON ERROR IS SET.
5456
5457      023740      DRSYNC:
5458      023740      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
5459      023742      005037      002326      CLR      BITCNT      ;CLEAR BIT COUNT
5460      023746      005037      001236      CLR      $ESCAPE      ;CLEAR ESCAPE
5461      023752      012700      000200      MOV      #128,R0      ;SET COUNT TO CLOCK TO READ GATE
5462      023756      005037      002320      CLR      PR.BIT      ;CLEAR PRESENT BIT
5463      023762      005037      002322      CLR      M1.BIT      ;CLEAR PREVIOUS BIT
5464      023766      012737      022040      002244      MOV      #DMD!MEWD!ECCW,E.MR1 ;SET UP EXPECTED MR1
5465
5466      023774      004737      031616      3$:      JSR      PC,RDBIT      ;GO READ A BIT
5467      024000      016237      000026      002204      MOV      RKM1(R2),T.MR1 ;GET MR1
5468      024006      023737      002204      002244      CMP      T.MR1,E.MR1    ;CHECK IF IT IS CORRECT
5469      024014      001424      BEQ      2$            ;YES - SKIP
5470
5471      024016      012737      024066      001236      MOV      #2,$ESCAPE    ;SET ESCAPE FOR CONTINUE TEST
5472
5473      024024      005037      001236      4$:      CLR      $ESCAPE      ;CLEAR ESCAPE
5474      024030      032777      001000      155102      BIT      #BIT9,@SWR     ;CHECK IF LOOP ON ERROR
5475      024036      001011      BNF      1$            ;YES SKIP
5476      024040      013700      001254      MOV      $TESTN,R0     ;ELSE SET UP TO SKIP TO NEXT TEST
5477      024044      006300      ASL      R0
5478      024046      016037      033700      001236      MOV      $SW08TB(R0),$ESCAPE
5479      024054      162737      000002      001236      SUB      #2,$ESCAPE
5480
5481      024062      1$:      MOV      (SP)+,R0      ;;POP STACK INTO R0
5482      024064      000204      RTS      R4
5483
5484      024066      005237      002326      2$:      INC      BITCNT      ;INCREMENT THE BIT COUNT
5485      024072      005300      DEC      R0            ;LOOP UNTIL READY FOR READ GATE
5486      024074      001337      BNE      3$
5487
5488      024076      012700      000177      MOV      #127,R0      ;SET COUNT FOR REST OF SYNC
5489      024102      012737      122040      002244      MOV      #DMD!MEWD!ECCW!RDGATE,E.MR1 ;SET NEW EXPECTED MR1
5490
5491      024110      004737      031616      5$:      JSR      PC,RDBIT      ;GO READ BIT
5492      024114      016237      000026      002204      MOV      RKM1(R2),T.MR1 ;GET MR1
5493      024122      023737      002204      002244      CMP      T.MR1,E.MR1    ;CHECK IF CORRECT
5494      024130      001404      BEQ      6$            ;YES - SKIP
5495      024132      012737      024142      001236      MOV      #6,$ESCAPE    ;SET FOR CONTINUE TESTING
5496      024140      000445      BR       50$          ;GO TO ERROR EXIT
5497      024142      005237      002326      6$:      INC      BITCNT      ;BUMP BIT COUNTER
5498      024146      005300      DEC      R0            ;LOOP UNTIL READY FOR SYNC 1
5499      024150      001357      BNE      5$
5500
5501      024152      013737      002320      002322      MOV      PR.BIT,M1.BIT ;MOV PRESENT TO PREVIOUS
5502      024160      012737      000001      002320      MOV      #1,PR.BIT     ;SET SYNC 1 BIT
5503      024166      004737      031616      JSR      PC,RDBIT      ;GO READ BIT
5504      024172      016237      000026      002204      MOV      RKM1(R2),T.MR1 ;GET MR1
5505      024200      023737      002204      002244      CMP      T.MR1,E.MR1    ;CHECK IF CORRECT
5506      024206      001404      BEQ      7$            ;YES - SKIP
5507
```

```

5508 024210 012737 024220 001236      MOV    #7$, $ESCAPE      ;SET RETURN FOR CONTINUE TEST
5509 024216 000416                BR     50$              ;GO TO ERROR EXIT
5510
5511 024220 004437 032356      7$:   JSR    R4, GETREG    ;GO GET RK611 REGISTERS
5512 024224 005237 002326                INC    BITCNT           ;BUMP BIT COUNTER
5513 024230 005724                TST    (R4)+            ;BUMP TO NEXT ERROR CALL
5514 024232 013737 002260 002220      MOV    L.CS1, E.CS1     ;GET EXPECTED CS1
5515 024240 023737 002160 002220      LMP   T.CS1, E.CS1     ;CHECK IF CORRECT
5516 024246 001266                BNE   4$               ;NO - SKIP
5517
5518 024250 005724                TST   (R4)+            ;BUMP TO NO ERROR RETURN
5519 024252 000703                BR    1$               ;SKIP TO RETURN
5520
5521 024254 032777 001000 154656 50$:   BIT   #BIT9, @SWR      ;TEST IF LOOP ON ERROR
5522 024262 001403                BEQ   51$              ;NO - SKIP
5523 024264 005037 001236                CLR   $ESCAPE          ;CLEAR FOR SKIP TO NEXT TEST
5524 024270 000674                BR    1$
5525 024272 010446      51$:   MOV    R4, -(SP)       ;SET UP STACK FOR RETURN
5526 024274 000204                RTS   R4
5527
5528      .SBTTL  DIAGNOSTIC MODE HEADER READ AND COMPARE
5529      ;*      THIS ROUTINE SIMULATES THE THREE HEADER WORDS READ AND
5530      ;*      COMPARE THAT IS PART OF ALL DATA TRANSFER OPERATIONS.
5531      ;*
5532      ;*      THE EXPECTED HEADER IS CALCULATED FROM THE CYLINDER, TRACK,
5533      ;*      AND SECTOR SPECIFIED IN THE 'L' REGISTERS. THE READING OF THE
5534      ;*      3 WORDS IS DONE AND MR1 IS MONITORED TO INSURE IT DOES
5535      ;*      NOT CHANGE. AT THE END OF THE READ/COMPARE, THE CONTENTS
5536      ;*      OF RKCS1 IS CHECKED AND THE INCREMENT OF RKDCYL AND RKDA IS
5537      ;*      VERIFIED. IF THE INCREMENT IS GOOD, THE 'L' REGISTERS ARE
5538      ;*      UPDATED TO THE NEW PACK ADDRESS IN CASE OF MULTI-SECTOR
5539      ;*      OPERATIONS. THE BAD SECTOR ERROR DESIRED (BSEDES) SWITCH
5540      ;*      IS TESTED AND IF IT IS NON-ZERO, THE 2ND AND 3RD HEADER
5541      ;*      WORDS ARE CHANGED ACCORDING TO BSEDES CONTENTS. THIS
5542      ;*      CAN BE USED TO FORCE HEADER TYPE ERROR.
5543      ;*
5544      ;*      CALL:
5545      ;*      JSR    R4, DHDCMP
5546      ;*      ERROR  12      ;ERROR IN MR1
5547      ;*      ERROR  13      ;ERROR IN CS1
5548      ;*      ERROR  14      ;ERROR IN PACK ADDRESS
5549
5550      DHDCMP:
5551 024276 010046      MOV    R0, -(SP)       ;;PUSH R0 ON STACK
5552 024300 010146      MOV    R1, -(SP)       ;;PUSH R1 ON STACK
5553 024302 010346      MOV    R3, -(SP)       ;;PUSH R3 ON STACK
5554 024304 010546      MOV    R5, -(SP)       ;;PUSH R5 ON STACK
5555 024306 005037 002326      CLR    BITCNT          ;CLEAR BIT COUNTER
5556 024312 012700 000005      MOV    #5, R0          ;SET A SHIFT COUNT
5557 024316 113701 002267      MOVB   L.DI, R1        ;GET DESIRED TRACK
5558 024322 042701 177400      BIC    #177400, R1     ;CLEAR UNUSED BITS
5559
5560 024326 006301      1$:   ASL    R1              ;SHIFT TRACK TO ALIGN TO
5561 024330 005300                DEC    R0              ;HEADER WORD FORMAT
5562 024332 001375                BNE   1$
5563

```

```
5564 024334 153701 002266      BISB  L.DS,R1      ;INSERT DESIRED SECTOR
5565 024340 052701 140000      BIS   #BIT15,BIT14,R1 ;SET BS BITS
5566 024344 013700 002274      MOV   L.DCYL,R0    ;GET DESIRED CYLINDER
5567 024350 032737 010000 002260  BIT   #CFMT,L.CS1  ;TEST IF 22 SECTOR FORMAT
5568 024356 001402              BEQ   2$          ;NO - SKIP
5569 024360 052701 001000      BIS   #BIT9,R1     ;ELSE SET FORMAT BIT
5570 024364 010003              2$:  MOV   R0,R3       ;COMPUTE VRC
5571 024366 010105              MOV   R1,R5
5572 024370 040005              BIC   R0,R5
5573 024372 040103              BIC   R1,R3
5574 024374 050503              BIS   R5,R3
5575 024376 005737 002352      TST   BSEDES      ;TEST IF BSE DESIRED
5576 024402 001404              BEQ   23$        ;NO - SKIP
5577 024404 043701 002352      BIC   BSEDES,R1   ;CLEAR BIT IN WORD 2
5578 024410 043703 002352      BIC   BSEDES,R3   ;CORRECT HVRC
5579 024414              23$:
5580
5581      ;          RO, R1, AND R3 HAVE THE EXPECTED HEADERS
5582
5583 024414 005037 001204      CLR   $TMP1       ;CLEAR A PASS COUNTER
5584 024420 012737 122040 002244  MOV   #DMD!MEWD!ECCW!RDGATE,E.MR1 ;SET EXPECTED MR1
5585
5586 024426 012705 000001 11$:  MOV   #BIT0,R5    ;ESTABLISH BEGINNING OF DATA WORD
5587 024432 013737 002320 002322  7$:  MOV   PR.BIT,M'.BIT ;STORE PREVIOUS BIT
5588 024440 005037 002320      CLR   PR.BIT     ;PRESET FOR PRESENT BIT 0
5589 024444 030500      BIT   R5,R0     ;TEST IF BIT TO BE A ONE
5590 024446 001403              BEQ   3$          ;NO - SKIP
5591 024450 012737 000001 002320  MOV   #BIT0,PR.BIT ;ELSE CHANGE TO ONE
5592 024456 004737 031616 3$:  JSR   PC,RDBIT   ;GO READ A BIT
5593 024462 016237 000026 002204  MOV   RKMRI(R2),T.MR1 ;GET MR1 FOR TESTING
5594 024470 023737 002204 002244  CMP   T.MR1,E.MR1 ;CHECK IF CORRECT
5595 024476 001431              BEQ   6$          ;YES - SKIP
5596 024500 012737 024562 001236  MOV   #6$,$ESCAPE ;SET RETURN TO CONTINUE TEST
5597 024506 000137 025160      JMP   50$
5598
5599 024512 005037 001236 4$:  CLR   $ESCAPE    ;CLEAR TO JUMP TO NEXT TEST
5600 024516 032777 001000 1544'4  BIT   #BIT9,@SWR  ;CHECK IF LOOP ON ERROR
5601 024524 001011              BNE   5$         ;YES - SKIP
5602 024526 013705 001254      MOV   $TESTN,R5  ;GET CURRENT TEST NUMBER
5603 024532 006305      ASL   R5         ;AND SET UP TO ESCAPE TO NEXT TEST
5604 024534 016537 033700 001236  MOV   $SW08TB(R5),$ESCAPE
5605 024542 162737 000002 001236  SUB   #2,$ESCAPE
5606
5607      5$:
5608 024550      MOV   (SP)+,R5   ;;POP STACK INTO R5
5609 024552 012603      MOV   (SP)+,R3   ;;POP STACK INTO R3
5610 024554 012601      MOV   (SP)+,R1   ;;POP STACK INTO R1
5611 024556 012600      MOV   (SP)+,R0   ;;POP STACK INTO R0
5612 024560 000204      RTS   R4
5613
5614 024562 005237 002326 6$:  INC   BITCNT     ;BUMP BIT COUNT
5615 024566 005705      TST   R5        ;CHECK IF LAST BIT OF THIS WORD
5616 024570 100402      BMI   8$        ;HAS BEEN SIMULATED - YES - SKIP
5617 024572 006305      ASL   R5        ;SHIFT TO NEXT BIT
5618 024574 000716      BR   7$         ;GO DO THIS BIT
5619
```

```

5620 024576 005737 001204      8$:  TST      $TMP1      ;TEST IF PASS FOR 1ST WORD JUST DONE
5621 024602 001004                BNE      9$          ;NO - SKIP
5622 024604 010100                MOV     R1,R0       ;PUT WORD 2 IN R0 FOR SIMULATION
5623 024606 005237 001204      INC     $TMP1       ;BUMP PASS COUNT
5624 024612 000705                BR      11$
5625
5626 024614 023727 001204 000002  9$:  CMP     $TMP1,#2    ;JUST DONE WITH WORD 2?
5627 024622 002004                BGE     10$        ;NO SKIP
5628 024624 010300                MOV     R3,R0       ;PUT 3RD WORD IN R0
5629 024626 005237 001204      INC     $TMP1       ;BUMP PASS COUNT
5630 024632 000675                BR      11$
5631
5632 024634 023727 001204 000003 10$:  CMP     $TMP1,#3    ;JUST DONE WITH WORD 3?
5633 024642 002006                BGE     16$        ;NO - SKIP
5634 024644 005000                CLR     R0          ;PUT IN WORD 0 OF GAP
5635 024646 012705 100000      MOV     #BIT15,R5   ;SET TO GIVE IT 1ST GAP BIT
5636 024652 005237 001204      INC     $TMP1       ;BUMP PASS COUNT
5637 024656 000665                BR      7$
5638
5639 024660 023727 001204 000004 16$:  CMP     $TMP1,#4    ;TEST IF READY FOR GAP
5640 024666 001436                BFO     12$        ;YES - SKIP
5641
5642      ;
5643      ; THE FOLLOWING BLOCK OF CODE COMPUTES THE NUMBER OF CLOCK
5644      ; PULSES (WHICH ARE ACTUALLY CALLS TO RDBIT) THE OPERATION
5645      ; BEING PERFORMED REQUIRES TO UPDATE THE ADDRESS COUNTERS
5646      ; (RKDC AND RKDA). THIS CALCULATION IS BASED ON THE WHAT WAS
5647      ; IN THESE COUNTERS WHEN THE OPERATION STARTED (L.DC AND L.DA).
5648 024670 012701 000025                MOV     #25,R1      ;PRESET R1 FOR 24 SECTOR MODE
5649 024674 032737 010000 002260      BIT     #CFMT,L.CS1 ;IS IT 24 SECTOR MODE OPERATION?
5650 024702 001402                BEQ     21$        ;YES - SKIP
5651 024704 012701 000023                MOV     #23,R1      ;ELSE CHANGE FOR 22 SECTOR MODE
5652 024710 052701 001000      21$:  BIS     #BIT9,R1    ;SET FOR MAX HEAD OF 2
5653 024714 023701 002266                CMP     L.DS,R1     ;TEST IF TRACK/SECTOR ADDRESS WAS MAX
5654 024720 001003                BNE     17$        ;NO - SKIP
5655 024722 012705 000100      MOV     #BIT6,R5    ;SET FOR 8 BITS OF GAP TO UPDATE
5656 024726 000410                BR      19$
5657 024730 123701 002266      17$:  CMPB   L.DS,R1     ;TEST IF SECTOR WAS MAX
5658 024734 001003                BNE     18$        ;NO - SKIP
5659 024736 012705 000400      MOV     #BIT8,R5    ;SET FOR 6 BITS OF GAP TO UPDATE
5660 024742 000402                BR      19$
5661 024744 012705 010000      18$:  MOV     #BIT12,R5   ;SET TO GIVE 1ST 4 BITS OF GAP
5662 024750 005237 001204      19$:  INC     $TMP1       ;BUMP PASS COUNT
5663 024754 042737 100000 002244      BIC     #RDGATE,E.MR1 ;CLEAR READ GATE
5664 024762 000623                BR      7$
5665      ;
5666      ; END OF BLOCK
5667 024764 004437 032356      12$:  JSR     R4,GETREG   ;GET RK611 REGISTERS
5668 024770 005724                TST     (R4)+       ;BUMP TO NEXT ERROR RETURN
5669 024772 013737 002260 002220      MOV     L.CS1,E.CS1 ;SET EXPECTED CS1
5670 025000 023737 002160 002220      CMP     T.CS1,F.CS1 ;CHECK IF CORRECT
5671 025006 001241                BNE     4$          ;NO SKIP
5672
5673 025010 005724                TST     (R4)+       ;BUMP TO NEXT ERROR RETURN
5674 025012 013737 002274 002240      MOV     L.DCYL,E.DCYL ;GET STARTING CYLINDER
5675 025020 013737 002266 002226      MOV     L.DA,E.DA   ;
  
```

```

5676 025026 105237 002226          INCB   E.DA          ;BUMP SECTOR
5677 025032 012700 000026          MOV    #26,R0      ;PRESET FOR 26 SEC TRACK
5678 025036 032737 010000 002260  BIT    #CFMT,L.CS1 ;TEST IF IN 26 SECTOR MODE
5679 025044 001402                BEQ    13$         ;YES - SKIP
5680 025046 012700 000024          MOV    #24,R0      ;ELSE CHANGE FOR 24 SECTOR MODE
5681 025052 120037 002226          13$:  CMPB   R0,E.DA     ;CHECK IF SECTOR TO LARGE
5682 025056 001015                BNE    15$         ;NO - SKIP
5683 025060 105037 002226          CLRB   E.DA        ;SET SECTOR TO 0
5684 025064 105237 002227          INCB   E.DA+1      ;BUMP TRACK
5685 025070 012700 000003          MOV    #3,R0       ;PRESET FOR 3 TRACKS (RK06)
5686 025074 123700 002227          CMPB   E.DA+1,R0   ;CHECK IF TRACK INC TO LARGE
5687 025100 001004                BNE    15$         ;NO - SKIP
5688
5689 025102 105037 002227          CLRB   E.DA+1      ;CLEAR TRACK TO 0
5690 025106 005237 002240          INC    E.DCYL      ;BUMP CYLINDER
5691
5692 025112 023737 002240 002200  15$:  CMP    E.DCYL,T.DCYL ;CHECK IF CYL OK
5693 025120 001004                BNE    25$         ;NO - SKIP
5694 025122 023737 002226 002166  CMP    E.DA,T.DA    ;CHECK IF DA OK
5695 025130 001402                BEQ    26$         ;YES - SKIP
5696 025132 000137 024512          25$:  JMP    4$         ;NO - SKIP
5697
5698 025136 013737 002240 002274  26$:  MOV    E.DCYL,L.DCYL ;STORE COMPUTED NEXT CYL
5699 025144 013737 002226 002260  MOV    E.DA,L.DA    ;AND NEXT DA
5700
5701 025152 005724                TST    (R4)+        ;BUMP TO NO ERROR RETURN
5702 025154 000137 024550          JMP    5$         ;GO TO GOOD EXIT
5703
5704 025160 032777 001000 153752  50$:  BIT    #BIT9,@SWR   ;TEST IF LOOP ON ERROR
5705 025166 001404                BEQ    51$         ;NO - SKIP
5706 025170 005037 001236          CLR    $ESCAPE     ;CLEAR TO LOOP ON ERROR
5707 025174 000137 024550          JMP    5$         ;YES - SKIP
5708
5709 025200 010440          51$:  MOV    R4,-(SP)     ;SET STACK TO CONTINUE TEST
5710 025202 000204          RTS    R4
5711          .SBTTL  DIAGNOSTIC MODE GAP READ AND SYNC WRITE
5712          :*    THE READING OF THE GAP AND WRITING OF THE SYNC (OR PREAMBLE)
5713          :*    IS HANDLED IN THIS ROUTINE.
5714          :*
5715          :*    THE FIRST GAP BITS ARE PROCESSED IN THE HEADER COMPARE
5716          :*    ROUTINE AND THE REMAINDER ARE PROCESSED HERE. THEN A
5717          :*    CHECK IS MADE THAT WRITE GATE CAME ON AND 255 '0' BITS
5718          :*    AND A SINGLE '1' BIT IS WRITTEN.
5719          :*
5720          :*    CALL:
5721          :*    JSR    R4,DWGPSN
5722          :*    ERROR 15      ;MR1 IN ERROR IN READ GAP
5723          :*    ERROR 16      ;CS1 IN ERROR AFTER READ GAP
5724          :*    ERROR 17      ;MR1 IN ERROR IN WRITE SYNC
5725          :*    ERROR 20      ;CS1 IN ERROR AFTER SYNC WRITE
5726          :*
5727          :*    ROUTINE CALLED:
5728          :*    JSR    PC,WRTBIT
5729          :*    RETURN POINT IF ERROR IN WRTBIT
5730          :*    NO ERROR RETURN
5731
    
```

```
5732 025204 DWGPN:
5733 025204 010046 MOV R0,-(SP) ;;PUSH R0 ON STACK
5734 : THE FOLLOWING CODE ADJUST THE COUNT FOR THE REMAINDER OF
5735 : THE GAP. THE ADJUSTMENT DEPENDS ON HOW MANY GAP BITS WERE
5736 : READ IN THE HEADER COMPARE ROUTINE FOR ADDRESS REGISTER UPDATE.
5737 025206 005737 002266 TST L,DA ;TEST IF DESIRED TRACK/SECTOR NOW 0
5738 025212 001006 BNE 8$ ;NO - SKIP
5739 025214 012700 000066 MOV #54,R0 ;ELSE ADJUST COUNT FOR CYL CHANGE
5740 025220 012737 000011 002326 MOV #11,BITCNT ;PRESET BIT COUNT FOR REST OF GAP
5741 025226 000416 BR 10$
5742 025230 105737 002266 8$: TSTB L,DS ;TEST IF NEW SECTOR IS 0
5743 025234 001006 BNE 9$ ;NO - SKIP
5744 025236 012700 000070 MOV #56,R0 ;ELSE ADJUST COUNT FOR TRACK CHANGE
5745 025242 012737 000007 002326 MOV #7,BITCNT ;PRESET BIT COUNT FOR REST OF GAP
5746 025250 000405 BR 10$
5747 025252 012737 000005 002326 9$: MOV #5,BITCNT ;PRESET BIT COUNTER FOR REST OF GAP
5748 025260 012700 000074 MOV #60,R0 ;SET COUNT FOR 60 BITS
5749 :
5750 : END OF BLOCK
5751 025264 012737 022040 002244 10$: MOV #DMD!ECCW.MEWD,E.MR1 ;SET EXPECTED MR1
5752 025272 005037 002320 CLR PR.BIT ;CLEAR PRESENT BIT
5753 025276 005037 002322 CLR M1.BIT ;CLEAR PREVIOUS BIT
5754 025302 004737 031616 1$: JSR PC,RDBIT ;GO READ BIT
5755 025306 016237 000026 002204 MOV RKMRI(R2),T.MR1 ;GET MR1
5756 025314 023737 002244 002204 CMP E.MR1,T.MR1 ;CHECK IF CORRECT
5757 025322 001404 BEQ 11$ ;YES - SKIP
5758 025324 012737 025346 001236 MOV #2$, $ESCAPE ;SET RETURN TO CONT TEST
5759 025332 000513 BR 50$
5760 025334 005237 002326 11$: INC BITCNT ;BUMP BIT COUNT
5761 025340 005300 DEC R0 ;DEC LOOP COUNT
5762 025342 001357 BNE 1$ ;LOOP IF NOT ZERO
5763 025344 000421 BR 4$ ;ELSE SKIP
5764 :
5765 025346 005037 001236 2$: CLR $ESCAPE ;CLEAR TO EXIT OR LOOP
5766 025352 022777 001000 153560 CMP #BIT9,@SWR ;ERROR RETURN: CHECK IF LOOP ON ERROR
5767 025360 001011 BNE 3$ ;YES - SKIP
5768 025362 013700 001254 MOV $TESTN,R0 ;GET NUMBER OF NEXT TEST
5769 025366 006300 ASL R0 ;SHIFT FOR INDEXING
5770 025370 016037 033700 001236 MOV $$W08TB(R0),$ESCAPE ;SET ESCAPE FOR GOT TO NEXT TEST
5771 025376 162737 000002 001236 SUB #2,$ESCAPE ;SET TO FIRST INST (SCOPE)
5772 025404 3$:
5773 025404 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
5774 025406 000204 RTS R4
5775 :
5776 025410 004437 032356 4$: JSR R4,GETREG ;GET RK611 REGS
5777 025414 005724 TST (R4)+ ;BUMP TO NEXT ERROR RETURN
5778 025416 013737 002260 002220 MOV L.CS1,E.CS1 ;SET EXPECTED CS1
5779 025424 023737 002160 002220 CMP T.CS1,E.CS1 ;CHECK IF CS1 OK
5780 025432 001345 BNE 2$ ;NO - SKIP
5781 :
5782 025434 005724 TST (R4)+ ;BUMP RETURN POINTER
5783 :
5784 025436 005037 002326 CLR BITCNT ;CLEAR BIT COUNTER
5785 025442 005037 002322 CLR M1.BIT ;CLEAR ALL WRITE BIT INDICATORS
5786 025446 005037 002324 CLR M2.BIT
5787 025452 005037 002316 CLR P1.BIT
```



```
5788 025456 005037 002320          CLR    PR,BIT
5789 025462 012737 062040 002244  MOV    #DMD!ECCW!MEWD.WRTGATE,E.MR1 ;SET EXPECTED MR1
5790 025470 012700 000400          MOV    #256.,RC ;SET COUNT TO WRITE 256 '0'
5791
5792 025474 004737 030652          5$:   JSR    PC,WRTBIT ;CALL WRITE BIT
5793 025500 000405          BR     6$ ;ERROR RETURN - SKIP TO HANDLER
5794 025502 005237 002326          INC    BITCNT ;BUMP BIT COUNT
5795 025506 005300          DEC    R0 ;LOOP UNTIL ALL 256
5796 025510 001371          BNE    5$ ;BITS ARE WRITTEN
5797 025512 000402          BR     7$
5798
5799 025514 010446          6$:   MOV    R4,-(SP) ;WRITE ERROR HANDLER-SET
5800 025516 000204          RTS    R4 ;STACK FOR REPORTING SUBSEQUENT ERRORS
5801
5802 025520 012737 000001 002316  7$:   MOV    #1,P1.BIT ;SET SYNC BIT OF 1 AS NEXT BIT
5803 025526 004737 030652          JSR    PC,WRTBIT ;GO WRITE IT
5804 025532 000770          BR     6$ ;ERROR RETURN
5805 025534 005237 002326          INC    BITCNT ;BUMP BIT COUNT
5806
5807 025540 004437 032356          JSR    R4,GETREG ;GET RK611 REGS
5808 025544 005724          TST   (R4)+ ;BUMP TO NEXT ERROR RETURN
5809 025546 023737 002160 002220  CMP    T.CS1,E.CS1 ;CHECK IF CORRECT
5810 025554 001274          BNE    2$ ;NO - SKIP
5811
5812 025556 005724          TST   (R4)+ ;BUMP PAST ERROR RETURNS
5813 025560 000711          BR     3$ ;GO TO EXIT
5814
5815 025562 032777 001000 153350  50$:  BIT    #BIT9,@SWR ;TEST IF LOOP ON ERROR
5816 025570 001403          BEQ   51$ ;YES - SKIP
5817 025572 005037 001236          CLR   $ESCAPE ;ELSE SET TO LOOP
5818 025576 000702          BR     3$
5819 025600 010446          51$:  MOV    R4,-(SP) ;PRESET STACK FOR RETURN
5820 025602 000204          RTS    R4
5821          .SBT'L  DIAGNOSTIC WRITE DATA AND ECC ROUTINE
5822          ;* THE DATA IN THE OUPUT BUFFER IS WRITE SIMULATED, ECC IS
5823          ;* GENERATED AND WRITTEN, AND THE POSTAMBLE IS WRITTEN.
5824          ;*
5825          ;* CALL:
5826          ;* JSR    R4,DWRITE
5827          ;* ERROR 17 ;MR1 IN ERROR IN WRITING
5828          ;* ERROR 21 ;ECC ERROR IN WRITING
5829          ;* ERROR 22 ;CS1 ERROR AFTER WRITE
5830          ;* NO ERROR RETURN
5831          ;*
5832          ;* ROUTINE CALLED:
5833          ;* JSR    PC,WRTBIT
5834          ;* RETURN POINT IF ERROR IN WRTBIT
5835          ;* NO ERROR RETURN FROM WRTBIT
5836
5837 025604          DWRITE:
5838 025604 010046          MOV    R0,-(SP) ;:PUSH R0 ON STACK
5839 025606 010146          MOV    R1,-(SP) ;:PUSH R1 ON STACK
5840 025610 010346          MOV    R3,-(SP) ;:PUSH R3 ON STACK
5841 025612 010546          MOV    R5,-(SP) ;:PUSH R5 ON STACK
5842 025614 005037 002326          CLR    BITCNT ;:CLEAR BIT COUNTER
5843 025620 005037 002342          CLR    ECCLO ;:CLEAR ECC LOW
```

```

5844 025624 005037 002340      CLR      ECCHI          :CLEAR ECC HI
5845 025630 012700 000400      MOV      #400,R0        :SET WORD COUNT FOR ONE SECTOR
5846 025634 012703 050130      MOV      #OBUFF,R3     :GET POINTER TO OUTPLT DATA
5847 025640 010437 001216      MOV      R4,$TMP6      :STORE MR1 ERROR RETURN
5848 025644 005724          TST      (R4)+         :BUMP TO NEXT ERROR RETURN
5849 025646 010437 001220      MOV      R4,$TMP7      :STORE ECC ERROR RETURN
5850
5851      :
5852      : THE FOLLOWING CODE SIMULATES THE WRITING OF 400 WORDS.
5853      : THE PROCEDURE IS TO LOAD THE CURRENT WORD INTO R5 AND
5854      : TO SHIFT A BIT FROM RIGHT TO LEFT TO POINT TO THE CURRENT BIT.
5855 025652 005037 001214      CLR      $TMP5         :CLEAR SWITCH
5856 025656 012701 000001      3$: MOV      #BIT0,R1    :LOAD BIT POINTER
5857 025662 012305          MOV      (R3)+,R5     :GET A WORD FOR WRITING
5858
5859 025664 013737 002322 002324 4$: MOV      M1.BIT,M2.BIT :SHIFT CURRENT-1 TO CURRENT-2
5860 025672 013737 002320 002322      MOV      PR.BIT,M1.BIT :PRESENT TO CURRENT-1
5861 025700 013737 002316 002320      MOV      P1.BIT,PR.BIT :NEXT TO PRESENT
5862 025706 005037 002316          CLR      P1.BIT      :CLEAR NEXT BIT
5863 025712 030105          BIT      R1,R5       :TEST IF NEXT BIT IS ONE
5864 025714 001403          BFO      5$         :NO - SKIP
5865 025716 012737 000001 002316      MOV      #1,P1.BIT    :ELSE SET NEXT BIT
5866 025724 004737 030652 5$: JSR      PC,WRTBIT    :GO WRITE BIT
5867 025730 000441          BR       33$        :ERROR RETURN - SKIP
5868 025732 004737 030462 43$: JSR      PC,ECCGEN    :GO GENERATE ECC FOR THIS BIT
5869 025736 016237 000032 002214      MOV      RKECPT(R2),T.ECPT :GET PATTERN
5870 025744 023737 002254 002214      CMP      E.ECPT,T.ECPT :CHECK IF ECC CORRECT
5871 025752 001402          BEQ     40$        :YES - SKIP
5872 025754 000137 026352          JMP     13$        :JUMP TO ERROR EXIT
5873
5874 025760 005237 002326 40$: INC      BITCNT      :BUMP BIT COUNT
5875 025764 005701          TST     R1          :TEST BIT POINTER
5876 025766 100424          BMI     7$         :IF NEGATIVE - SKIP
5877 025770 006301          ASL     R1          :ELSE SHIFT LEFT AND LOOP
5878 025772 020027 000001      CMP     R0,#1       :TEST IF WRITING LAST WORD
5879 025776 001332          BNE     4$         :NO - GO TO LOOP
5880 026000 032737 010000 002260      BIT     #CFMT,L.CS1  :TEST IF 22 SECTOR MODE
5881 026006 001404          BEQ     32$        :NO - SKIP
5882 026010 005737 001214          TST     $TMP5       :TEST IF 18 BIT SWITCH SET
5883 026014 001001          BNE     32$        :YES - SKIP
5884 026016 000722          BR     4$         :ELSE LOOP
5885 026020 005701 32$: TST     R1          :ELSE TEST IF LAST BIT OF LAST WORD
5886 026022 100320          BPL     4$         :NO - GO TO LOOP
5887 026024 042737 020000 002244      BIC     #ECCW,E.MR1  :ELSE SET MR FOR ECC WRITE
5888 026032 000714          BR     4$         :
5889 026034 000137 026374 33$: JMP     20$        :
5890
5891 026040 032737 010000 002260 7$: BIT     #CFMT,L.CS1  :TEST IF 22 SECTOR MODE
5892 026046 001413          BEQ     31$        :NO - SKIP
5893 026050 005737 001214          TST     $TMP5       :TEST IF 18 BIT SWITCH SET
5894 026054 001006          BNE     30$        :YES - SKIP
5895 026056 012701 040000      MOV     #BIT14,R1    :SET TO SUPPLY 2 MORE BITS (16 & 17)
5896 026062 005005          CLR     R5         :MAKE SURE THEY ARE 0'S
5897 026064 005137 001214      COM     $TMP5       :SET THE 18 BIT SWITCH
5898 026070 000675          BR     4$         :GO WRITE THE BITS
5899

```

5900	026072	005037	001214		30\$:	CLR	\$TMP5	:CLEAR THE SWITCH
5901	026076	005300			31\$:	DEC	R0	:DEC WORD COUNT
5902	026100	001266				BNE	3\$:IF NOT ZERO - GO GET NEXT WORD
5903	026102	005037	001214			CLR	\$TMP5	:CLEAR FOR PASS CONTROL
5904	026106	013705	002340			MOV	ECCHI,R5	:GET FIRST ECC WORD
5905	026112	012701	000C01			MOV	#BIT0,R1	:SET BIT POINTER
5906								
5907	026116	013737	002322	002324	8\$:	MOV	M1.BIT,M2.BIT	:SHIFT BIT INDICATORS AGAIN
5908	026124	013737	002320	002322		MOV	PR.BIT,M1.BIT	
5909	026132	013737	002316	002320		MOV	P1.BIT,PR.BIT	
5910	026140	005037	002316			CLR	P1.BIT	:CLEAR NEXT BIT
5911	026144	030105				BIT	R1,R5	:TEST IF BIT IS ZERO
5912	026146	001403				BEQ	9\$:YES - SKIP
5913	026150	012737	000001	002316		MOV	#1,P1.BIT	:ELSE CHANGE TO ONE
5914	026156	004737	030652		9\$:	JSR	PC,WRTBIT	:GO WRITE BIT
5915	026162	000501				BR	14\$:WRTBIT ERROR - SKIP
5916								
5917	026164	005237	002326		41\$:	INC	BITCNT	:BUMP BIT COUNT
5918	026170	005701				TST	R1	:TEST BIT POINTER
5919	026172	100413				BMI	10\$:IF NEGATIVE - SKIP
5920	026174	006301				ASL	R1	:ELSE SHIFT AND LOOP
5921	026176	023727	001214	000001		CMP	\$TMP5,#1	:TEST IF WRITING LAST ECC WORD
5922	026204	001344				BNE	8\$:NO - SKIP TO LOOP
5923	026206	005701				TST	R1	:ELSE TEST IF WRITING LAST BIT OF LAST ECC
5924	026210	100342				BPL	8\$:NO - SKIP TO LOOP
5925	026212	052737	020000	002244		BIS	#ECCW,E.MR1	:ELSE SET ECCW IN MR1
5926	026220	000736				BR	8\$	
5927								
5928	026222	005737	001214		10\$:	TST	\$TMP5	:TEST PASS 0 JUST DONE
5929	026226	001007				BNE	11\$:NO - SKIP
5930	026230	005237	001214			INC	\$TMP5	:BUMP PASS COUNT
5931	026234	012701	000001			MOV	#BIT0,R1	:PRESET BIT POINTER
5932	026240	013705	002342			MOV	ECCLO,R5	:GET SECOND ECC WORD
5933	026244	000724				BR	8\$:LOOP
5934								
5935	026246	012701	000017		11\$:	MOV	#15.,R1	:SET A BIT COUNT
5936								
5937	026252	013737	002322	002324	12\$:	MOV	M1.BIT,M2.BIT	:SHIFT BIT INDICATORS
5938	026260	013737	002320	002322		MOV	PR.BIT,M1.BIT	
5939	026266	013737	002316	002320		MOV	P1.BIT,PR.BIT	
5940	026274	005037	002316			CLR	P1.BIT	:SET ZERO FOR WRITE POSTAMBLE
5941	026300	004737	030652			JSR	PC,WRTBIT	:GO WRITE BIT
5942	026304	000436				BR	15\$:WRITE BIT ERROR - SKIP
5943	026306	005237	002326		42\$:	INC	BITCNT	:BUMP BIT COUNT
5944	026312	005301				DEC	R1	:DEC BIT COUNT
5945	026314	001356				BNE	12\$:LOOP UNTTIL BIT COUNT 0
5946								
5947	026316	004437	032356			JSR	R4,GETREG	:GET RK611 REGS
5948	026322	013704	001220			MOV	\$TMP7,R4	:SET R4 FOR ECC ERROR RETURN
5949	026326	005724				TST	(R4)+	:BUMP TO CS1 ERROR RETURN
5950	026330	013737	002260	002220		MOV	L.CS1,E.CS1	:GET EXPECTED CS1
5951								
5952	026336	023737	002160	002220		CMP	T.CS1,E.CS1	:CHECK IF CORRECT
5953	026344	001031				BNE	18\$:NO - SKIP
5954	026346	005724				TST	(R4)+	:BUMP TO GOOD RETURN
5955	026350	000446				BR	19\$:GO TO RETURN

```
5956  
5957 026352 013704 001220  
5958 026356 012737 025760 001236  
5959 026364 000410  
5960  
13$: MOV $TMP7,R4 ;SET FOR ECC ERROR RETURN  
MOV #40$, $ESCAPE ;STORE RETURN POINT FOR NO LOOP ON ERROR  
BR 16$
```

```

5961 026366 013704 001216      14$:  MOV    $TMP6,R4      ;SET FOR "R1 ERROR RETURN
5962 026372 000405                BR      16$
5963
5964 026374 013704 001216      20$:  MOV    $TMP6,R4      ;SET FOR MR1 ERROR RETURN
5965 026400 000402                BR      16$
5966
5967 026402 013704 001216      15$:  MOV    $TMP6,R4      ;SET FOR MR1 ERROR RETURN
5968
5969 026406 032777 001000 152524 16$:  BIT    #BIT9,@SWR      ;CHECK IF LOOP ON ERROR
5970 026414 001403                BEQ    17$              ;NO - SKIP
5971
5972 026416 005037 001236                CLR    $ESCAPE        ;CLEAR ESCAPE
5973 026422 000421                BR     19$            ;SKIP TO EXIT
5974
5975 026424 010446                17$:  MOV    R4,-(SP)      ;CONDITION STACK
5976 026426 000204                RTS    R4             ;GO REPORT ERROR
5977
5978 026430 005037 001236      18$:  CLR    $ESCAPE        ;CLEAR FOR LOOP OR EXIT
5979 026434 032777 001000 152476  BIT    #BIT9,@SWR      ;CHECK IF LOOP ON ERROR
5980 026442 001011                BNE    19$            ;YES - SKIP
5981 026444 013700 001254                MOV    $TESTN,R0      ;SET UP TO SKIP TO NEXT TEST
5982 026450 006300                ASL    R0
5983 026452 016037 033700 001236  MOV    $SW08TB(R0),$ESCAPE
5984 026460 162737 000002 001236  SUB    #2,$ESCAPE
5985 026466
5986 026466 012605      19$:  MOV    (SP)+,R5      ;;POP STACK INTO R5
5987 026470 012603                MOV    (SP)+,R3      ;;POP STACK INTO R3
5988 026472 012601                MOV    (SP)+,R1      ;;POP STACK INTO R1
5989 026474 012600                MOV    (SP)+,R0      ;;POP STACK INTO R0
5990 026476 000204                RTS    R4
5991
5992      .SBTTL  DIAGNOSTIC MODE READ GAP AND DATA SYNC
5993      :*      THE READING OF THE GAP AND READING OF THE SYNC (OR PREAMBLE)
5994      :*      IS HANDLED IN THIS ROUTINE.
5995      :*
5996      :*      THE FIRST GAP BITS ARE PROCESSED IN THE HEADER COMPARE
5997      :*      ROUTINE AND THE REMAINING GAP BITS ARE PROCESSED HERE.
5998      :*      THEN A CHECK IS MADE THAT READ GATE CAME ON
5999      :*      AT THE 129TH BIT OF SYNC. ;*
6000      :*      CALL:
6001      :*      JSR    R4,DRGPSN
6002      :*      ERROR  15      ;MR1 IN ERROR IN READ GAP
6003      :*      ERROR  16      ;CS1 IN ERROR AFTER READ GAP
6004      :*      ERROR  32      ;MR1 IN ERROR IN READ SYNC
6005      :*      ERROR  33      ;CS1 IN ERROR AFTER SYNC READ
6006      :*
6007      :*      ROUTINE CALLED:
6008      :*      JSR    PC,RDBIT
6009
6010 026500      DRGPSN:
6011 026500 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
6012      :      THE FOLLOWING CODE ADJUST THE COUNT FOR THE REMAINDER OF
6013      :      THE GAP. THE ADJUSTMENT DEPENDS ON HOW MANY GAP BITS WERE
6014      :      READ IN THE HEADER COMPARE ROUTINE FOR ADDRESS REGISTER UPDATE.
6015 026502 005737 002266      TST    L,DA          ;TEST IF DESIRED TRACK/SECTOR NOW 0
6016 026506 001006                BNE    8$            ;NO - SKIP

```

```

6017 026510 012700 000066          MOV    #54.,RO      ;ELSE ADJUST COUNT FOR CYL CHANGE
6018 026514 012737 000011 002326  MOV    #11,BITCNT  ;PRESET BIT COUNT FOR REST OF GAP
6019 026522 000416          BR     10$         ;
6020 026524 105737 002266          8$:   TSTB   L.DS     ;TEST IF NEW SECTOR IS 0
6021 026530 001006          BNE   9$         ;NO - SKIP
6022 026532 012700 000070          MOV    #56.,RO     ;ELSE ADJUST COUNT FOR TRACK CHANGE
6023 026536 012737 000007 002326  MOV    #7,BITCNT   ;PRESET BIT COUNT FOR REST OF GAP
6024 026544 000405          BR     10$         ;
6025 026546 012737 000005 002326  9$:   MOV    #5,BITCNT  ;PRESET BIT COUNTER FOR REST OF GAP
6026 026554 012700 000074          MOV    #60.,RO    ;SET COUNT FOR 60 BITS
6027          ;
6028          END OF BLOCK
6029 026560 012737 022040 002244 10$:   MOV    #DMD!ECCW.MEWD,E.MR1 ;SET EXPECTED MR1
6030 026566 005037 002320          CLR    PR.BIT     ;CLEAR PRESENT BIT
6031 026572 005037 002322          CLR    M1.BIT    ;CLEAR PREVIOUS BIT
6032 026576 004737 031616          1$:   JSR    PC,RDBIT   ;GO READ BIT
6033 026602 016237 000026 002204  MOV    RKMRI(R2),T.MR1 ;GET MR1
6034 026610 023737 002244 002204  CMP    E.MR1,T.MR1 ;CHECK IF CORRECT
6035 026616 001404          BEQ   11$        ;YES - SKIP
6036 026620 012737 026642 001236  MOV    #2$, $ESCAPE ;SET RETURN TO CONT TEST
6037 026626 000552          BR     50$        ;
6038 026630 005237 002326          11$:  INC    BITCNT     ;BUMP BIT COUNT
6039 026634 005300          DEC    RO         ;DEC LOOP COUNT
6040 026636 001357          BNE   1$         ;LOOP IF NOT ZERO
6041 026640 000421          BR     4$         ;ELSE SKIP
6042
6043 026642 005037 001236          2$:   CLR    $ESCAPE    ;CLEAR TO EXIT OR LOOP
6044 026646 022777 001000 152264  CMP    #BIT9,@SWR  ;ERROR RETURN: CHECK IF LOOP ON ERROR
6045 026654 001011          BNE   3$         ;YES - SKIP
6046 026656 013700 001254          MOV    $TESTN,RO  ;GET NUMBER OF NEXT TEST
6047 026662 006300          ASL   RO         ;SHIFT FOR INDEXING
6048 026664 016037 033700 001236  MOV    $$W08TB(RO), $ESCAPE ;SET ESCAPE FOR GOT TO NEXT TEST
6049 026672 162737 000002 001236  SUB    #2,$ESCAPE ;SET TO FIRST INST (SCOPE)
6050 026700
6051 026700 012600          MO,   (SP)+,RO   ;:POP STACK INTO RO
6052 026702 000204          RTS   R4
6053
6054 026704 004437 032356          4$:   JSR    R4,GETREG  ;GET RK611 REGS
6055 026710 005724          TST   (R4)+      ;BUMP TO NEXT ERROR RETURN
6056 026712 013737 002260 002220  MOV    L.CS1,E.CS1 ;SET EXPECTED CS1
6057 026720 023737 002160 002220  CMP    T.CS1,E.CS1 ;CHECK IF CS1 OK
6058 026726 001345          BNE   2$         ;NO - SKIP
6059
6060 026730 005724          TST   (R4)+      ;BUMP RETURN POINTER
6061
6062 026732 005037 002326          CLR    BITCNT     ;CLEAR BIT COUNTER
6063 026736 012737 022040 002244  MOV    #DMD!ECCW!MEWD,E.MR1 ;SET EXPECTED MR1
6064 026744 012700 000200          MOV    #128.,RO   ;SET COUNT TO READ 128 'C'
6065
6066 026750 004737 031616          5$:   JSR    PC,RDBIT   ;CALL READ BIT
6067 026754 016237 000026 002204  MOV    RKMRI(R2),T.MR1 ;GET MR1 FOR TESTING
6068 026762 023737 002244 002204  CMP    E.MR1,T.MR1 ;TEST IF CORRECT
6069 026770 001404          BEQ   22$        ;YES - SKIP
6070
6071 026772 012737 02700? 001236  MOV    #22$, $ESCAPE ;SET RETURN FOR CONTINUE TEST
6072 027000 001465          BR     50$        ;GO TO ERROR END

```

```

6073 027002 005237 002326      22$: INC BITCNT ;BUMP BIT COUNT
6074 027006 005300              DEC R0 ;LOOP UNTIL ALL 126
6075 027010 001357              BNE 5$ ;BITS ARE READ
6076 027012 012700 000177      MOV #127,R0 ;SET COUNT FOR REST OF SYNC
6077 027016 052737 100000 002244  BIS #RDGATE,E.MR1 ;SET EXPECTED MR1
6078 027024 004737 031616      6$: JSR PC,RDBIT ;GO CALL READ BIT
6079 027030 016237 050026 002204  MOV RKMR1(R2),T.MR1 ;GET MR1 FOR TESTING
6080 027036 023737 002244 002204  CMP E.MR1,T.MR1 ;TEST IF CORRECT
6081 027044 001404              BEQ 23$ ;YES - SKIP
6082 027046 012737 027056 001236  MOV #23$, $ESCAPE ;ELSE SET ESCAPE
6083 027054 000437              BR 50$ ;GO TO ERROR REPORT EXIT
6084
6085 027056 005237 002326      23$: INC BITCNT ;BUMP BIT COUNTER
6086 027062 005300              DEC R0 ;DEC BIT COUNT
6087 027064 001357              BNE 6$ ;LOOP UNTIL ALL 127 READ
6088
6089 027066 012737 000001 002320  MOV #1,PR.BIT ;SET SYNC BIT OF 1 AS NEXT BIT
6090 027074 004737 031616      JSR PC,RDBIT ;GO READ IT
6091 027100 016237 000026 002244  MOV RKMR1(R2),E.MR1 ;GET MR1 FOR TESTING
6092 027106 023737 002244 002204  CMP E.MR1,T.MR1 ;CHECK IF CORRECT
6093 027114 001404              BEQ 24$ ;YES - SKIP
6094 027116 012737 027126 001236  MOV #24$, $ESCAPE ;ELSE SET RETURN FOR CONTINUE TEST
6095 027124 000413              BR 50$
6096
6097 027126 005237 002326      24$: INC BITCNT ;BUMP BIT COUNT
6098
6099 027132 004437 032356      JSR R4,GETREG ;GET RK611 REGS
6100 027136 005724              TST (R4)+ ;BUMP TO NEXT ERROR RETURN
6101 027140 023737 002160 002220  CMP T.CS1,E.CS1 ;CHECK IF CORRECT
6102 027146 001235              BNE 2$ ;NO - SKIP
6103
6104 027150 005724              TST (R4)+ ;BUMP PAST ERROR RETURNS
6105 027152 000652              BR 3$ ;GO TO EXIT
6106
6107 027154 032777 001000 151756 50$: BIT #BIT9,$S,R ;TEST IF LOOP ON ERROR
6108 027162 001403              BEQ 51$ ;YES - SKIP
6109 027164 005037 001236      CLR $ESCAPE ;ELSE SET TO LOOP
6110 027170 000643              BR 3$
6111 027172 010446      51$: MOV R4,-(SP) ;PRESET STACK FOR RETURN
6112 027174 000204              RTS R4
6113      .SBTL_ DIAGNOSTIC READ DATA OR DIAGNOSTIC WRITE CHECK ROUTINE
6114      ;* THIS ROUTINE IS USED TO SIMULATE THE READ OR WRITE CHECK
6115      ;* OF DATA. THE ROUTINE WILL TRANSFER A FULL OR PARTIAL SECTOR.
6116      ;* THE FIRST WORD AFTER THE CALL SPECIFIES HOW MANY WORDS
6117      ;* ARE TO BE TRANSFERRED.
6118      ;*
6119      ;* IF LESS THAN A FULL SECTOR, THE ECC AT THE END
6120      ;* OF THE TRANSFER IS NOT CHECKED. THE ECC IS CHECKED AT THE TIME
6121      ;* EACH BIT IS TRANSFERED.
6122      ;*
6123      ;* EITHER 16 OR 18 BIT WORDS CAN BE SIMULATED
6124      ;* DEPENDING ON THE STATE OF CFMT BIT IN L.CS1. IF 18 BITS
6125      ;* ARE REQUIRED, BITS 16 AND 17 ARE ASSUMED TO BE ZERO.
6126      ;*
6127      ;* THE CONTENTS OF LOCATION ECPOSX MUST BE LOADED WITH THE ECC
6128      ;* COUNT EXPECTED AT THE END OF THE DATA TRANSFER. THIS

```

6129
6130
6131
6132
6133
6134
6135
6136
6137
6138
6139
6140
6141
6142
6143
6144
6145
6146
6147
6148
6149
6150
6151
6152
6153
6154
6155
6156
6157
6158
6159
6160
6161
6162
6163
6164
6165 027176
6166 027176
6167 027176 010046
6168 027200 010146
6169 027202 010346
6170 027204 010546
6171 027206 012400
6172 027210 010437 001216
6173 027214 005724
6174 027216 010437 001220
6175 027222 005724
6176 027224 010437 001210
6177 027230 005724
6178 027232 010437 001212
6179 027236 005724
6180 027240 010437 001222
6181 027244 005724
6182 027246 010437 001224
6183 027252 012703 050130
6184 027256 005037 002326

:* LOADING MUST OCCUR BEFORE THE ROUTINE IS CALLED.
:*
:* ANOTHER LOCATION, ECPATX, IS SET TO 0 IF THE ECC IS EXPECTED
:* TO BE CORRECT AND SET TO 1 IF AN ERROR IS BEING FORCED.
:*
:* A 3RD LOCATION, ECCSRC, SPECIFIES THE SOURCE OF THE ECC
:* WORDS TO BE USED IN CHECKING THE OPERATION. IF ECCSRC IS A
:* 1, THE ECC WORDS FED INTO THE DECODER ARE TAKEN FROM THE
:* LAST TWO LOCATIONS OF OBUFF, IF ECCSRC IS 0, THE COMPUTED
:* ECC WORDS FROM ECCHI AND ECCLO ARE USED.
:* CALL:
:* JSR R4,DREAD
:* LENGTH OF TRANSFER
:* ERROR 34 :MR1 IN ERROR READING DATA OR ECC
:* ERROR 35 :ECC ERROR READING DATA
:* ERROR 36 :CS1 ERROR AFTER READING DATA OR ECC
:* ERROR 41 :ECC REGISTER INCORRECT AFTER READ ECC
:* ERROR 42 :ERR IN ECC PAT CALCULATION
:* ERROR 43 :ERR IN ECC POS COUNTING
:*
:* OR
:*
:* JSR R4,DWRTCK
:* LENGTH OF TRANSFER
:* ERROR 53 :MR1 IN ERROR WRITE CHK OR ECC READ
:* ERROR 54 :ECC ERROR IN WRITE CHECK
:* ERROR 55 :CS1 ERROR AFTER IN WRT CHK OR ECC READ
:* ERROR 41 :ECC REGISTER INCORRECT AFTER READ ECC
:* ERROR 42 :ERR IN ECC PAT CALCULATION
:* ERROR 43 :ERR IN ECC POS COUNTING
:*
:* ROUTINES CALLED:
:* RDBIT
:* ECCGEN

DWRTCK:
DREAD:
MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R5,-(SP) ;;PUSH R5 ON STACK
MOV (R4)+,R0 ;;STORE WORD COUNT
MOV R4,\$TMP6 ;;STORE MR1 ERROR RETURN
TST (R4)+ ;;BUMP TO NEXT RETURN
MOV R4,\$TMP7 ;;STORE ECC ERROR RETURN
TST (R4)+ ;;BUMP TO NEXT ERROR RETURN
MOV R4,\$TMP3 ;;STORE CS1 ERROR RETURN
TST (R4)+ ;;BUMP TO NEXT ERROR RETURN
MOV R4,\$TMP4 ;;STORE ECC REG ERROR RETURN
TST (R4)+ ;;BUMP TO NEXT ERROR RETURN
MOV R4,\$TMP10 ;;STORE ECC PAT ERR RETURN
TST (R4)+ ;;BUMP TO NEXT ERROR RETURN
MOV R4,\$TMP11 ;;STORE ECC POS ERROR RETURN
MOV #OBUFF,R3 ;;GET ADDRESS OF BUFFER
CLR BITCNT ;;CLEAR BIT COUNTER

6185	027262	005037	002340		CLR	ECCHI	:CLEAR ECC COMPUTED LOCATIONS
6186	027266	005037	002342		CLR	ECCLC	
6187	027272	005037	001214		CLR	\$TMP5	:CLEAR FOR 18 BIT MODE SWITCH
6188	027276	032737	010000	002260	BIT	#CFMT,L.CS1	:TEST IF 18 BIT MODE
6189	027304	001412			BEQ	32\$:NO - SKIP
6190	027306	012737	011000	001226	MOV	#11000,\$TMP12	:ELSE SET UP BIT COUNTS FOR 18 BIT MODE
6191	027314	012737	011001	001230	MOV	#11001,\$TMP13	
6192	027322	012737	011041	001232	MOV	#11041,\$TMP14	
6193	027330	000411			BR	3\$	
6194	027332	012737	010000	001226	32\$: MOV	#10000,\$TMP12	:SET BIT COUNTS FOR 16 BIT MODE
6195	027340	012737	010001	001230	MOV	#10001,\$TMP13	
6196	027346	012737	010041	001232	MOV	#10041,\$TMP14	
6197							
6198	027354	012701	000001		3\$: MOV	#BIT0,R1	:PRESET BIT POINTER
6199	027360	012305			MOV	(R3)+,R5	:GET DATA WORD
6200							
6201	027362	013737	002320	002322	4\$: MOV	PR.BIT,M1.BIT	:SHIFT PRESENT INTO PREVIOUS
6202	027370	005037	002320		CLR	PR.BIT	:CLEAR PRESENT
6203	027374	030105			BIT	R1,R5	:TEST IF PRESENT BIT IS 0
6204	027376	001403			BEQ	1\$:YES - SKIP
6205	027400	012737	000001	002320	MOV	#1,PR.BIT	:ELSE INSERT A 1
6206							
6207	027406	004737	031616		1\$: JSR	PC,RDBIT	:GO READ BIT
6208	027412	016237	000026	002204	MOV	RKMR1(R2),T.MR1	:GET MR1
6209	027420	023737	002204	002244	CMP	T.MR1,E.MR1	:CHECK IF CORRECT
6210	027426	001402			BEQ	43\$:YES - SKIP
6211	027430	000137	030262		JMP	9\$	
6212							
6213	027434	005737	002326		43\$: TST	BITCNT	:TEST IF FIRST BIT OF DATA
6214	027440	001413			BEQ	44\$:YES - SKIP ECC GEN
6215	027442	004737	030462		JSR	PC,ECCGEN	:GO GENERATE ECC
6216	027446	016237	000032	002214	MOV	RKECPT(R2),T.ECPT	:TEST IF ECC CORRECT
6217	027454	023737	002214	002214	CMP	T.ECPT,E.ECPT	
6218	027462	001402			BEQ	44\$	
6219	027464	000137	030276		JMP	10\$	
6220							
6221	027470	005237	002326		44\$: INC	BITCNT	:BUMP BIT COUNTER
6222	027474	005701			TST	R1	:TEST IF THIS WORD IS DONE
6223	027476	100402			BMI	2\$:YES - SKIP
6224	027500	006301			ASL	R1	:ELSE SHIFT BIT POINTER
6225	027502	000727			BR	4\$	
6226							
6227	027504	032737	010000	002260	2\$: BIT	#CFMT,L.CS1	:TEST IF 18 BIT MODE
6228	027512	001412			BEQ	30\$:NO - SKIP
6229	027514	005737	001214		TST	\$TMP5	:TEST IF PROCESSING BIT 16 OR 17 DONE
6230	027520	001007			BNE	30\$:YES - SKIP
6231	027522	012701	040000		MOV	#BIT14,R1	:SET POINTER FOR 2 MORE BITS
6232	027526	013705	002354		MOV	HIBITS,R5	:INSERT HI ORDER BITS
6233	027532	005137	001214		COM	\$TMP5	:SET 18 BIT SWITCH
6234	027536	000711			BR	4\$	
6235	027540	005037	001214		30\$: CLR	\$TMP5	:CLEAR 18 BIT SWITCH
6236							
6237	027544	005300			31\$: DEC	R0	:DECREMENT WORD COUNT
6238	027546	001302			BNE	3\$:IF NOT ZERO - LOOP
6239							
6240	027550	023737	002326	001226	CMP	BITCNT,\$TMP12	:ELSE TEST IF FULL SECTOR WAS READ

```

6241 027556 001402          BEQ      33$      :YES - SKIP
6242 027560 000137 030234   JMP      8$      :ELSE JUMP TO EXIT - DO NOT CHECK
6243                                     :      ECC WORDS
6244
6245 027564 013737 002320 002322 33$:   MOV      PR.BIT,M1.BIT :MOVE LAST DATA BIT FOR ECC GEN
6246 027572 004737 030462     JSR      PC,ECCGEN   :GENERATE ECC
6247 027576 005037 001214     CLR      $TMP5      :CLEAR FOR PASS COUNTING
6248 027602 005737 002344     TST      ECCSRC     :TEST WHERE ECC WORDS ARE
6249 027606 001006          BNE      36$      :THEY ARE IN BUFFER - SKIP
6250 027610 013737 002340 051130   MOV      ECCHI,OBUFF+1000 ;ELSE MOVE THEM INTO BUFFER
6251 027616 013737 002342 051132   MOV      ECCLO,OBUFF+1002
6252
6253 027624 012305          36$:   MOV      (R3)+,R5    :GET ECC FROM BUFFER
6254 027626 012701 000001     MOV      #BIT0,R1   :SET BIT POINTER
6255
6256 027632 013737 002320 002322 5$:   MOV      PR.BIT,M1.BIT :SHIFT PRESENT TO PREVIOUS
6257 027640 005037 002320     CLR      PR.BIT    :AND SET UP PRESENT
6258 027644 030105          BIT      R1,R5
6259 027646 001403          BEQ      34$
6260 027650 012737 000001 002320   MOV      #1,PR.BIT
6261 027656 004737 031676     JSR      PC,RDBIT   :GO READ BIT
6262 027662 016237 000026 002204   MOV      RKMRI(R2),T.MR1 :GET MR1
6263 027670 023737 002204 002244   CMP      T.MR1,E.MR1 :CHECK IF CORRECT
6264 027676 001402          BEQ      38$      :YES - SKIP
6265 027700 000137 030312     JMP      11$      :ELSE JUMP TO REPORT EXIT
6266 027704 023737 002326 001226 38$:   CMP      BITCNT,$TMP12 :TEST IF FIRST ECC BIT
6267 027712 001402          BEQ      49$      :YES - SKIP ECC GEN (ALREADY DONE)
6268 027714 004737 030462     JSR      PC,ECCGEN   :GENERATE ECC FOR BIT
6269 027720 016237 000032 002214 4$:   MOV      RKECPT(R2),T.ECPT ;GET PATTERN
6270 027726 023737 002214 002254   CMP      T.ECPT,E.ECPT :TEST IF ECC PAT CORRECT
6271 027734 001402          BEQ      45$      :YES SKIP
6272 027736 000137 030276     JMP      10$      :ELSE CO REPORT ERROR
6273
6274 027742 005237 002326          45$:   INC      BITCNT     :BUMP BIT COUNT
6275 027746 023737 002326 001230   CMP      BITCNT,$TMP13 :TEST IF 1ST BIT OF ECC
6276 027754 001003          BNE      46$      :NO - SKIP
6277 027756 042737 020000 002244   BIC      #ECCW,E.MR1 :ELSE CLR ECCW FOR ECC READ
6278 027764 023737 002326 001232 46$:   CMP      BITCNT,$TMP14 :TEST IF 1ST BIT OF POSTAMBLE
6279 027772 001027          BNE      47$      :NO - SKIP
6280 027774 052737 020000 002244   BIS      #ECCW,E.MR1 :ELSE SET ECCW FOR ECC DONE
6281 030002 042737 100000 002244   BIC      #RDGATE,E.MR1 :CLEAR READ GATE
6282
6283 030010 016237 000032 002214   MOV      RKECPT(R2),T.ECPT ;STORE ACTUAL PATTERN
6284 030016 016237 000030 002212   MOV      RKECPS(R2),T.ECPS ;STORE ACTUAL POSITION
6285 030024 013737 002350 002252   MOV      ECPOSX,E.ECPS :GET EXPECTED POSITION FOR REPORT
6286 030032 023737 002214 002254   CMP      T.ECPT,E.ECPT :CHECK IF CORRECT
6287 030040 001163          BNE      20$      :NO SKIP
6288 030042 023737 002212 002252   CMP      T.ECPS,E.ECPS :CHECK IF POSITION CORRECT
6289 030050 001157          BNE      20$      :NO - SKIP
6290
6291 030052 023737 002326 001232 4$:   CMP      BITCNT,$TMP14 :TEST IF ECC DONE
6292 030060 002436          BLT      48$      :NO - SKIP
6293 030062 005737 002346          TST      ECPATX    :TEST IF EXPECTED ECC IS 0
6294                                     :      (NO ERROR)
6295 030066 001433          BEQ      48$      :YES - SKIP
6296 030070 023737 001232 002326   CMP      $TMP14,BITCNT :TEST IF FIRST POSTAMBLE BIT

```

```

6297 030076 001402          BEQ      52$          :YES - SKIP
6298 030100 005237 002350      INC      ECPOSX       :ELSE BUMP POS COUNT EXPECTED
6299 030104 016237 000032 002214 52$:  MOV     RKECPT(R2),T.ECPT :GET PATTERN
6300 030112 016237 000030 002212      MOV     RKECPS(R2),T.ECPS :GET POSITION
6301 030120 013737 002350 002252      MOV     ECPOSX,E.ECPS    :GET EXPECTED POSITION
6302 030126 023737 002214 002254      CMP     T.ECPT,E.ECPT   :TEST IF PATTERN CORRECT
6303 030134 001402          BEQ      51$          :YES - SKIP
6304 030136 000137 030326      JMP     22$          :ELSE GO REPORT
6305 030142 023737 002212 002252 51$:  CMP     T.ECPS,E.ECPS   :CHECK IF POSITION CORRECT
6306 030150 001402          BEQ      48$          :YES - SKIP
6307 030152 000137 030342      JMP     23$          :ELSE GO REPORT
6308
6309 030156 005701          48$:  TST     R1           :TEST IF WORD DONE
6310 030160 100403          BMI     6$           :YES SKIP
6311 030162 006301          ASL     R1           :ELSE SHIFT BIT POINTER
6312 030164 000137 027632      JMP     5$           :AND LOOP
6313
6314 030170 005737 001214          6$:  TST     $TMP5        :TEST PASS COUNT. IF NOT 0
6315 030174 001004          BNE     7$           : (NOT THE FIRST ECC WORD JUST WRITTE)
6316 030176 005237 001214          INC     $TMP5        :BUMP PASS COUNT
6317 030202 000137 027624          JMP     36$
6318
6319 030206 023727 001214 000001 7$:  CMP     $TMP5,#1     :TEST PASS COUNT. IF NOT 1 (NOT THE
6320 030214 001007          BNE     8$           :2ND ECC WORD JUST READ) - SKIP
6321 030216 005005          CLR     R5           :CLEAR WORD (POSTAMBLE)
6322 030220 012701 000001          MOV     #BIT0,R1    :SET BIT POINTER
6323 030224 005237 001214          INC     $TMP5        :BUMP PASS COUNT
6324 030230 000137 027632          JMP     5$
6325
6326 030234 004437 032356          8$:  JSR     R4,GETREG    :GET RK611 REGS
6327 030240 013737 002260 002220      MOV     L.CS1,E.CS1  :SET EXPECTED CS1
6328 030246 023737 002220 002160      CMP     E.CS1,T.CS1  :CHECK IF CORRECT
6329 030254 001060          BNE     21$          :NO - SKIP
6330 030256 005724          TST     (R4)+       :ELSE BUMP RETURN POINTER TO NO ERR
6331 030260 000446          BR      13$
6332
6333 030262 013704 001216          9$:  MOV     $TMP6,R4    :SET RETURN FOR MR1 ERR.
6334 030266 012737 027434 001236      MOV     #43$, $ESCAPE :SET RETURN TO CONTINUE TEST
6335 030274 000427          BR      12$
6336
6337 030276 013704 001220          10$: MOV     $TMP7,R4    :SET RETURN FOR ECC ERROR
6338 030302 012737 027470 001236      MOV     #44$, $ESCAPE :SET RETURN TO CONTINUE TEST
6339 030310 000421          BR      12$
6340
6341 030312 013704 001216          11$: MOV     $TMP6,R4    :SET RETURN FOR MR1 ERROR
6342 030316 012737 027742 001236      MOV     #45$, $ESCAPE :SET RETURN TO CONTINUE TEST
6343 030324 000413          BR      12$
6344
6345 030326 013704 001222          22$: MOV     $TMP10,R4   :GET RETURN FOR PATTERN ERROR
6346 030332 012737 030142 001236      MOV     #51$, $ESCAPE :SET RETURN TO CONTINUE TEST
6347 030340 000405          BR      12$
6348
6349 030342 013704 001224          23$: MOV     $TMP11,R4   :RET RETURN FOR POSITION ERROR
6350 030346 012737 030156 001236      MOV     #48$, $ESCAPE :SET RETURN TO CONTINUE TEST

```

```
6353
6354
6355 030354 013746 001224 12$: MOV $TMP11,-(SP) ;PREP STACK FOR RETURN
6356 030360 032777 001000 150552 BIT #BIT9,@SWR ;TEST IF LOOP ON ERROR
6357 030366 001407 BEQ 14$ ;NO SKIP
6358
6359 030370 005726 TST (SP)+ ;REMOVE LAST ENTRY FROM STACK
6360 030372 005037 U01236 CLR $ESCAPE ;NO LOOP - SLEAR ESCAPE
6361
6362 030376 13$:
6363 030376 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
6364 030400 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
6365 030402 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
6366 030404 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
6367 030406 000204 14$: RTS R4
6368
6369 030410 013704 001212 20$: MOV $TMP4,R4 ;GET RETURN FOR ECC ERROR
6370 030414 000402 BR 15$
6371 030416 013704 001210 21$: MOV $TMP3,R4 ;GET CS1 ERROR RETURN
6372 030422 005037 001236 5$: CLR $ESCAPE ;TEST IF LOOP ON ERROR
6373 030426 032777 001000 150504 BIT #BIT9,@SWR ;TEST IF LOOP ON ERROR
6374 030434 001360 BNE 13$ ;YES - SKIP
6375
6376 030436 013700 001254 MOV $TESTN,R0 ;SET REGISTERS TO LOOP ON ERROR OR
6377 030442 006300 ASL R0 ;ESCAPE TO NEXT TEST.
6378 030444 016037 033700 001236 MOV $SW08TB(R0),$ESCAPE ;SET UP TO ESCAPE TO NEXT TEST
6379
6380 030452 162737 000002 001236 SUB #2,$ESCAPE ;ADJUST RETURN
6381 030460 000746 BR 13$ ;GO TO EXIT
6382
6383 .SBTTL GENERATE ECC WORD
6384 :* EACH TIME THIS ROUTINE IS CALLED THE ECC IS CALCULATED FOR THE
6385 :* NEXT BIT TO BE READ OR WRITTEN AND THE BITS PREVIOUSLY
6386 :* READ OR WRITTEN (STORED IN ECCHI AND ECCLO). THE RESULTS OF
6387 :* THE CALCULATION ARE PLACED IN ECCHI AND ECCLO. THE LOW ORDER
6388 :* 11 BITS OF ECCHI ARE PLACED IN E.ECPT FOR EASY COMPARISON
6389 :* TO RKECPT.
6390 :*
6391 :* CALL: JSR PC,ECCGEN
6392 :* RETURN: RTS R4
6393 ECCGEN: MOV R0,-(SP) ;STORE R0
6394 030464 012700 002316 MOV #PI.BIT,R0 ;GET ADDRESS OF NEXT BIT
6395 030470 032737 000002 002260 BIT #BIT11,L.CS1 ;CHECK IF WRITING
6396 030476 001002 BNE 12$ ;YES - SKIP
6397 030500 012700 002322 MOV #M1.BIT,R0 ;ELSE CHANGE TO PRESENT BIT
6398
6399 030504 005710 12$: TST (R0) ;CHECK IF NEXT BIT 0
6400 030506 001410 BEQ 3$ ;YES - SKIP
6401 030510 032737 000001 002340 BIT #1,ECCHI ;NO, CHECK IF BIT 31 = 1
6402 030516 001410 BEQ 5$ ;NO, SET ECC XORED BIT
6403 030520 005037 002336 1$: CLR ECCXOR ;CLEAR ECC XORED BIT
6404 030524 000241 CLC ;CLEAR CARRY FLOP
6405 030526 000410 BR 7$ ;SHIFT IN ECC BIT
6406
6407 030530 032737 000001 002340 3$: BIT #1,ECCHI ;CHECK IF BIT 31 = 1
6408 030536 001770 BEQ 1$ ;NO, CLEAR ECC XORED BIT
```

```

6409 030540 012737 000001 002336 5$: MOV #1,ECCXOR ;SET ECC XOR
6410 030546 000261 SEC ;SET CARRY FLCP
6411 030550 006037 002342 7$: ROR ECCLO
6412 030554 006037 002340 ROR ECCHI
6413 030560 005737 002336 TST ECCXOR ;CHECK IF XOR NEEDED
6414 030564 001422 BEQ 10$ ;NO, RETURN
6415 030566 012746 020020 MOV #BIT13:BIT4,-(SP) ;DO XOR OF ECC BITS
6416 030572 043716 002342 BIC ECCLO,(SP) ; 0, 2, 11, 21, 23
6417 030576 042737 020020 002342 BIC #BIT13:BIT4,ECCLO
6418 030604 052637 002342 BIS (SP)+,ECCLO
6419 030610 012746 002400 MOV #BIT10:BIT8,-(SP)
6420 030614 043716 002340 BIC ECCHI,(SP)
6421 030620 042737 002400 002340 BIC #BIT10:BIT8,ECCHI
6422 030626 052637 002340 BIS (SP)+,ECCHI
6423 030632 013737 002340 002254 10$: MOV ECCHI,E.ECPT ;STORE ECC PATTERN
6424 030640 042737 174000 002254 BIC #174000,E.ECPT ;MASK UNSEEN BITS
6425 030646 012600 MOV (SP)+,R0 ;RESTORE R0
6426 030650 000207 RTS PC ;RETURN
6427
6428 .SBTTL SIMULATE ONE BIT OF WRITE DATA IN MAINTANENCE MODE
6429 :* ONE BIT OF DATA IS WRITE SIMULATED. THE CONTENTS OF
6430 :* RKMRI IS COMPUTED (PRECOMP ADVANCE AND DELAY, MAINTANENCE
6431 :* ENCODED WRITE DATA) FOR THE PROPER TRANSITION OF THE
6432 :* MAINTANENCE CLOCK AND CHECKED AT EACH TRANSITION. IF MR1
6433 :* IS INCORRECT AT ANY TRANSITION, AN ERROR IS SET UP TO
6434 :* CONTAIN A MESSAGE THAT IDENTIFIES THE TRANSITION WHEN THE
6435 :* ERROR OCCURRED.
6436 :*
6437 :* CALL: JSR PC,WRTBIT
6438 :* RETURN: RTS PC+2 FOR NO ERROR RETURN
6439 :* RTS PC FOR ERROR IN MR1
6440 030652 052737 002400 002244 WRTBIT: BIS #MCLK!MEWD,E.MR1 ;CREATE EXPECTED MAINT. REG. 1
6441 030660 012762 000440 000C26 MOV #DMD!MCLK,RKMR1(R2) ;PROVIDE 1ST UPWARD TRANSITION
6442 030666 016237 000026 002204 MOV RKMR1(R2),T.MR1 ;STORE MAINT. REG. 1
6443 030674 023737 002244 002204 CMP E.MR1,T.MR1 ;CHECK IF MAINT REG 1 CORRECT
6444 030702 001416 BEQ 3$ ;YES, PROVIDE DOWNWARD TRANSITION
6445 030704 012737 030724 001236 MOV #1$,SESCAPE ;LOAD ESCAPE FOR LOOP ON ERROR
6446 030712 012737 044136 001512 MOV #EMW1,EMW+2 ;LOAD ERROR MESSAGE
6447 030720 011646 MOV (SP),-(SP) ;SAVE RETURN
6448 030722 000207 RTS PC ;MR1 INCORRECT ON UPWARD TRANSITION
6449
6450 030724 032777 001000 150206 1$: BIT #SW9,@SWR ;CHECK IF LOOP ON ERROR
6451 030732 001402 BEQ 3$ ;NO, CONTINUE
6452 030734 000137 031606 JMP 63$ ;YES, LOOP ON ERROR
6453
6454 030740 042737 014400 002244 3$: BIC #MCLK!PCA PCD,E.MR1 ;INITIALIZE MAINTANENCE REG. 1
6455 030746 052737 042000 002244 BIS #MEWD!WRTGAT,E.MR1
6456 030754 005737 002320 TST PR.BIT ;CHECK IF ONE
6457 030760 001152 BNE 20$ ;YES, SIMULATE ONE
6458 030762 005737 002322 TST M1.BIT ;CHECK IF PREVIOUS ONE
6459 030766 001023 BNE 10$ ;YES, NO TRANSITION
6460 030770 042737 002000 002244 BIC #MEWD,E.MR1 ;INDICATE TRANSITION
6461 030776 005737 002316 TST P1.BIT ;CHECK IF NEXT BIT = 1
6462 031002 001007 BNE 5$ ;YES, CHECK FOR PRECOMP ADVANCE
6463 031004 005737 002324 TST M2.BIT ;CHECK FOR PRECOMP. ADVANCE
6464 031010 001412 BEQ 10$ ;NO, CLOCK IN ZERO
  
```

6465	031012	052737	010000	002244		BIS	#PCD,E.MR1	:SET PRECOMP. DELATY
6466	031020	000406				BR	10\$:CLOCK IN ZERO
6467								
6468	031022	005737	002324		5\$:	TST	M2.BIT	:CHECK FOR PRECOMP. ADVANCE
6469	031026	001003				BNE	10\$:CLOCK IN ZERO
6470	031030	052737	004000	002244		BIS	#PCA,E.MR1	:SET PRECOMP. ADVANCE
6471	031036	012762	000040	000026	10\$:	MOV	#DMD,RKMR1(R2)	:CLOCK IN DATA BIT
6472	031044	016237	000026	002204		MOV	RKMR1(R2),T.MR1	:STORE MR1
6473	031052	023737	002204	002244		COMP	T.MR1,E.MR1	:CHECK IF MR1 CORRECT
6474	031060	001416				BEQ	12\$:YES, CONTINUE
6475	031062	012737	031102	001236		MOV	#11\$, \$ESCAPE	:LOAD ESCAPE FOR LOOP ON ERROR
6476	031070	012737	044224	001512		MOV	#EMW2,EMW+2	:LOAD ERROR MESSAGE
6477	031076	011646				MOV	(SP),-(SP)	:SAVE RETURN
6478	031100	000207				RTS	PC	:MR1 INCORRECT
6479								
6480	031102	032777	001000	150030	11\$:	BIT	#SW9,@SWR	:CHECK IF LOOP ON ERROR
6481	031110	001402				BEQ	12\$:NO, CONTINUE
6482	031112	000137	031606			JMP	63\$:YES, LOOP ON ERROR
6483								
6484	031116	052737	002400	002244	12\$:	BIS	#MCLK!MFWD,E.MR1	:CREATE EXPECTED MAINT REG 1
6485	031124	012762	000440	000026		MOV	#DMD!MCLK,RKMR1(R2)	:PROVIDE 2ND UPWARD TRANSITION
6486	031132	016237	000026	002204		MOV	RKMR1(R2),T.MR1	:STORE MAINT REG. 1
6487	031140	023737	002244	002204		COMP	E.MR1,T.MR1	:CHECK IF MAINT REG. 1 CORRECT
6488	031146	001416				BEQ	15\$:YES, CONTINUE
6489	031150	012737	031170	001236		MOV	#13\$, \$ESCAPE	:LOAD ESCAPE FOR LOOP ON ERROR
6490	031156	012737	044314	001512		MOV	#EMW3,EMW+2	:LOAD ERROR MESSAGE
6491	031164	011646				MOV	(SP),-(SP)	:SAVE RETURN
6492	031166	000207				RTS	PC	:MR1 INCORRECT
6493								
6494	031170	032777	001000	147742	3\$:	BIT	#SW9,@SWR	:CHECK IF LOOP ON ERROR
6495	031176	001402				BEQ	15\$:NO, CONTINUE
6496	031200	000137	031606			JMP	63\$:YES, LOOP ON ERROR
6497								
6498	031204	052737	002000	002244	15\$:	BIS	#MEWD,E.MR1	:RESET TRANSITION INDICATION
6499	031212	042737	000400	002244		BIC	#MCLK,E.MR1	
6500	031220	012762	000040	000026		MOV	#DMD,RKMR1(R2)	:SUPPLY LAST PART OF DATA
6501	031226	016237	000026	002204		MOV	RKMR1(R2),T.MR1	:STORE MR1
6502	031234	023737	002204	002244		COMP	T.MR1,E.MR1	:CHECK IF MR1 CORRECT
6503	031242	001414				BEQ	18\$:YES, RETURN
6504	031244	012737	031264	001236		MOV	#17\$, \$ESCAPE	:LOAD ESCAPE FOR LOOP ON ERROR
6505	031252	012737	044402	001512		MOV	#EMW4,EMW+2	:LOAD ERROR MESSAGE
6506	031260	011646				MOV	(SP),-(SP)	:SAVE RETURN
6507	031262	000207				RTS	PC	:MR1 INCORRECT
6508								
6509	031264	032777	001000	147646	17\$:	BIT	#SW9,@SWR	:CHECK IF LOOP ON ERROR
6510	031272	001145				BNE	63\$:YES, LOOP ON ERROR
6511	031274	005037	001236		18\$:	CLR	\$ESCAPE	:CLEAR ESCAPE
6512	031300	062716	000002			ADD	#2,(SP)	:ADJUST RETURN
6513	031304	000207				RTS	PC	:RETURN
6514								
6515	031306	005737	002316		20\$:	TST	P1.BIT	:CHECK IN NEXT BIT A ONE
6516	031312	001007				BNE	30\$:YES, CHECK IF PRECOMP DELAY
6517	031314	005737	002322			TST	M1.BIT	:CHECK FOR PRECOMP ADVANCE
6518	031320	001415				BEQ	40\$:NO, CHECK FO MR1
6519	031322	052737	004000	002244		BIS	#PCA,E.MR1	:SET PRECOMP. ADVZNCE
6520	031330	000411				BR	40\$:CHECK MR1

```
6521
6522 031332 042737 000400 002244 30$: BIC #MCLK,E.MR1 ;RESET MAINT CLOCK IN EXPECTED MR1
6523 031340 005737 002322 TST M1.BIT ;CHECK FOR PRECOMP DELAY
6524 031344 001003 BNE 40$ ;NO, CHECK MR1
6525 031346 052737 010000 002244 BIS #PCD,E.MR1 ;SET SET PRECOMP DELAY
6526 031354 012762 000040 000026 40$: MOV #DMD,RKMR1(R2) ;CLOCK IN DATA BIT
6527 031362 016237 000026 002204 MOV RKMR1(R2),T.MR1 ;STORE MR1
6528 031370 023737 002204 002244 CMP T.MR1,E.MR1 ;CHECK MR1 CORRECT
6529 031376 001414 BEQ 42$ ;YES, CLOCK IN REST OF BIT
6530 031400 012737 031420 001236 MOV #41$, $ESCAPE ;LOAD ESCAPE FOR LOOP ON ERROR
6531 031406 012737 044224 001512 MOV #EMW2,EMW+2 ;LOAD ERROR MESSAGE
6532 031414 011646 MOV (SP),-(SP) ;SAVE RETURN
6533 031416 000207 RTS PC ;MR1 INCORRECT
6534
6535 031420 032777 001000 147512 41$: BIT #SW9,@SWR ;CHECK IF LOOP ON ERROR
6536 031426 001067 BNE 63$ ;YES, LOOP ON ERROR
6537 031430 052737 000400 002244 42$: BIS #MCLK,E.MR1 ;CHREATE EXPECTED MAINT. REG. 1
6538 031436 012762 000440 000026 MOV #DMD!MCLK,RKMR1(R2) ;PROVIDE 2ND UPWARD TRANSITION
6539 031444 016237 000026 002204 MOV RKMR1(R2),T.MR1 ;STORE MAINT REG 1
6540 031452 023737 002244 002204 CMP E.MR1,T.MR1 ;CHECK IF MAINT REG 1 CORRECT
6541 031460 001414 BFO 45$ ;YES, CONTINUE
6542 031462 012737 031502 001236 MOV #43$, $ESCAPE ;LOAD ESCAPE
6543 031470 012737 044314 001512 MOV #EMW3,EMW+2 ;LOAD ERROR MESSAGE
6544 031476 011646 MOV (SP),-(SP) ;SAVE RETURN
6545 031500 000207 RTS PC ;MR1 INCORRECT
6546
6547 031502 032777 001000 147430 43$: BIT #SW9,@SWR ;CHECK IF LOOP ON ERROR
6548 031510 001036 BNE 63$ ;YES, LOOP ON ERROR
6549 031512 042737 002400 002244 45$: BIC #MEWD!MCLK,E.MR1 ;SET TRANSITION
6550 031520 012762 000040 000026 MOV #DMD,RKMR1(R2) ;CLOCK TRANSITION
6551 031526 016237 000026 002204 MOV RKMR1(R2),T.MR1 ;STORE MR1
6552 031534 023737 002204 002244 CMP T.MR1,E.MR1 ;CHECK IF MR1 CORRECT
6553 031542 001414 BEQ 50$ ;YES, RETURN
6554 031544 012737 031564 001236 MOV #47$, $ESCAPE ;LOAD ESCAPE FOR LOOP ON ERROR
6555 031552 012737 044402 001512 MOV #EMW4,EMW+2 ;LOAD ERROR MESSAGE
6556 031560 011646 MOV (SP),-(SP) ;SAVE RETURN
6557 031562 000207 RTS PC ;MR1 INCORRECT
6558
6559 031564 032777 001000 147346 47$: BIT #SW9,@SWR ;CHECK IF LOOP ON ERROR
6560 031572 001005 BNE 63$ ;YES, LOOP ON ERROR
6561 031574 005037 001236 50$: CLR $ESCAPE ;CLEAR ESCAPE
6562 031600 062716 000002 ADD #?,(SP) ;ADJUST RETURN
6563 031604 000207 RTS PC ;RETURN
6564
6565 031606 012706 001100 63$: MOV #STACK,SP ;FORCE STACK
6566 031612 000177 147272 JMP @SLPERR ;LOOP ON ERROR
6567
6568 .SBTTL SIMULATE ONE BIT OR READ DATA IN MAINTANENCE MODE
6569 :* ONE BIT OF DATA IS READ SIMULATED. NO CHECKING IS
6570 :* DONE BY THIS ROUTINE.
6571 :*
6572 :* CALL: JSR PC,RDBIT
6573 :* RETURN: RTS PC
6574 031616 005737 002320 RDBIT: TST PR.BIT ;CHECK IF ONE
6575 031622 001024 BNE 10$ ;YES, SIMULATE ONE
6576 031624 005737 002322 TST M1.BIT ;CHECK IF PREVIOUS ONE
```

```

6577 031630 001404          BEQ      4$          ;NO, INSERT TRANSITION
6578 031632 012762 000440 000026  MOV     #DMD!MCLK,RKMR1(R2) ;YES, DO NOT INSERT TRANSITION
6579 031640 000403          BR       5$          ;CLOCK IN ZERO
6580
6581 031642 012762 001440 000026 4$: MOV     #DMD!MCLK!MERD,RKMR1(R2) ;INSERT TRANSITION
6582 031650 012762 000040 000026 5$: MOV     #DMD,RKMR1(R2) ;CLOCK IN ZERO
6583 031656 012762 000440 000026  MOV     #DMD!MCLK,RKMR1(R2)
6584 031664 012762 000040 000026  MOV     #DMD,RKMR1(R2)
6585 031672 000207          RTS      PC          ;RETURN
6586
6587 031674 012762 000440 000026 10$: MOV     #DMD!MCLK,RKMR1(R2) ;CLOCK IN ONE
6588 031702 012762 000040 000026  MOV     #DMD,RKMR1(R2)
6589 031710 012762 001440 000026  MOV     #DMD!MCLK!MERD,RKMR1(R2)
6590 031716 012762 000040 000026  MOV     #DMD,RKMR1(R2)
6591 031724 000207          RTS      PC          ;RETURN
6592
6593          .SBTTL MEMORY CHECK ENABLE TRAP
6594          ;* IF THE PROCESSOR IN USE HAS MEMORY PARITY OPTION,
6595          ;* IT WILL BE ENABLED. THIS ROUTINE WILL PROCESS TRAPS CAUSED
6596          ;* BY MEMORY PARITY ERRORS.
6597 031726 012737 031742 001236 MEMERR: MOV     #10$, $ESCAPE ;LOAD ESCAPE
6598 031734 011637 002310          MOV     (SP),TRAPPC ;STORE PC
6599 031740 104001          ERROR   1 ;REPORT MEM PARITY ERROR
6600 031742 005037 001236 10$: CLR     $ESCAPE ;CLEAR ESCAPE
6601 031746 032777 001000 147164 BIT     #SW9,@SWR ;CHECK IF LOOP ON ERROR
6602 031754 001001          BNE    15$          ;YES, FORCE STACK AND TRY AGAIN
6603 031756 000002          RTI     ;NO, RETURN
6604
6605 031760 012706 001100 15$: MOV     #STACK,SP ;INITIALIZE STACK
6606 031764 000177 147120          JMP     @ $LPERR ;LOOP ON ERROR
6607
6608          .SBTTL CLEAR INPUT BUFFER
6609 031770 012700 047124 CLRIBF: MOV     #IBUFF,R0 ;GET BUFFER POINTER
6610 031774 010146          MOV     R1,-(SP) ;STORE R1
6611 031776 012701 000400          MOV     #400,R1 ;SET COUNT FOR CLEAR
6612 032002 005020 1$: CLR     (R0)+ ;CLEAR BUFFER WORD
6613 032004 005301          DEC     R1 ;DEC COUNT
6614 032006 001375          BNE    1$          ;LOOP UNTIL COUNT IS ZERO
6615 032010 012601          MOV     (SP)+,R1 ;RESTORE R1
6616 032012 000204          RTS     R4 ;RETURN
6617
6618          .SBTTL BUILD DATA BUFFER
6619          ;* THE PATTERN SPECIFIED IN THE CALL IS LOADED INTO THE DATA
6620          ;* BUFFER. THE ENTIRE BUFFER IS ALWAYS LOADED (400(8) WORDS).
6620 032014          BLDDAT:
6621 032014 010046          MOV     R0,-(SP) ;:PUSH R0 ON STACK
6622 032016 010146          MOV     R1,-(SP) ;:PUSH R1 ON STACK
6623 032020 010346          MOV     R3,-(SP) ;:PUSH R3 ON STACK
6624 032022 010546          MOV     R5,-(SP) ;:PUSH R5 ON STACK
6625 032024 012700 050130          MOV     #OBUFF,R0 ;LOAD R0 TO POINT TO BUFFER
6626 032030 012701 000400          MOV     #400,R1 ;SET DATA LENGTH
6627 032034 021427 000007          CMP     (R4),#1 ;TEST IF PATTERN 1(ALL ZERO)
6628 032040 001004          BNE    2$          ;NO - SKIP
6629 032042 005020 1$: CLR     (R0)+ ;ELSE CLEAR OBUFF
6630 032044 005301          DEC     R1
6631 032046 001375          BNE    1$
6632 032050 000467          BR     13$

```



```
6633 032052 021427 000007 2$: CMP (R4),#7 ;TEST IF PAT 7 (ALL ONES)
6634 032056 001006 BNE 4$ ;NO SKIP
6635 032060 012703 177777 MOV #177777,R3 ;ELSE SET BUFFER TO ALL ONES
6636 032064 010320 3$: MOV R3,(R0)+
6637 032066 005301 DEC R1
6638 032070 001375 BNE 3$
6639 032072 000450 BR 13$
6640 032074 021427 000006 4$: (MP (R4),#6 ;TEST IF PAT 6 (COMPOSIT ROTATING)
6641 032100 001003 BNE 5$ ;NO - SKIP
6642 032102 012705 047064 MOV #PAT6,R5 ;ELSE GET ADDRESS OF PATTERN 6
6643 032106 000427 BR 10$ ;GO LOAD BUFFER
6644 032110 021427 000005 5$: (MP (R4),#5 ;TEST IF PATTERN 5 (ALT 1 & C)
6645 032114 001006 BNE 7$ ;NO SKIP
6646 032116 012703 125252 MOV #125252,R3 ;ELSE LOAD OBUFF WITH PAT 5
6647 032122 010320 6$: MOV R3,(R0)+
6648 032124 005301 DEC R1
6649 032126 001375 BNE 6$
6650 032130 000431 BR 13$
6651 032132 021427 000004 7$: CMP (R4),#4 ;TEST IF PATTERN 4 (HI-LO FREQ MIX)
6652 032136 001003 BNE 8$ ;NO - SKIP
6653 032140 012705 047024 MOV #PAT4,R5 ;LOAD POINTER WITH ADD OF PAT 4
6654 032144 000410 BR 10$
6655 032146 021427 000003 8$: CMP (R4),#3 ;TEST IF PAT 3 (MAX PRECOMP PHASE MIX)
6656 032152 001003 BNE 9$ ;NO - SKIP
6657 032154 012705 046764 MOV #PAT3,R5 ;LOAD POINTER WITH ADD OF PAT 3
6658 032160 000402 BR 10$ ;GO LOAD BUFFER
6659 032162 012705 046724 9$: MOV #PAT2,R5 ;GET ADDRESS OF PAT 2
6660 032166 012703 000020 10$: MOV #16.,R3 ;SET PATTERN LENGTH COUNT
6661 032172 012520 11$: MOV (R5)+,(R0)+ ;MOV PATTERN INTO OBUFF
6662 032174 005301 DEC R1 ;DEC COUNTERS
6663 032176 005303 DEC R3
6664 032200 001374 BNE 11$ ;LOOP UNTIL PATTERN IS MOVED
6665 032202 012705 050130 MOV #OBUFF,R5 ;SET R5 TO START OF BUFFFF
6666 032206 012520 12$: MOV (R5)+,(R0)+ ;REPEAT PATTERN THROUGH BUFFER
6667 032210 005301 DEC R1
6668 032212 001375 BNE 12$
6669 032214 005724 13$: TST (R4)+ ;BUMP PAST PARAMETER
6670 032216 012605 MOV (SP)+,R5 ;;POP STACK INTO R5
6671 032220 012603 MOV (SP)+,R3 ;;POP STACK INTO R3
6672 032222 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
6673 032224 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
6674 032226 000204 RTS R4
6675
6676 .SBTTL LOAD "L" REGISTERS
6677 ;* THE "L" REGISTERS ARE LOADED FORM THE PARAMETERS FOLLOWING
6678 ;* THE SUBROUTINE CALL. DIAGNOSTIC MODE IS SET IN L.MR1
6679 ;* AND L.CS1 IS LOADED WITH THE COMMAND.
6680 ;*
6681 ;* CALL:
6682 ;* JSR R4,LOADRK
6683 ;* .WORD CYLINDER
6684 ;* .BYTE ;SECTOR
6685 ;* .BYTE ;TRACK
6686 ;* .WORD ;BUFFER ADDRESS
6687 ;* .WORD ;WORD COUNT
6688 ;* .WORD ;COMMAND,FORMAT,DRIVE TYPE,&UPPER BUS ADDRESS BITS
```

```

6689
6690
6691
6692
6693 032230 012437 002274
6694 032234 012437 002266
6695 032240 012437 002264
6696 032244 012437 002262
6697 032250 012737 000040 002276
6698 032256 005037 002272
6699 032262 005037 002270
6700 032266 012437 002260
6701 032272 000204
6702
6703
6704
6705
6706
6707 032274 013762 002262 000002
6708 032302 013762 002264 000004
6709 032310 013762 002274 000020
6710 032316 013762 002266 000006
6711 032324 013762 002270 000010
6712 032332 013762 002276 000026
6713 032340 013762 002272 000016
6714 032346 013762 002260 000000
6715 032354 000204
6716
6717
6718
6719
6720
6721 032356
6722 032356 010046
6723 032360 010146
6724 032362 010346
6725 032364 010200
6726 032366 012701 002160
6727 032372 012703 000011
6728 032376 012021
6729 032400 005303
6730 032402 001375
6731
6732 032404 062700 000004
6733 032410 005021
6734 032412 012021
6735 032414 012037 002212
6736 032420 012037 002214
6737 032424 012037 002206
6738 032430 012037 002210
6739 032434 012603
6740 032436 012601
6741 032440 012600
6742 032442 000204
6743
6744

;*
;* RETURN:
;* RTS R4

LOADRK: MOV (R4)+,L.DCYL ;LOAD CYLINDER
MOV (R4)+,L.DA ; TRACK AND SECTOR
MOV (R4)+,L.BA ; BUS ADDRESS
MOV (R4)+,L.WC ; WORD COUNT
MOV #DMD,L.MR1 ;SET DIAGNOSTIC MODE
CLR L.ASOF ;CLEAR OFFSET
CLR L.CS2 ;CLEAR CS2
MOV (R4)+,L.CS1 ;LOAD CS1
RTS R4 ;RETURN

.SBTTL START THE OPERATION
;* THE INFORMATION IN THE 'L' REGISTERS ARE LOADED INTO THE
;* RK611 REGISTERS IN A STRAIGHT TRANSFER.

OPSTRT: MOV L.WC,RKWC(R2) ;MOVE THE 'L' REGISTERS INTO
MOV L.BA,RKBA(R2) ;CORRESPONDING RK611 REGISTERS.
MOV L.DCYL,RKDCYL(R2)
MOV L.DA,RKDA(R2)
MOV L.CS2,RKCS2(R2)
MOV L.MR1,RKMR1(R2)
MOV L.ASOF,RKASOF(R2)
MOV L.CS1,RKCS1(R2)
RTS R4

.SBTTL GET THE RK611 REGISTERS
;* ALL THE RK611 REGISTERS EXCEPT THE DATA BUFFER ARE
;* STORED IN THE 'T' REGISTERS.

GETREG: MOV R0,-(SP) ;;PUSH R0 ON STACK
MOV R1,-(SP) ;;PUSH R1 ON STACK
MOV R3,-(SP) ;;PUSH R3 ON STACK
MOV R2,R0 ;GET RK BASE
MOV #T.CS1,R1 ;GET START OF REGS
MOV #11,R3 ;SET COUNT
1$: MOV (R0)+,(R1)+ ;GET CS1 THRU DCYL
DEC R3
BNE 1$ ;LOOP UNTIL DONE

ADD #4,R0 ;SKIP OVER DB AND SPARE
CLR (R1)+ ;CLEAR T.DB
MOV (R0)+,(R1)+ ;STORE MR1
MOV (R0)+,T.ECPS ; ECC POSITION
MOV (R0)+,T.ECPT ; ECC PATTERN
MOV (R0)+,T.MR2 ; MR2
MOV (R0)+,T.MR3 ; MR3
MOV (SP)+,R3 ;;POP STACK INTO R3
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
RTS R4

.SBTTL 'FIRST SHALL BE LAST, LAST SHALL BE FIRST' SUBROUTINE

```

```

6745      ;*      THE CONTENTS OF R3 IS SWAPPED END FOR END, I.E. BIT 15
6746      ;*      BECOMES BIT 0 AND VICE VERSA,BIT 14 BECOMES BIT
6747      ;*      1 AND VICE VERSA, ETC.
6748      ;*
6749      ;*      CALL:
6750      ;*      JSR      R4,FSBLVV
6751      ;*      WITH R3 LOADED WITH THE WORD TO BE SWAPPED
6752      ;*
6753      ;*      RETURN:
6754      ;*      RTS      R4
6755      ;*      WITH R3 SWAPPED.
6756
6757      032444      FSB_LVV:
6758      032444      010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
6759      032446      010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
6760      032450      010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
6761      032452      005004      CLR      R4          ;;CLEAR R4 FOR SWAPPED WORD
6762      032454      012700      000001      MOV      #BIT0,R0      ;;SET FOR BIT TEST
6763      032460      012701      100000      MOV      #BIT15,R1     ;;SET FOR BIT SET
6764      032464      030003      1$:      BIT      R0,R3      ;;TEST IF BIT SET
6765      032466      001401      BFO      2$          ;;NO - SKIP
6766      032470      050104      BIS      R1,R4      ;;SET CORRESPONDING BIT
6767      032472      006300      2$:      ASL      R0          ;;SHIFT FOR NEXT BIT TEST
6768      032474      001403      BEQ      3$          ;;LAST BIT TESTED - YES - EXIT
6769      032476      000241      CLC
6770      032500      006001      ROR      R1          ;;SHIFT FOR NEXT BIT SET
6771      032502      000770      BR
6772      032504      010403      3$:      MOV      R4,R3      ;;STORED SWAPPED WORD
6773      032506      012604      MOV      (SP)+,R4     ;;POP STACK INTO R4
6774      032510      012601      MOV      (SP)+,R1     ;;POP STACK INTO R1
6775      032512      012600      MOV      (SP)+,R0     ;;POP STACK INTO R0
6776      032514      000204      RTS      R4
6777
6778
6779      .SBTTL CHECK FOR MEMORY CHECK ENABLE
6780
6781      032516      012737      032602      000004      PARCHK: MOV      #20$,ERRVEC      ;;SET VECTOR FOR MEMORY PARITY CHECK
6782      032524      012737      000340      000006      MOV      #PR7,ERRVEC+2
6783      032532      012737      000000      002334      MOV      #0,BADPAR      ;;LOAD GOOD PARITY
6784      032540      005037      002332      CLR      MEMPAR      ;;CLEAR FLAG
6785      032544      012703      172100      MOV      #MEMBAS,R3     ;;LOAD REGISTER TO DETERMINE IF
6786      ;*      MEMORY CHECK ENABLE AVAILIBLE
6787      032550      012704      000001      MOV      #1,R4          ;;INITIALIZE MASK
6788      032554      012713      000001      16$:      MOV      #PAR.EN,(R3)   ;;ENABLE MEMORY CHECK
6789      032560      005713      TST      (R3)
6790      032562      050437      002332      BIS      R4,MEMPAR     ;;SET FLAG
6791      032566      062703      000002      ADD      #2,R3
6792      032572      000241      CLC
6793      032574      006104      ROL      R4          ;;CHECK IF FINISHED
6794      032576      001366      BNE      16$          ;;NO, SET UP NEXT MEMORY PARITY MODULE
6795      032600      000406      BR      22$          ;;RESTORE TRAP VECTOR
6796
6797      032602      022626      20$:      CMP      (SP)+,(SP)+   ;;ADJUST STACK
6798      032604      062703      000002      ADD      #2,R3
6799      032610      000241      CLC
6800      032612      006104      ROL      R4          ;;CHECK IF FINISHED

```

```

6801 032614 001357          BNE      16$          ;NO, GET NEXT LOCATION
6802 032616 012737 000006 000004 22$:  MOV     @#ERRVEC+2,EPRVF ;RESTORE TRAP CATCHER
6803 032624 005037 000006          CLR     ERRVEC+2
6804 032630 005737 002332          TST     MEMPAR          ;CHECK IF MEMORY CHECK ENABLE AVAILIABLE
6805 032634 001005          BNE     25$          ;YES, RETURN
6806 032636 012737 000116 000114  MOV     @#MEMVEC+2,MEMVEC ;RESTORE TRAP CATCHER
6807 032644 005037 000116          CLR     MEMVEC+2
6808 032650 000207 25$:   RTS      PC          ;RETURN
6809          .SBTTL  ROUTINE TO SIZE MEMORY
6810
6811          ;*****
6812          ;*CALL:
6813          ;*   JSR      PC,$SIZE
6814          ;*   RETURN
6815          ;*$LSTAD WILL CONTAIN:
6816          ;*   WITH KT11 OPTION      -- LAST VIRTUAL ADDRESS OF THE LAST BANK
6817          ;*   WITHOUT KT11 OPTION   -- LAST ABSOLUTE ADDRESS OF AVAILABLE MEMORY
6818          ;*$LSTBK WILL CONTAIN THE LAST BANK AS A SAF
6819          ;*$KT11 IS THE MEMORY MANAGEMENT KEY
6820          ;*$BIT07 = 0 DON'T USE MEMORY MANAGEMENT
6821          ;*   MUST BE SETUP BEFORE THE CALL
6822          ;*$BIT15 = 0 DON'T HAVE MEMORY MANAGEMENT OPTION
6823          ;*   DETERMINED BY ROUTINE
6824
6825 032652 010046  $SIZE:  MOV     R0,-(SP)          ;;SAVE R0 ON THE STACK
6826 032654 010146          MOV     R1,-(SP)          ;;SAVE R1 ON THE STACK
6827 032656 010246          MOV     R2,-(SP)          ;;SAVE R2 ON THE STACK
6828 032660 010346          MOV     R3,-(SP)          ;;SAVE R3 ON THE STACK
6829 032662 010446          MOV     R4,-(SP)          ;;SAVE R4 ON THE STACK
6830 032664 013746 000114  MOV     @#114,-(SP)       ;;SAVE MEMORY ERROR VECTOR PS & PC
6831 032670 013746 000116  MOV     @#116,-(SP)
6832 032674 012737 000116 000114  MOV     #116,@#114       ;;IGNORE PARITY ERRORS WHILE SIZING
6833 032702 012737 000002 000116  MOV     #RTI,@#116
6834 032710 013746 000004          MOV     @#ERRVEC,-(SP)   ;;SAVE PRESENT ERROR VECTOR PS & PC
6835 032714 013746 000006          MOV     @#ERRVEC+2,-(SP)
6836 032720 010600          MOV     SP,R0           ;;SAVE THE STACK POINTER
6837          ;;SET THE ERRVEC PS TO THE PRESENT PS
6838 032722 104400          TRAP
6839 032724 012637 000006          MOV     (SP)+,@#ERRVEC+2 ;;PUSH OLD PSW AND PC ON STACK
6840 032730 012701 003776          MOV     #3776,R1        ;;SAVE THE PSW IN @#ERRVEC+2
6841 032734 105727          MOV     #3776,R1        ;;SETUP ADDRESS
6842 032736 000200          TSTB   (PC)+           ;;USE MEMORY MANAGEMENT?
6843 032740 100145          $KT11: .WORD 200        ;;SET TO USE MEMORY MANAGEMENT
6844 032742 012737 033246 000004  BPL    $SCORE          ;;BR IF NO
6845 032750 005737 177572          MOV     @#SKTNEX,@#ERRVEC ;;SET FOR TIMEOUT
6846 032754 052737 100000 032736  TST    @#SRO           ;;KT11 ARE YOU THERE?
6847 032762 012737 033012 000004  BIS    #100000,$KT11    ;;YES--SET KT11 KEY
6848 032770 005737 170200          MOV     #100$,@#ERRVEC  ;;SET FOR TIMEOUT
6849 032774 012737 176200 033032  TST    @#170200        ;;UNIBUS MAP ARE YOU THERE?
6850 033002 012737 000200 033030  MOV     #176200,@#$STOP ;;YES-SET COMPARISON VALUE FOR 11/70
6851 033010 000411          MOV     #200,@#$MAP     ;;TURN ON MAP INDICATOR
6852 033012 012737 006200 033032  BR     $MAPRG          ;;GO SET UP MAP REGISTERS
6853 033020 022626          MOV     #6200,@#$STOP  ;;COMPARISON VALUE FOR 18 BIT MAPPING
6854 033022 005037 033030          CMP     (SP)+,(SP)+    ;;CLEAN OFF STACK
6855 033026 000412          CLR     @#$MAP         ;;MAKE SURE MAP INDICATOR TURNED OFF
6856 033030 000000          BR     $NOMAP         ;;
          $MAP:  .WORD 0          ;;=200 IF MAP PRESENT

```

BK001

BK001
BK001
BK001
BK001
BK001
BK001
BK001
BK001
BK001
BK001

6857	C33032	000000			\$STOP: .WORD	0	:: FILLED WITH APPROPRIATE COMPARISON VALUE	BK001
6858	033034	012703	000037		\$MAPRG: MOV	#37,R3	:: SET UP COUNTER	BK001
6859	033040	012702	170200		MOV	#170200,R2	:: START WITH MAPLO	BK001
6860	033044	005022			100\$: CLR	(R2)+	:: LOAD ALL MAP REGISTERS	BK001
6861	033046	012722	000074		MOV	#74,(R2)+	:: WITH THE VALUE 1700000	BK001
6862	033052	077304			SOB	R3,100\$:: DO ALL 31 REGISTERS	BK001
6863	033054				\$NOMAP:			
6864	033054	005046			CLR	-(SP)	:: INITIALIZE FOR 'PAR' LOADING	
6865	033056	012702	172340		MOV	#KIPAR0,R2	:: ADDRESS OF FIRST 'PAR'	
6866	033062	012703	000010		MOV	#^D8,R3	:: LOAD EIGHT 'PAR.'S' AND EIGHT 'PDR.'S'	
6867	033066	012762	077406	177740	1\$: MOV	#77406,-40(R2)	:: PDR = 4K, UP, READ/WRITE	
6868	033074	011622			MOV	(SP),(R2)+	:: LOAD 'PAR'	
6869	033076	062716	000200		ADD	#200,(SP)	:: UPDATE FOR NEXT 'PAR'	
6870	033102	077307			SOB	R3,1\$:: LOOP UNTIL ALL EIGHT ARE LOADED	
6871	033104	012742	177600		MOV	#177600,-(R2)	:: SETUP KIPAR7 FOR I/O	
6872	033110	005042			CLR	-(R2)	:: SETUP KIPAR6 FOR TESTING	
6873	033112	012737	033130	000004	MOV	#2\$,@#ERRVEC	:: CATCH TIMEOUT IF NO SR3	
6874	033120	012737	000060	172516	MOV	#60,@#SR3	:: ENABLE 22 BIT MODE AND UNIBUS MAP	BK001
6875	033126	000401			BR	3\$:: THIS PDP-11 HAS A SR3 REGISTER	
6876	033130	022626			2\$: CMP	(SP)+,(SP)+	:: CLEAN OFF THE STACK--NO SR3	
6877	033132	005237	177572		3\$: INC	@#SR0	:: TURN ON MEMORY MANAGEMENT	
6878	033136	012737	033204	000004	MOV	#\$KTOUT,@#ERRVEC	:: SET FOR TIME OUT	
6879	033144	105737	033030		TSTB	@#\$MAP	:: IS MAP THERE?	BK001
6880	033150	100006			BPI	4\$:: NO-SKIP	BK001
6881	C33152	012737	033226	000114	MOV	#\$MMOUT,@#114	:: SET UP MEMORY ERROR VECTOR	BK001
6882	033160	013737	000006	000116	MOV	@#ERRVEC+2,@#116	:: LOCK OUT INTERRUPTS	BK001
6883	033166	005737	143776		4\$: TST	@#143776	:: TRAP ON NON-EX-MEM	
6884	033172	062712	000040		ADD	#40,(R2)	:: MAKE A 1K STEP	
6885	033176	023712	033032		CMP	@#\$STOP,(R2)	:: LAST ONE?	
6886	033202	101371			BHI	4\$:: NO--TRY IT	
6887	033204	011202			\$KTOUT: MOV	(R2),R2	:: GET LAST BANK+1	
6888	033206	005037	177572		CLR	@#SR0	:: TURN OFF MEMORY MANAGEMENT	
6889	033212	105737	033030		TSTB	@#\$MAP	:: IS MAP THERE?	BK001
6890	033216	100034			BPL	\$SIZEX	:: NO-SKIP	BK001
6891	033220	005037	172516		CLR	@#SR3	:: TURN OFF MAP	BK001
6892	033224	000431			BR	\$SIZEX		
6893	033226	013704	177744		\$MMOUT: MOV	@#177744,R4	:: SAVE MEMORY ERROR REGISTER	BK001
6894	033232	010437	177744		MOV	R4,@#177744	:: CLEAR BITS IN REGISTER	BK001
6895	033236	032704	000001		BIT	#1,R4	:: MEMORY TIMEOUT?	BK001
6896	033242	001360			BNE	\$KTOUT	:: YES-EXIT	BK001
6897	033244	000002			RTI		:: MUST BE PARITY ERROR-IGNORE IT	BK001
6898	033246	042737	100000	032736	\$KTNEX: BIC	#100000,\$KT11	:: KT11 NON-EXISTENT	
6899	033254	012737	033304	000004	\$SCORE: MOV	#\$SCROUT,@#ERRVEC	:: SET FOR TIMEOUT	
6900	033262	005002			CLR	R2	:: SET UP BANK	
6901	033264	062701	004000		1\$: ADD	#4000,R1	:: INCREMENT BY 1K	
6902	033270	062702	000040		ADD	#40,R2	:: 1K STEP	
6903	033274	005711			TST	(R1)	:: TRAP ON TIME OUT	
6904	033276	022701	177776		CMP	#177776,R1	:: LAST ONE	
6905	033302	001370			BNE	1\$:: NO--TRY AGAIN	
6906	033304	162701	004000		\$SCROUT: SUB	#4000,R1		
6907	033310	162702	000040		\$SIZEX: SUB	#40,R2	:: DROP BACK	
6908	033314	010006			MOV	R0,SP	:: RESTORE THE STACK	
6909	033316	012637	000006		MOV	(SP)+,@#ERRVEC+2	:: RESTORE ERROR VECTOR	
6910	033322	012637	000004		MOV	(SP)+,@#ERRVEC		
6911	033326	012637	000116		MOV	(SP)+,@#116	:: RESTORE MEMORY ERROR VECTOR	
6912	033332	012637	000114		MOV	(SP)+,@#114		

```

6913 033336 010137 033362      MOV      R1,$LSTAD      ;;LAST ADDRESS
6914 033342 010237 033364      MOV      R2,$LSTBK     ;;LAST BANK
6915 033346 012604              MOV      (SP)+,R4      ;;RESTORE R4
6916 033350 012603              MOV      (SP)+,R3      ;;RESTORE R3
6917 033352 012602              MOV      (SP)+,R2      ;;RESTORE R2
6918 033354 012601              MOV      (SP)+,R1      ;;RESTORE R1
6919 033356 012600              MOV      (SP)+,R0      ;;RESTORE R0
6920 033360 000207              RTS        PC
6921 033362 000000      $LSTAD: .WORD 0        ;;CONTAINS THE LAST ADDRESS
6922 033364 000000      $LSTBK: .WORD 0        ;;CONTAINS THE LAST BANK
6923                               .SBTTL  SCOPE HANDLER ROUTINE
6924
6925                               ;*****
6926                               ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
6927                               ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(D:SPRAY<7:0>)
6928                               ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
6929                               ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
6930                               ;*SW14=1      LOOP ON TEST
6931                               ;*SW11=1      INHIBIT ITERATIONS
6932                               ;*SW09=1      LOOP ON ERROR
6933                               ;*SW08=1      LOOP ON TEST IN SWR<7:0>
6934                               ;*CALL
6935                               ;*      SCOPE      ;;SCOPE=IOT
6936
6937                               $SCOPE:
6938 033366 104407      1$:      CKSWR      ;;TEST FOR CHANCE IN SOFT-SWR
6939 033370 032777 040000 145542 1$:      BIT      #BIT14,@SWR  ;;LOOP ON PRESENT TEST?
6940 033376 001131              BNE      $OVER      ;;YES IF SW14=1
6941                               ;*****START OF CODE FOR THE XOR TESTER*****
6942 033400 000416      $XTSTR: BR      6$
6943
6944 033402 013746 000004              MOV      @#ERRVEC,-(SP) ;;SAVE THE CONTENTS OF THE ERROR VECTOR
6945 033406 012737 033426 000004              MOV      #5$,@#ERRVEC  ;;SET FOR TIMEOUT
6946 033414 005737 177060              TST      @#177060      ;;TIME OUT ON XOR?
6947 033420 012637 000004              MOV      (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
6948 033424 000500              BR      $SVLAD      ;;GO TO THE NEXT TEST
6949 033426 022626      5$:      CMP      (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
6950 033430 012637 000004              MOV      (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
6951 033434 000440              BR      7$          ;;LOOP ON THE PRESENT TEST
6952 033436
6953 033436 032777 000400 145474      6$:;*****END OF CODE FOR THE XOR TESTER*****
6954 033444 001421              BIT      #BIT08,@SWR  ;;LOOP ON SPEC. TEST?
6955 033446 005046              BEQ      2$          ;;BR IF NO
6956 033450 117716 145464              CLR      -(SP)      ;;CLEAR A TEMP. LOCATION
6957 033454 001414              MOV      @SWR,(SP)  ;;PICKUP THE DESIRED TEST NUMBER
6958 033456 022716 000036              BEQ      8$          ;;BRANCH IF BAD TEST NUMBER IN SWR
6959 033462 002411              CMP      #36,(SP)   ;;CHECK THE NUMBER IN THE SWR
6960 033464 011637 001102              BLT      8$          ;;BRANCH IF TEST NUMBER IS OUT OF RANGE
6961 033470 005316              MOV      (SP),$TSTNM ;;UPDATE THE TEST NUMBER
6962 033472 006316              DEC      (SP)      ;;BACKUP BY ONE
6963 033474 062716 033700              ASL      (SP)      ;;SCALE THE TEST NUMBER AS AN INDEX
6964 033500 013637 001106              ADD      #$$SW08TBL,(SP) ;;FORM THE ADDRESS OF TEST POINTER
6965 033504 000466              MOV      @(SP)+,$LPADR ;;SET LOOP ADDRESS TO DESIRED TEST
6966 033506 005726      8$:      BR      $OVER      ;;GO LOOP ON THE TEST
6967 033510 105737 001103      2$:      TST      (SP)+    ;;CLEAN THE BAD TEST NUMBER OFF OF THE STACK
6968 033514 001421      2$:      TSTB   $ERFLG    ;;HAS AN ERROR OCCURRED?
                               3$:      BEQ      3$          ;;BR IF NO

```

BK001

6969	033516	123737	001115	001103		CMPB	\$ERMAX,\$ERFLG	::MAX. EPRORS FOR THIS TEST OCCURRED?
6970	033524	101015				BHI	3\$::BR IF NO
6971	033526	032777	001000	145404		BIT	#BIT09,@SWR	::LOOP ON ERROR?
6972	033534	001404				BEQ	4\$::BR IF NO
6973	033536	013737	001110	001106	7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
6974	033544	000446				BR	\$OVER	
6975	033546	105037	001103		4\$:	CLRB	\$ERFLG	::ZFRO THE ERROR FLAG
6976	033552	005037	001234			CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE
6977	033556	000415				BR	1\$::ESCAPE TO THE NEXT TEST
6978	033560	032777	004000	145352	3\$:	BIT	#BIT11,@SWR	::INHIBIT ITERATIONS?
6979	033566	001011				BNE	1\$::BR IF YES
6980	033570	005737	001256			TST	\$PASS	::IF FIRST PASS OF PROGRAM
6981	033574	001406				BEQ	1\$:: INHIBIT ITERATIONS
6982	033576	005237	001104			INC	\$ICNT	::INCREMENT ITERATION COUNT
6983	033602	023737	001234	001104		CMP	\$TIMES,\$ICNT	::CHECK THE NUMBER OF ITERATIONS MADE
6984	033610	002024				BGE	\$OVER	::BR IF MORE ITERATION REQUIRED
6985	033612	012737	000001	001104	1\$:	MOV	#1,\$ICNT	::REINITIALIZE THE ITERATION COUNTER
6986	033620	013737	033676	001234		MOV	\$MXCNT,\$TIMES	::SET NUMBER OF ITERATIONS TO DO
6987	033626	105237	001102		\$SVLAD:	INCB	\$TSTNM	::COUNT TEST NUMBERS
6988	033632	113737	001102	001254		MOVB	\$TSTNM,\$TSTNM	::SET TEST NUMBER IN APT MAILBOX
6989	033640	011637	001106			MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS
6990	033644	011637	001110			MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS
6991	033650	005037	001236			CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
6992	033654	112737	000001	001115		MOVB	#1,\$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
6993	033662	013777	001102	145252	\$OVER:	MOV	\$TSTNM,@DISPLAY	::DISPLAY TEST NUMBER
6994	033670	013716	001106			MCV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS
6995	033674	000002				RTI		::FIXES PS
6996	033676	003720			\$MXCNT:	2000.		::MAX. NUMBER OF ITERATIONS
6997	033700				\$SWCSTBL:			
6998	033700	003414			.WORD	TST1+2		::STARTING ADDRESS OF TEST 1
6999	033702	004032			.WORD	TST2+2		::STARTING ADDRESS OF TEST 2
7000	033704	004444			.WORD	TST3+2		::STARTING ADDRESS OF TEST 3
7001	033706	005056			.WORD	TST4+2		::STARTING ADDRESS OF TEST 4
7002	033710	005470			.WORD	TST5+2		::STARTING ADDRESS OF TEST 5
7003	033712	006112			.WORD	TST6+2		::STARTING ADDRESS OF TEST 6
7004	033714	006362			.WORD	TST7+2		::STARTING ADDRESS OF TEST 7
7005	033716	006624			.WORD	TST10+2		::STARTING ADDRESS OF TEST 10
7006	033720	007140			.WORD	TST11+2		::STARTING ADDRESS OF TEST 11
7007	033722	007432			.WORD	TST12+2		::STARTING ADDRESS OF TEST 12
7008	033724	010160			.WORD	TST13+2		::STARTING ADDRESS OF TEST 13
7009	033726	010706			.WORD	TST14+2		::STARTING ADDRESS OF TEST 14
7010	033730	011434			.WORD	TST15+2		::STARTING ADDRESS OF TEST 15
7011	033732	012162			.WORD	TST16+2		::STARTING ADDRESS OF TEST 16
7012	033734	012716			.WORD	TST17+2		::STARTING ADDRESS OF TEST 17
7013	033736	013452			.WORD	TST20+2		::STARTING ADDRESS OF TEST 20
7014	033740	014200			.WORD	TST21+2		::STARTING ADDRESS OF TEST 21
7015	033742	014514			.WORD	TST22+2		::STARTING ADDRESS OF TEST 22
7016	033744	015006			.WORD	TST23+2		::STARTING ADDRESS OF TEST 23
7017	033746	015534			.WORD	TST24+2		::STARTING ADDRESS OF TEST 24
7018	033750	016270			.WORD	TST25+2		::STARTING ADDRESS OF TEST 25
7019	033752	016604			.WORD	TST26+2		::STARTING ADDRESS OF TEST 26
7020	033754	017266			.WORD	TST27+2		::STARTING ADDRESS OF TEST 27
7021	033756	017602			.WORD	TST30+2		::STARTING ADDRESS OF TEST 30
7022	033760	020352			.WORD	TST31+2		::STARTING ADDRESS OF TEST 31
7023	033762	020662			.WORD	TST32+2		::STARTING ADDRESS OF TEST 32
7024	033764	021220			.WORD	TST33+2		::STARTING ADDRESS OF TEST 33

```
7025 033766 021550          .WORD TST34+2          ;;STARTING ADDRESS OF TEST 34
7026 033770 022104          .WORD TST35+2          ;;STARTING ADDRESS OF TEST 35
7027 033772 022460          .WORD TST36+2          ;;STARTING ADDRESS OF TEST 36
7028 *****
7029 .SBTTL LOOP ON INTERNAL ERROR
7030
7031 033774 032777 001000 145136 SCOP1$: BIT #SW9,@SWR      ;CHECK IF LOOP ON ERROR
7032 034002 001405          BEQ 5$                ;NO RETURN
7033 034004 105737 001103          TSTB $ERFLG          ;CHECK IF ERROR OCCURED
7034 034010 001402          BEQ 5$                ;NO, RETURN
7035 034012 013716 001110          MOV $LPERR,(SP)      ;GO BACK TO BEGINNING OF LOOP
7036 034016 000002          5$: RTI              ;RETURN
7037 .SBTTL APT COMMUNICATIONS ROUTINE
7038
7039 *****
7040 034020 112737 000001 034264 $ATY1: MOVB #1,$FFLG   ;;TO REPORT FATAL ERROR
7041 034026 112737 000001 034262 $ATY3: MOVB #1,$MFLG   ;;TO TYPE A MESSAGE
7042 034034 000403          BR $ATYC
7043 034036 112737 000001 034264 $ATY4: MOVB #1,$FFLG   ;;TO ONLY REPORT FATAL ERROR
7044 034044          $ATYC:
7045 034044 010046          MOV R0,-(SP)         ;;PUSH R0 ON STACK
7046 034046 010146          MOV R1,-(SP)         ;;PUSH R1 ON STACK
7047 034050 105737 034262          TSTB $MFLG          ;;SHOJLD TYPE A MESSAGE?
7048 034054 001450          BEQ 5$                ;IF NOT: BR
7049 034056 122737 000001 001270          CMPB #APTENV,$ENV   ;;OPERATING UNDER APT?
7050 034064 001031          BNE 3$                ;IF NOT: BR
7051 034066 132737 000100 001271          BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
7052 034074 001425          BEQ 3$                ;IF NOT: BR
7053 034076 017600 000004          MOV @4(SP),R0        ;;GET MESSAGE ADDR.
7054 034102 062766 000002 000004          ADD #2,4(SP)         ;;BUMP RETURN ADDR.
7055 034110 005737 001250          1$: TST $MSGTYPE     ;;SEE IF DONE W/ LAST XMISSION?
7056 034114 001375          BNE 1$                ;IF NOT: WAIT
7057 034116 010037 001264          MOV R0,$MSGAD        ;;PUT ADDR IN MAILBOX
7058 034122 105720          2$: TSTB (R0)+        ;;FIND END OF MESSAGE
7059 034124 001376          BNE 2$                ;
7060 034126 163700 001264          SUB $MSGAD,R0        ;;SUB START OF MESSAGE
7061 034132 006200          ASR R0                ;GET MESSAGE LNGTH IN WORDS
7062 034134 010037 001266          MOV R0,$MSGGLT       ;;PUT LENGTH IN MAILBOX
7063 034140 012737 000004 001250          MOV #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
7064 034146 000413          BR 5$
7065 034150 017637 000004 034174 3$: MOV @4(SP),4$        ;;PUT MSG ADDR IN JSR LINKAGE
7066 034156 062766 000002 000004          ADD #2,4(SP)         ;;BUMP RETURN ADDRESS
7067 034164 013746 177776          MOV 177776,-(SP)    ;;PUSH 17776 ON STACK
7068 034170 004737 035044          JSR PC,$TYPE        ;;CALL TYPE MACRO
7069 034174 000000          4$: .WORD 0
7070 034176          5$:
7071 034176 105737 034264          10$: TSTB $FFLG        ;;SHOULD REPORT FATAL ERROR?
7072 034202 001416          BEQ 12$               ;IF NOT: BR
7073 034204 005737 001270          TST $ENV             ;;RUNNING UNDER APT?
7074 034210 001413          BEQ 12$               ;IF NOT: BR
7075 034212 005737 001250          11$: TST $MSGTYPE     ;;FINISHED LAST MESSAGE?
7076 034216 001375          BNE 11$              ;IF NOT: WAIT
7077 034220 017637 000004 001252          MOV @4(SP),$FATAL   ;;GET ERROR #
7078 034226 062766 000002 000004          ADD #2,4(SP)         ;;BUMP RETURN ADDR.
7079 034234 005237 001250          INC $MSGTYPE        ;;TELL APT TO TAKE ERROR
7080 034240 105037 034264          12$: CLRB $FFLG     ;;CLEAR FATAL FLAG
```



```

7081 034244 105037 034263          CLR# $LFLG          ;;CLEAR LOG FLAG
7082 034250 105037 034262          CLR# $MFLG          ;;CLEAR MESSAGE FLAG
7083 034254 012601                   MOV (SP)+,R1        ;;POP STACK INTO R1
7084 034256 012600                   MOV (SP)+,R0        ;;POP STACK INTO R0
7085 034260 000207                   RTS PC              ;;RETURN
7086 034262 000          $MFLG: .BYTE 0      ;;MESSG. FLAG
7087 034263 000          $LFLG: .BYTE 0      ;;LOG FLAG
7088 034264 000          $FFLG: .BYTE 0      ;;FATAL FLAG
7089          034266                   .EVEN
7090          000200          APTSIZE=200
7091          000001          APTENV=001
7092          000100          APTSPool=100
7093          000040          APTCSUP=040
7094          .SBTTL ERROR HANDLER ROUTINE
7095
7096          *****
7097          *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
7098          *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
7099          *AND GO TO TYPERR ON ERROR
7100          *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
7101          *SW15=1          HALT ON ERROR
7102          *SW13=1          INHIBIT ERROR TYPEOUTS
7103          *SW10=1          BELL ON ERROR
7104          *SW09=1          LOOP ON ERROR
7105          *CALL
7106          *          ERROR N          ;;ERROR=EMT AND N=ERROR ITEM NUMBER
7107
7108 034266          $ERROR:
7109 034266 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
7110 034270 105237 001103          /$: INCB $ERFLG          ;;SET THE ERROR FLAG
7111 034274 001775          BEQ 7$          ;;DON'T LET THE FLAG GO TO ZERO
7112 034276 013777 001102 144636          MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
7113 034304 032777 002000 144626          BIT #BIT10,@SWR   ;;BELL ON ERROR?
7114 034312 001402          BEQ 1$          ;;NO - SKIP
7115 034314 104401 001240          TYPE $BELL       ;;RING BELL
7116 034320 005237 001112          1$: INC $ERTTL    ;;COUNT THE NUMBER OF ERRORS
7117 034324 011637 001116          MOV (SP), $ERRPC ;;GET ADDRESS OF ERROR INSTRUCTION
7118 034330 162737 000002 001116          SUB #2, $ERRPC
7119 034336 117737 144554 001114          MOV#B @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE
7120 034344 032777 020000 144566          BIT #BIT13,@SWR   ;;SKIP TYPEOUT IF SET
7121 034352 001004          BNE 20$          ;;SKIP TYPEOUTS
7122 034354 004737 034466          JSR PC, TYPERR   ;;GO TO USER ERROR ROUTINE
7123 034360 104401 001245          TYPE $CRLF
7124 034364
7125 034364 122737 000001 001270          20$: CMPB #APTENV, $ENV ;;RUNNING IN APT MODE
7126 034372 001007          BNE 2$          ;;NO, SKIP APT ERROR REPORT
7127 034374 113737 001114 034406          MOV#B $ITEMB, 21$ ;;SET ITEM NUMBER AS ERROR NUMBER
7128 034402 004737 034036          JSR PC, $ATY4    ;;REPORT FATAL ERROR TO APT
7129 034406 000          21$: .BYTE 0
7130 034407 000          .BYTE 0
7131 034410 000777          22$: BR 22$          ;;APT ERROR LOOP
7132 034412 005777 144522          2$: TST @SWR      ;;HALT ON ERROR
7133 034416 100002          BPL 3$          ;;SKIP IF CONTINUE
7134 034420 000000          HALT           ;;HALT ON ERROR.
7135 034422 104407          CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
7136 034424 032777 001000 144506          3$: BIT #BIT09,@SWR ;;LOOP ON ERROR SWITCH SET?

```

```

7137 034432 001402          BEQ      4$          ;;BR IF NO
7138 034434 013716 001110    MOV      $LPERR,(SP)    ;;FUDGE RETURN FOR LOOPING
7139 034440 005737 001236    4$:    TST      $ESCAPE    ;;CHECK FOR AN ESCAPE ADDRESS
7140 034444 001402          BEQ      5$          ;;BR IF NONE
7141 034446 013716 001236    MOV      $ESCAPE,(SP)  ;;FUDGE RETURN ADDRESS FOR ESCAPE
7142 034452          5$:
7143 034452 C22737 023276 000042  CMP      #SENDAD,@#42  ;;ACT-11 AUTO-ACCEPT?
7144 034460 001001          BNE      6$          ;;BRANCH IF NO
7145 034462 000000          HALT
7146 034464          6$:
7147 034464 000002          RTI          ;;RETURN
    
```

```

7148
7149
7150          ;*****
7151          ;SBTTL TYPE ERROR ROUTINE
7152          ;*ENTRY JSR PC,TYPERR
7153          ;*RETURN RTS PC
7154          ;*
7155          ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
7156          ;*ERROR IS TO BE REPORTED. IT THEN USES THE "ERROR TABLE" ($ERRTB)
7157          ;*ENTRY TO DEFINE WHAT INFORMATION IS TO BE REPORTED CONCERNING
7158          ;*THE ERROR.
    
```

```

7159 034466 104413          TYPERR: SAVREG
7160 034470 113737 001102 001254  MOVB     $STNM,$TESTN  ;SAVE TEST NUMBER FOR REPORT
7161 034476 042737 177400 001254  BIC      #177400,$TESTN ;CLEAR UNUSED BITS
7162 034504 113700 001114          MOVB     $ITEMB,R0     ;ENTER ERROR NUMBER
7163 034510 042700 177400          BIC      #177400,R0     ;CLEAR UNUSED BITS
7164 034514 005300          DEC     R0            ;FORM INDEX FOR ERROR TABLE
7165 034516 006300          ASL     R0
7166 034520 006300          ASL     R0
7167 034522 006300          ASL     R0
7168 034524 062700 001330 1$:    ADD     #ERRTB,R0      ;FORM ADDRESS OF ERROR ENTRY
7169 034530 012037 034544          MOV     (R0)+,2$      ;GET EM POINTER
7170 034534 001404          BEQ     3$            ;BRANCH IF THERE ISN'T ONE
7171 034536 104401 001245          TYPE   .$CRLF        ;TYPE CARRIAGE RETURN LINE FEED
7172 034542 104401          TYPE   ;TYPE ERROR MESSAGE (EM)
7173 034544 000000          2$:    .WORD 0          ;EM POINTER GOES HERE
7174 034546 012037 034562 3$:    MOV     (R0)+,4$      ;GET DH POINTER
7175 034552 001404          BEQ     5$            ;BRANCH IF THERE ISN'T ONE
7176 034554 104401 001245          TYPE   .$CRLF        ;TYPE CR-LF
7177 034560 104401          TYPE   ;TYPE DATA HEADER
7178 034562 000000          4$:    .WORD 0          ;DH POINTER GOES HERE
7179 034564 012001 5$:    MOV     (R0)+,R1      ;GET DT POINTER
7180 034566 001445          BEQ     20$          ;BRANCH IF THERE ARE NONE
7181 034570 005004          CLR     R4          ;RESET INDENT SWITCH
7182 034572 012000          MOV     (R0)+,R0      ;GET DF POINTER
7183 034574 012002          MOV     (R0)+,R2      ;STORE NUMBER OF DH'S
7184 034576 104401 001245          TYPE   .$CRLF
7185 034602 112003 10$:    MOVB     (R0)+,R3      ;GET & STORE NUMBER OF DATA WORDS
7186 034604 105720          TSTB   (R0)+        ;BUMP PAST FORMAT WORD
7187 034606 005703          TST     R3          ;TEST IF ANY DATA FOR THIS HEADER
7188 034610 001416          BEQ     14$          ;NO - SKIP DATA PRINT
7189 034612 005704          TST     R4          ;CHECK IF INDENT WORDS
7190 034614 001004          BNE     12$          ;YES, GO INDENT
7191 034616 013146 11$:    MOV     @ (R1)+,-(SP) ;PUT FIRST DATA WORD ON STACK
7192 034620 104402          TYPOC          ;TYPE IT
    
```

```
7193 034622 005303          DEC      R3          ;MORE DATA WORDS
7194 034624 001403          BEQ      13$         ;NO-BRANCH
7195 034626 104401 041410    12$:    TYPE     ,SPACE2 ;TYPE SEPARATORS
7196 034632 000771          BR       11$         ;LOOP
7197 034634 104401 001245    13$:    TYPE     ,$CRLF   ;TYPE <CR><LF>
7198 034640 005710          TST     (R0)        ;CHECK IF NEXT HEADER AVAILBLE
7199 034642 001401          BEQ     14$         ;NO, DO NOT CHANGE INDENT
7200 034644 005104          COM     R4          ;CHANGE INDENT
7201 034646 005302          14$:    DEC     R2          ;MORE DH'S?
7202 034650 003414          BLE     20$         ;NO-BRANCH
7203 034652 012037 034672    15$:    MOV     (R0)+,18$   ;GET NEXT DH POINTER
7204 034656 001751          BEQ     10$         ;IF NO HEADER GET DATA
7205 034660 005704          TST     R4          ;INDENT?
7206 034662 001402          BEQ     17$         ;NO-BRANCH
7207 034664 104401 041410    TYPE     ,SPACE2   ;INDENT
7208 034670 104401          17$:    TYPE     TYPE DH
7209 034672 000000          18$:    .WORD    0          ;DH POINTER GOES HERE
7210 034674 104401 001245    TYPE     ,$CRLF
7211 034700 000740          BR       10$         ;LOOP
7212 034702 104414          20$:    RESREG
7213 034704 005237 002314    INC     ERRCNT      ;INCREMENT ERROR COUNT
7214 034710 032777 010000 144222  BIT     #SW12,@SWR   ;CHECK IF ABORT AFTER 20 ERRORS
7215 034716 001421          BEQ     25$         ;NO, RETURN
7216 034720 022737 000024 002314  CMP     #20.,ERRCNT ;CHECK IF EROR THRESHOLD EXCEEDED
7217 034726 103015          BHIS   25$         ;NO, RETURN
7218 034730 104401 041413    TYPE     ,ABORT     ;TYPE 'PROGRAM HAS BEEN ABORTED BECAUSE
7219                                ; ERROR THRESHOLD EXCEEDED'
7220 034734 005737 000042    TST     42          ;CHECK IF IN CHAIN MODE
7221 034740 001407          BEQ     30$         ;NO, HALT
7222 034742 012737 000001 023134  MOV     #1,$EOPCT   ;FORCE END OF PASS COUNT TO ONE FOR ABORT
7223 034750 012706 001100    MOV     #STACK,SP  ;INITIALIZE STACK
7224 034754 000137 023106    JMP     $EOP        ;BRING IN NEXT PROGRAM IN CHAIN
7225 034760 000000          30$:    HALT
7226 034762 000207          25$:    RTS     PC
7227
7228                                .SBTTL CONTROLLED PROGRAM HALT
7229
7230 034764 013702 001324 000010  CTRHI T: MOV     $BASE,R2 ;SET '611 BASE
7231 034770 012762 005000    MOV     #CLR,RKCS2(R2) ;CLEAR SUBSYSTEM
7232 034776 005037 001236    CLR     $ESCAPE     ;CLEAR ESCAPE
7233 035002 105037 001103    CLRB   $ERFLG      ;CLEAR ERROR FLAG
7234 035006 012706 001100    MOV     #STACK,SP  ;CLEAR STACK
7235 035012 104401 041345    TYPE   ,OPR007     ;TYPE HALT MESSAGE
7236 035016 005737 000042    TST     42          ;TEST IF MONITOR PRESENT
7237 035022 001405          BEQ     5$          ;NO SKIP
7238 035024 012737 000001 023134  MOV     #1,$EOPCT   ;FORCE END OF PROGRAM
7239 035032 000137 023106    JMP     $EOP        ;JUMP TO END OF PASS
7240
7241 035036 000000          5$:    HALT PROGRAM
7242 035040 000137 002404    JMP     START1     ;RESTART IF CONTINUE
7243
7244                                .SBTTL TYPE ROUTINE
7245
7246                                ;*****
7247                                ;*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A C BYTE.
7248                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
```

```
7249 ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
7250 ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
7251 ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
7252 ;
7253 ;*CALL:
7254 ;*) USING A TRAP INSTRUCTION
7255 ;*   TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
7256 ;*OR
7257 ;*   TYPE
7258 ;*   MESADR
7259 ;*
7260 ;
7261 035044 105737 001157 $TYPE: TSTB $TFPLG      ;; IS THERE A TERMINAL?
7262 035050 100002      BPL 1$      ;; BR IF YES
7263 035052 000000      HALT      ;; HALT HERE IF NO TERMINAL
7264 035054 000430      BR 3$      ;; LEAVE
7265 035056 010046      1$: MOV R0,-(SP)      ;; SAVE R0
7266 035060 017600 000002 MOV @2(SP),R0      ;; GET ADDRESS OF ASCIZ STRING
7267 035064 122737 000001 001270 CMPB #APTENV,$ENV      ;; RUNNING IN APT MODE
7268 035072 001011      BNE 62$      ;; NO, GO CHECK FOR APT CONSOLE
7269 035074 132737 000100 001271 BITB #APTSPOOL,$ENVM      ;; SPOOL MESSAGE TO APT
7270 035102 001405      BEQ 62$      ;; NO, GO CHECK FOR CONSOLE
7271 035104 010037 035114 MOV R0,61$      ;; SETUP MESSAGE ADDRESS FOR APT
7272 035110 004737 034026 JSR PC,$ATY3      ;; SPOOL MESSAGE TO APT
7273 035114 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
7274 035116 132737 000040 001271 62$: BITB #APTCSUP,$ENVM      ;; APT CONSOLE SUPPRESSED
7275 035124 001003      BNE 60$      ;; YES, SKIP TYPE OUT
7276 035126 112046      2$: MOVB (R0)+,-(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
7277 035130 001005      BNE 4$      ;; BR IF IT ISN'T THE TERMINATOR
7278 035132 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
7279 035134 012600      60$: MOV (SP)+,R0      ;; RESTORE R0
7280 035136 062716 000002 3$: ADD #2,(SP)      ;; ADJUST RETURN PC
7281 035142 000002      RTI      ;; RETURN
7282 035144 122716 000011 4$: CMPB #HT,(SP)      ;; BRANCH IF <HT>
7283 035150 001430      BEQ 8$      ;;
7284 035152 122716 000200      CMPB #CRLF,(SP)      ;; BRANCH IF NOT <CRLF>
7285 035156 001006      BNE 5$      ;;
7286 035160 005726      TST (SP)+      ;; POP <CR><LF> EQUIV
7287 035162 104401      TYPE      ;; TYPE A CR AND LF
7288 035164 001245      $CRLF
7289 035166 105037 035374      CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
7290 035172 000755      BR 2$      ;; GET NEXT CHARACTER
7291 035174 004737 035256 5$: JSR PC,$TYPEC      ;; GO TYPE THIS CHARACTER
7292 035200 123726 001156 6$: CMPB $FILLC,(SP)+      ;; IS IT TIME FOR FILLER CHARS.?
7293 035204 001350      BNE 2$      ;; IF NO GO GET NEXT CHAR.
7294 035206 013746 001154      MOV $NULL,-(SP)      ;; GET # OF FILLER CHARS. NEEDED
7295 ;*AND THE NULL CHAR.
7296 035212 105366 000001 7$: DECB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
7297 035216 002770      BLT 6$      ;; BR IF NO--GO POP THE NULL OFF OF STACK
7298 035220 004737 035256 JSR PC,$TYPEC      ;; GO TYPE A NULL
7299 035224 105337 035374      DECB $CHARCNT      ;; DO NOT COUNT AS A COUNT
7300 035230 000770      BR 7$      ;; LOOP
7301 ;
7302 ;HORIZONTAL TAB PROCESSOR
7303 ;
7304 035232 112716 000040 8$: MOVB #' ,(SP)      ;; REPLACE TAB WITH SPACE
```

```

7305 035236 004737 035256 9$: JSR PC,$TYPEC ;;TYPE A SPACE
7306 035242 132737 000007 035374 BITB #7,$CHARCNT ;;BRANCH IF NOT AT
7307 035250 001372 BNE 9$ ;;TAB STOP
7308 035252 005726 TST (SP)+ ;;POP SPACE OFF STACK
7309 035254 000724 BR 2$ ;;GET NEXT CHARACTER
7310 035256 $TYPEC:
7311 035256 105777 143662 TSTB @$TKS ;;CHAR IN KYBD BUFFER? ;MJDC01
7312 035262 100022 BPL 10$ ;;BR IF NOT ;MJDC01
7313 035264 017746 143656 MOV @$TKB,-(SP) ;;GET CHAR ;MJDC01
7314 035270 042716 177600 BIC #177600,(SP) ;;STRIP EXTRANEIOUS BITS ;MJDC01
7315 035274 122716 000023 CMPB #XOFF,(SP) ;;WAS CHAR XOFF ;MJDC01
7316 035300 001012 BNE 102$ ;;BR IF NOT ;MJDC01
7317 035302 101$:
7318 035302 105777 143636 TSTB @$TKS ;;WAIT FOR CHAR ;MJDC01
7319 035306 100375 BPL 101$ ;MJDC01
7320 035310 117716 143632 MOVB @$TKB,(SP) ;;GET CHAR ;MJDC01
7321 035314 042716 177600 BIC #177600,(SP) ;;STRIP IT ;MJDC01
7322 035320 122716 000021 CMPB #XON,(SP) ;;WAS IT XON? ;MJDC01
7323 035324 001366 BNE 101$ ;MJDC01
7324 035326 102$:
7325 035326 005726 TST (SP)+ ;;FIX STACK ;MJDC01
7326 035330 10$:
7327 035330 105777 143614 TSTB @$TPS ;;WAIT UNTIL PRINTER IS READY ;MJDC01
7328 035334 100375 BPI 10$ ;MJDC01
7329 035336 116677 000002 143606 MOVB 2(SP),@$TPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
7330 035344 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
7331 035352 001003 BNE 1$ ;;BRANCH IF NO
7332 035354 105037 035374 CLRB $CHARCNT ;;YES--CLEAR CHARACTER COUNT
7333 035360 000406 BR $TYPEX ;;EXIT
7334 035362 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
7335 035370 001402 BEQ $TYPEX ;;BRANCH IF YES
7336 035372 105227 INCB (PC)+ ;;COUNT THE CHARACTER
7337 035374 000000 $CHARCNT: WORD 0 ;;CHARACTER COUNT STORAGE
7338 035376 000207 $TYPEX: RTS PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

7339
7340
7341
7342 *****
7343 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
7344 *OCTAL (ASCII) NUMBER AND TYPE IT.
7345 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
7346 *CALL:
7347 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7348 *      TYPOS      ;;CALL FOR TYPEOUT
7349 *      .BYTE    N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
7350 *      .BYTE    M              ;;M=1 OR 0
7351 *                               ;;1=TYPE LEADING ZEROS
7352 *                               ;;0=SUPPRESS LEADING ZEROS
7353 *
7354 *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
7355 *$TYPOS OR $TYPOC
7356 *CALL:
7357 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7358 *      TYPON      ;;CALL FOR TYPEOUT
7359 *
7360 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

7361          ;*CALL:
7362          ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
7363          ;*      TYPOC      ;;CALL FOR TYPEOUT
7364
7365 035400 017646 000000          $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
7366 035404 116637 000001 035623  MOVB     1(SP),$OFILL      ;;LOAD ZERO FILL SWITCH
7367 035412 112637 035625          MOVB     (SP)+,$OMODE+1    ;;NUMBER OF DIGITS TO TYPE
7368 035416 062716 000002          ADD      #2,(SP)          ;;ADJUST RETURN ADDRESS
7369 035422 000406          BR       $TYPON
7370 035424 112737 000001 035623  $TYPOC: MOVB     #1,$OFILL      ;;SET THE ZERO FILL SWITCH
7371 035432 112737 000006 035625  MOVB     #6,$OMODE+1    ;;SET FOR SIX(6) DIGITS
7372 035440 112737 000005 035622  $TYPON: MOVB     #5,$OCNT      ;;SET THE ITERATION COUNT
7373 035446 010346          MOV      R3,-(SP)        ;;SAVE R3
7374 035450 010446          MOV      R4,-(SP)        ;;SAVE R4
7375 035452 010546          MOV      R5,-(SP)        ;;SAVE R5
7376 035454 113704 035625          MOVB     $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
7377 035460 005404          NEG      R4
7378 035462 062704 000006          ADD      #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
7379 035466 110437 035624          MOVB     R4,$OMODE      ;;SAVE IT FOR USE
7380 035472 113704 035623          MOVB     $OFILL,R4      ;;GET THE ZERO FILL SWITCH
7381 035476 016605 000012          MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
7382 035502 005003          CLR      R3             ;;CLEAR THE OUTPUT WORD
7383 035504 006105          1$:     ROL      R5           ;;ROTATE MSB INTO 'C'
7384 035506 000404          BR       3$             ;;GO DO MSB
7385 035510 006105          2$:     ROL      R5           ;;FORM THIS DIGIT
7386 035512 006105          ROL      R5
7387 035514 006105          ROL      R5
7388 035516 010503          MOV      R5,R3
7389 035520 006103          3$:     ROL      R3           ;;GET LSB OF THIS DIGIT
7390 035522 105337 035624          DECB     $OMODE         ;;TYPE THIS DIGIT?
7391 035526 100016          BPL      7$             ;;BR IF NO
7392 035530 042703 177770          BIC      #177770,R3     ;;GET RID OF JUNK
7393 035534 001002          BNE      4$             ;;TEST FOR 0
7394 035536 005704          TST      R4             ;;SUPPRESS THIS 0?
7395 035540 001403          BEQ      5$             ;;BR IF YES
7396 035542 005204          4$:     INC      R4           ;;DON'T SUPPRESS ANYMORE 0'S
7397 035544 052703 000060          BIS      #'0,R3        ;;MAKE THIS DIGIT ASCII
7398 035550 052703 000040          5$:     BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
7399 035554 110337 035620          MOVB     R3,8$          ;;SAVE FOR TYPING
7400 035560 104401 035620          TYPE     ,8$           ;;GO TYPE THIS DIGIT
7401 035564 105337 035622          7$:     DECB     $OCNT      ;;COUNT BY 1
7402 035570 003347          BGT      2$             ;;BR IF MORE TO DO
7403 035572 002402          BLT      6$             ;;BR IF DONE
7404 035574 005204          INC      R4             ;;INSURE LAST DIGIT ISN'T A BLANK
7405 035576 000744          BR       2$             ;;GO DO THE LAST DIGIT
7406 035600 012605          6$:     MOV      (SP)+,R5     ;;RESTORE R5
7407 035602 012604          MOV      (SP)+,R4     ;;RESTORE R4
7408 035604 012603          MOV      (SP)+,R3     ;;RESTORE R3
7409 035606 016666 000002 000004  MOV      2(SP),4(SP)    ;;SET THE STACK FOR RETURNING
7410 035614 012616          MOV      (SP)+,(SP)
7411 035616 000002          RTI
7412 035620          8$:     .BYTE     0           ;;RETURN
7413 035621          .BYTE     0           ;;STORAGE FOR ASCII DIGIT
7414 035622          .BYTE     0           ;;TERMINATOR , OR TYPE ROUTINE
7415 035623          $OCNT:   .BYTE     0           ;;OCTAL DIGIT COUNTER
7416 035624 000000          $OFILL:  .BYTE     0           ;;ZERO FILL SWITCH
          $OMODE: .WORD     0           ;;NUMBER OF DIGITS TO TYPE

```

```
7417 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
7418
7419 ;*****
7420 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
7421 ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
7422 ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
7423 ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
7424 ;*REPLACED WITH SPACES.
7425 ;*CALL:
7426 ;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
7427 ;*      TYPDS                    ;;GO TO THE ROUTINE
7428
7429 $TYPDS:
7430      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
7431      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
7432      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
7433      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
7434      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
7435      MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
7436      MOV      20(SP),R5    ;;GET THE INPUT NUMBER
7437      BPL      1$          ;;BR IF INPUT IS POS.
7438      NEG      R5          ;;MAKE THE BINARY NUMBER POS.
7439      MOVVB   #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
7440      CLR      R0          ;;ZERO THE CONSTANTS INDEX
7441      MOV      #SDBLK,R3    ;;SETUP THE OUTPUT POINTER
7442      MOVVB   #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
7443      CLR      R2          ;;CLEAR THE BCD NUMBER
7444      MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
7445      SUB      R1,R5        ;;FORM THIS BCD DIGIT
7446      BLT      4$          ;;BR IF DONE
7447      INC      R2          ;;INCREASE THE BCD DIGIT BY 1
7448      BR      3$
7449      ADD      R1,R5        ;;ADD BACK THE CONSTANT
7450      TST      R2          ;;CHECK IF BCD DIGIT=0
7451      BNE      5$          ;;FALL THROUGH IF 0
7452      TSTB   (SP)          ;;STILL DOING LEADING 0'S?
7453      BMI      7$          ;;BR IF YES
7454      ASLB   (SP)          ;;MSD?
7455      BCC      6$          ;;BR IF NO
7456      MOVVB   1(SP),-1(R3) ;;YES--SET THE SIGN
7457      BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
7458      BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
7459      MOVVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
7460      TST      (R0)+      ;;JUST INCREMENTING
7461      CMP      R0,#10      ;;CHECK THE TABLE INDEX
7462      BLT      2$          ;;GO DO THE NEXT DIGIT
7463      BGT      8$          ;;GO TO EXIT
7464      MOV      R5,R2      ;;GET THE LSD
7465      BR      6$          ;;GO CHANGE TO ASCII
7466      TSTB   (SP)+      ;;WAS THE LSD THE FIRST NON-ZERO? -
7467      BPL      9$          ;;BR IF NO
7468      MOVVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
7469      CLRB   (R3)          ;;SET THE TERMINATOR
7470      MOV      (SP)+,R5    ;;POP STACK INTO R5
7471      MOV      (SP)+,R3    ;;POP STACK INTO R3
7472      MOV      (SP)+,R2    ;;POP STACK INTO R2
```

```
7473 036010 012601          MOV      (SP)+,R1          ;;POP STACK INTO R1
7474 036012 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
7475 036014 104401 036042    TYPE      $DBLK           ;;NOW TYPE THE NUMBER
7476 036020 016666 000002 000004    MOV      2(SP),4(SP)      ;;ADJUST THE STACK
7477 036026 012616          MOV      (SP)+,(SP)
7478 036030 000002          RTI                       ;;RETURN TO USER
7479 036032 023420          $DTBL: 10000.
7480 036034 001750          1000.
7481 036036 000144          100.
7482 036040 000012          10.
7483 036042 000004          $DBLK: .BLKW 4
7484          .SBTTL TTY INPUT ROUTINE
7485
7486          ;:*****
7487          .ENABL LSB
7488 036052 000000          $TKCNT: .WORD 0           ;;NUMBER OF ITEMS IN QUEUE
7489 036054 000000          $TKQIN: .WORD 0          ;;INPUT POINTER
7490 036056 000000          $TKQOUT: .WORD 0         ;;OUTPUT POINTER
7491 036060 000001          $TKQSRT: .BLKB 1        ;;TTY KEYBOARD QUEUE
7492          $TKQEND=.
7493          .EVEN
7494
7495          ;*TK INITIALIZE ROUTINE
7496          ;*THIS ROUTINE WILL INITIALIZE THE TTY KEYBOARD INPUT QUEUE
7497          ;*SETUP THE INTERRUPT VECTOR AND TURN ON THE KEYBOARD INTERRUPT
7498
7499          ;*CALL:
7500          ;*      JSR      PC,$TKINT
7501          ;*      RETURN
7502
7503 036062 005037 036052          $TKINI: CLR      $TKCNT           ;;CLEAR COUNT OF ITEMS IN QUEUE
7504 036066 012737 036060 036054    MOV      # $TKQSRT,$TKQIN    ;;MOVE THE STARTING ADDRESS OF THE
7505 036074 013737 036054 036056    MOV      $TKQIN,$TKQOUT     ;;QUEUE INTO THE INPUT & OUTPUT POINTERS.
7506 036102 012737 036132 000060    MOV      # $TKSRV,@ $TKVEC  ;;INITIALIZE THE KEYBOARD VECTOR
7507 036110 012737 000200 000062    MOV      #200,@ $TKVEC+2    ;;'BR' LEVEL 4
7508 036116 005777 143024          TST      @ $TKB            ;;CLEAR DONE FLAG
7509 036122 012777 000100 143014    MOV      #100,@ $TKS        ;;ENABLE TTY KEYBOARD INTERRUPT
7510 036130 000207          RTS      PC                ;;RETURN TO CALLER
7511
7512          ;*TK SERVICE ROUTINE
7513          ;*THIS ROUTINE WILL SERVICE THE TTY KEYBOARD INTERRUPT
7514          ;*BY READING THE CHARACTER FROM THE INPUT BUFFER AND PUTTING
7515          ;*IT IN THE QUEUE.
7516          ;*IF THE CHARACTER IS A "CONTROL-C" (^C) $TKINT IS CALLED AND
7517          ;*UPON RETURN EXIT IS MADE TO THE "CONTROL-C" RESTART ADDRESS (CTRL^C)
7518
7519 036132 117746 143010          $TKSRV: MOVW     @ $TKB, -(SP)  ;;PICKUP THE CHARACTER
7520 036136 042716 177600          BIC     #^C(177), (SP)     ;;STRIP THE JUNK
7521 036142 021627 000021          CMP     (SP), # $XON       ;;IS IT A RANDOM XON?
7522 036146 001002          BNE     308                ;;BRANCH IF NO
7523 036150 005726          TST     (SP)+              ;;CLEAN RANDOM XON OFF STACK
7524 036152 000002          RTI                       ;;RETURN
7525 036154          308:
7526 036154 021627 000003          CMP     (SP), #3           ;;IS IT A CONTROL C?
7527 036160 001007          BNE     'S                 ;;BRANCH IF NO
7528 036162 104401 037260          TYPE     .SBTTL           ;;TYPE A CONTROL-
```



```

7529 036166 004737 036062      JSR    PC,$TKINT      ;;INIT THE KEYBOARD
7530 036172 005726              TST    (SP)+          ;;CLEAN UP STACK
7531 036174 000137 034764      JMP    CTRHLT         ;;CONTROL C RESTART
7532 036200 021627 000007      1$:   CMP    (SP),#7   ;;IS IT A CONTROL G?
7533 036204 001004              BNE    2$            ;;BRANCH IF NO
7534 036206 022737 000176 001140  CMP    #SWREG,SWR    ;;IS SOFT-SWR SELECTED?
7535 036214 001500              BEQ    6$            ;;GO TO SWR CHANGE
7536
7537 036216              2$:
7538 036216 022737 000001 036052  CMP    #1,$TKCNT     ;;IS THE QUEUE FULL?
7539 036224 001004              BNE    3$            ;;BRANCH IF NO
7540 036226 104401 001240      TYPE  ,$BELL         ;;RING THE TTY BELL
7541 036232 005726              TST    (SP)+          ;;CLEAN CHARACTER OFF OF STACK
7542 036234 000451              BR     5$            ;;EXIT
7543 036236 021627 000023      3$:   CMP    (SP),#23     ;;IS IT A CONTROL-S?
7544 036242 001021              BNE    32$           ;;BRANCH IF NO
7545 036244 005077 142674      CLR    @$TKS         ;;DISABLE TTY KEYBOARD INTERRUPTS
7546 036250 005726              TST    (SP)+          ;;CLEAN CHAR OFF STACK
7547 036252 105777 142666      31$:  TSTB  @$TKS         ;;WAIT FOR A CHAR
7548 036256 100375              BPL    31$           ;;LOOP UNTIL ITS THERE
7549 036260 117746 142662      MOVB  @$TKB,-(SP)    ;;GET THE CHARACTER
7550 036264 042716 177600      BIC   #^C177,(SP)   ;;MAKE IT 7-BIT ASCII
7551 036270 022627 000021      CMP    (SP)+,#21     ;;IS IT A CONTROL-Q?
7552 036274 001366              BNF    31$           ;;BRANCH IF NO
7553 036276 012777 000100 142640  MOV    #100,@$TKs   ;;REENABLE TTY KEYBOARD INTERRUPTS
7554 036304 000002              RTI                    ;;RETURN
7555 036306 005237 036052      32$:  INC    $TKCNT        ;;COUNT THIS CHARACTER
7556 036312 021627 000140      CMP    (SP),#140     ;;IS IT UPPER CASE?
7557 036316 002405              BLT    4$            ;;BRANCH IF YES
7558 036320 021627 000175      CMP    (SP),#175     ;;IS IT A SPECIAL CHAR?
7559 036324 003002              BGT    4$            ;;BRANCH IF YES
7560 036326 042716 000040      BIC   #40,(SP)      ;;MAKE IT UPPER CASE
7561 036332 112677 177516      4$:   MOVB  (SP)+,@$TKQIN ;;AND PUT IT IN QUEUE
7562 036336 005237 036054      INC    $TKQIN        ;;UPDATE THE POINTER
7563 036342 023727 036054 036061  CMP    $TKQIN,#$TKQEND ;;GO OFF THE END?
7564 036350 001003              BNE    5$            ;;BRANCH IF NO
7565 036352 012737 036060 036054  MOV    #$TKQSRRT,$TKQIN ;;RESET THE POINTER
7566 036360 000002      5$:   RTI                    ;;RETURN
7567
7568
7569
7570
7571
7572
7573 036362 022737 000176 001140  $CKSWR: CMP    #SWREG,SWR ;;IS THE SOFT-SWR SELECTED
7574 036370 001124              BNE    15$           ;;EXIT IF NOT
7575 036372 105777 142546      TSTB  @$TKS         ;;IS A CHAR WAITING?
7576 036376 100121              BPL    15$           ;;IF NOT, EXIT
7577 036400 117746 142542      MOVB  @$TKB,-(SP)    ;;YES
7578 036404 042716 177600      BIC   #^C177,(SP)   ;;MAKE IT 7-BIT ASCII
7579 036410 021627 000007      CMP    (SP),#7      ;;IS IT A CONTROL-G?
7580 036414 001300              BNE    2$            ;;IF NOT, PUT IT IN THE TTY QUEUE
7581
7582
7583
7584

```

```

*****
;*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.
*****
;*CONTROL IS PASSED TO THIS POINT FROM EITHER THE TTY INTERRUPT SERVICE

```

```
7585      ;*ROUTINE OR FROM THE SOFTWARE SWITCH REGISTER TRAP CALL, AS A RESULT OF A  
7586      ;*CONTROL-G BEING TYPED, AND THE SOFTWARE SWITCH REGISTER BEING SELECTED.  
7587 036416 123727 001134 000001 6$: CMPB $AUTOB,#1      ;;ARE WE RUNNING IN AUTO-MODE?  
7588 036424 001674      BEQ 2$              ;;BRANCH IF YES  
7589 036426 005726      TST (SP)+          ;;CLEAR CONTROL-G OFF STACK  
7590 036430 004737 036062 JSR PC,$TKINT      ;;FLUSH THE TTY INPUT QUEUE  
7591 036434 005077 142504 CLR @TKS           ;;DISABLE TTY KEYBOARD INTERRUPTS  
7592 036440 112737 000001 001135 MOVB #1,$INTAG     ;;SET INTERRUPT MODE INDICATOR  
7593  
7594 036446 104401 037272      TYPE , $CNTLG      ;;ECHO THE CONTROL-G (^G)  
7595 036452 104401 037277 $GTSWR: TYPE , $MSWR ;;TYPE CURRENT CONTENTS  
7596 036456 013746 000176      MOV SWREG,-(SP)    ;;SAVE SWREG FOR TYPEOUT  
7597 036462 104402      TYPEOC           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)  
7598 036464 104401 037310      TYPE , $MNEW      ;;PROMPT FOR NEW SWR  
7599 036470 005046      CLR -(SP)         ;;CLEAR COUNTER  
7600 036472 005046      CLR -(SP)         ;;THE NEW SWR  
7601 036474 105777 142444 7$: TSTB @TKS          ;;CHAR THERE?  
7602 036500 100375      BPL 7$            ;;IF NOT TRY AGAIN  
7603  
7604 036502 117746 142440      MOVB @TKB,-(SP)   ;;PICK UP CHAR  
7605 036506 042716 177600      BIC #^C177,(SP)  ;;MAKE IT 7-BIT ASCII  
7606  
7607 036512 021627 000003      CMP (SP),#3       ;;IS IT A CONTROL-C?  
7608 036516 001015      BNF 9$            ;;BRANCH IF NOT  
7609 036520 104401 037260      TYPE , $CNTLC     ;;YES, ECHO CONTROL-C (^C)  
7610 036524 062706 000006      ADD #6,SP         ;;CLEAN UP STACK  
7611 036530 123727 001135 000001 CMPB $INTAG,#1     ;;REENABLE TTY KEYBOARD INTERRUPTS?  
7612 036536 001003      BNE 8$            ;;BRANCH IF NO  
7613 036540 012777 000100 142376 MOV #100,@TKS     ;;ALLOW TTY KEYBOARD INTERRUPTS  
7614 036546 000137 034764 8$: JMP CTRHLT        ;;CONTROL-C RESTART  
7615  
7616  
7617 036552 021627 000025 9$: CMP (SP),#25     ;;IS IT A CONTROL-U?  
7618 036556 001005      BNE 10$           ;;BRANCH IF NOT  
7619 036560 104401 037265      TYPE , $CNTLU     ;;YES, ECHO CONTROL-U (^U)  
7620 036564 062706 000006 20$: ADD #6,SP         ;;IGNORE PREVIOUS INPUT  
7621 036570 000737      BR 19$            ;;LET'S TRY IT AGAIN  
7622  
7623  
7624 036572 021627 000015 10$: CMP (SP),#15     ;;IS IT A <CR>?  
7625 036576 001022      BNE 16$           ;;BRANCH IF NO  
7626 036600 005766 000004      TST 4(SP)         ;;YES, IS IT THE FIRST CHAR?  
7627 036604 001403      BEQ 11$           ;;BRANCH IF YES  
7628 036606 016677 000002 142324 MOV 2(SP),@SWR     ;;SAVE NEW SWR  
7629 036614 062706 000006 11$: ADD #6,SP         ;;CLEAN UP STACK  
7630 036620 104401 001245 14$: TYPE , $CRLF     ;;ECHO <CR> AND <LF>  
7631 036624 123727 001135 000001 CMPB $INTAG,#1     ;;RE-ENABLE TTY KBD INTERRUPTS?  
7632 036632 001003      BNE 15$           ;;BRANCH IF NOT  
7633 036634 012777 000100 142302 MOV #100,@TKS     ;;RE-ENABLE TTY KBD INTERRUPTS  
7634 036642 000002 15$: RTI              ;;RETURN  
7635 036644 004737 035256 16$: JSR PC,$TYPEC     ;;ECHO CHAR  
7636 036650 021627 000060      CMP (SP),#60     ;;CHAR < 0?  
7637 036654 002420      BLT 18$           ;;BRANCH IF YES  
7638 036656 021627 000067      CMP (SP),#67     ;;CHAR > ??  
7639 036662 003015      BGT 18$           ;;BRANCH IF YES  
7640 036664 042726 000060      BIC #60,(SP)+    ;;STRIP-OFF ASCII
```

```
7641 036670 005766 000002      TST      2(SP)      ;; IS THIS THE FIRST CHAR
7642 036674 001403      BEQ      17$      ;; BRANCH IF YES
7643 036676 006316      ASL      (SP)      ;; NO, SHIFT PRESENT
7644 036700 006316      ASL      (SP)      ;; CHAR OVER TO MAKE
7645 036702 006316      ASL      (SP)      ;; ROOM FOR NEW ONE.
7646 036704 005266 000002      17$: INC      2(SP)      ;; KEEP COUNT OF CHAR
7647 036710 056616 177776      BIS      -2(SP),(SP) ;; SET IN NEW CHAR
7648 036714 000667      BR       7$       ;; GET THE NEXT ONE
7649 036716 104401 001244      18$: TYPE    ,9QUES  ;; TYPE ?<CR><LF>
7650 036722 000720      BP       20$      ;; SIMULATE CONTROL-U
7651      .USABL  LSB
7652
7653
7654
7655      ;*****
7656      ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
7657      ;*CALL:
7658      ;*      RDCHR      ;; GET A CHARACTER FROM THE QUEUE
7659      ;*      RETURN HERE ;; CHARACTER IS ON THE STACK
7660      ;*      ;; WITH PARITY BIT STRIPPED OFF
7661      ;
7662 036724 011646 000002      $RDCHR: MOV      (SP),-(SP) ;; PUSH DOWN THE PC AND
7663 036726 016666 000004 000002      MOV      4(SP),2(SP) ;; THE PS
7664 036734 005066 000004      CLP      4(SP)      ;; GET READY FOR A CHARACTER
7665 036740 005046      CLR      -(SP)      ;; PUT NEW PS ON STACK
7666 036742 012746 036750      MOV      #64$,-(SP) ;; PUT NEW PC ON STACK
7667 036746 000002      RTI      ;; POP NEW PC AND PS
7668 036750
7669 036750 005737 036052      64$: TST      $STKCNT  ;; WAIT ON A CHARACTER
7670 036754 001775      1$: BEQ      1$      ;;
7671 036756 005337 036052      DEC      $STKCNT  ;; DECREMENT THE COUNTER
7672 036762 117766 177070 000004      MOVVB   @STKQOUT,4(SP) ;; GET ONE CHARACTER
7673 036770 005237 036056      INC      $STKQOUT  ;; UPDATE THE POINTER
7674 036774 023727 036056 036061      CMP      $STKQOUT,#$STKQEND ;; DID IT GO OFF OF THE END?
7675 037002 001003      BNE     2$       ;; BRANCH IF NO
7676 037004 012737 036060 036056      MOV      #$STKQRT,$STKQOUT ;; RESET THE POINTER
7677 037012 000002      2$: RTI      ;; RETURN
7678
7679      ;*****
7680      ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
7681      ;*CALL:
7682      ;*      RDLIN     ;; INPUT A STRING FROM THE TTY
7683      ;*      RETURN HERE ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
7684      ;*      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
7685 037014 010346      $RDLIN: MOV      R3,-(SP) ;; SAVE R3
7686 037016 005046      CLR      -(SP)      ;; CLEAR THE RUBOUT KEY
7687 037020 012703 037250      1$: MOV      #$TTYIN,R3 ;; GET ADDRESS
7688 037024 022703 037260      2$: CMP      #$TTYIN+8.,R3 ;; BUFFER FULL?
7689 037030 101456      BLOS     4$       ;; BR IF YES
7690 037032 104410      RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
7691 037034 112613      MOVVB   (SP)+,(R3)  ;; GET CHARACTER
7692 037036 122713 000177      10$: CMPB   #177,(R3) ;; IS IT A RUBOUT
7693 037042 001022      BNE     5$       ;; BR IF NO
7694 037044 005716      TST     (SP)      ;; IS THIS THE FIRST RUBOUT?
7695 037046 001007      BNE     6$       ;; BR IF NO
7696 037050 112737 000134 037246      MOVVB   #'\,9$    ;; TYPE A BACK SLASH
```

```

7697 037056 104401 037246      TYPE      ,9$
7698 037062 012716 177777      MOV      #-1,(SP)      ;;SET THE RUBOUT KEY
7699 037066 005303      6$:      DEC      R3          ;;BACKUP BY ONE
7700 037070 020327 037250      CMP      R3,#$TTYIN    ;;STACK EMPTY?
7701 037074 103434      BLO      4$            ;;BR IF YES
7702 037076 111337 037246      MOVVB    (R3),9$       ;;SETUP TO TYPEOUT THE DELETED CHAR.
7703 037102 104401 037246      TYPE     ,9$          ;;GO TYPE
7704 037106 000746      BR       2$            ;;GO READ ANOTHER CHAR.
7705 037110 005716      5$:      TST      (SP)         ;;RUBOUT KEY SET?
7706 037112 001406      BEQ      7$            ;;BR IF NO
7707 037114 112737 000134 037244      MOVVB    #'\",9$       ;;TYPE A BACK SLASH
7708 037122 104401 037246      TYPE     ,9$
7709 037126 005016      CLR      (SP)         ;;CLEAR THE RUBOUT KEY
7710 037130 122713 000025      7$:      CMPB    #25,(R3)      ;;IS CHARACTER A CTRL U?
7711 037134 001003      BNE      8$            ;;BR IF NO
7712 037136 104401 037265      TYPE     ,SCNTLU      ;;TYPE A CONTROL 'U'
7713 037142 000726      BR       1$            ;;GO START OVER
7714 037144 122713 000022      8$:      CMPB    #22,(R3)      ;;IS CHARACTER A 'R'?
7715 037150 001011      BNE      3$            ;;BRANCH IF NO
7716 037152 105013      CLRB    (R3)          ;;CLEAR THE CHARACTER
7717 037154 104401 001245      TYPE     ,SCLF        ;;TYPE A 'CR' & 'LF'
7718 037160 104401 037250      TYPE     ,STTYIN      ;;TYPE THE INPUT STRING
7719 037164 000717      BR       2$            ;;GO PICKUP ANOTHER CHACTER
7720 037166 104401 001244      4$:      TYPE     ,SQUES      ;;TYPE A '?'
7721 037172 000712      BR       1$            ;;CLEAR THE BUFFER AND LOOP
7722 037174 111337 037246      3$:      MOVVB    (R3),9$       ;;ECHO THE CHARACTER
7723 037200 104401 037246      TYPE     ,9$
7724 037204 122723 000015      CMPB    #15,(R3)+     ;;CHECK FOR RETURN
7725 037210 001305      BNE      2$            ;;LOOP IF NOT RETURN
7726 037212 105063 177777      CLRB    -1(R3)        ;;CLEAR RETURN (THE 15)
7727 037216 104401 001246      TYPE     ,SLF         ;;TYPE A LINE FEED
7728 037222 005726      TST     (SP)+         ;;CLEAN RUBOUT KEY FROM THE STACK
7729 037224 012603      MOV     (SP)+,R3      ;;RESTORE R3
7730 037226 011646      MOV     (SP),-(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
7731 037230 016666 000004 000002      MOV     4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
7732 037236 012766 037250 000004      MOV     #$TTYIN,4(SP)
7733 037244 000002      RTI                    ;;RETURN
7734 037246      C00      9$:      .BYTE    0            ;;STORAGE FOR ASCII CHAR. TO TYPE
7735 037247      C00      .BYTE    0            ;;TERMINATOR
7736 037250 000010      $TTYIN: .BLKB    8.        ;;RESERVE 8 BYTES FOR TTY INPUT
7737 037260 041536 005015      000      $CNTLC: .ASCIZ  /^C/<15><12>  ;;CONTROL 'C'
7738 037265      136      006525 000012      $CNTLU: .ASCIZ  /^U/<15><12>  ;;CONTROL 'U'
7739 037272 043536 005015      000      $CNTLG: .ASCIZ  /^G/<15><12>  ;;CONTROL 'G'
7740 037277      015      051412 051127      $MSWR:  .ASCIZ  <15><12>/SWR - /
7741 037304 036440 000040
7742 037310 020040 042516 020127      $MNEW:  .ASCIZ  / NEW = /
7743 037316 020075      000
7744      037322
7745
7746
7747
7748      ;;*****
7749      ;;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
7750      ;;*CHANGE IT TO BINARY.
7751      ;;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
7752      ;;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
      ;;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST

```

```

7753      ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
7754      ;*CALL:
7755      ;*      RDOCT          ;;READ AN OCTAL NUMBER
7756      ;*      RETURN HERE    ;;LOW ORDER BITS ARE ON TOP OF THE STACK
7757      ;*                      ;;HIGH ORDER BITS ARE IN $HIOCT
7758
7759 037322 011646          $RDOCT: MOV      (SP),-(SP)    ;;PROVIDE SPACE FOR THE
7760 037324 016666 000004 000002 MOV      4(SP),2(SP)    ;;INPUT NUMBER
7761 037332 010046          MOV      R0,-(SP)      ;;PUSH R0 ON STACK
7762 037334 010146          MOV      R1,-(SP)      ;;PUSH R1 ON STACK
7763 037336 010246          MOV      R2,-(SP)      ;;PUSH R2 ON STACK
7764 037340 104411          1$:  RDLIN          ;;READ AN ASCII LINE
7765 037342 012600          MOV      (SP)+,R0      ;;GET ADDRESS OF 1ST CHARACTER
7766 037344 010037 037450  MOV      R0,$$      ;;AND SAVE IT
7767 037350 005001          CLR      R1          ;;CLEAR DATA WORD
7768 037352 005002          CLR      R2
7769 037354 112046          2$:  MOVB     (R0)+,-(SP)    ;;PICKUP THIS CHARACTER
7770 037356 001420          BEQ      3$          ;;IF ZERO GET OUT
7771 037360 122716 000060  CMPB     #'0,(SP)      ;;MAKE SURE THIS CHARACTER
7772 037364 003026          BGT      4$          ;;IS AN OCTAL DIGIT
7773 037366 122716 000067  CMPB     #'7,(SP)
7774 037372 002423          BLT      4$
7775 037374 006301          ASL     R1          ;;*2
7776 037376 006102          ROI     R2
7777 037400 006301          ASL     R1          ;;*4
7778 037402 006102          ROL     R2
7779 037404 006301          ASL     R1          ;;*8
7780 037406 006102          ROL     R2
7781 037410 042716 177770  BIC     #'^C7,(SP)    ;;STRIP THE ASCII JUNK
7782 037414 062601          ADD     (SP)+,R1     ;;ADD IN THIS DIGIT
7783 037416 000756          BR      2$          ;;LOOP
7784 037420 005726          3$:  TST     (SP)+      ;;CLEAN TERMINATOR FROM STACK
7785 037422 010166 000012  MOV     R1,12(SP)    ;;SAVE THE RESULT
7786 037426 010237 037460  MOV     R2,$HIOCT
7787 037432 012602          MOV     (SP)+,R2     ;;POP STACK INTO R2
7788 037434 012601          MOV     (SP)+,R1     ;;POP STACK INTO R1
7789 037436 012600          MOV     (SP)+,R0     ;;POP STACK INTO R0
7790 037440 000002          RTI          ;;RETURN
7791 037442 005726          4$:  TST     (SP)+      ;;CLEAN PARTIAL FROM STACK
7792 037444 105010          CLRB   (R0)         ;;SET A TERMINATOR
7793 037446 104401          TYPE          ;;TYPE UP THRU THE BAD CHAR.
7794 037450 000000          5$:  .WORD   0
7795 037452 104401 001244  TYPE     $QUES      ;;'"' 'CR' & 'LF'
7796 037456 000730          BR      1$          ;;TRY AGAIN
7797 037460 000000          $HIOCT: .WORD   0    ;;HIGH ORDER BITS GO HERE
7798
7799      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
7800
7801      ;******
7802      ;*SAVE R0-R5
7803      ;*CALL:
7804      ;*      SAVREG
7805      ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
7806      ;*
7807      ;*TOP---(+16)
7808      ;* +2---(+18)
7809      ;* +4---R5

```

```
7809      ;* +6---R4
7810      ;* +8---R3
7811      ;*+10---R2
7812      ;*+12---R1
7813      ;*+14---R0
7814
7815      $SAVREG:
7816      037462 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
7817      037464 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
7818      037466 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
7819      037470 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
7820      037472 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
7821      037474 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
7822      037476 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
7823      037502 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
7824      037506 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PS OF CALL
7825      037512 016646 000022      MOV      22(SP),-(SP)  ;;SAVE PC OF CALL
7826      037516 000002      RTI
7827
7828      ;*RESTORE R0-R5
7829      ;*CALL:
7830      ;*      RESREG
7831      $RESREG:
7832      037520 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF CALL
7833      037524 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF CALL
7834      037530 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
7835      037534 012666 000022      MOV      (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
7836      037540 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
7837      037542 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
7838      037544 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
7839      037546 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
7840      037550 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
7841      037552 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
7842      037554 000002      RTI
7843
7844      .SBTTL POWER DOWN AND UP ROUTINE
7845
7846      ;*****
7847
7848      ;POWER DOWN ROUTINE
7849      037556 017737 141356 002356 $PWRDN: MOV      @SWR,SAVSWR  ;SAVE SWITCH REGISTER
7850      037564 012737 037604 000024      MOV      #PWRUP,PWRVEC  ;SET UP VECTOR
7851      037572 012737 000340 000026      MOV      #PR7,PWRVEC+2
7852      037600 000000      HALT
7853      037602 000776      BR      .-2      ;HANG UP
7854
7855      ;*****
7856
7857      ;POWER UP ROUTINE
7858      037604 005037 037700 $PWRUP: CLR      $PWRCT      ;LOAD WAIT COUNT
7859      037610 012737 000144 037702      MOV      #100,$PWRCT+2
7860      037616 005237 037700      1$: INC      $PWRCT      ;WAIT FOR TELETYPE
7861      037622 001375      BNE      1$
7862      037624 005337 037702      DEC      $PWRCT+2
7863      037630 001372      BNE      1$
7864      037632 012737 037556 000024      MOV      #PWRDN,PWRVEC  ;SET UP FOR POWER DOWN VECTOR
```

```

7865 037640 012737 000340 000026      MOV      #PR7,PWRVEC+2
7866 037646 012706 001100      MOV      #STACK,SP          ;FORCE STACK
7867 037652 104401 037704      TYPE     $POWER            ;TYPE POWER
7868 037656 004737 032516      JSR      PC,PARCHK         ;REINITIALIZE MEMORY CHECK ENABLE
7869 037662 013777 002356 141250      MOV      SAVSWR,@SWR       ;RESTORE SWITCH REGISTER
7870 037670 013702 001324      MOV      $BASE,R2         ;REINITIALISE R2 FOR '611 BASE
7871 037674 000177 141206      JMP      @SLPADR ;GO BACK TO LAST TEST
7872
7873 037700 000000 000000      $PWRCT: .WORD 0,0          ;TELETYPE TIME OUT
7874 037704 047520 042527 000122      $POWER: .ASCIZ /POWER/
7875
7876      .SBTTL TRAP DECODER
7877
7878      ;*****
7879      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
7880      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
7881      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
7882      ;*GO TO THAT ROUTINE.
7883
7884 037712 010046      $TRAP:  MOV      R0,-(SP)    ;;SAVE R0
7885 037714 016600 000002      MOV      2(SP),R0         ;;GET TRAP ADDRESS
7886 037720 005740      TST      -(R0)           ;;BACKUP BY 2
7887 037722 111000      MOV      (R0),R0         ;;GET RIGHT BYTE OF TRAP
7888 037724 006300      ASL      R0              ;;POSITION FOR INDEXING
7889 037726 016000 037746      MOV      $TRPAD(R0),R0   ;;INDEX TO TABLE
7890 037732 000200      RTS      R0              ;;GO TO ROUTINE
7891
7892
7893      ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
7894
7895 037734 011646      $TRAP2: MOV      (SP),-(SP)  ;;MOVE THE PC DOWN
7896 037736 016666 000004 000002      MOV      4(SP),2(SP)    ;;MOVE THE PSW DOWN
7897 037744 000002      RTI                    ;;RESTORE THE PSW
7898
7899      .SBTTL TRAP TABLE
7900
7901      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
7902      ;*BY THE "TRAP" INSTRUCTION.
7903
7904      :      ROUTINE
7905      :      -----
7906 037746 037734      $TRPAD: .WORD  $TRAP2
7907 037750 035044      $TYPE   ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
7908 037752 035424      $TYPOC  ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
7909 037754 035400      $TYPOS  ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
7910 037756 035440      $TYPON  ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
7911 037760 035626      $TYPDS  ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
7912
7913 037762 036452      $GTSWR  ;;CALL=GTSWR     TRAP+6(104406)  GET SOFT-SWR SETTING
7914
7915 037764 036362      $CKSWR  ;;CALL=CKSWR     TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
7916 037766 036724      $RDCHR  ;;CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
7917 037770 037014      $RDLIN  ;;CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
7918 037772 037322      $RDOCT  ;;CALL=RDOCT     TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
7919 037774 037462      $SAVREG ;;CALL=SAVREG        TRAP+13(104413) SAVE R0-R5 ROUTINE
7920 037776 037520      $RESREG ;;CALL=RESREG        TRAP+14(104414) RESTORE R0-R5 ROUTINE
  
```

CZR6ECO RK611 DSALS CTRL PRYS
CZR6EC.P11 14-SEP-81 13:57

MACV11 30(1046) 14-SEP-81 15:14 PAGE 152
TRAP TABLE

1 12

SEQ 0151

CZ
CZ

7921 040000 033774

SCOPI8 ;:CALL=SCOPI TRAP+15(104415) INTERNAL LOOP ON ERROR

					.SBITL DATA TABLE FOR PRINT OUT	
7922						
7923						
7924	040002	001254	002310		DT000:	.WORD \$TESTN,TRAPPC
7925	040006	001254	001116	002220	DT0004:	.WORD \$TESTN,\$ERRPC,E.CS1,T.CS1,T.CS2,T.ER,T.DS
7926	040014	002160	002170	002174		
7927	040022	002172				
7928	040024	001254	001116	002244	DT0005:	.WORD \$TESTN,\$ERRPC,E.MR1,T.MR1
7929	040032	002204				
7930	040034	001254	001116	002246	DT0006:	.WORD \$TESTN,\$ERRPC,E.MR2,T.MR2
7931	040042	002206				
7932	040044	001254	001116	002250	DT0007:	.WORD \$TESTN,\$ERRPC,E.MR3,T.MR3
7933	040052	002210				
7934	040054	001254	001116	002244	DT0010:	.WORD \$TESTN,\$ERRPC,E.MR1,T.MR1,PR.BIT,M1.BIT,BITCNT
7935	040062	002204	002320	002322		
7936	040070	002326				
7937	040072	001254	001116	002220	DT0011:	.WORD \$TESTN,\$ERRPC,E.CS1,T.CS1,PR.BIT,M1.BIT,BITCNT,T.CS2,T.ER,T.DS
7938	040100	002160	002320	002322		
7939	040106	002326	002170	002174		
7940	040114	002172				
7941	040116	001254	001116	002240	DT0014:	.WORD \$TESTN,\$ERRPC,E.DCYL,T.DCYL,E.DA,T.DA
7942	040124	002200	002226	002166		
7943	040132	001254	001116	002244	DT0017:	.WORD \$TESTN,\$ERRPC,E.MR1,T.MR1,P1.BIT,PR.BIT,M1.BIT,M2.BIT,BITCNT
7944	040140	002204	002316	002320		
7945	040146	002322	002324	002326		
7946	040154	001254	001116	002254	DT0021:	.WORD \$TESTN,\$ERRPC,E.ECPT,T.ECPT,BITCNT
7947	040162	002214	002326			
7948	040166	001254	001116	002160	DT0026:	.WORD \$TESTN,\$ERRPC,T.CS1,T.CS2,T.ER,T.DS,T.WC,T.DCYL,T.DA
7949	040174	002170	002174	002172		
7950	040202	002162	002200	002166		
7951	040210	001254	001116		DT0037:	.WORD \$TESTN,\$ERRPC
7952	040214	001162	001164	001166	DT0040:	.WORD \$REG0,\$REG1,\$REG2
7953	040222	001254	001116	002254	DT0041:	.WORD \$TESTN,\$ERRPC,E.ECPT,T.ECPT,E.ECPS,T.ECPS,BITCNT
7954	040230	002214	002252	002212		
7955	040236	002326				
7956	040240	001254	001116	002252	DT0043:	.WORD \$TESTN,\$ERRPC,E.ECPS,T.ECPS,BITCNT
7957	040246	002214	002326			
7958	040252	001254	001116	002234	DT0045:	.WORD \$TESTN,\$ERRPC,E.ER,T.ER
7959	040260	002174				
7960	040262	001254	001116	002224	DT0046:	.WORD \$TESTN,\$ERRPC,E.BA,T.BA
7961	040270	002164				
7962	040272	001254	001116	002222	DT0062:	.WORD \$TESTN,\$ERRPC,E.WC,T.WC
7963	040300	002162				
7964	040302	001254	001116	001162	DT0063:	.WORD \$TESTN,\$ERRPC,\$REG0,\$REG3,\$REG1,\$REG4,\$REG2
7965	040310	001170	001164	001172		
7966	040316	001166				

7967
7968
7969 040320 000001
7970 040322 002 000
7971 040324 000006
7972 040326 000 000
7973 040330 041524
7974 040332 000 000
7975 040334 041542

.SBTTL DATA FORMAT FOR PRINT OUT
DF000: .WORD 1
.BYTE 2.0
DF0004: .WORD 6
.BYTE 0.0
.WORD DH000A
.BYTE 0.0
.WORD DH000B

7976	040336	002	000	.BYTE	2,0
7977	040340	041606		.WORD	DH004
7978	040342	000	000	.BYTE	0,0
7979	040344	041625		.WORD	DH004A
7980	040346	002	000	.BYTE	2,0
7981	040350	041715		.WORD	DH004E
7982	040352	003	000	.BYTE	3,0
7983	040354	000005		DF0005: .WORD	5
7984	040356	000	000	.BYTE	0,0
7985	040360	041524		.WORD	DH000A
7986	040362	000	000	.BYTE	0,0
7987	040364	041542		.WORD	DH000B
7988	040366	002	000	.BYTE	2,0
7989	040370	041606		.WORD	DH004
7990	040372	000	000	.BYTE	0,0
7991	040374	041643		.WORD	DH004B
7992	040376	002	000	.BYTE	2,0
7993	040400	000005		DF0006: .WORD	5
7994	040402	000	000	.BYTE	0,0
7995	040404	041524		.WORD	DH000A
7996	040406	000	000	.BYTE	0,0
7997	040410	041542		.WORD	DH000B
7998	040412	002	000	.BYTE	2,0
7999	040414	041606		.WORD	DH004
8000	040416	000	000	.BYTE	0,0
8001	040420	041661		.WORD	DH004C
8002	040422	002	000	.BYTE	2,0
8003	040424	000005		DF0007: .WORD	5
8004	040426	000	000	.BYTE	0,0
8005	040430	041524		.WORD	DH000A
8006	040432	000	000	.BYTE	0,0
8007	040434	041542		.WORD	DH000B
8008	040436	002	000	.BYTE	2,0
8009	040440	041606		.WORD	DH004
8010	040442	000	000	.BYTE	0,0
8011	040444	041677		.WORD	DH004D
8012	040446	002	000	.BYTE	2,0
8013	040450	000005		DF0010: .WORD	5
8014	040452	000	000	.BYTE	0,0
8015	040454	041524		.WORD	DH000A
8016	040456	000	000	.BYTE	0,0
8017	040460	041542		.WORD	DH000B
8018	040462	002	000	.BYTE	2,0
8019	040464	042011		.WORD	DH010
8020	040466	000	000	.BYTE	0,0
8021	040470	042055		.WORD	DH010A
8022	040472	005	000	.BYTE	5,0
8023	040474	000006		DF0011: .WORD	6
8024	040476	000	000	.BYTE	0,0
8025	040500	041524		.WORD	DH000A
8026	040502	000	000	.BYTE	0,0
8027	040504	041542		.WORD	DH000B
8028	040506	002	000	.BYTE	2,0
8029	040510	042011		.WORD	DH010
8030	040512	000	000	.BYTE	0,0
8031	040514	042123		.WORD	DH010B

8032	040516	005	000		.BYTE	5.0
8033	040520	041715			.WORD	DH004E
8034	040522	003	000		.BYTE	3.0
8035	040524	000005		DF0014:	.WORD	5
8036	040526	000	000		.BYTE	0.0
8037	040530	041524			.WORD	DH000A
8038	040532	000	000		.BYTE	0.0
8039	040534	041542			.WORD	DH000B
8040	040536	002	000		.BYTE	2.0
8041	040540	042171			.WORD	DH014
8042	040542	000	000		.BYTE	0.0
8043	040544	042230			.WORD	DH014A
8044	040546	004	000		.BYTE	4.0
8045	040550	000005		DF0017:	.WORD	5
8046	040552	000	000		.BYTE	0.0
8047	040554	041524			.WORD	DH000A
8048	040556	000	000		.BYTE	0.0
8049	040560	041542			.WORD	DH000B
8050	040562	002	000		.BYTE	2.0
8051	040564	042265			.WORD	DH0017
8052	040566	000	000		.BYTE	0.0
8053	040570	042351			.WORD	DH017A
8054	040572	007	000		.BYTE	7.0
8055	040574	000005		DF0021:	.WORD	5
8056	040576	000	000		.BYTE	0.0
8057	040600	041524			.WORD	DH000A
8058	040602	000	000		.BYTE	0.0
8059	040604	041542			.WORD	DH000B
8060	040606	002	000		.BYTE	2.0
8061	040610	042437			.WORD	DH0021
8062	040612	000	000		.BYTE	0.0
8063	040614	042463			.WORD	DH021A
8064	040616	003	000		.BYTE	3.0
8065	040620	000004		DF0026:	.WORD	4
8066	040622	000	000		.BYTE	0.0
8067	040624	041524			.WORD	DH000A
8068	040626	000	000		.BYTE	0.0
8069	040630	041542			.WORD	DH000B
8070	040632	002	000		.BYTE	2.0
8071	040634	042537			.WORD	DH0026
8072	040636	007	000		.BYTE	7.0
8073	040640	000003		DF0027:	.WORD	3
8074	040642	00	000		.BYTE	0.0
8075	040644	041542			.WORD	DH000B
8076	040646	002	000		.BYTE	2.0
8077	040650	042537			.WORD	DH0026
8078	040652	007	000		.BYTE	7.0
8079	040654	000004		DF0037:	.WORD	4
8080	040656	000	000		.BYTE	0.0
8081	040660	041542			.WORD	DH000B
8082	040662	002	000		.BYTE	2.0
8083	040664	042624			.WORD	DH037A
8084	040666	000	000		.BYTE	0.0
8085	040670	042651			.WORD	DH037B
8086	040672	003	000		.BYTE	3.0
8087	040674	000000		DF0040:	.WORD	0

8088	040676	007	000		.BYTE	3,0
8089	040700	000004		DF0041:	.WORD	4
8090	040702	000	000		.BYTE	0,0
8091	040704	041542			.WORD	DH000B
8092	040706	002	000		.BYTE	2,0
8093	040710	042700			.WORD	DH0041
8094	040712	000	000		.BYTE	0,0
8095	040714	042744			.WORD	DH041A
8096	040716	005	000		.BYTE	5,0
8097	040720	000004		DF0042:	.WORD	4
8098	040722	000	000		.BYTE	0,0
8099	040724	041542			.WORD	DH000B
8100	040726	002	000		.BYTE	2,0
8101	040730	042437			.WORD	DH0021
8102	040732	000	000		.BYTE	0,0
8103	040734	042463			.WORD	DH021A
8104	040736	003	000		.BYTE	3,0
8105	040740	000004		DF0043:	.WORD	4
8106	040742	000	000		.BYTE	0,0
8107	040744	041542			.WORD	DH000B
8108	040746	002	000		.BYTE	2,0
8109	040750	042437			.WORD	DH0021
8110	040752	000	000		.BYTE	0,0
8111	040754	042511			.WORD	DH021B
8112	040756	003	000		.BYTE	3,0
8113	040760	000005		DF0045:	.WORD	5
8114	040762	000	000		.BYTE	0,0
8115	040764	041524			.WORD	DH000A
8116	040766	000	000		.BYTE	0,0
8117	040770	041542			.WORD	DH000B
8118	040772	002	000		.BYTE	2,0
8119	040774	041606			.WORD	DH004
8120	040776	000	000		.BYTE	0,0
8121	041000	041742			.WORD	DH004F
8122	041002	002	000		.BYTE	2,0
8123	041004	000004		DF0046:	.WORD	4
8124	041006	000	000		.BYTE	0,0
8125	041010	041542			.WORD	DH000B
8126	041012	002	000		.BYTE	2,0
8127	041014	041606			.WORD	DH004
8128	041016	000	000		.BYTE	0,0
8129	041020	041757			.WORD	DH004G
8130	041022	002	000		.BYTE	2,0
8131	041024	000005		DF0050:	.WORD	5
8132	041026	000	000		.BYTE	0,0
8133	041030	041524			DH000A	
8134	041032	000	000		.BYTE	0,0
8135	041034	041542			.WORD	DH000B
8136	041036	002	000		.BYTE	2,0
8137	041040	041606			.WORD	DH004
8138	041042	000	000		.BYTE	0,0
8139	041044	041625			.WORD	DH004A
8140	041046	002	000		.BYTE	2,0
8141	041050	000004		DF0060:	.WORD	4
8142	041052	000	000		.BYTE	0,0
8143	041054	041542			.WORD	DH000B


```
8179 .SBTTL ASCII MESSAGES
8180
8181 041164 005015 045522 030466 OPR001: .ASCIZ <15><2>/RK611 VECTOR ADDRESS ( /
8182 041172 020061 042526 052103
8183 041200 051117 040440 042104
8184 041206 042522 051523 024040
8185 041214 000040
8186 041216 024440 036440 000040 OPR002: .ASCIZ / ) = /
8187 041224 045522 030466 020061 OPR003: .ASCIZ /RK611 VECTOR ADDRESS ( /
8188 041232 042526 052103 051117
8189 041240 040440 042104 042522
8190 041246 051523 024040 000040
8191 041254 045522 030466 020061 OPR004: .ASCIZ /RK611 PRIORITY ( /
8192 041262 051120 047511 044522
8193 041270 054524 024040 000040
8194 041276 005015 047062 020104 OPR006: .ASCIZ <15><12>/2ND PASS RUN TIME APPROX 4 MINUTES/<15><12>
8195 041304 040520 051523 051040
8196 041312 047125 052040 046511
8197 041320 020105 050101 051120
8198 041326 054117 032040 046440
8199 041334 047111 052125 051505
8200 041342 005015 000
8201 041345 015 025012 025052 OPR007: .ASCIZ <15><12>/***** PROGRAM HALTED *****/<15><12>
8202 041352 025052 020040 050040
8203 041360 047522 051107 046501
8204 041366 044040 046101 042524
8205 041374 020104 020040 025052
8206 041402 025052 006452 000012
8207 041410 020040 000
8208 041413 015 050012 047522 SPACE2: .ASCIZ / /
8209 041420 051107 046501 040440 ABORT: .ASCIZ <15><12>/PROGRAM ABORTED BECAUSE ERROR THRESHOLD EXCEEDED/<15><12>
8210 041426 047502 052122 042105
8211 041434 041040 041505 052501
8212 041442 042523 042440 051122
8213 041450 051117 052040 051110
8214 041456 051505 047510 042114
8215 041464 042440 041530 042505
8216 041472 042504 006504 000012
8217 041500 005015 042524 052123 TSTBY1: .ASCIZ <15><12>/TEST /
8218 041506 000040
8219 041510 041040 050131 051501 TSTBY2: .ASCIZ / BYPASSED/<15><12>
8220 041516 042523 006504 000012
```


8432	043624	040504	044440	020116	
8433	043632	051105	047522	000122	
8434	043640	051105	047522	020122	EM5015: .ASCIZ /ERROR ATTEMPTING TO READ GAP/
8435	043646	052101	042524	050115	
8436	043654	044524	043516	052040	
8437	043662	020117	042522	042101	
8438	043670	043440	050101	000	
8439	043675	105	051122	051117	EM5020: .ASCIZ /ERROR ATTEMPTING TO WRITE DATA SYNC (PREAMBLE)/
8440	043702	040440	052124	046505	
8441	043710	052120	047111	020107	
8442	043716	047524	053440	044522	
8443	043724	042524	042040	052101	
8444	043732	020101	054523	041516	
8445	043740	024040	051120	040505	
8446	043746	041115	042514	000051	
8447	043754	051105	047522	020122	EM5021: .ASCIZ /ERROR ATTEMPTING TO WRITE DATA OR ECC/
8448	043762	052101	042524	050115	
8449	043770	044524	043516	052040	
8450	043776	020117	051127	052111	
8451	044004	020105	040504	040524	
8452	044012	047440	020122	041505	
8453	044020	000103			
8454	044022	051105	047522	020122	E5021A: .ASCIZ /ERROR IN ECC GENERATION/
8455	044030	047111	042440	041503	
8456	044036	043440	047105	051105	
8457	044044	052101	047511	000116	
8458	044052	051105	047522	020122	EM5023: .ASCIZ /ERROR BETWEEN SECTORS ATTEMPTING MULTI-SECTOR WRITE/
8459	044060	042502	053524	042505	
8460	044066	020116	042523	052103	
8461	044074	051117	020123	052101	
8462	044102	042524	050115	044524	
8463	044110	043516	046440	046125	
8464	044116	044524	051455	041505	
8465	044124	047524	020122	051127	
8466	044132	052111	000105		
8467	044136	051115	020061	047111	EMW1: .ASCIZ /MR1 INCORRECT ON 1ST UPWARD TRANSITION OF MAINT CLOCK/
8468	044144	047503	051122	041505	
8469	044152	020124	047117	030440	
8470	044160	052123	052440	053520	
8471	044166	051101	020104	051124	
8472	044174	047101	044523	044524	
8473	044202	047117	047440	020106	
8474	044210	040515	047111	020124	
8475	044216	046103	041517	000113	
8476	044224	051115	020061	047111	EMW2: .ASCIZ /MR1 INCORRECT ON 1ST DOWNWARD TRANSITION OF MAINT CLOCK/
8477	044232	047503	051122	041505	
8478	044240	020124	047117	030440	
8479	044246	052123	042040	053517	
8480	044254	053516	051101	020104	
8481	044262	051124	047101	044523	
8482	044270	044524	047117	047440	
8483	044276	020106	040515	047111	
8484	044304	020124	046103	041517	
8485	044312	000113			
8486	044314	051115	020061	047111	EMW3: .ASCIZ /MR1 INCORRECT ON 2ND UPWARD TRANSITION OF MAINT CLOCK/
8487	044322	047503	051122	041505	

8544	045034	046106	053517	053440	
8545	045042	044522	044524	043516	
8546	045050	046040	051501	020124	
8547	045056	044120	051531	041511	
8548	045064	046101	040440	042104	
8549	045072	042522	051523	000	
8550	045077	125	042516	050130	EM5031: .ASCIZ /UNEXPECTED ERROR WHILE WRITING LAST PHYSICAL ADDRESS/
8551	045104	041505	042524	020104	
8552	045112	051105	047522	020122	
8553	045120	044127	046111	020105	
8554	045126	051127	052111	047111	
8555	045134	020107	040514	052123	
8556	045142	050040	054510	044523	
8557	045150	040503	020114	042101	
8558	045156	051104	051505	000123	
8559	045164	051105	047522	020122	EM5032: .ASCIZ /ERROR ATTEMPTING TO READ DATA SYNC (PREAMBLE)/
8560	045172	052101	042524	050115	
8561	045200	044524	043516	052040	
8562	045206	020117	042522	042101	
8563	045214	042040	052101	020101	
8564	045222	054523	041516	024040	
8565	045230	051120	040505	041115	
8566	045236	042514	000051		
8567	045242	051105	047522	020122	EM5034: .ASCIZ /ERROR ATTEMPTING TO READ DATA OR ECC/
8568	045250	052101	042524	050115	
8569	045256	044524	043516	052040	
8570	045264	020117	042522	042101	
8571	045272	042040	052101	020101	
8572	045300	051117	042440	041503	
8573	045306	000			
8574	045307	105	041503	042440	EM5035: .ASCIZ /ECC ERROR IN READ DATA OPERATION/
8575	045314	051122	051117	044440	
8576	045322	020116	042522	042101	
8577	045330	042040	052101	020101	
8578	045336	050117	051105	052101	
8579	045344	047511	000116		
8580	045350	040504	040524	041440	EM5037: .ASCIZ /DATA COMPARE ERROR AFTER INPUT NPR TRANSFER/
8581	045356	046517	040520	042522	
8582	045364	042440	051122	051117	
8583	045372	040440	052106	051105	
8584	045400	044440	050116	052125	
8585	045406	047040	051120	052040	
8586	045414	040522	051516	042506	
8587	045422	000122			
8588	045424	041505	020103	042522	EM5041: .ASCIZ /ECC REGISTER NOT CORRECT AFTER ECC READ IN READ DATA/
8589	045432	044507	052123	051105	
8590	045440	047040	052117	041440	
8591	045446	051117	042522	052103	
8592	045454	040440	052106	051105	
8593	045462	042440	041503	051040	
8594	045470	040505	020104	047111	
8595	045476	051040	040505	020104	
8596	045504	040504	040524	000	
8597	045511	105	051122	051117	EM5042: .ASCIZ /ERROR IN ECC PATTERN CALCULATION AFTER ECC ERROR/
8598	045516	044440	020116	041505	
8599	045524	020103	040520	052124	

CZ
Z
PA
PA
PA
PA
PA
PA
PC
PC
PG
PI
PI
PI
PR

PR
PR
PR
PR
PR
PR
PR
FR
PS
PS
PW
PI

RD

RD
RD

RD
RD
RD
RD
RD
RD
RD
RE
RE
RE
RE
RK
RK
RK
RK
RK
RK

8768 047050 177777
8769 047052 052525
8770 047054 177252
8771 047056 177252
8772 047060 172765
8773 047062 172765
8774
8775
8776
8777
8778
8779 047064
8780 047064 072307
8781 047066 135143
8782 047070 156461
8783 047072 167230
8784 047074 073514
8785 047076 035646
8786 047100 016723
8787 047102 107351
8788 047104 143564
8789 047106 061672
8790 047110 030735
8791 047112 114356
8792 047114 046167
8793 047116 123073
8794 047120 151453
8795 047122 164616
8796
8797
8798
8799 047124 000402
8800 050130 000402
8801 000001

* PATTERN 5 - ALTERNATING 1'S AND 0'S

* PATTERN 6 - COMPOSIT ROTATING

PAT6:

072307
135143
156461
167230
073514
035646
016723
107351
143564
061672
030735
114356
046167
123073
151453
164616

* PATTERN 7 - ALL ONES

IBUFF: .BLKW 402
OBUFF: .BLKW 402
.END

CZ
CZ

SA
SA
SA
SA
SA
SA
SA

SB
SB

SB
SB

SB
SC

SC
SC

SC
SC

SC
SC

SC
SC

SC
SC

SC
SC

SC
SC

SC
SC

SC
SD

SD
SD

SD
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

SE
SE

ABASE = 177440	1054#	1307	1348
ABORT = 041413	7218	8208#	
ACDW1 = 000000	1307		
ACDW2 = 000000	1307		
ACLO = 000010	1149#		
ACPUOP = 000000	1307	1322	
ADDW0 = 000000	1307		
ADDW1 = 000000	1307		
ADDW10 = 000000	1307		
ADDW11 = 000000	1307		
ADDW12 = 000000	1307		
ADDW13 = 000000	1307		
ADDW14 = 000000	1307		
ADDW15 = 000000	1307		
ADDW2 = 000000	1307		
ADDW3 = 000000	1307		
ADDW4 = 000000	1307		
ADDW5 = 000000	1307		
ADDW6 = 000000	1307		
ADDW7 = 000000	1307		
ADDW8 = 000000	1307		
ADDW9 = 000000	1307		
ADEVCT = 000000	1307	1313	
ADEVN = 000000	1307	1349	
AENV = 000000	1307	1318	
AENVN = 000000	1307	1319	
AFATAL = 000000	1307	1310	
AMADR1 = 000000	1307	1335	
AMADR2 = 000000	1307	1339	
AMADR3 = 000000	1307	1342	
AMADR4 = 000000	1307	1345	
AMAMS1 = 000000	1307	1329	
AMAMS2 = 000000	1307	1337	
AMAMS3 = 000000	1307	1340	
AMAMS4 = 000000	1307	1343	
AMSGAD = 000000	1307	1315	
AMSGLG = 000000	1307	1316	
AMSGTY = 000000	1307	1309	
AMTYP1 = 000000	1307	1330	
AMTYP2 = 000000	1307	1338	
AMTYP3 = 000000	1307	1341	
AMTYP4 = 000000	1307	1344	
APASS = 000000	1307	1312	
APRIOR = 000005	1053#	1307	
APTCSU = 000040	7093#	7274	
APTENV = 000001	7049	7091#	7125 7267
APTSIZ = 000200	1806	7090#	
APTSPO = 000100	7051	7092#	7269
ASWREG = 000000	1307	1320	
ATESTN = 000000	1307	1311	
AUNIT = 000000	1307	1314	
AUSWR = 000000	1307	1321	
AVECT1 = 120210	1052#	1307	1346
AVECT2 = 000000	1307	1347	
BADPAR = 002334	1740#	6783*	
BAI = 000020	1112#		

DT0021	040154	1463	1535	1565	1625	7946#								
DT0026	040166	1493	1499	1505	1511	7948#								
DT0037	040210	1547	1649	7951#										
DT0040	040214	1553	7952#											
DT0041	040222	1559	7953#											
DT0043	040240	1571	7956#											
DT0045	040252	1583	1607	7958#										
DT0046	040262	1589	1613	1655	7960#									
DT0062	040272	1661	7962#											
DT0063	040302	1667	7964#											
DWGPSN	025204	1950	2068	2185	2302	2423	2538	2621	5732#					
DWRITE	025604	1957	2075	2192	2309	2430	2545	2628	5837#					
DWRTCK	027176	4521	4625	4750	4841	4942	5039	5132	5245	6165#				
ECCGEN	030462	2937	3067	3200	3331	3462	3593	3725	4048	4179	4641	5868	6215	6246
		6268	6393#											
ECCMI	002340	1742#	5844*	5904	6185*	6250	6401	6407	6412*	6420	6421*	6422*	6423	
ECCLO	002342	1743#	5843*	5932	6186*	6251	6411*	6416	6417*	6418*				
ECCSRC	002344	1744#	2674*	2780*	2781*	2887*	3017*	3150*	3283*	3413*	3544*	3675*	3787*	3891*
		3892*	3998*	4130*	4243*	4338*	4472*	4473*	4590*	4702*	4793*	4898*	4995*	5087*
		5210*	6248											
ECCW =	020000	1173#	5394	5464	5489	5584	5751	5789	5887	5925	6029	6063	6277	6280
ECCXOR	002336	1741#	6403*	6409*	6413									
ECH =	000100	1133#	2974	3104	3237	3368	3499	3501	3630	3632	3762	4085	4216	4218
		4678												
ECPATX	002346	1745#	2714*	2821*	2919*	3049*	3182*	3313*	3444*	3575*	3707*	3827*	3932*	4030*
		4161*	4283*	4378*	4519*	4623*	4748*	4839*	4940*	5037*	5130*	5243*	6293	
ECP0SX	002350	1746#	2715*	2822*	2920*	2940*	2941	2953*	2961*	3050*	3070*	3071	3083*	3091*
		3183*	3203*	3204	3216*	3224*	3314*	3334*	3335	3347*	3355*	3445*	3465*	3466
		3478*	3486*	3576*	3596*	3597	3609*	3617*	3708*	3728*	3729	3741*	3749*	3828*
		3933*	4031*	4051*	4052	4064*	4072*	4162*	4182*	4183	4195*	4203*	4284*	4379*
		4520*	4624*	4644*	4645	4657*	4665*	4749*	4840*	4941*	5038*	5131*	5244*	6285
		6298*	6301											
EH026	001602	1492#	2649*											
EH030	001622	1504#	2563*											
EH031	001632	1510#	2567*											
EMTVEC=	000030	1008#	1777*	1778*										
EMW	001510	1449#	1949*	1956*	2067*	2074*	2184*	2191*	2301*	2308*	2422*	2429*	2537*	2544*
		2620*	2627*	6446*	6476*	6490*	6505*	6531*	6543*	6555*				
EMW1	044136	6446	8467#											
EMW2	044224	6476	6531	8476#										
EMW3	044314	6490	6543	8486#										
EMW4	044402	6505	6555	8495#										
EM000	043176	1368	8378#											
EM5004	043243	1383	1389	1395	1401	1587	8385#							
EM5010	043455	1407	1413	1443	8413#									
EM5011	043535	1419	1425	1431	8422#									
EM5015	043640	1437	8434#											
EM5020	043675	1455	1949	2067	2184	2301	2422	2537	2620	8439#				
EM5021	043754	1461	1467	1956	2074	2191	2308	2429	2544	2627	8447#			
EM5023	044052	1473	1479	1485	8458#									
EM5026	044472	1491	8505#											
EM5027	044632	1497	8522#											
EM5030	045004	1503	8540#											
EM5031	045077	1509	8550#											
EM5032	045164	1515	1521	8559#										
EM5034	045242	1527	1539	1635	8567#									

CI
CC
EE
E
F
GI
LI
M

E5004E	043403	1582	1606	8404#										
E5004F	043421	1654	8407#											
E5004G	043437	1660	1666	8410#										
E5011A	043610	1432	1594	8430#										
E5021A	044022	1462	1534	1624	8454#									
E5026A	044545	2563	2567	2649	8513#									
E5026B	044716	8531#												
FMTE =	000020	1131#												
FSBLVV	032444	1969	1983	2087	2101	2204	2218	2321	2335	2442	2456	5378	5390	6757#
GETREG	032356	2557	2640	2841	2966	3096	3229	3360	3491	3622	3754	3951	4077	4208
		4308	4424	4438	4538	4670	4767	4864	4957	5054	5147	5261	5395	5511
		5667	5776	5807	5947	6054	6099	6328	5721#					
GNS =	***** U	1193	1830	5307	5314	7907	7908	7909	7910	7911	7913	7915	7916	7917
		7918	7919	7920	7921									
GO =	000001	1094#	2842	2969	3099	3232	3363	3494	3625	3757	3953	4080	4211	4310
		4439	4540	4673	4769	4866								
GTSWR =	104406	1825	7913#											
HIBITS	002354	1748#	2675*	2782*	2869*	2999*	3132*	3265*	3395*	3526*	3657*	3788*	3893*	3980*
		4112*	4244*	4339*	4469*	4566*	4701*	4792*	4897*	4994*	5096*	5154	5163	5165*
		5185*	6232											
HT =	000011	91#	7282	7340										
HVRC =	000400	1135#												
IBUFF	047124	2680	2727	2787	2874	3004	3137	3270	3400	3531	3662	3793	3840	3898
		3985	4117	4249	4296	4299	4344	4446	4478	4488	4571	4581	4707	4717
		4798	4808	4852	4855	4903	4909*	4910	4961	4967*	5000	5006*	5007	5058
		5064*	5065*	5092	5097*	5151	5191	5201	5265	5270	6609	8799#		
IDAE =	002000	1137#												
IE =	000100	1095#												
ILF =	000001	1127#												
INTR =	000300	1090#												
IOTVEC =	000020	1006#	1775*	1776*										
IR =	000100	1114#												
KIPAR0 =	172340	1039#	685	6865										
KIPAR1 =	172342	1040#												
KIPAR2 =	172344	1041#												
KIPAR3 =	172346	1042#												
KIPAR4 =	172350	1043#												
KIPAR5 =	172352	1044#												
KIPAR6 =	172354	1045#												
KIPAR7 =	172356	1046#												
KIPDR0 =	172300	1028#												
KIPDR1 =	172302	1029#												
KIPDR2 =	172304	1030#												
KIPDR3 =	172306	1031#												
KIPDR4 =	172310	1032#												
KIPDR5 =	172312	1033#												
KIPDR6 =	172314	1034#												
KIPDR7 =	172316	1035#												
LF =	000012	912#	7334	7340										
LOADRK	032230	1919	2037	2154	2271	2392	2507	2590	2676	2783	2870	3000	3133	3266
		3396	3527	3658	3789	3894	3981	4113	4245	4340	4474	4567	4703	4794
		4899	4996	5088	5187	6693#								
L.ASOF	002272	1715#	6698*	6713										
L.BA	002264	1710#	6695*	6708										
L.CS1	002260	1708#	1979	1988	2097	2106	2214	2223	2331	2340	2452	2461	2967	3097
		3230	3361	3492	3623	3755	3952	4078	4209	4309	4425	4539	4671	4768

SW12 = 010000	948#	7214								
SW13 = 020000	947#									
SW14 = 040000	946#									
SW15 = 100000	945#									
SW2 = 000004	968#									
SW3 = 000010	967#									
SW4 = 000020	966#									
SW5 = 000040	965#									
SW6 = 000100	964#									
SW7 = 000200	963#									
SW8 = 000400	962#									
SW9 = 001000	961#	6450	6480	6494	6509	6535	6547	6559	6601	7031
S.CLR = 000400	1183#									
S.FMT = 001000	1184#									
S.PACK= 004000	1186#									
S.RECL= 000040	1180#									
S.RTC = 000200	1182#									
S.SEEK= 000020	1179#									
S.STSP= 000100	1181#									
S.UNLD= 002000	1185#									
TBITVE= 000014	1003#									
TKVEC = 000060	1010#	7506*	7507*							
TPVEC = 000064	1011#									
TRAPPC 002310	1727#	6598*	7924							
TRAPVE= 000034	1009#	1779*	1780*							
TRTVEC= 000014	1004#									
TSTBY1 041500	8217#									
TSTBY2 041510	8219#									
TST1 003412	1891	1910#	6999							
TST10 006622	2668#	7005								
TST11 007136	2775#	7006								
TST12 007430	2863#	7007								
TST13 010156	2994#	7008								
TST14 010704	3126#	7009								
TST15 011432	3258#	7010								
TST16 012160	3389#	7011								
TST17 012714	3520#	7012								
TST2 004030	1992	1997	2002	2030#	6999					
TST20 013450	3651#	7013								
TST21 014176	3781#	7014								
TST22 014512	3886#	7015								
TST23 015004	3974#	7016								
TST24 015532	4106#	7017								
TST25 016266	4237#	7018								
TST26 016602	4332#	7019								
TST27 017264	4422	4429	4444	4467#	7020					
TST3 004442	2110	2115	2120	2147#	7000					
TST30 017600	4562#	7021								
TST31 020350	4697#	7022								
TST32 020660	4788#	7023								
TST33 021216	4893#	7024								
TST34 021546	4990#	7025								
TST35 022102	5083#	7026								
TST36 022456	5182#	7027								
TST4 005054	2227	2232	2237	2264#	7001					
TST5 005466	2344	2349	2354	2382#	7002					

\$POWER	037704	7867	7874#																	
\$PWRCT	037700	7858*	7859*	7860*	7862*	7873#														
\$PWRDN	037556	1781	7849#	7864																
\$PWRUP	037604	7850	7858#																	
\$QUES	001244	1300#	7148	7340	7649	7720	7737	7795	7798											
\$RDCHR	036724	7662#	7916																	
\$RDDEC=	***** U	7919																		
\$RDLIN	037014	7685#	7917																	
\$RDOCT	037322	7759#	7918																	
\$RDSZ =	000010	7678#																		
\$REGAD	001160	1274#																		
\$REGO	001162	1276#	2746*	3859*	4960*	5057*	5150*	5264*	7952	7964										
\$REG1	001164	1277#	2747*	3860*	4961*	5058*	5151*	5265*	7952	7964										
\$REG2	001166	1278#	2748*	3861*	4962*	5059*	5152*	5266*	7952	7964										
\$REG3	001170	1279#	5153*	7964																
\$REG4	001172	1280#	5154*	5156*	7964															
\$REG5	001174	1281#																		
\$REG6	001176	1282#																		
\$REG7	001200	1283#																		
\$SRE	037520	7831#	7920																	
\$PINAD	023310	5330#																		
\$R2A =	***** U	7921																		
\$SAVRE	037462	7815#	7919																	
\$SCOPE	033366	1775	6937#																	
\$SETUP=	000137	1752#	1774	1775	1777	1779	1781	1783	1784	1785	1787	1815	1818	5295						
		6938	7109	7135	7143	7532	7537	7538	7568	7744										
\$SIZE	032652	1885	6825#																	
\$SIZEX	033310	5890	6892	6907#																
\$STOP	033032	6849*	6852*	6857#	6885															
\$STUP =	177777	1752#																		
\$SVLAD	033626	6948	6987#																	
\$SVPC =	000220	1205#	1210																	
\$SWR =	167400	879#	891	895	896	897	898	899	900	901	902	1297	1298	1299						
		1784	1785	1787	1788	1911	2031	2148	2265	2383	2502	2585	2669	2776						
		2864	2995	3127	3259	3390	3521	3652	3782	3887	3975	4107	4238	4333						
		4468	4563	4698	4789	4894	4991	5084	5183	5290	5296	5323	5329	5331						
		6929	6930	6931	6932	6933	6939	6951	6953	6954	6967	6968	6969	6976						
		6977	6978	6990	6993	6996	7100	7101	7102	7103	7104	7113	7120	7132						
		7136	7148																	
\$SWREG	001272	1320#	1808																	
\$SWRMK-	000000	902	903	6933	6934	6957														
\$SWOBT	033700	5478	5604	5770	5983	6048	6378	6963	6997#											
\$TESTN	001254	1311#	5476	5602	5768	5981	6046	6376	6988*	7160*	7161*	7924	7925	7928						
		7930	7932	7934	7937	7941	7943	7946	7948	7951	7953	7956	7958	7960						
		7962	7964																	
\$TIMES	001234	1297#	1784*	1911*	2031*	2148*	2265*	2383*	2502*	2585*	2669*	2776*	2864*	2995*						
		3127*	3259*	3390*	3521*	3652*	3782*	3887*	3975*	4107*	4238*	4333*	4468*	4563*						
		4698*	4789*	4894*	4991*	5084*	5183*	5296*	6976*	6983	6986*	6996								
\$TKB	001146	1267#	7313	7320	7340	7487	7508	7519	7549	7577	7604									
\$TKCNT	036052	7488#	7503*	7538	7555*	7669	7671*													
\$TKINT	036062	1766	7503#	7529	7590															
\$TKQEN=	036061	7492#	7563	7674																
\$TKQIN	036054	7489#	7504*	7505	7561*	7562*	7563	7565*												
\$TKQOU	036056	7490#	7505*	7672	7673*	7674	7676*													
\$TKQSR	036060	7491#	7504	7565	7676															
\$TKS	001144	1266#	7311	7318	7340	7487	7509*	7545*	7547	7553*	7575	7591*	7601	7613*						

\$TKSRV	036132	7633*	7519#											
\$TMP0	001202	7506	1842*	1844	1846	1852*	1854	1857	1869*	1871	1873	1875*	1876*	1977*
		1284#	1878*	1881										
\$TMP1	001204	1285#	5583*	5620	5623*	5626	5629*	5632	5636*	5639	5662*			
\$TMP10	001222	1292#	6180*	6347										
\$TMP11	001224	1293#	6182*	6351	6355									
\$TMP12	001226	1294#	6190*	6194*	6240	6266								
\$TMP13	001230	1295#	6191*	6195*	6275									
\$TMP14	001232	1296#	6196*	6278	6291	6296								
\$TMP2	001206	1286#												
\$TMP3	001210	1287#	6176*	6371										
\$TMP4	001212	1288#	6178*	6369										
\$TMP5	001214	1289#	2934*	2957	2962*	3064*	3087	3092*	3197*	3220	3225*	3328*	3351	3356*
		3459*	3482	3487*	3590*	3613	3618*	3722*	3745	3750*	4045*	4068	4073*	4176*
		4199	4204*	4638*	4661	4666*	5855*	5882	5893	5897*	5900*	5903*	5921	5928
		5930*	6187*	6229	6233*	6235*	6247*	6314	6318*	6321	6325*			
\$TMP6	001216	1290#	5847*	5961	5964	5967	6172*	6335	6343					
\$TMP7	001220	1291#	2731*	2750	2752*	3844*	3863	3865*	5849*	5948	5957	6174*	6339	
\$TN =	000037	880#	891	1895	1911#	1992	1997	2002	2015	2031#	2110	2115	2120	2133
		2148#	2227	2232	2237	2250	2265#	2344	2349	2354	2368	2383#	2465	2470
		2475	2488	2502#	2572	2585#	2656	2669#	2763	2776#	2848	2864#	2980	2995#
		3111	3127#	3244	3259#	3375	3390#	3506	3521#	3637	3652#	3769	3782#	3874
		3887#	3959	3975#	4092	4107#	4225	4238#	4318	4333#	4422	4429	4444	4455
		4468#	4547	4563#	4685	4698#	4775	4789#	4874	4894#	4971	4991#	5069	5084#
		5169	5183#	6958	6998									
\$TPB	001152	1269#	7329*	7340										
\$TPFLG	001157	1273#	7261	7340										
\$TPS	001150	1268#	7327	7340										
\$TRAP	037712	1779	7884#											
\$TRAP2	037734	7895#	7906											
\$TRP =	000016	7899#	7908#	7909#	7910#	7911#	7912#	7913	7914#	7915	7916#	7917#	7918#	7919#
		7920#	7921#	7922#										
\$TRPAD	037746	7889	7906#											
\$STSM	001004	1233#												
\$STNM	001102	1246#	5295*	6928	6960*	6987*	6988	6993	6997	7112	7148	7160		
\$TTYIN	037250	7687	7688	7700	7718	7732	7736#							
\$TYPAN=	***** U	7912												
\$TYPDS	035626	7429#	7911											
\$TYPE	035044	7068	7261#	7899	7907									
\$YPEC	035256	7291	7298	7305	7310#	7635								
\$YPEX	035376	7333	7335	7338#										
\$YPOC	035424	7370#	7908											
\$YPON	035440	7369	7372#	7910										
\$YPOS	035400	7365#	7909											
\$UNIT	001262	1314#												
\$UNITM	001010	1235#												
\$USWR	001274	1321#												
\$VECT1	001320	1346#	1848	1856*	1857*	1860	1880*	1881*	1882	1884				
\$VECT2	001322	1347#												
\$XOFF =	000023	7315	7340											
\$XON =	000021	7322	7340	7521										
\$XTSTR	033400	6942#												
\$\$GET4=	000000	5323#												
\$\$SW08=	000037	6997#	6998	6999#	7000#	7001#	7002#	7003#	7004#	7005#	7006#	7007#	7008#	7009#
		7010#	7011#	7012#	7013#	7014#	7015#	7016#	7017#	7018#	7019#	7020#	7021#	7022#

BYPASS	1215#															
COMMEN	1#	1013#														
ENDCOM	1#	1013#														
ERROR	907#	1933	1934	1935	1936	1941	1942	1945	1946	1947	1951	1952	1953	1954	1958	
	1959	1960	1991	1996	2001	2007	2008	2011	2012	2013	2051	2052	2053	2054	2059	
	2060	2063	2064	2065	2069	2070	2071	2072	2076	2077	2078	2109	2114	2119	2125	
	2126	2129	2130	2131	2168	2169	2170	2171	2176	2177	2180	2181	2182	2186	2187	
	2188	2189	2193	2194	2195	2226	2231	2236	2242	2243	2246	2247	2248	2285	2286	
	2287	2288	2293	2294	2297	2298	2299	2303	2304	2305	2306	2310	2311	2312	2343	
	2348	2353	2359	2360	2363	2364	2365	2406	2407	2408	2409	2414	2415	2418	2419	
	2420	2424	2425	2426	2427	2431	2432	2433	2464	2469	2474	2480	2481	2484	2485	
	2486	2521	2522	2523	2524	2529	2530	2533	2534	2535	2539	2540	2541	2542	2546	
	2547	2548	2564	2568	2604	2605	2606	2607	2612	2613	2616	2617	2618	2622	2623	
	2624	2625	2629	2630	2631	2646	2650	2692	2693	2694	2695	2700	2701	2704	2705	
	2706	2709	2710	2711	2712	2718	2719	2720	2721	2722	2723	2753	2756	2794	2800	
	2801	2802	2807	2808	2811	2812	2813	2816	2817	2818	2819	2825	2826	2827	2828	
	2829	2830	2846	2897	2898	2899	2900	2905	2906	2909	2910	2911	2914	2915	2916	
	2917	2923	2924	2925	2926	2927	2928	2944	2947	2972	2977	3027	3028	3029	3030	
	3035	3036	3039	3040	3041	3044	3045	3046	3047	3053	3054	3055	3056	3057	3058	
	3074	3077	3102	3107	3160	3161	3162	3163	3168	3169	3172	3173	3174	3177	3178	
	3179	3180	3186	3187	3188	3189	3190	3191	3207	3210	3235	3240	3291	3292	3293	
	3294	3299	3300	3303	3304	3305	3308	3309	3310	3311	3317	3318	3319	3320	3321	
	3322	3338	3341	3366	3371	3422	3423	3424	3425	3430	3431	3434	3435	3436	3439	
	3440	3441	3442	3448	3449	3450	3451	3452	3453	3469	3472	3497	3502	3553	3554	
	3555	3556	3561	3562	3565	3566	3567	3570	3571	3572	3573	3579	3580	3581	3582	
	3583	3584	3600	3603	3628	3633	3685	3686	3687	3688	3693	3694	3697	3698	3699	
	3702	3703	3704	3705	3711	3712	3713	3714	3715	3716	3732	3735	3760	3765	3805	
	3806	3807	3808	3813	3814	3817	3818	3819	3822	3823	3824	3825	3831	3832	3833	
	3834	3835	3836	3866	3869	3910	3911	3912	3913	3918	3919	3922	3923	3924	3927	
	3928	3929	3930	3936	3937	3938	3939	3940	3941	3957	4008	4009	4010	4011	4016	
	4017	4020	4021	4022	4025	4026	4027	4028	4034	4035	4036	4037	4038	4039	4055	
	4058	4083	4088	4139	4140	4141	4142	4147	4148	4151	4152	4153	4156	4157	4158	
	4159	4165	4166	4167	4168	4169	4170	4186	4189	4214	4219	4261	4262	4263	4264	
	4269	4270	4273	4274	4275	4278	4279	4280	4281	4287	4288	4289	4290	4291	4292	
	4300	4314	4356	4357	4358	4359	4364	4365	4368	4369	4370	4373	4374	4375	4376	
	4382	4383	4384	4385	4386	4387	4401	4402	4405	4406	4415	4421	4428	4443	4449	
	4497	4498	4499	4500	4505	4506	4509	4510	4511	4514	4515	4516	4517	4523	4524	
	4525	4526	4527	4528	4544	4601	4602	4603	4604	4609	4610	4613	4614	4615	4618	
	4619	4620	4621	4627	4628	4629	4630	4631	4632	4648	4651	4676	4681	4726	4727	
	4728	4729	4734	4735	4738	4739	4740	4743	4744	4745	4746	4752	4753	4754	4755	
	4756	4757	4773	4817	4818	4819	4820	4825	4826	4829	4830	4831	4834	4835	4836	
	4837	4843	4844	4845	4846	4847	4848	4856	4870	4918	4919	4920	4921	4926	4927	
	4930	4931	4932	4935	4936	4937	4938	4944	4945	4946	4947	4948	4949	4963	5015	
	5016	5017	5018	5023	5024	5027	5028	5029	5032	5033	5034	5035	5041	5042	5043	
	5044	5045	5046	5060	5108	5109	5110	5111	5116	5117	5120	5121	5122	5125	5126	
	5127	5128	5134	5135	5136	5137	5138	5139	5159	5221	5222	5223	5224	5229	5230	
	5233	5234	5235	5238	5239	5240	5241	5247	5248	5249	5250	5251	5252	5267	5274	
	5280	6599														
ESCAPE	1#	1013#														
FSECWT	1720#	1919	2037	2154	2271	2392										
GETPRI	1#	1013#	6838													
GETSWR	1#	1013#	1818#													
LDLPER	1215#															
MSG	1895#	1897	2015#	2017	2133#	2135	2250#	2252	2368#	2370	2488#	2490	2572#	2574	2656#	
	2658	2763#	2765	2848#	2850	2980#	2982	3111#	3113	3244#	3246	3375#	3377	3506#	3509	
	3637#	3639	3769#	3771	3874#	3876	3959#	3961	4092#	4094	4225#	4227	4318#	4320	4455#	

.\$APTB	1#	1304#	
.\$APTH	1#	879#	1215
.\$APTY	1#	879#	7037
.\$ASTA	1#		
.\$CATC	1#	879#	1187
.\$CMTA	1#	879#	1237
.\$DB2D	1#		
.\$DB2O	1#		
.\$DIV	1#		
.\$EOP	1#	879#	5284
.\$ERRO	1#	879#	7094
.\$ERRT	1#	879#	
.\$MULT	1#		
.\$POWE	1#	879#	
.\$RAND	1#		
.\$RDDE	1#		
.\$RDOC	1#	879#	7745
.\$READ	1#	879#	7484
.\$R2AZ	1#		
.\$SAVE	1#	879#	7798
.\$SB2D	1#		
.\$SB2O	1#		
.\$SCOP	1#	879#	6923
.\$SIZE	1#	879#	6809
.\$SUPR	1#		
.\$TRAP	1#	879#	7876
.\$TYFB	1#		
.\$TYPD	1#	879#	7417
.\$TYPE	1#	879#	7244
.\$TYPO	1#	879#	7340
.\$40CA	1#		
.\$170	1#		

. ABS. 051134 000

ERRORS DETECTED: 0

CZR6EC,CZR6EC.LST/SOL/CRF/NL:TOC=SYSMAC.SML,CZR6EC.P11
RUN-TIME: 28 33 2 SECONDS
RUN-TIME RATIO: 124/64=1.9
CORE USED: 40K (80 PAGES)