

RX11, RX01
RX02

RX11 SYS RELIAB
CZRXAFO

AH-9336F-MC
FICHE 1 OF 1

SEP 1982
COPYRIGHT © 75-82
MADE IN USA



A grid of approximately 15 columns and 15 rows of data. Each cell contains a small table or set of data points, likely representing system reliability metrics. The data is too faint to transcribe accurately but appears to be organized in a structured format.



.REM 8

IDENTIFICATION

PRODUCT CODE: AC-9334F-MC
PRODUCT NAME: CZRXAFO SYS RELIAB
DATE: 29-MAR-82
MAINTAINER: S.S.S.T.A.
AUTHOR: DAVID L. ADAMS

COPYRIGHT (C) 1975,1982
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY
ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS
SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT
FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE
LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL
AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DEC.

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

TABLE OF CONTENTS

57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98

1.0	GENERAL PROGRAM INFORMATION
1.1	ABSTRACT
1.2	SYSTEM REQUIREMENTS
1.2.1	HARDWARE
1.2.2	SOFTWARE
2.0	OPERATING INSTRUCTIONS
2.0.1	OUTLINE OF OPERATING PROCEDURE
2.1	LOADING PROCEDURE
2.2	STARTING ADDRESSES
2.3	OPERATOR ACTION BEFORE STARTING PROGRAM
2.3.1	DEVICE ADDRESS SELECTION
2.3.2	NON-STANDARD DISKETTE ADDRESS SELECTION
2.3.3	SOFTWARE SWITCH REGISTER (LOC. 176)
2.3.4	TEST PARAMETER SELECTION ('DTESTP' LOC. 1212)
2.4	OPERATOR ACTION TO RUN THE PROGRAM
2.4.1	STARTING THE PROGRAM
2.4.2	OPERATING CONDITIONS
2.4.3	ACT11 AND XXDP HOOKS
2.5	PROGRAM OPTIONS
2.5.1	PSEUDO SCOPE LOOP
2.5.2	DISKETTE COMPATABILITY
2.5.3	RX11 TO RX8 COMPATABILITY
2.6	RUN TIME
3.0	ERROR DETECTION
3.1	ERROR DEFINITIONS
3.2	DEFINITIVE ERROR CODES
3.3	UNEXPECTED OR MISSING ERROR CONDITIONS
3.4	POWER FAILURE
3.5	PROGRAM HUNG
4.0	ERROR REPORTING
5.0	HALTS

100
102
103
104
105
106
107
108
109
110
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
138
139
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157

1.0 GENERAL PROGRAM INFORMATION
1.1 ABSTRACT

THE RX11 SYSTEM RELIABILITY PROGRAM CONSISTS OF SELECTABLE TESTS THAT CHECK THE OPERATION OF THE RX11 SYSTEM, BY WRITING, READING AND VERIFYING VARIOUS DATA PATTERNS, UNDER VARIOUS (SELECTABLE) HEAD MOVEMENTS. IT CAN TRANSFER DATA AND CHECK FOR ERRORS OVER THE ENTIRE DISKETTE, ALL TRACKS AND SECTORS, OR BETWEEN SEPARATELY SELECTABLE TRACK AND SECTOR ADDRESS LIMITS. AS WELL AS TRANSFERRING DATA THE PROGRAM ACCUMULATES STATISTICAL DATA ON THE FOLLOWING:

- A. COMPLETED PASSES OF THE PROGRAM
- B. NUMBER OF RESTARTS
- C. NUMBER OF SECTORS WRITTEN/READ
- D. RECOVERABLE AND UNRECOVERABLE FAULTS AS FOLLOWS:
 - 1. PARITY ERRORS
 - 2. ERROR FLAG ERRORS
 - 3. INTERRUPT ERRORS
 - 4. SEEK ERRORS
 - 5. DATA CRC ERRORS
 - 6. CRC NO DATA ERRORS
 - 7. DATA NO CRC ERRORS
 - 8. DELETED DATA MARK ERRORS
 - 9. WRITE ERRORS
 - 10. READ ERRORS
- E. TOTAL OF EACH ERROR CODE DETECTED
- F. TOTAL OF TIMES EACH TRACK WAS ACCESSED
- G. TOTAL OF TIMES HEAD MOVED TO A TRACK.
- H. TOTAL OF ERRORS DETECTED PER TRACK PER DRIVE.

THE ABOVE IS REPORTED BY THE ERROR DUMP PROGRAM.

1.2 SYSTEM REQUIREMENTS
1.2.1 HARDWARE REQUIREMENTS

THE FOLLOWING EQUIPMENT IS REQUIRED.

- A. PDP-11 SERIES OF COMPUTER WITH MIN 8K MEMORY
- B. RX11 FLOPPY DISK SYSTEM INCLUDING DUAL OR SINGLE DRIVE RX01 AND PDP-11 INTERFACE. (SEE SECTION 2.3 FOR SELECTION OF REGISTER ADDRESSES AND VECTOR ADDRESS)
NOTE: A DISKETTE MUST BE INCLUDED WITH EACH DRIVE TESTED.
- C. CONSOLE TELEPRINTER

1.2.1 SOFTWARE REQUIREMENTS

THIS PROGRAM ASSUMES THAT THE RX11 INTERFACE DIAGNOSTIC (MAINDEC - 11 - DZRXB-*) HAS BEEN SUCCESSFULLY RUN ON THIS SYSTEM.

159
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215

2.0 OPERATING INSTRUCTIONS
2.0.1 OUTLINE OF OPERATING PROCEDURE

THE STANDARD OPERATING PROCEDURE FOR THE RX11 RELIABILITY TEST (TO RUN ALL TESTS ON BOTH DRIVES WITH NO OPERATOR INTERVENTION VIA THE SWITCH REGISTER) IS AS FOLLOWS:

- A. LOAD THE PROGRAM INTO MEMORY
 - 1. IF IT'S BEING LOADED FROM A DISKETTE, REPLACE THE 'LIBRARY DISKETTE' WITH A 'SCRATCH DISKETTE'
- B. START THE PROGRAM RUNNING AT LOCATION 200
- C. THE PROGRAM WILL TYPE OUT THE FOLLOWING:
 - 1. PROGRAM NAME AND REVISION
 - 2. RX11 REGISTER AND VECTOR ADDRESSES
 - 3. UNITS BEING TESTED
 - 4. 'P'ATTERN, 'T'EST, AND 'S'EQUENCEIT THEN STARTS RUNNING UNDER THOSE CONDITIONS
- D. IF THERE ARE NO ERRORS, AT THE END OF THE COMPLETED PASS A 'D' AND 'BELL' IS TYPED, AND THE PROGRAM WILL CONTINUE ON FOR ANOTHER PASS.
- E. HALT THE PROGRAM AS FOLLOWS:
 - 1. IF THERE IS A HARDWARE SWITCH REGISTER, PUT SW 14 UP TO HALT AT THE END OF PASS.
 - 2. IF THERE IS NO SWITCH REGISTER, OR TO HALT AT ANY TIME, HALT THE PROCESSOR.

2.1 LOADING PROCEDURE

LOAD THE PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR BINARY PAPER TAPES.

MAKE SURE THAT THE TOTAL SYSTEM IS READY FOR OPERATION, DISKETTE INSERTED CORRECTLY, DOORS CLOSED ON DRIVES TO BE TESTED, ETC.

2.2 STARTING ADDRESSES

THE PROGRAM HAS THREE (3) STARTING LOCATIONS.

2.2.1 INITIAL START (LOC. 200)

THIS STARTING LOCATION INITIALIZES THE PROGRAM AS FOLLOWS:

- A. CLEARS ALL ERROR LOGS
- B. RESETS COUNTERS AND CONSTANTS
- C. TYPES OUT RX11 REGISTER AND VECTOR ADDRESSES BEING USED.
- D. TYPES THE DRIVE AND TEST SELECTION IN 'DTESTP'
- E. INITIATES THE COLLECTION OF STATISTICAL DATA PERTAINING TO THE OPERATION OF THE RX11 SYSTEM.

217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260

2.2.2 RESTART (LOC. 202)

THIS STARTING LOCATION DIRECTS THE PROGRAM TO CONTINUE RUNNING USING DRIVE AND TEST SELECTIONS SPECIFIED IN THE PREVIOUS INITIAL START. IT ALSO CONTINUES TO ACCUMULATE STATISTICAL INFORMATION, AMENDING IT TO THE DATA ALREADY COLLECTED PRIOR TO THIS RESTART.

2.2.3 ERROR DUMP (LOC. 204)

THIS STARTING ADDRESS DIRECTS THE PROGRAM TO PRINT OUT ON THE CONSOLE TELEPRINTER ALL THE DATA ACCUMULATED FROM THE LAST "INITIAL START" TO THE TIME OF THE PRINTOUT. TYPING OUT THIS INFORMATION DOES NOT DESTROY IT, SO A RESTART CAN BE INITIATED AND INFORMATION WILL CONTINUE TO BE COLLECTED.

2.3 OPERATOR ACTION BEFORE STARTING PROGRAM

2.3.1 DEVICE ADDRESS SELECTION

LIKE MOST OPTIONS ON THE PDP11 THE RX11 INTERFACE CARD HAS JUMPERABLE REGISTER AND VECTOR ADDRESSES. THIS ALLOWS FOR DEVICES WITH THE SAME STANDARD ADDRESSES TO BE JUMPED TO AN OTHER ADDRESS SO THEY WILL RUN WITHOUT CONFLICT.

THE PROGRAM MUST KNOW WHAT ADDRESSES ARE BEING USED, AS IT IS THROUGH THESE REGISTERS AND VECTOR THAT ALL COMMUNICATION BETWEEN THE PDP11 AND RX11 IS HANDLED.

IF THE RX11 SYSTEM UNDER TEST IS JUMPED FOR REGISTER ADDRESS OTHER THAN STANDARD, WHICH IS RXCS=177170 AND RXDB=177172. PLACE IN THE MEMORY LOCATION CALLED "RXCS" (LOC. 1206) ITS NEW ADDRESS, AND IN LOCATION "RXDB" (LOC. 1210) ITS NEW ADDRESS. IF THERE AS A NONSTANDARD INTERRUPT VECTOR ADDRESS (STANDARD IS LOC 264) THEN PLACE IN MEMORY LOCATION CALLED "INTVEC" (LOC. 1204) ITS NEW ADDRESS.

IF THESE THREE MEMORY LOCATIONS DO NOT CONTAIN THE ADDRESSES THAT THE INTERFACE BOARD IS WIRED TO THE PROGRAM WILL REPORT "TEST HUNG" AND HALT, OR HALT AT THE VECTOR ADDRESS THAT IS JUMPED IN. (NOTE: THE VECTOR ADDRESSES CAN NOT BE JUMPED FOR LOCATIONS 200 THROUGH 220 AS THESE ARE USED BY THE PROGRAM)

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

2.3.2 NON-STANDARD DISKETTE ADDRESS SELECTION

IF IT IS DESIREABLE TO TEST THE DISKETTES BETWEEN TRACK AND SECTOR ADDRESS LIMITS OTHER THAN BETWEEN THE NORMAL OUTER DIAMETER (OD) AND INNER DIAMETER (ID) TRACK ADDRESSES, AND/OR MINIMUM (FIRST) AND MAXIMUM (LAST) SECTOR ADDRESS. THIS IS DONE BY THE OPERATOR MAKING CHANGES TO TWO (2) MEMORY LOCATIONS BEFORE THE PROGRAM IS STARTED. ONE LOCATION IS CALLED OD WHICH CONTAINS TWO BYTES ONE FOR OD AND THE OTHER ID TRACK ADDRESSES. THE OTHER LOCATION IS CALLED FIRST AND IT TOO CONTAINS TWO BYTES ONE FOR FIRST THE OTHER FOR LAST SECTOR ADDRESS. (IF THESE TWO LOCATIONS ARE LEFT CLEARED THE MAX AND MIN VALUES FOR THE ADDRESSES ARE ASSUMED.)

A. DEFINITIONS:

OD = ADDRESS OF TRACK AT OUTER DIAMETER (MIN VALUE=0)
ID = ADDRESS OF TRACK AT INNER DIAMETER (MAX VALUE=114 [OCTAL])
FIRST = ADDRESS OF FIRST SECTOR OF TRACK (MIN VALUE=1)
LAST = ADDRESS OF LAST SECTOR OF TRACK (MAX VALUE=32 [OCTAL])

B. LOCATIONS:

TRACKS LOC. 1200 BITS	14-----8	6-----0
	ID	OD
SECTORS LOC. 1202 BITS	12-----8	4-----0
	LAST	FIRST

C. RESTRICTIONS:

THE CONTENTS OF 'OD' MUST BE LESS THAN OR EQUAL TO THE CONTENTS OF 'ID'
THE CONTENTS OF FIRST MUST BE LESS THAN OR EQUAL TO THE CONTENTS OF 'LAST'

IF THESE LOCATIONS ARE CHANGED TO NEW LIMITS, THEN THE PROGRAM WILL ACCESS ONLY THOSE ADDRESSES INCLUSIVE OF AND BETWEEN THESE LIMITS.

300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339

2.3.3 SOFTWARE SWITCH REGISTER (LOC. 176)

FOR THE PDP 11 PROCESSORS THAT DO NOT HAVE A HARDWARE SWITCH REGISTER, OR IF THE OPERATOR WISHES TO SELECT THE SOFTWARE SWITCH REGISTER, BY PUTTING ALL THE SWITCHES UP TO A '1'. (THIS MUST BE DONE EACH TIME THE PROGRAM IS STARTED AT LOCATION 200, OTHERWISE THE PROGRAM WILL USE THE HARDWARE SWR.) LOCATION 176 IS ASSIGNED AS THE SWITCH REGISTER. BITS SET TO A '1' IN THIS LOCATION HAVE THE SAME FUNCTION AS THE CORRESPONDING SWITCH IN THE HARDWARE SWITCH REGISTER. ALL REFERENCES TO THE SWR ARE INDIRECT AND THE PROGRAM ASSIGNS THE CORRECT ADDRESS OF THE SWR AT 'INITIAL START'. SEE SECTION 2.4.2 FOR THE SELECTION OF OPERATING CONDITIONS.

TO CHANGE THE SOFTWARE SWR. WHILE THE PROGRAM IS RUNNING, TYPE 'CONTROL G'. EACH TIME THE SWR. IS TO BE TESTED THE PROGRAM WILL CHECK TO SEE IF THE SOFTWARE SWR IS SELECTED, AND THE PROGRAM IS NOT RUNNING IN AUTO MODE OF RXDP/ACT11. IF BOTH CONDITIONS EXIST THEN THE PROGRAM CHECKS FOR THE CTRL G IN THE KEYBOARD BUFFER. IF THE CTRL G IS THERE THE CONTENTS OF THE SOFTWARE SWR. ARE PRINTED AND A 'NEW =' IS ASKED FOR. THE OPERATOR MAY NOW TYPE IN THE NEW SWITCH REGISTER CONTENTS, TERMINATED BY A CARRIAGE RETURN (CR), OR IF HE DOESN'T WANT TO CHANGE THE SWR. JUST TERMINATE WITH THE (CR). NOTE SEE THE CHARACTER RESTRICTIONS BELOW.

WHEN THE PROGRAM DETECTS THE (CR) IT WILL REPLACE THE CONTENTS OF THE SOFTWARE SWR., IF A NEW ONE HAS BEEN TYPED IN, AND RETURN TO THE FLOW OF THE PROGRAM.

NOTE: CHARACTER RESTRICTIONS FOR CHANGING THE SOFTWARE SWR.

1. ONLY OCTAL NUMBERS 0 - 7 ARE ACCEPTED. ANY OTHER CHARACTER TYPED WILL BE PRINTED AS A ? AND THE WHOLE SWR MUST BE RETYPED.
2. TO WIPE OUT A 'NEW' CONTENTS JUST TYPED IN, TYPE CTRL U. NOW A NEW CONTENTS CAN BE RETYPED.
3. ONLY 6 OCTAL CHARACTERS WILL BE PUT INTO THE SWR. IF MORE THAN 6 CHARACTERS ARE TYPED IN ONLY THE LAST 6 WILL BE PUT INTO THE SWR.

340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392

2.3.4 TEST PARAMETER SELECTION ('DTESTP' LOC. 1212)

THE DRIVE AND TEST SELECTION MUST BE DONE BEFORE THE PROGRAM STARTS. 'DTESTP' (LOCATION 1212) IS WHERE THE BITS ARE SET TO TELL THE PROGRAM WHAT DRIVES ARE WANTED AND WHAT TEST TO RUN, AS INDICATED BELOW. WHEN THE PROGRAM STARTS IT WILL TYPE OUT THE TEST CONDITIONS IT IS RUNNING UNDER.

BIT 15 (1) SELECT DRIVE UNIT 1
BIT 14 (1) SELECT DRIVE UNIT 0

THEN SET THE TEST CONDITIONS IN BITS 8 THROUGH 0 AS SHOWN BELOW.

'DTESTP' BITS 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
 U1 U0 NOT USED D P P P T T T S S S

BIT 9 IF BIT 9 IS ON, ALL WRITE/READ FUNCTIONS WILL BE IN THE DELETED DATA MODE.
BIT 8,7,6 SELECTS A DATA PATTERN TO BE USED.
BIT 5,4,3 SELECTS A TEST TO BE PERFORMED.
BIT 2,1,0 SELECTS A HEAD MOVEMENT SEQUENCE.

THE SELECTIONS ARE DEFINED AS FOLLOWS:

P = DATA PATTERN SELECTION

- 0 DEFAULT TO 7
- 1 ZEROS
- 2 ONES
- 3 FLOATING ZERO
- 4 FLOATING ONE
- 5 125
- 6 314
- 7 RANDOM

T = FUNCTIONAL TESTS

- 0 DEFAULT TO 7
- 1 WRITE ONLY
- 2 WRITE/READ
- 3 WRITE/READ CHECK
- 4 READ CHECK ONLY
- 5 READ ONLY (CRC CHECK)
- 6 WRITE/READ CHECK ON ALTERNATING DRIVES *
- 7 WRITE/READ/READ CHECK **

* NOTE: TEST 6 WRITES THEN READ CHECKS ANY SELECTED DATA PATTERN USING ANY TRACK SEQUENCE, BUT ONE TRACK AT A TIME. FIRST ON UNIT 0 THEN UNIT 1. WHEN BOTH UNITS HAVE ACCESSED THAT TRACK, IT GOES BACK TO UNIT 0 FOR THE NEXT TRACK, ETC.

393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448

** NOTE: THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE TO INCREMENT UP THROUGH ALL TRACKS DOING WRITE/READ CHECK FUNCTIONS. THIS VERIFIES THAT ALL TRACKS ARE ACCESSABLE. THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE OPERATOR AS INDICATED BELOW, AND ONLY READ CHECK THE DATA JUST WRITEN. THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK AFTER THE HEAD HAS BEEN MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT PASS WILL BE RUN UNDER THIS NEW CONDITION.

S = TRACK SEQUENCING

- 0 DEFAULT TO 7
- 1 INCREMENT
- 2 DECREMENT
- 3 INCREMENT/DECREMENT
- 4 BOUNCE
- 5 DECREASING BOUNCE
- 6 STROBE
- 7 RANDOM

IF NO BITS ARE SET (ZEROED 'DTESTP:') THEN THE PROGRAM WILL SELECT ALL DRIVE UNITS THAT ARE READY AND DEFAULT TO TEST CONDITIONS AS INDICATED.

THE PROGRAM NEXT PRINTS THE REGISTER AND VECTOR ADDRESSES IT WILL USE IN COMMUNICATING WITH THE RX11 SYSTEM. AS EXPLAINED IN SECTION 2.3.1 THE OPERATOR MUST VERIFY THAT THESE ADDRESSES ARE THE ONES JUMPERED ON THE RX11 INTERFACE BOARD.

2.4 OPERATOR ACTION TO RUN THE PROGRAM

2.4.1 STARTING THE PROGRAM

SET THE DESIRED STARTING ADDRESS INTO THE SWITCH REGISTER, DEPENDING UPON THE TYPE OF CONSOLE AVAILABLE, LOAD ADDRESS, AND PRESS START.

THE PROGRAM WILL TYPE ITS 'MAINDEC' NUMBER AND REVISION, AND DEPENDING UPON THE STARTING ADDRESS DO THE FOLLOWING:

SA200 - THE PROGRAM WILL TYPE DRIVE AND TEST PARAMETERS, THE TWO REGISTER ADDRESSES, AND VECTOR ADDRESS. IT WILL THEN BEGIN FUNCTIONAL TESTING, AND INFORMATION COLLECTION.

SA202 - THE PROGRAM WILL CONFIGURE TO CONDITIONS SET IN PREVIOUS "INITIAL START", PRINT OUT THESE CONDITIONS AND CONTINUE FUNCTIONAL TESTING AND DATA COLLECTING. THE ONLY OPERATOR ACTION REQUIRED IS THE DYNAMIC SELECTION OF OPERATING CONDITIONS IN THE SWITCH REGISTER AS REQUIRED.

SA204 - THIS PROGRAM WILL REPORT VIA THE TELEPRINTER ALL DATA COLLECTED. NO OTHER OPERATOR ACTION IS REQUIRED.

450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506

2.4.2 OPERATING CONDITIONS

THE PROGRAM CHECKS FOR OPERATING CONDITIONS AT VARIOUS POINTS WHILE RUNNING. IF THERE IS A HARDWARE SWR THESE CONDITIONS CAN BE CHANGED AND SET WHILE THE PROGRAM IS RUNNING. IF THE SOFTWARE SWR IS IN USE, THEN THESE CONDITIONS MUST BE SET IN LOCATION 176 BEFORE THE PORGRAM STARTS.

SW15 = HALT ON ERROR
SW14 = HALT AT END OF PASS
SW13 = DON'T PRINT ERROR MESSAGE
SW12 = TYPE ONLY 10 DATA ERRORS
SW11 = NO RETRY ON ERROR. LOG HARD ERROR
SW08 = NO RECALIBRATION ON SEEK ERRORS
SW15-SW0 (1) = SELECT SOFTWARE SWITCH REGISTER

NOTE: IF THERE IS A HARDWARE SWITCH REGISTER, AND THE OPERATOR WANTS THE SOFTWARE SWITCH REGISTER. PUT ALL SWITCHES UP (1) BEFORE STARTING THE PROGRAM AT THE INITIAL START ADDRESS.

THE PROGRAM WILL PRINT A 'DRIVE(S)' SELECTED CONFIRMATION MESSAGE THAT IT WAS SUCCESSFUL IN FINDING AT LEAST ONE DRIVE READY (DRY) CONDITION ON AN OPERATOR SELECTED DRIVE OR EITHER OR BOTH IF NO SELECTION WAS MADE. THE DRIVES CONFIRMED AS BEING SELECTED BY THE PROGRAM MAY DIFFER FROM THAT SELECTED BY THE OPERATOR IF THE PROGRAM DETECTED A DRIVE TO BE NOT READY.

IF HOWEVER THERE ARE NO DRIVES IN THE DRIVE READY CONDITION THE PROGRAM WILL TYPE 'NO DRIVES READY' AND HALT, AS IT CAN'T FUNCTION WITH NO DRIVES READY. WHEN THE REASON FOR ALL DRIVES TO BE NOT READY IS FOUND AND CORRECTED, THE OPERATOR MAY PRESS CONTINUE AND THE PROGRAM WILL GO BACK TO 'INITIAL START' OR HE MAY RELOAD THE STARTING ADDRESS HIMSELF.

AS WELL AS 'DRIVE(S)' SELECTED THE PROGRAM WILL TYPE OUT THE TEST PARAMETERS SELECTED OR DEFAULTED TO. IF NON-STANDARD TRACK AND/OR SECTOR ADDRESS LIMITS WERE SELECTED (SEE SECTION 2.3.2 OF THIS DOCUMENT) THESE LIMITS WILL ALSO BE PRINTED OUT FOR VARIFICATION BY THE OPERATOR.

2.4.3 ACT11 AND XXDP HOOKS

THE PROGRAM HAS THE NECESSARY LOCATIONS SET UP FOR OPERATION UNDER ACT11 AND XXDP OPERATION. THE PROGRAM LOOKS AT THE LOADING MEDIA LOCATION AND IF IT CONTAINS THE NUMBER 10, INDICATING THE FLOPPY DISK LOADED THE PROGRAM, WILL TYPE THE FOLLOWING PROMPT MESSAGE AND WAIT FOR A USER RESONSE:

"CAUTION - IF YOU DESIRE TO TEST UNIT 0
REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE
THEN PRESS CONTINUE"

THUS, IF THE OPERATOR WISHES TO TEST DRIVE UNIT 0, AFTER LOADING THE PROGRAM, HE REMOVES THE LIBRARY DISKETTE FROM DRIVE 0, INSERTS A SCRATCH DISKETTE INTO DRIVE 0, AND PRESSES THE 'CONTINUE' SWITCH

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

2.5 PROGRAM OPTIONS

THERE ARE A COUPLE OF WAYS THE PROGRAM CAN BE SET UP TO RUN FOR OPTIONAL TESTING.

2.5.1 PSEUDO - SCOPE LOOP

BY SETTING THE ADDRESS LIMITS IN OD AND OR FIRST TO ONE OF TWO TRACKS AND OR SECTORS IT IS POSSIBLE TO PRODUCE A FAIRLY TIGHT PSEUDO - SCOPE LOOP. BY RUNNING ONLY A TEST THAT DOES WRITE ONLY OR READ ONLY ITS POSSIBLE TO SCOPE SPECIFIC FUNCTIONS, I.E. FILL BUFFER, WRITE, READ, EMPTY BUFFER.

2.5.2 DISKETTE COMPATABILITY

TO CHECK FOR DATA TRANSFER COMPATABILITY BETWEEN DRIVE UNITS USE THE FOLLOWING PROCEDURE. (IT IS ASSUMED THE SYSTEM HAS DUAL DRIVE RX01, IF NOT YOU MAY TEST COMPATABILITY BETWEEN DIFFERENT RX11 SYSTEMS BY RUNNING THE SAME TESTS ON BOTH.) THIS TEST INSURES PROPER HEAD ALIGNMENT.

- A. HAVE DISKETTES IN BOTH DRIVES AND DRIVES READY.
- B. CLEAR THE OD/ID AND FIRST/LAST MEMORY LOCATION, FOR TOTAL DISKETTE TRANSFERS.
- C. LOAD THE INITIAL START ADDRESS (LOC 200) AND START THE PROGRAM RUNNING.
- D. SET LOCATION 'DTESTP' FOR 'DRIVE AND TEST CONDITIONS' ON BOTH DRIVES, ANY 'P'ATTERN OF DATA, 'S'SEQUENCE #1 (INCREMENT TRACKS, TO INSURE ALL HEAD POSITIONS ARE ALIGNED) AND 'T'EST 3 (WRITE/READ CHECK). THIS WILL WRITE THEN READ AND VERIFY THE DATA ON BOTH DISKETTES.
- E. ALLOW THE PROGRAM TO RUN FOR AT LEAST 1 COMPLETE PASS.

NOTE: IF RANDOM PATTERN (0 OR 7) IS SELECTED YOU MUST HALT AT THE END OF THE FIRST PASS, AS ADDITIONAL PASSES CHANGES THE DATA AND YOU WILL GET DATA ERRORS WHEN YOU TRY TO REREAD.

TO HALT AT THE END OF PASS PUT SW14 UP (1) WHEN 'OPERATING CONDITIONS' ARE REQUESTED BY THE PROGRAM.

- F. AFTER COMPLETION OF THE PASS, PROGRAM HALTED. SWAP DISKETTES, AND AS YOU ONLY WANT TO READ VERIFY THE DATA, START THE PROGRAM AGAIN AT LOC 200 (INITIAL START).
- G. WHEN REQUESTED SELECT THE SAME DRIVES, PATTERN, AND SEQUENCE, BUT SELECT TEST 4 (READ CHECK ONLY).
- H. ALLOW THE PROGRAM TO RUN AS LONG AS YOU WISH TO VERIFY

564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607

THAT DATA WRITTEN AND CHECKED ON ONE DRIVE CAN BE
READ AND VERIFIED ON THE OTHER DRIVE.

2.5.3 RX11 TO RX8 COMPATABILITY

TO WRITE A DISKETTE ON THE RX11 AND READ/VERIFY THE SAME DATA ON A
RX8 REQUIRES THE FOLLOWING:

A. SET UP LOCATION 'DTESTP' (1212) WITH ONE OF THE
FOLLOWING DATA PATTERNS

P=1 (0'S)
P=2 (1'S)
P=5 (125) *
P=6 (314) *

* NOTE: IF ONE OF THESE PATTERNS IS SELECTED A MODIFICATION
TO THE PROGRAM MUST BE MADE TO ALLOW THIS DATA TO BE READ ON
A RX8 SYSTEM.
THE MODIFICATION IS IN THE 'PAT125' ROUTINE, THE TWO (2) LOCATIONS
CONTAINING THE 'COMB DATABYTE' INSTRUCTION MUST BE CHANGED
TO TWO (2) NOP'S (OCTAL 000240)

B. SET IN 'DTESTP' TEST 3 WHICH WILL WRITE AND READ CHECK
THE DATA THEREBY VERIFYING THE QUALITY OF THE DATA PRIOR
TO THE DISKETTE SWAP.

SET THE SEQUENCE TO 1 TO INSURE THAT ALL TRACKS HAVE BEEN WRITTEN ON.

C. HALT THE PROGRAM AT THE COMPLETION OF ONE PASS.

D. THE DISKETTE IS NOW READY TO BE READ ON THE RX8
SYSTEM. SEE MD-8-DIRXB-* FOR THE PROCEDURE TO READ THIS DISKETTE.

TO READ A DISKETTE THAT WAS WRITTEN ON A RX8 SYSTEM SET UP AS FOLLOWS:

E. SET IN 'DTESTP' THE DATA PATTERN THAT WAS USED BY THE
RX8 SYSTEM. (SEE NOTE UNDER SECTION A.)

F. SET 'DTESTP' FOR TEST 4 (READ CHECK ONLY) AND SEQUENCE 1
TO CHECK ALL TRACKS. START THE PROGRAM, IT SHOULD RUN WITHOUT
DATA ERRORS.

608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659

2.6 RUN TIME

RUN TIME PER PASS DEPENDS UPON NUMBER OF A FUNCTIONS IN THE TEST SELECTED AND THE TRACK SEQUENCE. EXAMPLES OF RUN TIMES FOLLOW. (THESE TIMES ARE FOR 1 DRIVE AND COMPLETE DISKETTE, MAXIMUM ID/OD, FIRST/LAST LIMITS. IF OPERATING ON 2 DRIVES DOUBLE THE TIME.) THESE TIMES ARE FOR A PDP 11/05 PROCESSOR AND MAY CHANGE SLIGHTLY FOR A FASTER PROCESSOR.

TEST SELECTION	P	T	S	TIME/PASS
	7	6	5	3 MIN.30 SEC.
	7	7	4	2 MIN.47 SEC.
	7	7	7	2 MIN.23 SEC.
	6	3	4	1 MIN.46 SEC.
	7	3	1	1 MIN.30 SEC.
	7	1	1	51 SEC.

NOTE: DUE TO THE SLOW SPEED OF THE LSI 11 PROCESSOR, THE RUN TIME IS ABOUT DOUBLE THAT LISTED ABOVE. TO SPEED UP THE RUNNING OF THIS PROGRAM IN THE LSI 11 YOU CAN CHANGE THE INTERLEAVE FACTOR, OF THE SECTORS, USED IN THE PROGRAM. TO DO THIS CHANGE THE CONTENTS OF LOCATION "THREE" FROM OCTAL 3 TO OCTAL 5.

3.0 ERROR DETECTION

3.1 PROGRAM DEFINITIONS

ON MOST ERRORS THE PROGRAM WILL TYPE OUT THE CONTENTS OF "STATUS A" AND "STATUS B".

STATUS A IS THE CONTENTS OF THE RXES (ERROR AND STATUS REGISTER) AT THE TIME THE ERROR IS DETECTED. IT SHOWS THE CRC, PAR, ETC. ERRORS.

STATUS B IS THE "DEFINITIVE ERROR CODES" THAT THE RX01 DETECTED, THAT MAY HAVE CAUSED THE ERROR CONDITION. THESE ERROR CODES ARE DEFINED IN SECTION 3.2.

THE PROGRAM HAS DEFINED THE FOLLOWING AS ERRORS.

3.1.1 WRITE ERROR

A WRITE ERROR IS A RETRIED READ ERROR IF THE DATA BEING READ IS OF UNKNOWN QUALITY (THE DATA READ IS BEING READ FOR THE FIRST TIME AFTER A WRITE FUNCTION)

3.1.2 READ (CRC) ERROR

A READ ERROR IS A RETRIED READ ERROR WHERE THE QUALITY OF THE DATA BEING READ IS KNOWN GOOD. (THE DATA HAS BEEN READ CORRECTLY SOME TIME PREVIOUSLY.)

660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705

- 3.1.3 CRC AND DATA ERROR
- 3.1.4 NO CRC ERROR BUT DATA ERROR
- 3.1.5 CRC ERROR BUT NO DATA ERROR

THE ABOVE THREE ERRORS ARE DETECTED WHEN THE PROGRAM IS VERIFYING THE DATA READ OFF THE DISKETTE AGAINST THE DATA THAT SHOULD HAVE BEEN READ.

UPON A NON-COMPARISON THE PROGRAM TYPES OUT THE 'BYTE' NUMBER IN THE SECTOR, THE DATA READ FROM THE DISKETTE 'BAD' AND THE EXPECTED DATA 'GOOD'.

BYTENS BAD GOOD
(THE DATA PATTERNS ARE FORMATTED AS SHOWN)

0 (TRACK ADDRESS; BITS 6 - 0)
1 (SECTOR ADDRESS; BITS 4 - 0)

BYTES 2 THROUGH 125 CONTAIN THE SELECTED 'P'ATTERN.

126 (THE SUM OF ALL BYTES 0 - 125)
127 (THE NEGATIVE OF 2 TIMES BYTE 126)

THE PROGRAM DETECTS A CHECKSUM ERROR BY SUMMING ALL THE DATA READ FROM THE DISKETTE AND COMPARING THAT SUM TO 0.

AT THE END OF THE DATA ERROR TYPEOUT THE PROGRAM TYPES OUT IF THE CHECK SUM ACCUMULATED IS 'GOOD' OR HAD 'ERRORS'. IF BYTES 0 OR 1 HAVE DATA ERRORS THE OPERATOR MUST CHECK THE RESULTS OF THE CHECK SUM. IF IT IS ALSO BAD, THEN THERE WAS A TRUE DATA ERROR. IF THE CHECK SUM IS GOOD, THEN IT MIGHT BE THAT THE HEAD IS NOT OVER THE TRACK EXPECTED, AND THERE IS A POSITIONING ERROR.

- 3.1.6 SEEK ERROR

A SEEK ERROR IS DEFINED AS NOT A CRC AND NOT A PARITY ERROR. A PROGRAMED RECALIBRATE IS ISSUED TO TRY TO CORRECT EACH SEEK ERROR.

- 3.1.7 PARITY ERROR

A PARITY ERROR RESULTS FROM AN INCORRECT TRANSFER OF A COMMAND WORD FROM THE RX11 INTERFACE TO THE RX01 MICRO-PROCESSOR CONTROLLER.

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

3.2 DEFINITIVE ERROR CODES

THE RX01 MICROCONTROLLER HAS DEFINED THE ERROR CODES AND MEANINGS WHICH ARE AVAILABLE TO THE PROGRAM BY ISSUING COMMAND #7 'READ THE B-CODE STATUS REGISTER'.

A DEFINITIVE ERROR CODE REPRESENTS [WHERE] WITHIN A FUNCTION AN ERROR WAS DETECTED.

THE FOLLOWING ARE THE DEFINITIVE ERROR CODES AND MEANINGS:

- 00 - NO ERROR
- 10 - DRIVE 0 FAILED TO SEE HOME FROM INITIALIZE
- 20 - DRIVE 1 FAILED TO SEE HOME FROM INITIALIZE
- 30 - HOME FOUND WHEN STEPPING OUT 10 TRACKS FROM INIT
- 40 - TRIED TO ACCESS; A TRACK GREATER THAN /7(DECIMAL)
- 50 - HOME WAS FOUND BEFORE DESIRED TRACK
- 60 - SELF DIAGNOSTIC ERROR
- 70 - DESIRED SECTOR NOT FOUND AFTER SAMPLING 52 HEADERS
- 100 - WRITE PROTECT ERROR
- 110 - MORE THAN 40US AND NO SEP CLOCK DETECTED
- 120 - A PREAMBLE COULD NOT BE FOUND
- 130 - PREAMBLE FOUND BUT NO ID MARO FOUND IN TIME
- 140 - CRC ERROR ON SUPPOSIDLY GOOD HEADER
- 150 - GOOD HEADER (NO CRC ERROR) BUT TRACK COMPARE ERROR
- 160 - IDAM NOT FOUND IN TOME
- 170 - DATA AM NOT FOUND IN TIME
- 200 - DATA CRC ERROR
- 210 - ALL PARITY ERRORS

3.3 UNEXPECTED OR MISSING ERROR CONDITIONS

3.3.1 MISSING DD MARK

AN ERROR WHEN THE PROGRAM WROTE DELETED DATA INFORMATION BUT NO DELETED DATA MARK WAS DETECTED WHEN THE DATA WAS READ.

3.3.2 UNEXPECTED DD MARK

AN ERROR WHEN A DELETED DATA MARK IS DETECTED BUT NO DELETED DATA WAS WRITTEN.

749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804

3.3.3 NO INTERRUPT ON DONE

THE INTERRUPT ENABLE BIT WAS SET AND THE DONE FLAG WAS SET BUT NO INTERRUPT OCCURRED.

3.3.4 UNKNOWN INTERRUPT

IF AN INTERRUPT OCCURS FROM ANY OTHER DEVICE WHILE THIS PROGRAM IS RUNNING IT WILL HALT AT THE INTERRUPT VECTOR LOCATION.

IF AN INTERRUPT OCCURS ON INTERRUPT VECTOR LOCATION 264 (RX11), AND THERE IS NO ERROR, DONE, OR ERROR STATUS CONDITION SET, THEN IT WILL BE TAGGED AS AN UNKNOWN INTERRUPT.

3.4 POWER FAILURE

THE PROGRAM TESTS FOR TWO TYPES OF POWER FAILURE, TOTAL SYSTEM POWER LOSS AND RX11 POWER LOSS RESULTING IN A RECALIBRATION OF THE DRIVES.

THE TOTAL SYSTEM POWER FAILURE IS DETECTED BY THE 'SYSMAC' SUBROUTINE .\$POWER. WHEN THE POWER IS DETECTED TO BE GOING DOWN, THE REGISTERS ARE SANED. WHEN THE POWER COMES BACK UP THE REGISTERS ARE RESTORED AND THE MESSAGE 'POWER' IS PRINTED. THE PROGRAM THEN AUTOMATICALLY DOES A RESTART.

LOSS OF POWER IN THE RX11 CAUSES A RECALIBRATION OF ALL DRIVES. WHEN THIS HAPPENS, THE 'INIT DONE' BIT IS SET IN THE THE RXES REGISTER ALONG WITH THE NORMAL 'DONE' FLAG. AT EACH INTERRUPT THE PROGRAM TESTS FOR INIT DONE. IF IT IS FOUND TRUE, THE FUNCTION WAS NOT COMPLETED AND A POWER LOSS MUST HAVE BEEN DETECTED. WHEN THIS HAPPENS THE PROGRAM TYPES OUT 'RX11 POWER' AND DOES AN AUTOMATIC RESTART.

IF THERE ARE REPEATED, NOTHING ELSE HAPPENS BUT, RX11 POWER MESSAGES THE INIT DONE FLAG MIGHT BE STUCK ON.

3.5 PROGRAM HUNG

THERE ARE MANY PLACES WHERE THE PROGRAM MUST WAIT FOR AN OPERATION TO BE COMPLETED IN THE RX01. THESE ARE WAITING FOR THE 'DONE' FLAG TO INDICATE A FUNCTION IS COMPLETED, OR WAITING FOR A TRANSFER REQUEST 'TR' FLAG TO SEND OR RECEIVE THE NEXT BYTE OF INFORMATION. IF THE RX11 DOES NOT COME BACK WITH EITHER OR BOTH OF THESE FLAGS THE PROGRAM WOULD HANG UP.

TO INHIBIT THIS FROM HAPPENING THERE ARE TWO SUBROUTINES USED TO CHECK FOR THESE TWO FLAGS.

THE 'DONECK' LOOKS FOR THE DONE FLAG AND IF IT IS NOT SEEN WITHIN A SPECIFIC TIME THE 'TEST HUNG' MESSAGE IS PRINTED AND THE PROGRAM HALTS. IN REGISTER 3 (177703) IS THE RETURN ADDRESS OF THE TEST THAT IS WAITING FOR THE DONE FLAG.

805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860

"TRCK" LOOKS FOR THE "TR" FLAG. IF IT IS NOT SEEN WITHIN A SPECIFIC TIME IT ALSO PRINTS THE "TEST HUNG" MESSAGE AND HALTS. IN THE SP (177706) IS THE RETURN ADDRESS OF THE TEST WAITING FOR THE TR FLAG.

THE WAITING TIME FOR THE TWO FLAGS DEPENDS UPON THE HOST PROCESSOR AS INDICATED BELOW:

"DONE" WAIT	11/45 BIPOL 14 SEC. 11/05 CORE 62 SEC.
"TR" WAIT	11/45 BIPOL .44 SEC. 11/05 CORE 2 SEC.

4.0 ERROR REPORTING

ALL ERRORS DETECTED WILL BE REPORTED IF SW13 = 0 FOR OPERATING CONDITIONS. IF SW12 = 1 THEN ONLY 10 DATA ERRORS FOR 1 SECTOR WILL BE REPORTED, AND A TOTAL OF DATA ERRORS FOR THAT SECTOR WILL BE REPORTED AT THE END OF THE SECTOR.

THE END OF PASS INDICATOR TYPED, ALSO INDICATES WEITHER ANY ERRORS OCCURED DURING THAT PASS. IF THERE WERE NO ERRORS THEN A "*" IS PRINTED. IF THERE WERE ERRORS THEN A "-" IS PRINTED. THE TOTAL ACCUMULATED ERRORS AND SYSTEMS OPERATION IS REPORTED BY THE "ERROR DUMP" PROGRAM.

5.0 HALTS

THERE ARE VARIOUS HALT LOCATIONS THROUGHOUT THE PROGRAM. SOME ARE THE RESULTS OF "HARD" ERRORS OTHERS ARE WAITING FOR INTERVENTION. THEY ARE LISTED BELOW:

HALT #	TYPE	DEFINITION
HLT1:	NO ERROR	:CAUTION - LOAD MEDIUM ON UNIT 0
HLT3:	ERROR	:ERRORS FOUND ON RECAL FUNCTIONS
HLT4:	ERROR	:HARD PARITY ERRORS
HLT5:	ERROR	:NO DRIVES READY
HLT6:	ERROR	:SW15 SET ON PARITY ERROR
HLT7:	ERROR	:SW15 SET ON SEEK ERROR
HLT10:	ERROR	:SW15 SET ON CRC GENERATOR ERROR (NO DATES ERROR
HLT11:	ERROR	:HARD CRC GENERATOR ERROR (NO DATA ERRORS)
HLT12:	ERROR	:SW15 SET ON DATA CRC ERROR
HLT13:	ERROR	:SW15 SET ON MISSING DETECTED DATA ERROR
HLT14:	ERROR	:SW15 SET ON DATA NO CRC ERROR
HLT15:	ERROR	:SW15 SET ON MISSING INTERRUPT AT DONE
HLT16:	NO ERROR	:HALT AT END OF PASS
HLT17:	NO ERROR	:HALT AT END OF ERROR DUMP
HLT20:	ERROR	:PROGRAM HUNG WAITING FOR DONE
HLT21:	ERROR	:PROGRAM HUNG WAITING FOR TR FLAG

.NLIST CND,MD,MC
.LIST ME

861
862
863
864
865
866
867
868
869
870
871
872
873
874
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

.ENABL ABS,AMA

```

.TITLE MAINDEC-11-CZRXA-F
.*COPYRIGHT (C) 09-APR-82
.*DIGITAL EQUIPMENT CORP.
.*MAYNARD, MASS. 01754
.*
.*PROGRAM BY DAVID L ADAMS
.*
.*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
.*PACKAGE (MAINDEC-11-DZQAC-C) FEB 1982.
.*
$TN=1
$SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
  
```

000001
160000

```

: COPYRIGHT (C) 1975,1982
: THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY
: ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
: THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS
: SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
: OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON
: EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO
: THESE LICENSE TERMS. TITLE TO OWNERSHIP OF THE
: SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.
:
: THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
: WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT
: BY DIGITAL EQUIPMENT CORPORATION.
:
: DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
: OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
: DEC.
  
```

.SBTTL BASIC DEFINITIONS

```

.*INITIAL ADDRESS OF THE STACK POINTER *** 1200 ***
STACK= 1200
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
  
```

001200

```

.*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
  
```

000011
000012
000015
000200
177776
177774
177772
177570
177570

```

917          ;*GENERAL PURPOSE REGISTER DEFINITIONS
918          000000 R0= %0          ;;GENERAL REGISTER
919          000001 R1= %1          ;;GENERAL REGISTER
920          000002 R2= %2          ;;GENERAL REGISTER
921          000003 R3= %3          ;;GENERAL REGISTER
922          000004 R4= %4          ;;GENERAL REGISTER
923          000005 R5= %5          ;;GENERAL REGISTER
924          000006 R6= %6          ;;GENERAL REGISTER
925          000007 R7= %7          ;;GENERAL REGISTER
926          000006 SP= %6          ;;STACK POINTER
927          000007 PC= %7          ;;PROGRAM COUNTER
928
929          ;*PRIORITY LEVEL DEFINITIONS
930          000000 PR0= 0           ;;PRIORITY LEVEL 0
931          000040 PR1= 40          ;;PRIORITY LEVEL 1
932          000100 PR2= 100         ;;PRIORITY LEVEL 2
933          000140 PR3= 140         ;;PRIORITY LEVEL 3
934          000200 PR4= 200         ;;PRIORITY LEVEL 4
935          000240 PR5= 240         ;;PRIORITY LEVEL 5
936          000300 PR6= 300         ;;PRIORITY LEVEL 6
937          000340 PR7= 340         ;;PRIORITY LEVEL 7
938
939          ;*'SWITCH REGISTER' SWITCH DEFINITIONS
940          100000 SW15= 100000
941          040000 SW14= 40000
942          020000 SW13= 20000
943          010000 SW12= 10000
944          004000 SW11= 4000
945          002000 SW10= 2000
946          001000 SW09= 1000
947          000400 SW08= 400
948          000200 SW07= 200
949          000100 SW06= 100
950          000040 SW05= 40
951          000020 SW04= 20
952          000010 SW03= 10
953          000004 SW02= 4
954          000002 SW01= 2
955          000001 SW00= 1
956          .EQUIV SW09,SW9
957          .EQUIV SW08,SW8
958          .EQUIV SW07,SW7
959          .EQUIV SW06,SW6
960          .EQUIV SW05,SW5
961          .EQUIV SW04,SW4
962          .EQUIV SW03,SW3
963          .EQUIV SW02,SW2
964          .EQUIV SW01,SW1
965          .EQUIV SW00,SW0
966
967          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
968          100000 BIT15= 100000
969          040000 BIT14= 40000
970          020000 BIT13= 20000
971          010000 BIT12= 10000
972          004000 BIT11= 4000
  
```

```

973      002000      BIT10= 2000
974      001000      BIT09= 1000
975      000400      BIT08= 400
976      000200      BIT07= 200
977      000100      BIT06= 100
978      000040      BIT05= 40
979      000020      BIT04= 20
980      000010      BIT03= 10
981      000004      BIT02= 4
982      000002      BIT01= 2
983      000001      BIT00= 1
984      .EQUIV      BIT09,BIT9
985      .EQUIV      BIT08,BIT8
986      .EQUIV      BIT07,BIT7
987      .EQUIV      BIT06,BIT6
988      .EQUIV      BIT05,BIT5
989      .EQUIV      BIT04,BIT4
990      .EQUIV      BIT03,BIT3
991      .EQUIV      BIT02,BIT2
992      .EQUIV      BIT01,BIT1
993      .EQUIV      BIT00,BIT0
  
```

```

995      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
996      000004      ERRVEC= 4          ;; TIME OUT AND OTHER FRRORS
997      000010      RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
998      000014      TBITVEC=14        ;; "T" BIT
999      000014      TRTVEC= 14         ;; TRACE TRAP
1000     000014      BPTVEC= 14         ;; BREAKPOINT TRAP (BPT)
1001     000020      IOTVEC= 20         ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
1002     000024      PWRVEC= 24         ;; POWER FAIL
1003     000030      EMTVEC= 30         ;; EMULATOR TRAP (EMT) **ERROR**
1004     000034      TRAPVEC=34         ;; "TRAP" TRAP
1005     000060      TKVEC= 60          ;; TTY KEYBOARD VECTOR
1006     000064      TPVEC= 64          ;; TTY PRINTER VECTOR
1007     000240      PIRQVEC=240        ;; PROGRAM INTERRUPT REQUEST VECTOR
  
```

```

1008
1009
1010      ;SPECIAL EQUATES
1011
1012
1013     000013      RDOSTAT =13
1014     000033      RD1STAT =33
1015     000017      RDER      =17
1016     000040      DONEBIT  =40
1017     000101      FBIE     =101
1018     000103      EBIE     =103
1019     000105      WRTIE    =105
1020     000107      RDIE     =107
1021     000115      WTDDIE   =115
1022     040001      RECAL    =40001
1023     000200      PR4      =200
1024     000340      PR7      =340
1025     000000      OPEN     =0
1026
1027     000000      .=0
1028     000000      000000      000000      .WORD 0,0
  
```

```

1029
1030          000024          . =24
1031 000024 014062          $PWRDN
1032 000026 000340          340
1033
1034          000034          . =34
1035 000034 014000          $TRAP          ;ADDRESS OF TRAP SERVICE
1036 000034 000340          340
1037
1038          000046          . =44
1039 000046 011130          LOGICAL          ;ACT11 EOP HOOK
1040
1041          000052          . =52
1042 000052 000000          .WORD 0
1043
1044          000174          . =174
1045 000174 000000          DISPREG:          0
1046 000176 000000          SWREG:           0
1047
1048
1049          ;:*****
1050
1051          ;STARTING ADDRESSES
1052
1053          ;INITIAL START =200 /CLEARS ALL ERROR LOGS,RESETS COUNTERS,AND ALLOWS FOR
1054          ; /SELECTION OF DRIVES AND TEST CONDITIONS
1055
1056          ;RESTART =202 /USES PREVIOUS INITIAL START DRIVES AND TEST
1057          ; /SELECTION AND CONTINUES TO ACCUMULATE STATISTICAL DATA
1058
1059          ;ERROR REPORT =204 /PRINTS OUT ALL RUN AND ERROR CONDITIONS
1060          ; /ACUMULATED OVER THE RUN OF THE PROGRAM.
1061
1062
1063
1064          . =200
1065 000200 000402          BR 1$
1066 000202 000403          BR 2$
1067 000204 000404          BR 3$
1068 000206 000137 001220          1$:          JMP SA200          ;OPERATOR SELECTED CONDITIONS
1069 000212 000137 002700          2$:          JMP RESTART          ;RESTART PROGRAM WITH PREVIOUS CONDITIONS
1070 000216 000137 021160          3$:          JMP ERDUMP          ;STATISTICAL ERROR PRINT OUT
1071
1072
1073          ;:*****
1074
1075          ;THE FOLLOWING LOCATIONS 'OD','ID','FIRST',AND 'LAST'
1076          ;MAY BE CHANGED BY THE OPERATOR MANUALLY HOWEVER FOLLOWING THESE RESTRICTIONS.
1077          ;(IF THESE LOCATIONS ARE LEFT CLEARED,MAX AND MIN VALUES ARE ASSUMED)
1078
1079          ;
1080          ; 1. DEFINITIONS:
1081          ; OD=ADDRESS OF TRACK AT OUTER DIAMETER (MIN VALUE=0)
1082          ; ID=ADDRESS OF TRACK AT INNER DIAMETER (MAX VALUE=114 [OCTAL])
1083          ; FIRST=ADDRESS OF FIRST SECTOR OF TRACK (MIN VALUE=1)
1084          ; LAST=ADDRESS OF LAST SECTOR OF TRACK (MAX VALUE=32 [OCTAL])

```

1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140

001200 001200
001200 000000
001202 001201
001202 000000
001202 001203

OD: . =1200 0
 ID=OD+1 0
 FIRST: 0
 LAST=FIRST+1

- 2. LOCATIONS:
 TRACKS LOC. 1200 BITS 14-----8 6-----0
 ID OD
- SECTORS LOC. 1202 BITS 12-----8 4-----0
 LAST FIRST
- 3. RESTRICTIONS:
 THE CONTENTS OF 'OD' MUST BE <= THE CONTENTS OF 'ID'
 THE CONTENTS OF FIRST MUST BE <= THE CONTENTS OF 'LAST'

::*****

: THE NEXT WORD IS THE LOCATION OF THE "INTERRUPT VECTOR" ADDRESS
 : IF THE HARDWARE IS JUMPED FOR OTHER THAN STANDARD (264) VECTOR
 : ADDRESS, THEN LOAD INTO THIS LOCATION THE NEW VECTOR ADDRESS.
 :
 : IF THIS ADDRESS IS INCORRECT (DOES NOT MATCH THE HARDWARE) THE
 : PROGRAM WILL HALT AT THE VECTOR ADDRESS THE HARDWARE IS JUMPED
 : TO, AS ALL OTHER VECTOR ADDRESSES WILL HAVE HALTS IN THEM.

:*****NOTE*****

: THE INTERRUPT VECTOR ADDRESS CAN NOT BE SET FOR ADDRESSES 200
 : THRU 243, AS THESE ADDRESS ARE USED BY THE PROGRAM

:*****

001204 000264

INTVEC: 264

::*****

: THE FOLLOWING TWO WORDS CONTAIN THE DEVICE CODES FOR
 : THE RX11 INTERFACE REGISTERS
 : RXCS = COMMAND STATUS REGISTER
 : RXDB = DATA BUFFER REGISTER (USED AT VARIOUS TIMES FOR:)
 : (RXTA = TRACK ADDRESS)
 : (RXSA = SECTOR ADDRESS)
 : (RXES = ERROR STATUS)
 : IF THE RX11 SYSTEM UNDER TEST IS JUMPED FOR REGISTER
 : ADDRESSES OTHER THAN STANDARD: RXCS = 177170
 : RXDB = 177172
 : PLACE IN THESE TWO WORDS THE CORRESPONDING NEW ADDRESSES FOR
 : THE REGISTERS. THE PROGRAM WILL TYPE OUT THE REGISTER
 : ADDRESSES FOR VERIFICATION BY THE OPERATOR. IF THE ADDRESSES
 : ARE INCORRECT (I.E. - DON'T MATCH THE HARDWARE) AND THE PROGRAM
 : IS RUN WITH THE INCORRECT ADDRESSES, A 'HUNG' CONDITION WILL BE

1141
1142
1143
1144
1145 001206 177170
1146 001210 177172
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196

:REPORTED AND THE PROGRAM WILL 'HALT' AS THE RX11 WILL NOT BE
:ABLE TO RESPOND TO COMMANDS.

::*****

RXCS: 177170
RXDB: 177172

:BIT ASSIGNMENT IN THE RXCS REGISTER.

:KEY: R - READ ONLY BIT
: W - WRITE ONLY BIT

- 15 - R - ERROR FLAG
- 14 - W - INITIALIZE (RECALIBRATE)
- 13 -
- 10 - NOT USED
- 8 -
- 7 - R - TRANSFER REQUEST (TR) FLAG
- 6 - R/W - INTERRUPT ENABLE
- 5 - R - DONE FLAG
- 4 - W - UNIT SELECT
- 3 - W - FUNCTION
- 2 - W - FUNCTION
- 1 - W - FUNCTION
- 0 - W - GO

:FUNCTION CODES:

- 0 + GO = FILL BUFFER
- 2 + GO = EMPTY BUFFER
- 4 + GO = WRITE SECTOR
- 6 + GO = READ SECTOR
- 10 (NOT USED)
- 12 + GO = READ STATUS 'A'
- 14 + GO = WRITE DELETED DATA
- 16 + GO = READ STATUS 'B' (CODES)

:THE FOLLOWING BIT ASSIGNMENTS REPRESENTS THE STATUS AT THE END OF A
:FUNCTION (EXCEPT FUNCTION 'READ STATUS B') DISPLAYED IN THE
:RX DATA BUFFER (RXDB).

- 15 -
- 10 - NOT USED
- 8 -
- 7 - SELECTED DRIVE READY *
- 6 - DELETED DATA
- 5 -
- 4 -
- 3 - WRITE PROTECT ERROR (WHEN AVAILABLE)
- 2 - INITIALIZE DONE **
- 1 - PARITY ERROR
- 0 - CRC ERROR

* VISIBLE ONLY IF THE FUNCTION WAS A #12 'READ STATUS A'

1197
1198
1199 C01212 000000
1200 001214 177570
1201 001216 177570
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252

;** VISIBLE ONLY AFTER AN INITIALIZE [KEY] OR [PROGRAMMED] WAS ISSUED.

DTESTP: .WORD 0 ;TEST SELECTION WORD
SWR: .WORD DSWR ;ADDRESS OF SWITCH REGISTER
DISPLAY: .WORD DDISP ; ADDRESS OF DISPLAY REGISTER

::*****

;START OF OPERATOR SELECTABLE TEST,DATA PATTERNS,AND DRIVE UNIT CONDITIONS
;SET THE TEST CONDITIONS WANTED (OR LEAVE 0) IN LOCATION CALLED
;'DTESTP' (LOC. 1212), BEFORE STARTING THE PROGRAM. WHEN THE PROGRAM
;STARTS IT WILL TYPE OUT THE TEST CONDITIONS IT IS OPERATING UNDER.

;SWITCH REGISTER BITS

: 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
: U1 U0 NOT USED D P P P T T T S S S

: U1 SELECT DRIVE UNIT 1
: U0 SELECT DRIVE UNIT 0
: IF NEITHER DRIVE IS SPECIFIED,PROGRAM WILL SELECT ALL DRIVES
: THAT ARE READY

: D = DELETED DATA FUNCTIONS

: IF THIS SWITCH IS ON ALL READ AND WRITE FUNCTIONS WILL BE IN THE
: DELETED DATA MODE

: P = DATA PATTERN SELECTION
: 0 DO PATTERN 7 (RANDOM)
: 1 ZEROS
: 2 ONES
: 3 FLOATING ZERO
: 4 FLOATING ONE
: 5 125
: 6 314
: 7 RANDOM

: T = FUNCTIONAL TESTS
: 0 DO TEST 7 (WRITE/READ/READ CHECK)
: 1 WRITE ONLY
: 2 WRITE/READ
: 3 WRITE/READ CHECK
: 4 READ CHECK ONLY
: 5 READ ONLY
: 6 WRITE/READ CHECK ON ALTERNATE DRIVES
: 7 WRITE/READ/READ CHECK *

: * NOTE: THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE
: TO INCRIMENT UP THROUGH ALL THE TRACKS DOING WRITE / READ CHECK FUNCTIONS.
: THIS VERIFIES THAT ALL THE TRACKS ARE ACCESSABLE.
: THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE
: OPERATOR AS INDICATED BELOW,AND ONLY READ CHECK THE DATA JUST WRITTEN.
: THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK,AFTER THE HEAD HAS BEEN

1253 ; MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS
 1254 ; THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT
 1255 ; PASS WILL BE RUN UNDER THIS NEW CONDITION.

1256 :
 1257 : S = TRACK SEQUENCING
 1258 : 0 DO SEQUENCE 7 (RANDOM)
 1259 : 1 INCREMENT
 1260 : 2 DECREMENT
 1261 : 3 INCREMENT/DECREMENT
 1262 : 4 BOUNCE
 1263 : 5 DECREASING BOUNCE
 1264 : 6 STROBE
 1265 : 7 RANDOM

1266 : *****

1267 :
 1268 :
 1269 :
 1270 ; SET THE OPERATING CONDITIONS IN THE SWITCH REGISTER (HARDWARE).
 1271 ; OR THE SOFTWARE SWITCH REGISTER (LOC. 176) BEFORE STARTING THE PROGRAM.

1272 :
 1273 ; SWITCHES DO THE FOLLOWING WHEN SET TO '1'.
 1274 :

1275 : SW 15 =HALT ON ERROR
 1276 : SW 14 =HALT AT END OF TEST
 1277 : SW 13 =DON'T PRINT ERROR MESSAGE
 1278 : SW 12 =TYPE ONLY 10 DATA ERRORS
 1279 : SW 11 =NO RETRY ON ERROR. LOG HARD ERROR
 1280 :
 1281 : SW 8 =NO RECALIBRATION ON SEEK ERRORS
 1282 :
 1283 : SW15 - SW0 =SELECT SOFTWARE SWITCH REGISTER
 1284 :
 1285 :
 1286 : *****

1287 :
 1288 :
 1289 :
 1290 001220 000005 SA200: RESET ; INITIALIZE THE RX01
 1291 001222 012737 177570 001214 MOV #177570,SWR ; RESET TO HARDWARE SWR.
 1292 001230 012706 001200 MOV #STACK,SP
 1293 001234 012746 000340 MOV #PR7,-(SP)
 1294 001240 012746 001246 MOV #2\$,-(SP)
 1295 001244 000002 RTI
 1296 001246 104401 016652 2\$: TYPE ,MREV ; PRINT NAME AND REVISION
 1297 001252 012737 001272 000004 MOV #3\$,4 ; SET TIME OUT VECTOR
 1298 001260 022777 177777 177726 CMP #177777,@SWR ; IS SOFTWARE SWR SELECTED
 1299 001266 001402 BEQ 4\$; YES, INSERT IT'S ADDRESS
 1300 001270 000423 BR 5\$; BRANCH IF NO TIMEOUT TRAP OCCURS
 1301 001272 022626 3\$: CMP (SP)+,(SP)+ ; RESTORE STACK AFTER TRAP
 1302 001274 012737 000176 001214 4\$: MOV #SWREG,SWR ; POINT TO SOFTWARE SWITCH REG.
 1303 001302 012737 000174 001216 MOV #DISPREG,DISPLAY ; POINT TO SOFTWARE DISP. REG.
 1304 :.SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
 1305 001310 005737 000042 TST @#42 ; ARE WE RUNNING UNDER XXDP/ACT?
 1306 001314 001006 BNE 64\$; BRANCH IF YES
 1307 001316 023727 001214 000176 CMP SWR,#SWREG ; SOFTWARE SWITCH REG SELECTED?
 1308 001324 001005 BNE 65\$; BRANCH IF NO

1309	001326	104405				GTSWR		::GET SOFT-SWR SETTINGS
1310	001330	000403				BR	65\$	
1311	001332	112737	000001	013776	64\$:	MOV	#1,\$AUTOB	::SET AUTO-MODE INDICATOR
1312	001340				65\$:			
1313	001340	012737	000006	000004	5\$:	MOV	#6,4	;RESTORE ERROR VECTOR
1314	001346	013703	001204			MOV	INTVEC,R3	;SET INTERRUPT VECTOR LOCATIONS
1315	001352	012723	007250			MOV	#INTSERV,(R3)+	
1316	001356	012713	000340			MOV	#340,(R3)	
1317	001362	012737	001234	012210		MOV	#001234,RAN1	;INITIALIZE CONSTANTS IN RANDOM NUMBER GENERATOR
1318	001370	012737	000765	012212		MOV	#000765,RAN2	
1319	001376	012705	017122			MOV	#PARLOG,R5	;SET UP TO CLEAR ALL COUNTERS
1320	001402	005025			1\$:	CLR	(R5)+	;OF ERRORS,ACCESSES,AND PASSES
1321	001404	022705	021142			CMP	#PASCNTR+2,R5	;HAVE ALL COUNTERS BEEN CLEARED
1322	001410	001374				BNE	1\$	
1323	001412	012705	017102		START1:	MOV	#RDRETRY,R5	;SET ALL RETRY COUNTERS TO 10
1324	001416	012725	000012		3\$:	MOV	#10,(R5)+	
1325	001422	022705	017122			CMP	#SRRETRY+2,R5	
1326	001426	001373				BNE	3\$	
1327	001430	104401	015265			TYPE	,MRXCS	;TYPE OUT DEVICE REGISTER ADDRESSES
1328	001434	013746	001206			MOV	RXCS,-(SP)	::SAVE RXCS FOR TYPEOUT
1329	001440	104403				TYPOS		::GO TYPE--OCTAL ASCII
1330	001442	006				.BYTE	6	::TYPE 6 DIGITS
1331	001443	000				.BYTE	0	::SUPPRESS LEADING ZEROS
1332	001444	104401	015275			TYPE	,MRXDB	
1333	001450	013746	001210			MOV	RXDB,-(SP)	::SAVE RXDB FOR TYPEOUT
1334	001454	104403				TYPOS		::GO TYPE--OCTAL ASCII
1335	001456	006				.BYTE	6	::TYPE 6 DIGITS
1336	001457	000				.BYTE	0	::SUPPRESS LEADING ZEROS
1337	001460	104401	015246			TYPE	,MINTVEC	;AND VECTOR ADDRESS
1338	001464	013746	001204			MOV	INTVEC,-(SP)	::SAVE INTVEC FOR TYPEOUT
1339	001470	104403				TYPOS		::GO TYPE--OCTAL ASCII
1340	001472	003				.BYTE	3	::TYPE 3 DIGIT(S)
1341	001473	000				.BYTE	0	::SUPPRESS LEADING ZEROS
1342	001474	004737	007122			JSR	PC,DONECK	;WAIT FOR DONE BIT AFTER RECAL
1343	001500	032777	000004	177502		BIT	#BIT2,@RXDB	;IS INIT DONE SET
1344	001506	001002				BNE	1\$;YES SKIP NEXT INSTRUCTION
1345	001510	104401	015633			TYPE	,MINIT1	;NO,PRINT NO INIT DONE ERROR
1346	001514	005777	177466		1\$:	TST	@RXCS	;WAS THERE AN ERROR ON START RECAL
1347	001520	100002				BPL	TESTDR	;IF NOT CONTINUE
1348	001522	004737	002550			JSR	PC,HOME	;YES,DO RECAL AGAIN
1349	001526	012737	000012	017112	TESTDR:	MOV	#10,@P2RETRY	;SET THE PARITY RETRY COUNTER
1350	001534	005037	010704			CLR	UNITSEL	;CLEAR UNIT SELECTION WORD
1351	001540	105777	177444			TSTB	@RXDB	;IS DRY BIT SET FOR UNIT 0 AFTER RECAL
1352	001544	100003				BPL	2\$;IF NOT SKIP NEXT INSTRUCTION
1353	001546	052737	000200	010704		BIS	#BIT7,UNITSEL	;YES,SET UNIT 0 READY BIT
1354	001554	032737	140000	001212	2\$:	BIT	#140000,DTESTP	;WERE ANY DRIVES SPECIFIED
1355	001562	001445				BEQ	NOSEL	;NO,SEE IF UNIT 1 IS READY
1356	001564	005737	001212			TST	DTESTP	;WAS UNIT 1 SELECTED
1357	001570	100044				BPL	OPCOND	;NO,IT MUST BE UNIT 0
1358	001572	004737	001614			JSR	PC,DRY1	;SEE IF UNIT 1 IS READY
1359	001576	032737	040000	001212		BIT	#40000,DTESTP	;WAS UNIT 0 SELECTED
1360	001604	001036				BNE	OPCOND	;YES,CONTINUE
1361	001606	105037	010704			CLRB	UNITSEL	;NO,CLEAR UNIT 0 READY BIT
1362	001612	000433				BR	OPCOND	
1363								
1364	001614	012777	000033	177364	DRY1:	MOV	#RD1STAT,@RXCS	;READ STATUS OF UNIT 1

```

1365 001622 004737 007122 JSR PC,DONECK ;WAIT FOR DONE BIT
1366 001626 132777 000002 177354 BITB #2,@RXDB ;IS PARITY ERROR BIT SET
1367 001634 001411 BEQ 2$ ;NO,CONTINUE
1368 001636 004737 002046 JSR PC,STATER ;YES,GO HANDLE PARITY ERROR
1369 001642 000764 BR DRY1 ;REISSUE FUNCTION
1370 001644 032777 000004 177336 BIT #BIT2,@RXDB ;INIT DONE SHOULD NOT BE SET
1371 001652 001402 BEQ 2$
1372 001654 104401 015665 TYPE ,MINIT2 ;INIT DONE WAS SET PRINT ERROR
1373 001660 105777 177324 2$: TSTB @RXDB ;IS UNIT 1 DRY BIT SET
1374 001664 100003 BPL 3$ ;IF NOT SKIP NEXT INSTRUCTION
1375 001666 052737 100000 010704 BIS #BIT15,UNITSEL ;YES,SET UNIT 1 READY BIT
1376 001674 000207 3$: RTS PC
1377
1378 001676 004737 001614 NOSEL: JSR PC,DRY1 ;TEST UNIT 1 FOR DRY SET
1379 001702 105737 010704 OPCOND: TSTB UNITSEL ;IS UNIT 0 SELECTED
1380 001706 100016 BPL 1$ ;NO
1381
1382 ;TEST ACT11 LOAD MEDIA INDICATOR
1383
1384 001710 123727 000041 000010 CMPB @#41,#10 ;DOES LOCATION 41 CONTAIN THE NUMBER 10?
1385 001716 001012 BNE 1$ ;BRANCH IF NOT
1386 001720 005737 000042 TST @#42 ;CHECK FOR RXDP OPERATION
1387 001724 001404 BEQ 2$
1388 001726 042737 000200 010704 BIC #200,UNITSEL ;IN CHAIN MODE, DESELECT UNIT 0
1389 001734 000403 BR 1$ ;AND DO NOT HALT
1390 001736 104401 014741 2$: TYPE, DOLOAD ;INFORM USER TO REMOVE LOAD MEDIUM
1391 ;FROM UNIT 0 AND REPLACE WITH
1392 ;A 'SCRATCH' DISKETTE IF HE
1393 001742 000000 HALT ;WISHES TO TEST UNIT 0
1394 001744 000241 1$: CLC ;CLEAR C BIT FOR ROR'S THAT FOLLOW
1395 001746 013737 001212 011370 MOV DTESTP,SEQUEN ;PUT INITIAL CONDITIONS IN SEQUENCE
1396 001754 042737 177770 011370 BIC #177770,SEQUEN ;CLEAR ALL BUT SEQUENCE BITS
1397 001762 013737 001212 002140 MOV DTESTP,TEST ;PUT INITIAL CONDITIONS IN TEST
1398 001770 042737 177707 002140 BIC #177707,TEST ;CLEAR ALL BUT TEST BITS
1399 001776 006037 002140 ROR TEST ;RIGHT JUSTIFY TEST BITS
1400 002002 006037 002140 ROR TEST
1401 002006 006037 002140 ROR TEST
1402 002012 013737 001212 010420 MOV DTESTP,PAT ;PUT INITIAL CONDITIONS IN PATTERN
1403 002020 006137 010420 ROL PAT ;RIGHT JUSTIFY PATTERN BITS
1404 002024 006137 010420 ROL PAT
1405 002030 000337 010420 SWAB PAT
1406 002034 042737 177770 010420 BIC #177770,PAT ;CLEAR ALL BUT PATTERN BITS
1407 002042 000137 002722 JMP XSTART
1408
1409 002046 005237 017122 STATER: INC PARLOG ;INC PARITY ERROR COUNTER
1410 002052 104401 015265 TYPE ,MRXCS ;PRINT THE RXCS CONTENTS
1411 002056 017746 177124 MOV @RXCS,-(SP) ;;SAVE @RXCS FOR TYPEOUT
1412 002062 104403 TYPOS ;;GO TYPE--OCTAL ASCII
1413 002064 003 .BYTE 3 ;;TYPE 3 DIGIT(S)
1414 002065 000 .BYTE 0 ;;SUPPRESS LEADING ZEROS
1415 002066 005337 017112 DEC P2RETRY ;HAVE 10 ERRORS OCCURED
1416 002072 001405 BEQ 1$ ;YES,LOG HARD ERROR
1417 002074 104401 016115 TYPE ,MPAR ;NO,TYPE PARITY ERROR
1418 002100 104401 015150 TYPE ,MCRLF
1419 002104 000207 RTS PC ;RETURN
1420 002106 005237 017124 1$: INC HPARLOG ;INC HARD PARITY ERROR LOG
  
```

1421	002112	104401	016053				TYPE ,MUNREC	:PRINT UNRECOVERABLE PARITY ERROR
1422	002116	104401	016115				TYPE ,MPAR	
1423	002122	104401	015150				TYPE ,MCRLF	
1424	002126	104401	016000				TYPE ,MHALT4	
1425	002132	000000				HLT4:	HALT	:HALT BECAUSE OF HARD PARITY ERROR
1426	002134	000137	001220				JMP SA200	:IF CONTINUE IS PRESSED GO TO START
1427	002140	000000				TEST:	0	
1428								
1429	002142	005002				TSTLIMITS:	CLR R2	:MESSAGE FLAG
1430	002144	005737	001200				TST OD	:TEST FOR NO SELECTION OF TRACKS
1431	002150	001441					BEQ 1\$:IT WILL BE STANDARD LIMITS
1432	002152	123727	001201	000114			CMPB ID,#114	:TEST FOR MAXIMUM ID LIMIT
1433	002160	101112					BHI 14\$	
1434	002162	001431					BEQ 2\$	
1435	002164	123737	001200	001201	3\$:		CMPB OD,ID	:OD MUST BE EQUAL OR LESS THAN ID
1436	002172	101105					BHI 14\$	
1437	002174	004737	002432				JSR PC,TSTMSG	:HAS INITIAL MESSAGE BEEN TYPED
1438	002200	104401	015606				TYPE ,MOD	:TYPE BOTH ID AND OD LIMITS
1439	002204	113737	001200	002216			MOVB OD,7\$	
1440	002212	004537	014544				JSR R5,SGLDEC	:TYPE SINGLE DECIMAL WORD
1441	002216	000000			7\$:		OPEN	
1442	002220	104401	015145				TYPE ,DBLSP	
1443	002224	104401	015612				TYPE ,MID	
1444	002230	113737	001201	002242			MOVB ID,10\$	
1445	002236	004537	014544				JSR R5,SGLDEC	
1446	002242	000000			10\$:		OPEN	
1447	002244	000403					BR 1\$	
1448	002246	105737	001200		2\$:		TSTB OD	:IS OD = 0
1449	002252	001344					BNE 3\$:NO,GO PRINT IT
1450	002254	005737	001202		1\$:		TST FIRST	:TEST FOR NO SELECTION OF SECTORS
1451	002260	001451					BEQ 13\$:IT WILL BE STANDARD LIMITS
1452	002262	123727	001203	000032			CMPB LAST,#32	:TEST FOR NORMAL LAST SECTOR LIMIT
1453	002270	101053					BHI 15\$	
1454	002272	001440					BEQ 4\$	
1455	002274	105737	001202		5\$:		TSTB FIRST	:THERE IS NO 0 SECTOR
1456	002300	001447					BEQ 15\$	
1457	002302	123737	001202	001203			CMPB FIRST, LAST	:FIRST MUST BE EQUAL TO OR LESS
1458	002310	101043					BHI 15\$:THAN LAST
1459	002312	004737	002432				JSR PC,TSTMSG	
1460	002316	104401	015145				TYPE ,DBLSP	
1461	002322	104401	015616				TYPE ,MFIRST	:TYPE BOTH FIRST AND LAST LIMITS
1462	002326	113737	001202	002340			MOVB FIRST,11\$	
1463	002334	004537	014544				JSR R5,SGLDEC	
1464	002340	000000			11\$:		OPEN	
1465	002342	104401	015145				TYPE ,DBLSP	
1466	002346	104401	015625				TYPE ,MLAST	
1467	002352	113737	001203	002364			MOVB LAST,12\$	
1468	002360	004537	014544				JSR R5,SGLDEC	
1469	002364	000000			12\$:		OPEN	
1470	002366	104401	015336				TYPE ,DBLLF	
1471	002372	000404					BR 13\$	
1472	002374	123727	001202	000001	4\$:		CMPB FIRST,#1	:IS FIRST SET TO 1
1473	002402	001334					BNE 5\$:NO, GO PRINT IT
1474	002404	000207			13\$:		RTS PC	
1475								
1476	002406	104401	016461		14\$:		TYPE ,OD2BIG	:PRINT TRACK LIMITS OUT OF RANGE

1477	002412	005037	001200		CLR OD	
1478	002416	000716			BR 1\$	
1479	002420	104401	016553	15\$:	TYPE ,S2BIG	:PRINT SECTOR LIMITS OUT OF RANGE
1480	002424	005037	001202		CLR FIRST	
1481	002430	000765			BR 13\$	
1482						
1483	002432	005702		TSTMSG:	TST R2	:TEST MESSAGE FLAG
1484	002434	001005			BNE 1\$	
1485	002436	104401	015544		TYPE ,MNONSTD	:MESSAGE HEADING HAD NOT BEEN TYPED
1486	002442	104401	015150		TYPE ,MCRLF	
1487	002446	005202			INC R2	:SET THE MESSAGE FLAG
1488	002450	000207		1\$:	RTS PC	
1489						
1490	002452	104401	015150	ICOND:	TYPE ,MCRLF	
1491	002456	104401	015515		TYPE ,MICON	:TYPE INITIAL CONDITIONS
1492	002462	105737	010704		TSTB UNITSEL	:TEST FOR DRIVE SELECTION
1493	002466	100002			BPL 1\$	
1494	002470	104401	015204		TYPE ,MUNIT0	:TYPE UNIT 0 SELECTED
1495	002474	005737	010704	1\$:	TST UNITSEL	
1496	002500	100002			BPL 2\$	
1497	002502	104401	015214		TYPE ,MUNIT1	:TYPE UNIT 1 SELECTED
1498	002506	104401	015136	2\$:	TYPE ,TAB	
1499	002512	004737	010012		JSR PC ,TYPESEL	:TYPE TEST SELECTIONS
1500	002516	104401	015150		TYPE ,MCRLF	
1501	002522	005737	010704		TST UNITSEL	:WERE ANY DRIVES READY
1502	002526	001005			BNE CONT12	:YES,CONTINUE
1503	002530	104401	015114		TYPE ,MNODRY	:NO,TYPE NO DRIVES MESSAGE
1504	002534	000000		HLT5:	HALT	:NO DRIVES CAN'T CONTINUE
1505	002536	000137	001220		JMP SA200	:IF CONTINUE IS PRESSED RESTART
1506	002542	004737	002142	CONT12:	JSR PC ,TSTLIMITS	:TYPE NONSTANDARD TRK AND SEC LIMITS
1507	002546	000207			RTS PC	
1508						
1509	002550	104406		HOME:	CKSWR	
1510	002552	032777	000400 176434		BIT #SW8,@SWR	:TEST THE NO RECAL SWITCH
1511	002560	001046			BNE RTN	:RETURN IF THE SWITCH IS SET
1512	002562	012737	000012 017112		MOV #10.,P2RETRY	:SET UP THE PARITY RETRY COUNTER
1513	002570	012737	000012 017106		MOV #10.,DDRETRY	:USE THE DD RETRY COUNTER FOR RECAL RETRIES
1514	002576	012777	040001 176402	2\$:	MOV #RECAL,@RXCS	:ISSUE RECAL FUNCTION
1515	002604	004737	007122		JSR PC,DONECK	:WAIT FOR DONE FLAG
1516	002610	032777	000002 176372		BIT #2,@RXDB	:WAS THERE A PARITY ERROR
1517	002616	001403			BEQ 1\$:NO,CHECK FOR ANY OTHER ERROR
1518	002620	004737	002046		JSR PC,STATER	:YES,GO REPORT IT
1519	002624	000764			BR 2\$:RETRY RECAL
1520	002626	005777	176354	1\$:	TST @RXCS	:IS THE ERROR FLAG SET
1521	002632	100016			BPL XHOME	:IF NOT RETURN
1522	002634	004737	010200		JSR PC,RDCODE	:YES,PRINT STATUS REGISTERS
1523	002640	123727	010072 000040		CMPB BSTAT,#40	:IS THE B CODE LESS THAN CODE 40
1524						: (RECAL CODES ARE 10,20,AND 30)
1525	002646	002010			BGE XHOME	:IF NOT RETURN
1526	002650	005337	017106		DEC DDRETRY	:HAVE 10 ERRORS OCCURED
1527	002654	001350			BNE 2\$:NO,RETRY
1528	002656	104401	016011		TYPE ,MHALT3	:YES,CAN'T CONTINUE AS RX11 WILL NOT RECAL
1529	002662	000000		HLT3:	HALT	
1530	002664	000137	001220		JMP SA200	:IF CONT. SWITCH IS PRESSED GO TO START
1531	002670	012737	000001 011362	XHOME:	MOV #1,PRESTRK	:RESET THE PRESENT TRACK TO TRACK 1
1532	002676	000207		RTN:	RTS PC	:RETURN

1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588

:DECODE TEST BITS OF INITIAL CONDITIONS

:BITS 3,4,AND 5 OF THE INITIAL SWR SELECTED THE TEST WHICH
 :IS TO BE PREFORMED. THEY ARE AS FOLLOWS:

	BITS			TESTS
	5	4	3	
	0	0	0	NO TEST SELECTED (DEFAULT TO TEST 7)
	0	0	1	WRITE ONLY
	0	1	0	WRITE FOLLOWED BY READ
	0	1	1	WRITE FOLLOWED BY READ AND VERIFY
	1	0	0	READ AND VERIFY ONLY
	1	0	1	READ ONLY (CRC CHECK)
	1	1	0	READ AND VERIFY DELETED DATA
	1	1	1	WRITE FOLLOWED BY READ FOLLOWED BY ANOTHER READ AND VERIFY

```

RESTART:      MOV #STACK,SP          :RESET THE STACK POINTER
              MOV #PR7,-(SP)       :AND INTERRUPT LEVEL
              MOV #1$,-(SP)
              RTI
1$:           INC RESTCNTR         :INC RESTART COUNTER
XSTART:      JSR PC,ICOND         :TYPE OUT INITIAL CONDITIONS
TESTSEL:     CLR CHARCT          :CLEAR EOP CHARACTER COUNTER
              MOV #10.,RDRETRY     :SET UP READ AND WRITE RETRY COUNTERS
              MOV #10.,WTRETRY
              CLR ERWRT            :CLEAR WRITE CAUSED BY ERROR FLAG
              CLR RDAFTWT         :CLEAR READ AFTER WRITE FLAG
              BICB #377,@#BRONTEST :CLEAR OUT BRANCH OFFSET
              TST TEST            :IF NO TEST SPECIFIED FORCE TEST 7
              BNE 1$
              MOV #7,TEST
1$:          MOV TEST,R4
              DEC R4              :ADJUST TEST BITS FOR CORRECT
              ASL R4              :BRANCH OFFSET
              BISB R4,@#BRONTEST  :INSERT OFFSET TO BRANCH INST.
BRONTEST:   BR .                :BRANCH BY TEST OFFSET
              JMP WRONLY         :WRITE ONLY FUNCTION
              JMP WRTRD         :WRITE THEN READ FUNCTION
              JMP WTRDCK        :WRITE THEN READ VERIFY
              JMP RDCHK         :READ VERIFY
              JMP RDONLY        :READ ONLY FUNCTION
              JMP DRVSWP        :WRITE,AND READ VERIFY ON ALTERNATING DRIVES
              JMP TEST7         :WRITE,READ,FOLLOWED BY READ VERIFY
  
```

:THIS IS A WRITE ONLY FUNCTION USING DATA PATTERN AND
 :TRACK SEQUENCE SPECIFIED BY INITIAL SWR SETTINGS

```

1589
1590
1591 003052 004737 010312 WRONLY: JSR PC,GETPATTERN ;SET UP SOFTWARE DATA BUFFER
1592 003056 004737 011144 XWRONLY: JSR PC,INITTRACKS ;SET UP ID,OD,AND TRACK COUNTER
1593 003062 004737 010614 JSR PC,GETUNIT ;SET UP DRIVE UNIT SELECTION
1594 003066 004737 011254 1$: JSR PC,GETTRACK ;PICK UP NEXT TRACK
1595 003072 004737 003112 JSR PC,WRITE ;DO THE WRITE FUNCTION
1596 003076 005337 011356 DEC TRKCNTR ;TEST TRACK COUNTER
1597 003102 001371 BNE 1$
1598 003104 004737 010732 JSR PC,STOP ;CHECK FOR LAST DRIVE AND EOP
1599 003110 000762 BR XWRONLY ;NEXT PASS
1600
1601 003112 004737 012216 WRITE: JSR PC,INITSECTOR ;SET UP FIRST, LAST, AND SECTOR COUNTER
1602 003116 004737 012316 XWRITE: JSR PC,GETSECTOR ;PICK UP NEXT SECTOR
1603 003122 012737 000012 017120 MOV #10.,SRETRY
1604 003130 012737 000012 017114 ONEWRT: MOV #10.,PRETRY ;SET RETRY COUNTER
1605 003136 004737 005656 FILLBUF: JSR PC,ADJSUM ;ADJUST DATA BUFFER AND CHECK SUM FOR ADDRESSES
1606 003142 012746 003372 MOV #FILLDONE,-(SP) ;PUT GOOD RETURN ON STACK
1607 003146 012746 003214 MOV #FILLER,-(SP) ;PUT ERROR RETURN ON STACK
1608 003152 005037 006434 CLR BYTECNTR
1609 003156 012746 000200 MOV #PR4,-(SP)
1610 003162 012746 003170 MOV #1$,-(SP)
1611 003166 000002 RTI
1612 003170 012777 000101 176010 1$: MOV #FBIE,@RXCS ;EXICUTE FILLBUFER COMMAND
1613 003176 004737 007014 FILLFLAG: JSR PC,TRCK ;TEST FOR TRANSFER REQUEST FLAG
1614 003202 112077 176002 XFRBYTE: MOV B(R0)+,@RXDB ;TRANSFER DATA BYTE
1615 003206 005237 006434 INC BYTECNTR
1616 003212 000771 BR FILLFLAG ;WAIT FOR NEXT TR FLAG
1617
1618 003214 005726 FILLER: TST (SP)+ ;REMOVE THE DONE RETURN FRGM THE STACK
1619 003216 012737 015464 003264 MOV #MFIL,PTYP1+2 ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYPOUT 1
1620 003224 012737 015464 003354 MOV #MFIL,PTYP2+2 ;PUT ADDR OF FILLBUF MESSAGE IN PAR ERR TYPOUT 2
1621 003232 012737 003136 003334 MOV #FILLBUF,PCONT+2 ;IF NOT HARD ERR RETURN THROUGH PCONT TO FILLBUF
1622 003240 000137 003244 JMP PARTEST ;PRINT OUT PAR ERR AND TEST CONDITIONS FOR RETRY
1623
1624
1625 003244 005237 017122 PARTEST: INC PARLOG ;INCREMENT PARITY ERROR
1626 003250 104406 CKSWR
1627 003252 032777 020000 175734 BIT #SW13,@SWR ;TEST DON'T PRINT ERROR SWITCH
1628 003260 001006 BNE CONT4
1629 003262 104401 000000 PTYP1: TYPE ,OPEN ;PRINT THE PARITY ERROR MESSAGE
1630 003266 104401 016115 TYPE ,MPAR
1631 003272 104401 015150 TYPE ,MCR LF
1632 003276 104406 CONT4: CKSWR
1633 003300 005777 175710 TST @SWR ;TEST HALT ON ERROR SWITCH
1634 003304 100001 BPL CONT13
1635 003306 000000 HLT6: HALT ;HALT ON ERROR
1636 003310 032777 004000 175676 CONT13: BIT #SW11,@SWR ;TEST NO RETRY SWITCH!
1637 003316 001007 BNE CONT5 ;IF SET LOG HARD ERROR
1638 003320 005337 017114 DEC PRETRY ;DECREMENT RETRY COUNTER
1639 003324 005737 017114 TST PRETRY ;HAVE 10 ERRORS OCCURED
1640 003330 001402 BEQ CONT5 ;IF CLEARED LOG HARD ERROR
1641 003332 000137 003136 PCONT: JMP FILLBUF ;RETURN TO RETRY TEST THROUGH HERE
1642 003336 005237 017124 CONT5: INC HPARLOG ;INC.HARD PARITY ERROR COUNTER
1643 003342 104401 015336 TYPE ,DBLLF
1644 003346 104401 016053 TYPE ,MUNREC ;TYPE HARD PARITY ERROR

```



```

1645 003352 104401 000000 P1YP2:          TYPE ,OPEN
1646 003356 104401 016115          TYPE ,MPAR
1647 003362 104401 015150          TYPE ,MCRLF
1648 003366 000137 010074          JMP DELUNIT          ;GO DELETE THE UNIT,CAN NOT CONTINUE
1649
1650          ;SWITCH 9 OF INITIAL TEST CONDITIONS SWITCH SETTINGS IS THE DELETED DATA
1651          ;FUNCTION INDICATOR. WHEN THIS BIT IS SET ALL WRITE / READ FUNCTIONS WILL
1652          ;SET AND CHECK THE DELETED DATA BIT ON THE DISKETTE.
1653          ;:*****
1654
1655 003372 012737 000012 017114 FILLDONE:      MOV #10,PRETRY          ;SET UP RETRY COUNTER
1656 003400 012746 003462 REWRITE:      MOV #WRITDONE,-(SP)    ;SET GOOD RETURN ON STACK
1657 003404 012746 003510          MOV #WRITER,-(SP)     ;SET ERROR RETURN ON STACK
1658 003410 112737 000105 004120          MOVB #WRTIE,FUNCTION  ;SET FUNCTION WORD TO WRITE
1659 003416 032737 001000 001212          BIT #BIT9,DTESTP     ;TEST FOR WRITE DELETED DATA
1660 003424 001403          BEQ 1$
1661 003426 112737 000115 004120          MOVB #WTDIE,FUNCTION
1662 003434 062737 000001 021126 1$:          ADD #1,WTCNTR          ;INC TOTAL WRITE FUNCTIONS COUNTER
1663 003442 005537 021130          ADC WTCNTR+2         ;DOUBLE PRECISION COUNTER
1664 003446 004737 004022          JSR PC,COMMWORD     ;TRANSFER COMMAND TO DRIVE
1665 003452 004737 007122          JSR PC,DONECK       ;TEST FOR DONE FLAG
1666 003456 000137 007204          JMP NOINTER         ;NO INTERRUPT ERROR
1667
1668 003462 005737 004124 WRTDONE:      TST ERWRT            ;IS THIS A REWRITE FROM A DATA ERROR
1669 003466 001004          BNE 1$              ;YES GO REREAD THIS SECTOR
1670 003470 005337 012306          DEC SECCNTR         ;NO,TEST SECTOR COUNTER
1671 003474 001003          BNE 2$              ;NOT LAST SECTOR GO TO NEXT ONE
1672 003476 000207          RTS PC
1673 003500 000137 004246 1$:          JMP REREAD          ;REREAD THE SECTOR
1674 003504 000137 003116 2$:          JMP XWRITE
1675
1676 003510 005726 WRITER:      TST (SP)+           ;REMOVE THE DONE RETURN FROM THE STACK
1677 003512 032737 000002 010070          BIT #BIT1,ASTAT     ;IS THIS A PARITY ERROR
1678 003520 001413          BEQ WRTSEK         ;NO, IT MUST BE A SEEK ERROR
1679          ;PARITY ERROR DURING A WRITE FUNCTION
1680 003522 012737 016106 003264          MOV #MWRITE,PTYP1+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 1
1681 003530 012737 016106 003354          MOV #MWRITE,PTYP2+2 ;PUT ADDR OF WRITE MESSAGE IN PAR ER TYP0UT 2
1682 003536 012737 003400 003334          MOV #REWRITE,PCONT+2 ;IF NOT HARD ERR RETURN THROUGH PCONT TO REWRITE
1683 003544 000137 003244          JMP PARTEST        ;GO INC LOG AND TEST FOR RETRY
1684
1685          ;SEEK ERROR DURING A WRITE FUNCTION
1686 003550 012737 003130 003674 WRTSEK:      MOV #ONEWRT,SEKRTY+2 ;SETUP FOR WRT RETRY ON SEEK ERROR
1687          ;(AFTER A RECAL. THE CONTENTS OF SECTOR 1,
1688          ;TRACK 1 ARE LOADED INTO THE SECTOR BUFFER.
1689          ;TO REWRITE THE CORRECT DATA THE PROGRAM
1690          ;MUST REFILL THE SECTOR BUFFER.
1691 003556 012737 016106 003630          MOV #MWRITE,STYP1+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYP0UT 1
1692 003564 012737 016106 003722          MOV #MWRITE,STYP2+2 ;PUT ADDR OF WRITE MESSAGE IN SEEK ER TYP0UT 2
1693 003572 004737 003576          JSR PC,SEEKER      ;RECORD SEEK ERROR
1694
1695 003576 012703 017134 SEEKER:      MOV #ZSEKLOG,R3     ;SETUP INC INSTRUCTION FOR UNIT 0 LOG
1696 003602 004737 021142          JSR PC,U1LOG        ;TEST FOR UNIT 1 LOG
1697 003606 005213          INC (R3)           ;INCREMENT SEEK ERROR LOG
1698 003610 004737 003770          JSR PC,TRKERR      ;INC ERROR PER TRACK COUNTERS
1699 003614 104406          CKSWR
1700 003616 032777 020000 175370          BIT #SW13,@SWR     ;CHECK DON'T PRINT ERROR SWITCH

```

1701	003624	0C1004			BNE SWHLT1	
1702	003626	104401	016106	STYP1:	TYPE ,MWRITE	:PRINT WRITE (READ) SEEK ERROR
1703	003632	004737	003736		JSR PC,SEKTYP	
1704	003636	104406		SWHLT1:	CKSWR	
1705	003640	005777	175350		TST @SWR	:TEST THE HALT ON ERROR SWITCH
1706	003644	100001			BPL CONT14	
1707	003646	000000		HLT7:	HALT	:HALT ON THE ERROR
1708	003650	032777	004000	175336	CONT14:	BIT #SW11,@SWR
1709	003656	001007			BNE HARDSK	:CHECK THE NO RETRY SWITCH
1710	003660	005337	017120		DEC SRETRY	:IF SET LOG HARD SEEK ERROR
1711	003664	001404			BEQ HARDSK	:HAVE 10 ERRORS BEEN LOGED
1712	003666	004737	002550		JSR PC,HOME	:YES LOG HARD ERROR
1713	003672	000137	003130		SEKRTY:	:RECALIBRATE DRIVES ON SEEK ERROR
1714	003676	012703	017174		HARDSK:	JMP ONEWRT
1715	003702	004737	021142		MOV #ZHSEKLOG,R3	:NO RETRY WRITE COMMAND (READ COMAND)
1716	003706	005213			JSR PC,U1LOG	:SET INC. INST.FOR UNIT 0 ERR LOG
1717	003710	104401	015336		INC (R3)	:TEST FOR UNIT 1 ERROR LOG
1718	003714	104401	016053		TYPE ,DBLLF	:HARD SEEK ERROR
1719	003720	104401	016106		TYPE ,MUNREC	:TYPE UNRECOVERABLE SEEK ERROR
1720	003724	004737	003736		STYP2:	TYPE ,MWRITE
1721	003730	062706	000002		JSR PC,SEKTYP	:ON WRITE (READ) COMMAND
1722	003734	000207			ADD #2,SP	
1723					RTS PC	:REMOVE SEEK ERROR FROM STACK POINTER
1724	003736	104401	016073		SEKTYP:	:RETURN TO NEXT SECTOR BY DONE RETURN ON STACK
1725	003742	104401	014665		TYPE ,MSEEK	:TYPE SEEK ERROR
1726	003746	013737	011362	003760	TYPE ,MPRES	:TYPE ADDRESS OF TRACK MOVED FROM
1727	003754	004537	014544		MOV PRESTRK,1\$	
1728	003760	000000			JSR R5,SGLDEC	
1729	003762	104401	015150		1\$:	OPEN
1730	003766	000207			TYPE ,MCRLF	
1731					RTS PC	
1732	003770	013705	011360		TRKERR:	MOV TARGET,R5
1733	003774	006305			ASL R5	:SET UP TO INC ERROR PER TRACK COUNTER
1734	003776	062705	020442		ADD #UOTRK,R5	:ADJUST FOR EVEN ADDRESS
1735	004002	032737	000020	010704	BIT #BIT4,UNITSEL	:ADDIN ADDRESS OF UNIT 0 LOG
1736	004010	001402			BEQ 1\$:CHECK THE UNIT SELECTION BIT
1737	004012	062705	000232		ADD #232,R5	:IF CLEARED UNIT 0 IS ACTIVE
1738	004016	005215			1\$:	ADD #232,R5
1739	004020	000207			INC (R5)	:ADJUST FOR UNIT 1 LOG
1740					RTS PC	:INC THE CORRECT COUNTER
1741	004022	013705	011360		COMMWORD:	MOV TARGET,R5
1742	004026	006305			ASL R5	:GET TRACK NUMBER
1743	004030	006305			ASL R5	:MULTIPLY BY 4 TO DOUBLE PRECISION
1744	004032	062705	017272		ADD #TKACC,R5	:INTERLEAVE COUNTER LOCATIONS
1745	004036	062715	000001		ADD #1,(R5)	:ADD ON ADDRESS OF TRACK ACCESS COUNTER
1746	004042	005565	000002		ADC 2(R5)	:INCREMENT THE COUNTER
1747	004046	153737	010704	004120	BISB UNITSEL,FUNCTION	:ADD CARRY TO HIGH ORDER WORD
1748	004054	013777	004120	175124	MOV FUNCTION,@RXCS	:SET UNIT SELECTION BIT IN COMMAND WORD
1749	004062	004737	007014		JSR PC,TRCK	:SEND OUT COMMAND TO DRIVE
1750	004066	113777	012310	175114	MOVB TSECTOR,@RXDB	:WAIT FOR TR FLAG
1751	004074	004737	007014		JSR PC,TRCK	:SEND OUT TARGET SECTOR
1752	004100	113777	011360	175102	MOVB TARGET,@RXDB	:WAIT FOR TR FLAG
1753	004106	005046			CLR -(SP)	:SEND OUT TARGET TEACK
1754	004110	012746	004116		MOV #1\$,-(SP)	:LOWER INTERUPT LEVEL TO ALLOW AN INTERUPT
1755	004114	000002			RTI	
1756	004116	000207			1\$:	RTS PC

```
1757
1758 004120 000000
1759 004122 000000
1760 004124 000000
1761 004126 000000
1762
1763
1764
1765
1766
1767
1768
1769 004130 005137 004122
1770 004134 004737 010312
1771 004140 004737 011144
1772 004144 004737 010614
1773 004150 004737 011254
1774 004154 004737 003112
1775 004160 004737 004200
1776 004164 005337 011356
1777 004170 001367
1778 004172 004737 010732
1779 004176 000760
1780
1781
1782
1783
1784
1785
1786 004200 004737 012216
1787 004204 004737 012316
1788 004210 012737 000012 017106
1789 004216 012737 000012 017110
1790 004224 012737 000012 017114
1791 004232 012737 000012 017120
1792 004240 012737 000012 017116
1793 004246 005037 004126
1794 004252 012746 004316
1795 004256 012746 004460
1796 004262 112737 000107 004120
1797 004270 062737 000001 021132
1798 004276 005537 021134
1799 004302 004737 004022
1800 004306 004737 007122
1801 004312 000137 007204
1802
1803
1804
1805 004316 022737 000012 017102
1806 004324 001420
1807 004326 012703 017150
1808 004332 004737 021142
1809 004336 005213
1810 004340 012737 000012 017102
1811 004346 104401 015336
1812 004352 104401 015725

FUNCTION: 0
RDAFTWT: 0 ;READ AFTER WRITE FUNCTION FLAG
ERWRT: 0 ;WRITE CAUSED BY DATA ERROR FLAG
DATAACK: 0 ;DATA CHECK ON CRC ERROR FLAG

:*****
:THIS IS A WRITE ALL SECTORS FOLLOWED BY A READ ALL SECTORS
:WITH DATA PATTERNS AND HEAD SEQUENCING SET BY INITIAL SWR
: T = 2

WRTRD: COM RDAFTWT ;SET READ AFTER WRITE FLAG
XWRTRD: JSR PC,GETPATTERN
JSR PC,INITTRACKS
1$: JSR PC,GETUNIT
JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READ
DEC TRKCNTR
BNE 1$
JSR PC,STOP
BR XWRTRD

:*****
:READ DATA FROM THE DISKETTE

READ: JSR PC,INITSECTOR
XREAD: JSR PC,GETSECTOR
MOV #10,DDRETRY ;SET UP RETRY COUNTERS FOR DELETED DATA
MOV #10,DATARETRY ;DATA ERROR
MOV #10,PRETRY ;PARITY
MOV #10,SRETRY ;SEEK
MOV #10,CRETRY ;AND CRC ERRORS
REREAD: CLR DATAACK ;CLEAR CRC DATA CHECK FLAG
MOV #RDDONE,-(SP) ;SET GOOD RETURN ON STACK
MOV #RDERR,-(SP) ;SET READ ERROR RETURN ON STACK
MOV# #RDIE,FUNCTION
ADD #1,RDCNTR ;INC TOTAL READ FUNCTIONS COUNTER
ADC RDCNTR+2 ;DOUBLE PRECISION COUNTER
JSR PC,COMMWORD
JSR PC,DONECK ;TEST FOR DONE
JMP NOINTER ;NO INTERRUPT ON DONE

RDDONE: CMP #10,DDRETRY ;IS READ RETRY EQUAL TO 10
BEQ CONT1 ;YES,NO ERRORS OCCURED
MOV #ZRDLOG,R3 ;SET INC. INST. FOR UNIT 0 ERROR LOG
JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOG
INC (R3) ;INC RECOVERABLE READ LOG
MOV #10,DDRETRY ;RESET READ RETRY COUNTER
TYPE ,DBLLF ;TYPE RECOVERABLE READ ERROR
TYPE ,MREC
```

```

1813 004356 104401 015772          TYPE ,MREAD
1814 004362 104401 015150          TYPE ,MCRLF
1815 004366 022737 000012 017104  CONT1:  CMP #10,WTRETRY          ;IS WRITE RETRY EQUAL TO 10
1816 004374 001420          BEQ CONT2                ;YES,NO ERRORS OCCURED
1817 004376 012703 017154          MOV #ZWRLOG,R3          ;SET INC.INST.FOR UNIT 0 ERROR LOG
1818 004402 004737 021142          JSR PC,U1LOG            ;TEST FOR UNIT 1 LOG
1819 004406 005213          INC (R3)                ;INC RECOVERABLE WRITE LOG
1820 004410 012737 000012 017104  MOV #10,WTRETRY          ;RESET THE WRITE RETRY COUNTER
1821 004416 104401 015336          TYPE ,DBLLF
1822 004422 104401 015725          TYPE ,MREC              ;TYPE RECOVERABLE WRITE ERROR
1823 004426 104401 016106          TYPE ,MWRITE
1824 004432 104401 015150          TYPE ,MCRLF
1825 004436 004737 005204          CONT2:  JSR PC,DDCHK            ;CHECK FOR DELETED DATA INDICATOR
1826 004442 005701          TST R1                  ;BIT 15 OF R1 IS READ 1 SECTOR FLAG
1827 004444 100001          BPL NEXTRD
1828 004446 000207          RTS PC                  ;IF SET,GO VERIFY DATA JUST READ
1829 004450 005337 012306          NEXTRD:  DEC SECCNTR
1830 004454 001253          BNE XREAD
1831 004456 000207          RTS PC                  ;READ FUNCTION IS DONE
1832
1833 004460 005726          RDERR:  TST (SP)+              ;REMOVE THE DONE RETURN FROM T E STACK
1834 004462 032737 000002 010070  BIT #BIT1,ASTAT          ;IS THIS A PARITY ERROR
1835 004470 001413          BEQ 1$                  ;NO, SEE IF ITS A CRC ERROR
1836          ;PARITY ERROR DURING A READ FUNCTION
1837 004472 012737 015772 003264  MOV #MREAD,PTYP1+2      ;PUT ADDR OF READ MESSAGE IN PAR ERR TYPEOUT 1
1838 004500 012737 015772 003354  MOV #MREAD,PTYP2+2      ;PUT ADDR OF READ MESSAGE IN PAR ER TYPEOUT 2
1839 004506 012737 004246 003334  MOV #REREAD,PCONT+2     ;IF HARD ERR RETURN THROUGH PCONT TO REREAD
1840 004514 000137 003244          JMP PARTEST             ;RECORD PARITY ERROR AND RETRY FUNCTION
1841 004520 032737 000001 010070  1$:  BIT #BIT0,ASTAT          ;IS THIS A CRC ERROR
1842 004526 001014          BNE CRCER              ;YES GO TEST AND LOG IT
1843          ;SEEK ERROR DURING A READ FUNCTION
1844 004530 012737 004246 003674  MOV #REREAD,SEKRTY+2    ;SET SEEK CONTINUE FOR READ RETRY
1845 004536 012737 015772 003630  MOV #MREAD,STYP1+2      ;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 1
1846 004544 012737 015772 003722  MOV #MREAD,STYP2+2      ;SET ADDR OF READ MESSAGE IN SEEK ER TYPEOUT 2
1847 004552 004737 003576          JSR PC,SEEKER           ;RECORD SEEK ERROR
1848 004556 000734          BR NEXTRD              ;GO TO NEXT SECTOR,CAN'T READ THIS ONE
1849          ;CRC ERROR DETECTED WHILE READING
1850 004560 004737 003770          CRCER:  JSR PC,TRKERR           ;INC ERROR PER TRACK COUNTER
1851 004564 005701          TST R1                  ;IF READ ONLY, REPORT DATA CRC ERROR
1852 004566 100061          BPL DATACRC
1853 004570 005237 004126          INC DATAK              ;SET DATA CHECK FLAG
1854 004574 004737 005536          JSR PC,EMPBLIFF         ;CHECK FOR A DATA ERROR
1855 004600 005737 006442          TST ERCNTR             ;WAS THERE A DATA ERROR
1856 004604 001052          BNE DATACRC            ;YES,REREAD AND/OR REWRITE THE DATA
1857 004606 012703 017144          MOV #ZCRCBAD,R3        ;SET INC.INST.FOR UNIT 0 ERROR LOG
1858 004612 004737 021142          JSR PC,U1LOG            ;TEST FOR UNIT 1 ERROR LOGS
1859 004616 005213          INC (R3)                ;NO,INC BAD CRC GENERATOR ERROR
1860 004620 104406          CKSWR
1861 004622 032777 020000 174364  BIT #SW13,@SWR          ;TEST DON'T SWITCH
1862 004630 001004          BNE 2$
1863 004632 104401 015742          TYPE ,MBADCRC           ;TYPE CRC GENERATOR ERROR
1864 004636 104401 015150          TYPE ,MCRLF
1865 004642 104406          2$:  CKSWR
1866 004644 005777 174344          TST @SWR                ;TEST HALT ON ERROR SWITCH
1867 004650 100001          BPL CONT15
1868 004652 000000          HLT10:  HALT                  ;HALT ON ERROR

```

1869	004654	032777	004000	174332	CONT15:	BIT #SW11,@SWR	:CHECK NO RETRY SWITCH
1870	004662	001005				BNE 3\$:IF SET LOG HARD ERROR
1871	004664	005337	017116			DEC CRETRY	:HAVE 10 ERRORS BEEN LOGED
1872	004670	001402				BEQ 3\$:YES,LOG HARD ERROR
1873	004672	000137	004246			JMP REREAD	:NO,GO REREAD DATA
1874	004676	012703	017204		3\$:	MOV #ZHCRCBAD,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1875	004702	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1876	004706	005213				INC (R3)	
1877	004710	104401	015336			TYPE ,DBLLF	
1878	004714	104401	015742			TYPE ,MBADCRC	:TYPE HARD CRC GENERATOR ERROR
1879	004720	104401	015150			TYPE ,MCRLF	
1880	004724	104401	016022			TYPE ,MHALT11	:AND HALT AS THE CRC GENERATOR DOESN'T WORK
1881	004730	000000			HLT11:	HALT	
1882						:DATA CRC ERROR REREAD AND/OR REWRITE TO GET GOOD DATA	
1883	004732	012703	017140		DATAARC:	MOV #ZCPCLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1884	004736	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1885	004742	005213				INC (R3)	:TRUE DATA CRC ERROR
1886	004744	104406				CKSWR	
1887	004746	032777	020000	174240		BIT #SW13,@SWR	:TEST DON'T PRINT ERROR SWITCH
1888	004754	001004				BNE 4\$	
1889	004756	104401	016034			TYPE ,MCRC	:TYPE DATA CRC ERROR
1890	004762	104401	015150			TYPE ,MCRLF	
1891	004766	104406			4\$:	CKSWR	
1892	004770	005777	174220			TST @SWR	:TEST HALT ON ERROR SWITCH
1893	004774	100001				BPL CONT16	
1894	004776	000000			HLT12:	HALT	:HALT ON ERROR
1895	005000	032777	004000	174206	CONT16:	BIT #SW11,@SWR	:TEST NO RETRY SWITCH
1896	005006	001005				BNE 5\$:IF SET LOG HARD ERROR
1897	005010	005337	017102			DEC RDRETRY	:HAVE 10 ERRORS OCCURED
1898	005014	001402				BEQ 5\$:YES LOG HARD ERROR OR REWRITE DATA
1899	005016	000137	004246			JMP REREAD	:NO,GO REREAD THIS SECTOR
1900	005022	012703	017200		5\$:	MOV #ZHCRCLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1901	005026	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1902	005032	005213				INC (R3)	:INC HARD CRC ERROR LOG
1903	005034	012737	000012	017102		MOV #10.,RDRETRY	:RESET READ RETRY COUNTER
1904	005042	005737	004122			TST RDAFTWT	:IS THIS A READ AFTER WRITE FUNCTION
1905	005046	001021				BNE CONT6	:YES,GO REWRITE IT
1906	005050	012703	017210			MOV #ZHRDLOG,R3	:NO,SET INC.INST.FOR UNIT 0 ERROR LOG
1907	005054	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1908	005060	005213				INC (R3)	:READ ONLY, HARD READ ERROR
1909	005062	104401	015336			TYPE ,DBLLF	
1910	005066	104401	016053			TYPE ,MUNREC	:TYPE UNRECOVERABLE READ ERROR
1911	005072	104401	015772			TYPE ,MREAD	
1912	005076	104401	015150		CGNT7:	TYPE ,MCRLF	
1913	005102	062706	000002			ADD #2,SP	:REMOVE READ DONE ADDRESS FROM STACK
1914	005106	000137	004450			JMP NEXTRD	:READ NEXT SECTOR CAN'T READ THIS ONE
1915	005112	104406			CONT6:	CKSWR	
1916	005114	032777	004000	174072		BIT #SW11,@SWR	:TEST NO RETRY SWITCH
1917	005122	001011				BNE 7\$:IF SET LOG HARD ERROR
1918	005124	005337	017104			DEC WRETRY	:HAVE 10 REWRITES BEEN DONE
1919	005130	001406				BEQ 7\$:YES,LOG HARD WRITE ERROR
1920	005132	005137	004124			COM ERWRT	:NO,SET THE WRITE CAUSED BY ERROR FLAG
1921	005136	062706	000002			ADD#2,SP	:REMOVE RDDONE ADDRESS FROM STACK
1922	005142	000137	003130			JMP ONEWRT	:REWRITE SECTOR THEN REREAD TO CHECK FOR ERROR
1923	005146	012703	017214		7\$:	MOV #ZHWRLOG,R3	:SET INC.INST.FOR UNIT 0 ERROR LOG
1924	005152	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS

1925	005156	005213				INC (R3)	:HARD WRITE ERROR
1926	005160	012737	000012	017104		MOV #10.,WRETRY	:RESET WRITE RETRY COUNTER
1927	005166	104401	015336			TYPE ,DBLLF	
1928	005172	104401	016053			TYPE ,MUNREC	:TYPE UNRECOVERABLE WRITE ERROR
1929	005176	104401	016106			TYPE ,MWRITE	
1930	005202	000735				BR CONT7	
1931							
1932							
1933							
1934	005204	032737	001000	001212	DDCHK:	BIT #BIT9,DTESTP	:TEST BIT 9 AS TO DELETED DATA TRANSFER
1935	005212	001472				BEQ CONT10	
1936	005214	132737	000100	010070		BITB #BIT6,ASTAT	:THIS IS A DELETED DATA FUNCTION
1937	005222	001114				BNE RETURN	:DD BIT SHOULD BE SET
1938	005224	012703	017164			MOV #ZDDMIS,R3	:SET INC INST FOR UNIT 0 ERROR LOG
1939	005230	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1940	005234	005213				INC (R3)	:INC MISSING DELETED DATA LOG
1941	005236	004737	003770			JSR PC,TRKERR	:INC ERROR PER TRACK COUNTER
1942	005242	104406				CKSWR	
1943	005244	032777	020000	173742		BIT #SW13,@SWR	:TEST DON'T PRINT ERROR SWITCH
1944	005252	001011				BNE CONT11	
1945	005254	104401	014604			TYPE ,MDDMIS	:TYPE MISSING DELETED DATA BIT
1946	005260	052737	000400	010704	DDERR:	BIS #BIT8,UNITSEL	:SET HAD ERROR FLAG
1947	005266	004737	007716			JSR PC,TYPADR	:TYPE ADDRESS OF ERROR
1948	005272	104401	015150			TYPE ,MCRLF	
1949	005276	104406			CONT11:	CKSWR	
1950	005300	005777	173710			TST @SWR	:TEST HALT ON ERROR SWITCH
1951	005304	100001				BPL CONT17	
1952	005306	000000			HLT13:	HALT	:HALT ON DELETED DATA ERROR
1953	005310	032777	004000	173676	CONT17:	BIT #SW11,@SWR	:TEST NO RETRY SWITCH
1954	005316	001005				BNE 4\$:IF SET LOG HARD ERROR
1955	005320	005337	017106			DEC DDRETRY	:HAVE 10 ERRORS BEEN LOGED
1956	005324	001402				BEQ 4\$:YES LOG HARD ERROR
1957	005326	000137	004246			JMP REREAD	:NO, REREAD SECTOR
1958	005332	012703	017224		4\$:	MOV #ZHDDLOG,R3	:SET INC INST.FOR UNIT 0 ERROR LOG
1959	005336	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1960	005342	005213				INC (R3)	
1961	005344	104401	015336			TYPE ,DBLLF	
1962	005350	104401	016053			TYPE ,MUNREC	:TYPE UNRECOVERABLE DELETED DATA ERROR
1963	005354	104401	015713			TYPE ,MDDER	
1964	005360	104401	015150			TYPE ,MCRLF	
1965	005364	004737	007716			JSR PC,TYPADR	
1966	005370	104401	015150			TYPE ,MCRLF	
1967	005374	000137	004450			JMP NEXTRD	:READ NEXT SECTOR
1968	005400	032737	000100	010070	CONT10:	BIT #BIT6,ASTAT	:THIS IS NOT A DELETED DATA TRANSFER
1969	005406	001422				BEQ RETURN	
1970	005410	012703	017170			MOV #ZUNXDD,R3	:SET INC INST.FOR UNIT 0 ERROR LOG
1971	005414	004737	021142			JSR PC,U1LOG	:TEST FOR UNIT 1 ERROR LOGS
1972	005420	005213				INC (R3)	:UNEXPECTED DD BIT SET
1973	005422	052737	000400	010704		BIS #BIT8,UNITSEL	:SET HAD ERROR FLAG
1974	005430	004737	003770			JSR PC,TRKERR	:INC ERROR PER TRACK COUNTER
1975	005434	104406				CKSWR	
1976	005436	032777	020000	173550		BIT #SW13,@SWR	:TEST DON'T PRINT ERROR SWITCH
1977	005444	001314				BNE CONT11	
1978	005446	104401	014560			TYPE ,MUNXDD	:TYPE UNEXPECTED DELETED DATA BIT
1979	005452	000702				BR DDERR	
1980	005454	000207			RETURN:	RTS PC	

1981
1982
1983
1984
1985
1986
1987
1988
1989 005456 005137 004122
1990 005462 004737 010312
1991 005466 004737 011144
1992 005472 004737 010614
1993 005476 004737 011254
1994 005502 004737 003112
1995 005506 004737 005526
1996 005512 005337 011356
1997 005516 001367
1998 005520 004737 010732
1999 005524 000760
2000
2001
2002
2003
2004
2005
2006
2007 005526 052701 100000
2008 005532 004737 004200
2009 005536 005737 012306
2010 005542 001002
2011 005544 000137 006432
2012 005550 005037 006434
2013 005554 005037 006442
2014 005560 052701 000200
2015 005564 004737 005656
2016 005570 005037 005746
2017 005574 012746 006212
2018 005600 012746 005750
2019 005604 005046
2020 005606 012746 005614
2021 005612 000002
2022 005614 012777 000103 173364 1\$:
2023 005622 004737 007014 EMPFLAG:
2024
2025 005626 117737 173356 006436 CKBYTE:
2026 005634 063737 006436 005746
2027 005642 123720 006436
2028 005646 001054
2029 005650 005237 006434
2030 005654 000762
2031
2032 005656 113737 011360 016702 ADJSUM:
2033 005664 113737 012310 016703
2034 005672 013737 010612 005746
2035 005700 063737 011360 005746
2036 005706 063737 012310 005746

::*****

;WRITE ALL SECTORS,READ AND VERIFY ALL SECTORS
; T = 3

WTRDCK: COM RDAFTWT ;SET READ AFTER WRITE FLAG
JSR PC,GETPATTERN
XWTRDCK: JSR PC,INITTRACKS
JSR PC,GETUNIT
1\$: JSR PC,GETTRACK
JSR PC,WRITE
JSR PC,READCHK
DEC TRKCNT
BNE 1\$
JSR PC,STOP
BR XWTRDCK

::*****

;READ A SECTOR,EMPTY THE SECTOR BUFFER AND VERIFY
;THE DATA READ AGAINST CORE DATA BUFFER

READCHK: BIS #BIT15,R1 ;SET READ ONE SECTOR FLAG
JSR PC,READ ;GO READ ONE SECTOR
EMPBUFF: TST SECCNTR ;IF CLEARED NO SECTOR WAS FOUND
BNE 2\$
JMP EXIT ;GO TO NEXT TRACK CAN'T READ THIS ONE
2\$: CLR BYTECNTR ;CLEAR THE BYTE AND ERROR COUNTERS
CLR ERCNTR
BIS #BIT7,R1 ;R1 BIT 7 IS USED AS FIRST ERROR FLAG
JSR PC,ADJSUM ;ADJUST DATA AND CK SUM FOR ADDRESSES
CLR CKSUM ;SET UP FOR CHECK SUM ACCUMULATION
MOV #EMPDONE,-(SP) ;SET UP RETURN ADDRESSES
MOV #EMPER,-(SP)
CLR -(SP) ;LOWER INTERRUPT LEVEL
MOV #1\$,-(SP)
RTI
1\$: MOV #EBIE,@RXCS ;LOAD EMPTY BUFFER FUNCTION
EMPFLAG: JSR PC,TRCK ;TEST FOR TR FLAG
CKBYTE: MOVB @RXDB,BADBYTE ;SAVE BYTE FROM DISKETTE
ADD BADBYTE,CKSUM ;ACCUMULATE CHECK SUM
CMPB BADBYTE,(R0)+ ;COMPARE AGAINST GOOD BYTE
BNE DATAER ;IF NOT EQUAL GO TO DATAER
INC BYTECNTR
BR EMPFLAG ;GET NEXT BYTE
ADJSUM: MOVB TARGET,BUFADR ;SET FIRST AND SECOND BYTES WITH ADDRESSES
MOVB TSECTOR,BUFADR+1
MOV SUM,CKSUM ;GET THE PATTERN SUM
ADD TARGET,CKSUM ;ADD TRACK ADDRESS TO CHECK SUM
ADD TSECTOR,CKSUM ;ADD SECTOR ADDRESS TO CHECK SUM

2037	005714	113737	005746	017100		MOVB CKSUM, BUFADR+176	:INSERT CHECK SUM TO DATA BUFFER
2038	005722	106337	005746			ASLB CKSUM	:GENERATE NEGITIVE CHECK SUM
2039	005726	105437	005746			NEGB CKSUM	
2040	005732	113737	005746	017101		MOVB CKSUM, BUFADR+177	:INSERT NEG SUM INTO DATA BUFFER
2041	005740	012700	016702			MOV #BUFADR, R0	:SET ADDRESS OF BYTE IN R0
2042	005744	000207				RTS PC	:RETURN
2043							
2044	005746	000000			CKSUM:	0	
2045							
2046	005750	005726			EMPER:	TST (SP)+	:REMOVE THE DONE RETURN FROM THE STACK
2047	005752	012737	015500	003264		MOV #EMPTY, P1+2	:PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYPOUT 1
2048	005760	012737	015500	003354		MOV #EMPTY, P2+2	:PUT ADDR OF EMPTYBUF MESSAGE IN PAR ER TYPOUT 2
2049	005766	012737	005536	003334		MOV #EMPBUF, PCONT+2	:NOT HARD ERR RETURN THROUGH PCONT TO EMPTYBUF
2050	005774	000137	003244			JMP PARTEST	:REPORT PARITY ERROR
2051							
2052	006000	052737	000400	010704	DATAER:	BIS #BIT8, UNITSEL	:SET THE HAD ERROR FLAG
2053	006006	005237	006442			INC ERCNTR	:INC THE BYTE ERROR COUNTER
2054	006012	104406				CKSWR	
2055	006014	032777	020000	173172		BIT #SW13, @SWR	:TEST PRINT ERROR SW IN SWR
2056	006022	001062				BNE NOERTYP	:DON'T PRINT THE ERROR
2057	006024	032777	010000	173162		BIT #SW12, @SWR	:TEST PRINT 10 ERRORS SWITCH
2058	006032	001404				BEQ 1\$:PRINT ALL ERRORS
2059	006034	023727	006442	000012		CMP ERCNTR, #10.	:HAVE 10 ERRORS BEEN TYPED
2060	006042	003052				BGT NOERTYP	:YES, DON'T PRINT ANY MORE
2061	006044	005737	004126		1\$:	TST DATAK	:WAS THIS A READ CHECK DUE TO CRC ERROR
2062	006050	001403				BEQ 2\$:NO, REPORT DATA ERRORS
2063	006052	105701				TSTB R1	:YES, TEST FIRST ERROR FLAG
2064	006054	100017				BPL TYPERR	:TYPE THE DATA CRC ERROR
2065	006056	000412				BR 3\$	
2066	006060	105701			2\$:	TSTB R1	:TEST FIRST ERROR FLAG
2067	006062	100014				BPL TYPERR	
2068	006064	104401	015150			TYPE ,MCRLF	
2069	006070	104401	014625			TYPE ,MDERHDR	:PRINT ERROR HEADER
2070	006074	104401	015150			TYPE ,MCRLF	
2071	006100	004737	007716			JSR PC, TYPADR	:PRINT TRACK AND SECTOR LOCATIONS
2072	006104	104401	014714		3\$:	TYPE ,MCLMUN	:SET UP COLMUN HEADINGS
2073	006110	042701	000200			BIC #BIT7, R1	:CLEAR FIRST ERROR FLAG
2074	006114	013737	006434	006126	TYPERR:	MOV BYTECNTR, 1\$:PRINT BYTE NUMBER
2075	006122	004537	014544			JSR R5, SGLDEC	
2076	006126	000000			1\$:	OPEN	
2077	006130	104401	015145			TYPE ,DBLSP	
2078	006134	013746	006436			MOV BADBYTE, -(SP)	:PRINT BYTE READ FROM DISKETTE
2079	006140	104403				TYPOS	
2080	006142	000003				.WORD 3	
2081	006144	104401	015145			TYPE ,DBLSP	
2082	006150	114037	006440			MOVB -(R0), GOODBYTE	:GET GOOD BYTE
2083	006154	005200				INC R0	:RETURN R0 TO NEXT BYTE IN BUFFER
2084	006156	013746	006440			MOV GOODBYTE, -(SP)	
2085	006162	104404				TYPON	:PRINT GOOD DATA
2086	006164	104401	015150			TYPE ,MCRLF	
2087	006170	104406			NOERTYP:	CKSWR	
2088	006172	005777	173016			TST @SWR	:TEST HALT ON ERROR SWITCH
2089	006176	100001				BPL CONT20	
2090	006200	000000			HLT14:	HALT	
2091	006202	005237	006434		CONT20:	INC BYTECNTR	
2092	006206	000137	005622			JMP EMPFLAG	


```
2093
2094 006212 005737 004126 EMPDONE: TST DATAK ;WAS THIS READCHECK CAUSED BY CRC ERROR
2095 006216 001401 BEQ 1$ ;NO,CONTINUE
2096 006220 000207 RTS PC ;YES,RETURN TO CRC ERROR HANDLER
2097 006222 005737 006442 1$: TST ERCNTR ;WAS THERE ERRORS
2098 006226 001471 BEQ CONT3 ;NO ERRORS
2099 006230 104406 CKSWR
2100 006232 032777 020000 172754 BIT #SW13,@SWR ;YES,TEST DON'T PRINT SWITCH
2101 006240 001024 BNE 2$ ;DON'T PRINT THE ERROR
2102 006242 104401 015431 TYPE ,MERCT ;PRINT THE TOTAL DATA ERROR COUNT
2103 006246 013737 006442 006260 MOV ERCNTR,3$
2104 006254 004537 014544 JSR R5,SGLDEC
2105 006260 000000 3$: OPEN
2106 006262 104401 016425 TYPE ,MSUM ;INDICATE IF CHECK SUM WAS GOOD OR HAD ERRORS
2107 006266 105737 005746 TSTB CKSUM
2108 006272 001403 BEQ 4$
2109 006274 104401 016165 TYPE ,MERS
2110 006300 000402 BR 5$
2111 006302 104401 016420 4$: TYPE ,MGOOD
2112 006306 104401 015150 5$: TYPE ,MCRLF
2113 006312 012703 017160 2$: MOV #ZDATALOG,R3 ;SET INC.INST.FOR UNIT 0 ERROR LOG
2114 006316 004737 021142 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
2115 006322 005213 INC (R3) ;INC DATA ERROR LOG
2116 006324 004737 003770 JSR PC,TRKERR ;INC ERROR PER TRACK COUNTER
2117 006330 104406 CKSWR
2118 006332 032777 004000 172654 BIT #SW11,@SWR ;TEST NO RETRY SWITCH
2119 006340 001007 BNE 6$ ;IF SET LOG HARD ERROR
2120 006342 005337 017110 DEC DATARETRY ;HAVE 10 ERRORS BEEN LOGED
2121 006346 001404 BEQ 6$
2122 006350 004737 004246 JSR PC,REREAD ;NO,GO REREAD THE DATA
2123 006354 000137 005536 JMP EMPBUFF ;GO RECHECK THE DATA
2124 006360 012703 017220 6$: MOV #ZHATALOG,R3 ;YES,SET INC.INST.FOR UNIT 0 ERROR LOG
2125 006364 004737 021142 JSR PC,U1LOG ;TEST FOR UNIT 1 ERROR LOGS
2126 006370 005213 INC (R3) ;INC HARD DATA NO STATUS ERROR
2127 006372 104401 015336 TYPE ,DBLLF
2128 006376 104401 016053 TYPE ,MUNREC ;TYPE UNRECOVERABLE DATA NO
2129 006402 104401 014625 TYPE ,MDERHDR ;STATUS ERROR
2130 006406 104401 015150 TYPE ,MCRLF
2131 006412 005337 012306 CONT3: DEC SECCNTR
2132 006416 001405 BEQ EXIT
2133 006420 004737 004204 JSR PC,XREAD ;READ THE NEXT SECTOR
2134 006424 000137 005536 JMP EMPBUFF
2135 006430 005001 3$: CLR R1 ;CLEAR THE ONE READ FLAG
2136 006432 000207 EXIT: RTS PC
2137
2138 006434 000000 BYTECNTR: 0
2139 006436 000000 BADBYTE: 0
2140 006440 000000 GOODBYTE: 0
2141 006442 000000 ERCNTR: 0
2142
2143 ;:*****
2144
2145 ;READ AND VERIFY ALL SECTORS WRITTEN BY
2146 ;PREVIOUS WRITE FUNCTION
2147
2148 ; T = 4
```

2149
2150 006444 004737 010312
2151 006450 004737 011144
2152 006454 004737 010614
2153 006460 004737 011254
2154 006464 004737 005526
2155 006470 005337 011356
2156 006474 001371
2157 006476 004737 010732
2158 006502 000762

RDCHK: JSR PC,GETPATTERN
XRDCHK: JSR PC,INITTRACKS
JSR PC,GETUNIT
1\$: JSR PC,GETTRACK
JSR PC,READCHK
DEC TRKNTR
BNE 1\$
JSR PC,STOP
BR XRDCHK

::*****

:DO A READ ONLY FUNCTION ON ALL SECTORS
:THIS DOES NOT VERIFY THE DATA, ONLY TESTS FOR CRC ERRORS
: T = 5

2166
2167 006504 004737 010312
2168 006510 004737 011144
2169 006514 004737 010614
2170 006520 004737 011254
2171 006524 004737 004200
2172 006530 005337 011356
2173 006534 001371
2174 006536 004737 010732
2175 006542 000762

RONLY: JSR PC,GETPATTERN
XRONLY: JSR PC,INITTRACKS
JSR PC,GETUNIT
1\$: JSR PC,GETTRACK
JSR PC,READ
DEC TRKNTR
BNE 1\$
JSR PC,STOP
BR XRONLY

::*****

:WRITE AND READ CHECK ANY DATA PATTERN,AND ANY TRACK SEQUENCE
:BUT ALTERNATE BETWEEN DRIVE UNITS ON EACH TRACK SELECTED.
: T = 6

2184
2185 006544 005137 004122
2186 006550 004737 010312
2187 006554 004737 011144
2188 006560 004737 010614
2189 006564 004737 011254
2190 006570 004737 003112
2191 006574 004737 005526
2192 006600 004737 010614
2193 006604 004737 003112
2194 006610 004737 005526
2195 006614 005337 011356
2196 006620 001357
2197 006622 004737 010732
2198 006626 000752

DRVSWP: COM RDAFTWT ;SET THE READ AFTER WRITE FLAG
2\$: JSR PC,GETPATTERN
1\$: JSR PC,INITTRACKS
JSR PC,GETUNIT ;SET UP FOR UNIT 0
JSR PC,GETTRACK ;GET THE TRACK TO BE ACCESSED
JSR PC,WRITE
JSR PC,READCHK
JSR PC,GETUNIT ;NOW GO TO UNIT 1 ON THE SAME TRACK
JSR PC,WRITE
JSR PC,READCHK
DEC TRKNTR
BNE 1\$
JSR PC,STOP
BR 2\$

::*****

:THE FIRST HALF OF TEST 7 FORCES THE TRACK SEQUENCE
:TO INCRIMENT UP THROUGH ALL THE TRACKS DOING WRITE / READ CHECK FUNCTIONS.
:THIS VERIFIES THAT ALL THE TRACKS ARE ACCESSABLE.

2199
2200
2201
2202
2203
2204

```

2205 ;THE SECOND HALF OF THE PASS WILL USE THE SEQUENCE SELECTED BY THE
2206 ;OPERATOR AS INDICATED BELOW,AND ONLY READ CHECK THE DATA JUST WRITTEN.
2207 ;THIS VERIFIES THAT THE DATA CAN BE READ FROM A TRACK,AFTER THE HEAD HAS BEEN
2208 ;MOVED AWAY FROM AND BACK TO THAT TRACK. AT THE COMPLETION OF THE PASS
2209 ;THE DELETED DATA BIT IN TEST CONDITIONS IS COMPLEMENTED AND THE NEXT
2210 ;PASS WILL BE RUN UNDER THIS NEW CONDITION.
2211 ; T = 0 OR 7
2212
2213 006630 004737 010312 TEST7: JSR PC,GETPATTERN
2214 006634 013737 011370 007012 MOV SEQUEN,SEQSAV ;SAVE THE SELECTED SEQUENCE
2215 006642 012737 000001 011370 TEST7X: MOV #1,SEQUEN ;FORCE INCREMENT SEQUENCE
2216 006650 005137 004122 COM RDAFTWT ;SET READ AFTER WRITE FLAG FOR ERROR TESTING
2217 006654 004737 011144 JSR PC,INITTRACKS
2218 006660 004737 010614 JSR PC,GETUNIT
2219 006664 004737 011254 1$: JSR PC,GETTRACK
2220 006670 004737 003112 JSR PC,WRITE
2221 006674 004737 005526 JSR PC,READCHK
2222 006700 005337 011356 DEC TRKCNT ;IS THE FIRST HALF OF THE PASS DONE
2223 006704 001367 BNE 1$ ;NO,CONTINUE WRITING AND READING
2224 006706 005037 004122 CLR RDAFTWT ;YES,NOW DO A READ ONLY
2225 006712 013737 007012 011370 MOV SEQUEN,SEQUEN ;USING SELECTED SEQUENCE
2226 006720 004737 011144 JSR PC,INITTRACKS
2227 006724 004737 011254 2$: JSR PC,GETTRACK
2228 006730 004737 005526 JSR PC,READCHK
2229 006734 005337 011356 DEC TRKCNT ;IS THIS HALF OF THE PASS DONE
2230 006740 001371 BNE 2$ ;NO,CONTINUE READ CHECKING
2231 006742 004737 010732 JSR PC,STOP ;YES,TEST FOR END OF PASS CONDITIONS
2232 006746 032737 040100 010704 BIT #40100,UNITSEL ;HAVE ALL SELECTED UNITS BEEN USED
2233 006754 001332 BNE TEST7X ;IF A USED BIT IS STILL SET GO TO NEXT DRIVE
2234 006756 012706 001200 MOV #1200,SP ;RESET STACK ADDRESS FOR ACT11 EOP
2235 006762 032737 001000 001212 BIT #BIT9,DTESTP ;NEXT PASS,COMPLEMENT D D BIT
2236 006770 001404 BEQ 3$
2237 006772 042737 001000 001212 BIC #BIT9,DTESTP ;IT WAS ON CLEAR IT
2238 007000 000720 BR TEST7X
2239 007002 052737 001000 001212 3$: BIS #BIT9,DTESTP ;IT WAS OFF SET IT
2240 007010 000714 BR TEST7X
2241
2242 007012 000000 SEQSAV: 0
2243 ;:*****
2244
2245 ;TEST FOR TRANSFER FLAG AND PROGRAM NOT HUNG
2246
2247 007014 005037 007200 TRCK: CLR HCNTR1 ;CLEAR HUNG COUNTER
2248 007020 012746 000340 MOV #PR7,-(SP) ;RAISE INTERUPT LEVEL TO INHIBIT INTERUPTS
2249 007024 012746 007032 MOV #3$,-(SP)
2250 007030 000002 RTI
2251 007032 105777 172150 3$: TSTB @RXCS ;TEST FOR TR FLAG
2252 007036 100001 BPL 1$ ;SKIP NEXT INST.IF NOT SET
2253 007040 000207 RTS PC ;RETURN TO TRANSFER DATA
2254 007042 023727 006434 000200 1$: CMP BYTECNTR,#128. ;HAVE 128 BYTES BEEN TRANSFERED
2255 007050 001403 BEQ 2$ ;ALLOW AN INTERUPT IF EQUAL
2256 007052 005237 007200 INC HCNTR1 ;IF THE HUNG COUNTER OVERFLOWES EITHER
2257 ;THERE WAS NO TR FLAG OR NO INTERUPT OCCURED
2258 ;AFTER 128 BYTES TRANSFERED
2259 007056 001365 BNE 3$ ;NO OVERFLOW RETEST FOR TR FLAG
2260 007060 132777 000040 172120 2$: BITB #DONEBIT,@RXCS ;TEST FOR A DONE FLAG WHICH WOULD INDICATE A

```


2317	007264	032737	000004	010070		BIT #BIT2,ASTAT	:IS INIT DONE SET
2318	007272	001402				BEQ 2\$:NO,CONTINUE
2319	007274	000137	007702			JMP RXPWR	:YES,REPORT POWER FAILED AND RESTART
2320	007300	032737	000003	010070	2\$:	BIT #3,ASTAT	:ARE PAR OR CRC BITS SET
2321	007306	001023				BNE 1\$:YES GO LOG ERROR
2322	007310	132777	000040	171670		BITB #DONEBIT,@RXCS	:IS DONE SET
2323	007316	001014				BNE 3\$:IF SET RETURN TO TEST
2324	007320	005237	017130			INC UKNINT	:INC UNKNOWN INTERUPT ERROR LOG
2325	007324	104406				CKSWR	
2326	007326	032777	020000	171660		BIT #SW13,@SWR	:TEST DON'T PRINT ERROR SWITCH
2327	007334	001004				BNE 4\$:DON'T PRINT
2328	007336	104401	015410			TYPE ,MUKNINT	:TYPE UNKNOWN INTERUPT
2329	007342	104401	015150			TYPE ,MCRLF	
2330	007346	000002			4\$:	RTI	:RETURN FROM THE INTERUPT
2331	007350	062706	000006		3\$:	ADD #6,SP	:BYPASS INTERUPT POINTERS ON STACK
2332	007354	000207				RTS PC	:RETURN TO PROGRAM
2333	007356	005237	017126		1\$:	INC NOERLOG	:NO STATUS ERROR FLAG ERROR
2334	007362	104406				CKSWR	
2335	007364	032777	020000	171622		BIT #SW13,@SWR	:TEST DON'T PRINT ERROR SWITCH
2336	007372	001004				BNE RXERROR	
2337	007374	104401	016132			TYPE ,MNOFLAG	:TYPE NO STATUS ERROR ERROR
2338	007400	104401	015150			TYPE ,MCRLF	
2339							
2340	007404	052737	000400	010704	RXERROR:	BIS #BIT8,UNITSEL	:SET HAD ERROR FLAG
2341	007412	012737	000012	017112		MOV #10.,P2RETRY	:SET RETRY COUNTER
2342	007420	012777	000017	171560	2\$:	MOV #RDR,@RXCS	:GET THE ERROR CODE
2343	007426	004737	007122			JSR PC,DONECK	:TEST FOR DONE FLAG
2344	007432	032777	000002	171550		BIT #2,@RXDB	:WAS THERE A PARITY ERROR
2345	007440	001403				BEQ 1\$:NO,CONTINUE
2346	007442	004737	002046			JSR PC,STATER	:YES,GO REPORT THE PARITY ERROR
2347	007446	000764				BR 2\$:REISSUE THE FUNCTION
2348	007450	117737	171534	010072	1\$:	MOVB @RXDB,BSTAT	:SAVE THE ERROR CODE IN B STATUS
2349	007456	005737	010072			TST BSTAT	:IS THERE A DEFINITE CODE
2350	007462	001407				BEQ NOPRNT	:NO,CONTINUE
2351	007464	013705	010072			MOV BSTAT,R5	:ADJUST ERROR CODE TO PRODUCE AN EVEN ADDR
2352	007470	006005				ROR R5	:OF THE CORRESPONDING COUNTER
2353	007472	006005				ROR R5	
2354	007474	062705	017226			ADD #ERCODE-2,R5	:ADD ON ADDR OF ERROR LOG -2,AS THERE IS NO 0
2355							:ERROR CODE. THE CONTENTS OF R5 WILL READJUST
2356							:ADDRESS TO CORRECT LOG.
2357	007500	005215				INC (R5)	:INC THE CORRECT ERROR CODE COUNTER
2358	007502	104406				CKSWR	
2359	007504	032777	020000	171502	NOPRNT:	BIT #SW13,@SWR	:TEST PRINT ERROR SWITCH IN SWR
2360	007512	001032				BNE 2\$	
2361	007514	104401	015150			TYPE ,MCRLF	
2362	007520	104401	015161			TYPE ,MERHEADER	:TYPE ERROR HEADER
2363	007524	104401	016240			TYPE ,MPASS	:TYPE PASSES COMPLETED AT ERROR
2364	007530	013737	021140	007542		MOV PASCNTR,1\$	
2365	007536	004537	014544			JSR R5,SGLDEC	
2366	007542	000000			1\$:	OPEN	
2367	007544	104401	015150			TYPE ,MCRLF	
2368	007550	104401	015265			TYPE ,MRXCS	:TYPE COMMAND STATUS REGISTER
2369	007554	013746	004120			MOV FUNCTION,-(SP)	::SAVE FUNCTION FOR TYPEOUT
2370	007560	104403				TYPOS	::GO TYPE--OCTAL ASCII
2371	007562	003				.BYTE 3	::TYPE 3 DIGIT(S)
2372	007563	000				.BYTE 0	::SUPPRESS LEADING ZEROS

2373	007564	004737	007716		JSR PC,TYPADR	;TYPE ADDRESSES AND RUN CONDITIONS
2374	007570	104401	015150		TYPE ,MCRLF	
2375	007574	004737	010250		JSR PC,TYPECODE	;PRINT THE STATUS REGISTERS
2376	007600	032737	000020	004120	2\$: BIT #BIT4,FUNCTION	;WHAT DRIVE IS BEING USED
2377	007606	001006			BNE 6\$	
2378	007610	012777	000013	171370	MOV #RD0STAT,@RXCS	;DRIVE 0 BEING USED
2379	007616	004737	007122		5\$: JSR PC,DONECK	;TEST FOR DONE FLAG
2380	007622	004040			BR 7\$	
2381	007624	012777	000033	171354	6\$: MOV #RD1STAT,@RXCS	;DRIVE 1 BEING USED
2382	007632	000771			BR 5\$	
2383	007634	032777	000002	171346	7\$: BIT #2,@RXDB	;WAS THERE A PARITY ERROR
2384	007642	001403			BEQ 3\$;NO,CONTINUE
2385	007644	004737	002046		JSR PC,STATER	;YES,REPORT THE ERROR
2386	007650	000753			BR 2\$;REISSUE THE COMMAND
2387	007652	105777	171332		3\$: TSTB @RXDB	;WAS DRIVE READY SET
2388	007656	100406			BMI 4\$;YES RETURN
2389	007660	104401	015114		TYPE ,MNODRY	
2390	007664	104401	015150		TYPE ,MCRLF	
2391	007670	004737	010074		JSR PC,DELUNIT	;NO GO DELETE THAT UNIT
2392	007674	062706	000004		4\$: ADD #4,SP	;MOVE ERROR RETURN TO TOP OF STACK
2393	007700	000207			RTS PC	
2394						
2395	007702	104401	016442		RXPWR: TYPE ,MRX11	;ONLY THE RX11 POWER HAS FAILED
2396	007706	104401	014234		TYPE ,\$POWER	;PRINT POWER FAILED
2397	007712	000137	002700		JMP RESTART	;GO TO RESTART
2398						
2399						
2400	007716	104401	014653		TYPADR: TYPE ,MTRK	;TYPE TRACK ADDRESS
2401	007722	013737	011360	007734	MOV TARGET,3\$	
2402	007730	004537	014544		JSR R5,SGLDEC	
2403	007734	000000			3\$: OPEN	
2404	007736	104401	014702		TYPE ,MSECT	;TYPE SECTOR ADDRESS
2405	007742	013737	012310	007762	MOV TSECTOR,2\$	
2406	007750	042737	177740	007762	BIC #177740,2\$;CLEAR ALL BUT SECTOR ADDRESS
2407	007756	004537	014544		JSR R5,SGLDEC	
2408	007762	000000			2\$: OPEN	
2409	007764	104401	015145		TYPE ,DBLSP	
2410	007770	032737	000020	010704	BIT #BIT4,UNITSEL	;WHICH DRIVE IS BEING USED
2411	007776	001003			BNE 1\$	
2412	010000	104401	015204		TYPE ,MUNIT0	;TYPE UNIT 0
2413	010004	000402			BR TYPSEL	
2414	010006	104401	015214		1\$: TYPE ,MUNIT1	;TYPE UNIT 1
2415	010012	104401	015342		TYPSEL: TYPE ,MPAT	;TYPE PATTERN CODE
2416	010016	013746	010420		MOV PAT,-(SP)	::SAVE PAT FOR TYPEOUT
2417	010022	104403			TYPOS	::GO TYPE--OCTAL ASCII
2418	010024	001			.BYTE 1	::TYPE 1 DIGIT(S)
2419	010025	000			.BYTE 0	::SUPPRESS LEADING ZEROS
2420	010026	104401	015145		TYPE ,DBLSP	
2421	010032	104401	015346		TYPE ,MTEST	;TYPE TEST CODE
2422	010036	013746	002140		MOV TEST,-(SP)	::SAVE TEST FOR TYPEOUT
2423	010042	104403			TYPOS	::GO TYPE--OCTAL ASCII
2424	010044	001			.BYTE 1	::TYPE 1 DIGIT(S)
2425	010045	000			.BYTE 0	::SUPPRESS LEADING ZEROS
2426	010046	104401	015145		TYPE ,DBLSP	
2427	010052	104401	015352		TYPE ,MSEQ	;TYPE SEQUENCE CODE
2428	010056	013746	011370		MOV SEQUEN,-(SP)	::SAVE SEQUEN FOR TYPEOUT

```
2429 010062 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
2430 010064      001        .BYTE          1          ;;TYPE 1 DIGIT(S)
2431 010065      000        .BYTE          0          ;;SUPPRESS LEADING ZEROS
2432 010066 000207          RTS PC
2433
2434 010070 000000          ASTAT:          0
2435 010072 000000          BSTAT:          0
2436
2437 010074 132737 000020 004120 DELUNIT:      BITB #BIT4,FUNCTION ;TEST FOR PRESENTLY USED UNIT
2438 010102 001016          BNE 3$
2439 010104 104401 015204          TYPE ,MUNIT0      ;UNIT 0 HAS BEEN DELETED FROM USE
2440 010110 104401 015224          TYPE ,MDELET
2441 010114 104401 015336          TYPE ,DBLLF
2442 010120 042737 000200 010704          BIC #BIT7,UNITSEL ;CLEAR UNIT 0 SELECTION BIT
2443 010126 005737 010704          TST UNITSEL      ;WAS UNIT 1 SELECTED FOR USE
2444 010132 100020          BPL 1$           ;UNIT 1 NOT SELECTED GO DUMP ERROR REPORT
2445 010134 000137 002726          JMP TESTSEL      ;CONTINUE ON OTHER UNIT AT BEGINING OF TEST
2446 010140 104401 015214          2$:              ;UNIT 1 HAS BEEN DELETED FROM USE
2447 010144 104401 015224          3$:              TYPE ,MUNIT1
2448 010150 104401 015336          TYPE ,MDELET
2449 010154 042737 100000 010704          TYPE ,DBLLF
2450 010162 105737 010704          BIC #BIT15,UNITSEL ;CLEAR UNIT 1 SELECTION BIT
2451 010166 100762          TSTB UNITSEL     ;WAS UNIT 0 SELECTED FOR USE
2452 010170 005137 022612          BMI 2$           ;YES CONTINUE ON UNIT 0
2453 010174 000137 021160          COM SHORTRPT    ;SET SHORT REPORT FLAG
2454          JMP ERDUMP      ;CAN'T CONTINUE TYPE OUT STATISTICAL REPORT
2455
2456 010200 117737 171004 010070 RDCODE:      MOVB @RXDB,ASTAT ;SAVE THE A STATUS
2457 010206 012777 000017 170772 2$:          MOV #ORDER,@RXCS ;READ THE B STATUS REGISTER
2458 010214 004737 007122          JSR PC,DONECK    ;WAIT FOR DONE FLAG
2459 010220 032777 000002 170762          BIT #2,@RXDB    ;WAS THERE A PARITY ERROR
2460 010226 001403          BEQ 1$           ;NO,CONTINUE
2461 010230 004737 002046          JSR PC,STATER   ;YES,REPORT THE PARITY ERROR
2462 010234 000764          BR 2$           ;RETRY READING STATUS B
2463 010236 117737 170746 010072 1$:          MOVB @RXDB,BSTAT ;SAVE THE B STATUS CODES
2464 010244 104401 015150          TYPE ,MCRLF
2465 010250 104401 015306          TYP CODE:      TYPE ,MASTAT      ;TYPE THE CONTENTS OF THE TWO STATUS REGISTERS
2466 010254 013746 010070          MOV           ASTAT,-(SP) ;SAVE ASTAT FOR TYPEOUT
2467 010260 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
2468 010262      003        .BYTE          3          ;;TYPE 3 DIGIT(S)
2469 010263      000        .BYTE          0          ;;SUPPRESS LEADING ZEROS
2470 010264 104401 015136          TYPE ,TAB
2471 010270 104401 015322          TYPE ,MBSTAT
2472 010274 013746 010072          MOV           BSTAT,-(SP) ;SAVE BSTAT FOR TYPEOUT
2473 010300 104403          TYPOS          ;;GO TYPE--OCTAL ASCII
2474 010302      003        .BYTE          3          ;;TYPE 3 DIGIT(S)
2475 010303      000        .BYTE          0          ;;SUPPRESS LEADING ZEROS
2476 010304 104401 015150          TYPE ,MCRLF
2477 010310 000207          RTS PC
2478
2479
2480          ;:*****
2481
2482          ;DECODES THE DATA PATERM SELECTED IN THE SWR
2483
2484          ;BITS 6,7,AND 8 OF THE INITIAL SWR SELECTED THE DATA PATERM
```

2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540

: TO BE USED AS FOLLOWS:

BITS			DATA PATTERN
8	7	6	NO PATTERN SPECIFIED (FORCE RANDOM DATA)
0	0	0	ALL ZEROS
0	0	1	ALL ONES
0	1	0	FLOATING ZERO
0	1	1	FLOATING ONE
1	0	0	ALTERNATING BITS
1	0	1	ALTERNATING PAIRS OF BITS
1	1	0	
1	1	1	RANDOM

: NOTE: ALL DATA PATTERNS WILL BE MODIFIED SO THE FIRST BYTE WILL
: CONTAIN THE TRACK ADDRESS. THE SECOND BYTE WILL CONTAIN THE UNIT
: NUMBER AND SECTOR ADDRESS IN WHICH THE DATA IS WRITTEN. THE LAST
: TWO BYTES CONTAIN THE CHECK SUM NUMBERS.

:*****

```

010360 GETPATTERN:      BICB #377,@#BRONPAT      ;CLEAR BRANCH OFFSET
                        CLR SUM                ;SET UP FOR ACCUMULATION OF CHECK SUM
                        TST PAT                ;IF NO PATTERN SPECIFIED FORCE PATTERN 7
                        BNE 1$
010420 1$:              MOV #7,PAT
                        MOV PAT,R4            ;GET PATTERN BITS
                        DEC R4                ;ADJUST FOR CORRECT OFFSET
                        ASL R4
                        BISB R4,@#BRONPAT     ;INSERT OFFSET
                        MOV #BUFADR+2,R4     ;SET UP ADDRESS OF FIRST BYTE
BRONPAT:                BR .                 ;BRANCH BY OFFSET SELECTED
                        JMP DATA0           ;000 DATA BYTE
                        JMP DATA1           ;377 DATA BYTE
                        JMP FLOAT0           ;FLOAT A 0 THROUGH ALL 1'S
                        JMP FLOAT1           ;FLOAT A 1 THROUGH ALL 0'S
                        JMP PAT125          ;125/052 DATA WORD
                        JMP PAT314          ;314/063 DATA WORD
                        JMP RANDATA         ;RANDOM DATA BYTE

```

```

DATA0:      0
PAT:        0

```

:*****

:LOAD SOFTWARE BUFFER WITH ALL ZEROS
: P = 1

```

DATA0:      CLR DATA0
PATGEN:     JSR PC,LOAD      ;GO LOAD THE DATA BUFFER
            BR PATGEN

```

:*****


```
2541 ;LOAD SOFTWARE BUFFER WITH ALL ONES
2542 ; P = 2
2543
2544
2545 010431 112737 000377 010416 DATA1:      MOVB #377,DATABYTE
2546 010442 000771                                BR PATGEN
2547
2548 ;:*****
2549
2550 ;FLOAT A 0 THROUGH ONES IN SOFTWARE BUFFER
2551 ; P = 3
2552
2553
2554 010444 112737 000376 010416 FLOAT0:      MOVB #376,DATABYTE      ;SET UP A ONES FIELD
2555 010452 000261 XPATGEN:      SEC                ;SET THE C BIT TO ROTATE THROUGH THE DATA
2556 010454 012702 000000 1$:      MOV #0,R2          ;CLR R2 (CAN'T USE "CLR" AS IT CLEARS "C" BIT)
2557 010460 103001                                BCC 2$              ;BR IF THE "C" BIT IS CLEARED
2558 010462 005202                                INC R2              ;SET R2 IF NOT
2559 010464 004737 010564 2$:      JSR PC,LOAD        ;GO LOAD THE DATA BUFFER
2560 010470 000241                                CLC
2561 010472 005702                                TST R2              ;IS R2 NONZERO
2562 010474 001401                                BEQ 3$
2563 010476 000261                                SEC                ;YES, SET THE "C" BIT
2564 010500 106137 010416 3$:      ROLB DATABYTE
2565 010504 000763                                BR 1$
2566
2567 ;:*****
2568
2569 ;FLOAT A 1 THROUGH ALL ZEROS IN SOFTWARE BUFFER
2570 ; P = 4
2571
2572 010506 005037 010416 FLOAT1:      CLR DATABYTE
2573 010512 000757                                BR XPATGEN
2574
2575 ;:*****
2576
2577 ;LOAD SOFTWARE BUFFER WITH ALTERNATING 1 AND 0 FOR
2578 ;ONE BYTE AND THE COMPLIMENT INTO THE NEXT
2579 ; P = 5
2580
2581 010514 112737 000125 010416 PAT125:     MOVB #125,DATABYTE
2582 010522 004737 010564 XXPATGEN:  JSR PC,LOAD
2583 010526 105137 010416                                COMB DATABYTE      ; * (SEE NOTE BELOW )
2584 010532 000773                                BR XXPATGEN
2585
2586 ; * NOTE: TO MAKE PATTERN 5 AND 6 COMPATABLE WITH THE RX8 PATTERNS
2587 ;NOP THIS INSTRUCTION. (CHANGE THESE 2 LOCATIONS TO 000240)
2588 ;THIS CHANGE IS FOR INTERPROCESSOR RX** (COMPATABILITY TESTING ONLY.
2589
2590 ;:*****
2591
2592 ;LOAD SOFTWARE BUFFER WITH ALTERNATING PAIRS OF 1 AND 0 AND
2593 ;COMPLIMENT INTO THE NEXT
2594 ; P = 6
2595
2596 010534 112737 000314 010416 PAT314:     MOVB #314,DATABYTE
```

```
2597 010542 0C0767 BR XXPATGEN
2598
2599 ::*****
2600
2601 ;LOAD SOFTWARE BUFFER WITH RANDOM DATA PATTERN
2602 ; P = 0 OR 7
2603
2604 010544 004737 012122 RANDATA: JSR PC,RANGEN ;GET RANDOM NUMBER
2605 010550 113737 012214 010416 MOVB RANUM,DATABYTE
2606 010556 004737 010564 JSR PC,LOAD
2607 010562 000770 BR RANDATA
2608
2609 010564 063737 010416 010612 LOAD: ADD DATABYTE,SUM ;ACCUMULATE THE PATTERN CHECK SUM
2610 010572 113724 010416 MOVB DATABYTE,(R4)+ ;LOAD THE DATA BUFFER
2611 010576 022704 017100 CMP #BUFADR+176,R4 ;HAVE 124 BYTES BEEN GENERATED
2612 010602 001401 BEQ 1$ ;IF YES,RETURN TO TEST
2613 010604 000207 RTS PC ;IF NO,RETURN TO PATTERN GENERATOR
2614 010606 005726 1$: TST (SP)+ ;TAKE PATTERN RETURN ADDRESS OF STACK
2615 010610 000207 RTS PC ;RETURN TO TEST
2616
2617 010612 000000 SUM: 0
2618
2619 ::*****
2620
2621 ;TEST FOR SELECTED UNITS,DRIVE READY,AND USED CONDITIONS
2622 ;ALSO CONTAINS A 'HAD ERROR' FLAG TO BE TESTED AT EOP.
2623 ;THE BITS IN UNITSEL ARE USED AS FOLLOWS
2624
2625 ;BIT15 =UNIT 1 SELECTED VIA SWR
2626 ;BIT14 =UNIT 1 USED BIT
2627 ;BIT8 =THIS PASS HAD AN ERROR
2628 ;BIT7 =UNIT 0 SELECTED VIA SWR
2629 ;BIT6 =UNIT 0 USED BIT
2630 ;BIT4 =UNIT SELECTION FOR FUNCTION WORD
2631
2632 ::*****
2633
2634 010614 032737 000100 010704 GETUNIT: BIT #BIT6,UNITSEL ;WAS UNIT 0 JUST USED
2635 010622 001012 BNE 1$ ;UNIT 0 USED CHECK UNIT 1
2636 010624 105737 010704 TSTB UNITSEL ;WAS UNIT 0 SELECTED
2637 010630 100007 BPL 1$ ;NO GO TO UNIT 1
2638 010632 042737 040020 010704 BIC #40020,UNITSEL ;CLEAR UNIT 1 USED BIT AND FUNCTION UNIT BIT
2639 010640 052737 000100 010704 BIS #BIT6,UNITSEL ;SET UNIT 0 USED BIT
2640 010646 000207 RTS PC
2641 010650 005737 010704 1$: TST UNITSEL ;WAS UNIT 1 SELECTED
2642 010654 100012 BPL 2$ ;NO RETURN
2643 010656 032737 040000 010704 BIT #BIT14,UNITSEL ;HAS UNIT 1 BEEN USED
2644 010664 001006 BNE 2$ ;YES RETURN
2645 010666 042737 000100 010704 BIC #BIT6,UNITSEL ;CLEAR UNIT 0 USED BIT
2646 010674 052737 040020 010704 BIS #40020,UNITSEL ;SET UNIT 1 USED BIT AND FUNCTION UNIT BIT
2647 010702 000207 2$: RTS PC
2648
2649
2650
2651 010704 000000 UNITSEL: 0
2652
```

```

2653                                     ;TEST THAT ALL UNITS HAVE BEEN ACCESSED
2654
2655 010706 005737 010704      DONE:      TST UNITSEL      ;IS UNIT 1 SELECTED
2656 010712 100006             BPL 1$           ;NO RETURN
2657 010714 032737 040000 010704      BIT #BIT14,UNITSEL ;YES HAS IT BEEN USED
2658 010722 001002             BNE 1$           ;YES RETURN
2659 010724 062706 000002             ADD #2,SP        ;BYPASS NOT DONE RETURE ON STACK
2660 010730 000207             1$:      RTS PC
2661
2662
2663 010732 004737 010706      STOP:      JSR PC,DONE      ;HAVE ALL DRIVES BEEN USED
2664 010736 005037 007200      CLR HCNTR1
2665 010742 005237 021140      INC PASCNTR     ;KEEP TOTAL NUMBER OF PASSES
2666 010746 005737 011142      TST CHARCT     ;HAVE 72 EOP INDICATORS BEEN TYPED
2667 010752 001005             BNE 3$
2668 010754 104401 015150      TYPE ,MCRLF    ;YES DO A CRLF
2669 010760 012737 177670 011142      MOV #-72,,CHARCT ;RESET CHARACTER COUNTER
2670 010766 005237 011142      3$:      INC CHARCT
2671 010772 032737 000400 010704      BIT #BIT8,UNITSEL ;TEST HAD ERROR FLAG
2672 011000 001403             BEQ 2$         ;NO ERROR PRINT D
2673 011002 104401 015155      TYPE ,MEREOP   ;HAD ERROR PRINT ?
2674 011006 000404             BR 1$
2675 011010 104401 015153      2$:      TYPE ,MEOP      ;PRINT EOP INDICATOR
2676 011014 104401 015157      TYPE ,MABELL
2677 011020 104406             1$:      CKSWR
2678 011022 032777 040000 170164      BIT #SW14,@SWR  ;TEST EOP HALT SWITCH
2679 011030 001417             BEQ CONT22
2680 011032 104401 015150      TYPE ,MCRLF
2681 011036 104401 016240      TYPE ,MPASS    ;AT HALT ON PASS PRINT NUMBER OF PASSES COMPLETE
2682 011042 013737 021140 011054      MOV PASCNTR,6$
2683 011050 004537 014544      JSR R5,SGLDEC
2684 011054 000000             6$:      OPEN
2685 011056 104401 015150      TYPE ,MCRLF
2686 011062 000000             HLT16:     HALT
2687 011064 005037 011142      CLR CHARCT
2688 011070 042737 040500 010704      CONT22:    BIC #40500,UNITSEL ;THIS CAUSES A CRLF AFTER HALT AT E.O.P.
2689 011076 022737 000007 010420      CMP #7,PAT     ;CLEAR USED BITS, AND 'HAD ERROR' BIT
2690 011104 001002             BNE 2$         ;IF RANDOM DATA GET NEW DATA FOR BUFFER
2691 011106 004737 010312      JSR PC,GETPATTERN
2692 011112 005237 007200      2$:      INC HCNTR1     ;WAIT FOR THE EOP INDICATOR TO BE PRINTED
2693 011116 001375             BNE 2$
2694 011120 013705 000042      MOV @#42,R5   ;ACT11 END OF PASS HOOKS
2695 011124 001405             BEQ HERE
2696 011126 000005             RESET
2697 011130 004715      LOGICAL:    JSR PC,(R5)
2698 011132 000240             NOP
2699 011134 000240             NOP
2700 011136 000240             NOP
2701 011140 000207      HERE:      RTS PC
2702
2703 011142 000000      CHARCT:    0
2704
2705      ;;*****
2706
2707      ;INITIALIZE TRACK SEQUENCE
2708

```

```

2709
2710 011144 042737 100200 001200 INITTRACK: BIC #100200,OD ;CLEAR FIRST USED BITS
2711 011152 005737 001200 TST OD ;TEST CONTENTS OF ID,OD FOR 0
2712 011156 001005 BNE 1$ ;ID,OD SPECIFIED USE THEM
2713 011160 112737 000114 001201 MOVB #114,ID ;NONE SPECIFIED SET ID TO MAXIMUM
2714 011166 105037 001200 CLRB OD ;SET OD TO MINIMUM
2715 011172 113737 001200 011360 1$: MOVB OD,TARGET ;INIT OD AS PRESENT TRACK
2716 011200 005037 011366 CLR XID ;INIT WORKING ID AND OD LOCATIONS
2717 011204 113737 001201 011366 MOVB ID,XID
2718 011212 005037 011364 CLR XOD
2719 011216 113737 001200 011364 MOVB OD,XOD
2720 011224 013737 011366 011356 MOV XID,TRKCNTR ;SET UP NUMBER OF TRACK MOVEMENTS
2721 011232 163737 011364 011356 SUB XOD,TRKCNTR
2722 011240 005237 011356 INC TRKCNTR
2723 011244 052737 100200 001200 BIS #100200,OD ;SET FIRST TIME BITS IN ID,OD
2724 011252 000207 RTS PC

```

::*****

```

;TEST FOR HEAD SEQUENCE SELECTED BY INITIAL SWR SETTING
;BITS 0,1,AND 2 OF THE INITIAL SWR SELECTED THE TRACK SEQUENCING
;TO BE FOLLOWED AS INITICATED BELOW

```

BITS	SEQUENCE
: 0 0 0	NO SEQUENCE SPECIFIED (DEFAULT TO SEQ 7)
: 0 0 1	INCREMENT FROM OD TO ID
: 0 1 0	DECREMENT FROM ID TO OD
: 0 1 1	DO PREVIOUS 2 SEQUENCES
: 1 0 0	BOUNCE BETWEEN ID AND OD
: 1 0 1	DECREASING BOUNCE
: 1 1 0	STROBE BETWEEN OD AND DECREMENTING ID
: 1 1 1	RANDOM TRACK SELECTION

::*****

```

2746
2747 011254 113737 011360 011362 GETTRACK: MOVB TARGET,PRESTRK ;RESET TO PRESENT TRACK
2748 011262 142737 000377 011320 BICB #377,@#BRONTRK ;CLEAR OUT BRANCH OFFSET
2749 011270 005737 011370 TST SEQUEN ;IF NO SEQUENCE SPECIFIED FORCE SEQUENCE 7
2750 011274 001003 BNE 1$
2751 011276 012737 000007 011370 MOV #7,SEQUEN
2752 011304 013704 011370 1$: MOV SEQUEN,R4 ;GET SEQUENCE BITS
2753 011310 005304 DEC R4 ;ADJUST FOR CORRECT OFFSET
2754 011312 006304 ASL R4
2755 011314 150437 011320 BISB R4,@#BRONTRK ;THIS BR INST.IS MODIFIED BY THE TEST CONDITIONS
2756 ;SELECTED ,TO BRANCH TO ONE OF THE SEQ. BELOW
2757
2758 011320 000777 BRONTRK: BR . ;BRANCH BY OFFSET SELECTED
2759 011322 000137 JMP SEQ1 ;INCREMENT
2760 011326 000137 JMP SEQ2 ;DECREMENT
2761 011332 000137 JMP SEQ3 ;INCREMENT/DECREMENT
2762 011336 000137 JMP SEQ4 ;BOUNCE ID TO OD
2763 011342 000137 JMP SEQ5 ;DECREASING BOUNCE
2764 011346 000137 JMP SEQ6 ;STROBE

```

```
2765 011352 0C0137 012022          JMP SEQ7          ;RANDOM
2766
2767 011356 000000          TRKCNTR:         0
2768 011360 000000          TARGET:         0
2769 011362 000000          PRESTRK:        0
2770 011364 000000          XOD:            0
2771 011366 000000          XID:            0
2772 011370 000000          SEQUEN:         0
2773
2774          ;:*****
2775
2776          ;INCREMENT FROM OD+1 TO ID AND RETURN TO OD
2777          ; S = 1
2778
2779 011372 042737 100200 001200 SEQ1:          BIC #100200,OD          ;CLEAR FIRST TIME BITS
2780 011400 123737 011366 011362          CMPB XID,PRESTRK          ;PRESENT TRACK EQUAL TO ID
2781 011406 001004          BNE 1$          ;NO GET NEW TRACK
2782 011410 113737 001200 011360          MOVB OD,TARGET          ;YES RETURN TO OD
2783 011416 000461          BR NEWTRK          ;NEWTRK INCREMENTS THE TRACK MOVED TO COUNTER
2784          ;AND RETURNS TO NEXT TRACK
2785 011420 005237 011360          1$:          INC TARGET          ;ADD 1 TO TARGET TRACK
2786 011424 000456          BR NEWTRK
2787
2788          ;:*****
2789
2790          ;DECREMENT FROM ID TO OD
2791          ; S = 2
2792
2793 011426 105737 001201          SEQ2:          TSTB ID          ;FIRST TIME BIT SET
2794 011432 100007          BPL 1$          ;NO GET NEXT TRACK
2795 011434 042737 100200 001200          BIC #100200,OD          ;YES CLEAR FIRST TIME BITS
2796 011442 113737 011366 011360          MOVB XID,TARGET          ;GO TO ID
2797 011450 000444          BR NEWTRK
2798 011452 005337 011360          1$:          DEC TARGET          ;MOVE TO NEXT TRACK
2799 011456 000441          BR NEWTRK
2800
2801          ;:*****
2802
2803          ;INCREMENT THEN DECREMENT TRACKS
2804          ; S = 3
2805
2806 011460 105737 001200          SEQ3:          TSTB OD          ;CHECK FIRST TIME BIT
2807 011464 100007          BPL 1$          ;NOT FIRST TIME THROUGH
2808 011466 042737 100200 001200          BIC #100200,OD          ;CLEAR FIRST TIME BITS
2809 011474 005337 011356          DEC TRKCNTR
2810 011500 006337 011356          ASL TRKCNTR          ;RESET TRACK COUNTER TO DOUBLE THE COUNT
2811 011504 013700 011366          1$:          MOV XID,RO
2812 011510 163700 011364          SUB XOD,RO          ;GET DIFFERENCE BETWEEN ID AND OD
2813 011514 163700 011356          SUB TRKCNTR,RO          ;IS TRACK COUNTER HALF DONE
2814 011520 002342          BGE SEQ2          ;YES GO DECREMENT TRACKS
2815 011522 000723          BR SEQ1          ;NO CONTINUE INCREMENTING TRACKS
2816
2817          ;:*****
2818
2819          ;BOUNCE BETWEEN ID AND OD ONLY
2820
```

```
2821 ; S = 4
2822
2823 011524 042737 100200 001200 SEQ4: BIC #100200,OD ;CLEAR THE FIRST TIME BITS
2824 011532 123737 011362 001200 CMPB PRESTRK,OD ;DID IT JUST DO OD
2825 011540 001404 BEQ IDNEXT ;YES
2826 011542 113737 001200 011360 MOVB OD,TARGET ;NO GO TO OD TRACK
2827 011550 000404 BR NEWTRK
2828 011552 113737 001201 011360 IDNEXT: MOVB ID,TARGET ;GO DO ID TRACK
2829 011560 000404 BR NEWTRK
2830
2831 ;:*****
2832
2833 ;INCREMENT THE 'HEAD MOVED TO' COUNTERS AND RETURN
2834 ;THROUGH HERE FROM ALL TRACK SEQUENCES
2835
2836 011562 013705 011360 NEWTRK: MOV TARGET,R5 ;GET TRACK ADDRESS
2837 011566 006305 ASL R5 ;MULTIPLY BY 4 TO DOUBLE PRECISION
2838 011570 006305 ASL R5 ;INTERLEAVE COUNTER ADDRESSES
2839 011572 062705 017756 ADD #HDMOVE,R5 ;ADD ON ADDRESS OF HEAD MOVE COUNTER
2840 011576 062715 000001 ADD #1,(R5) ;INC COUNTER
2841 011602 005565 000002 ADC 2(R5) ;ADD CARRY TO HIGH ORDER WORD
2842 011606 000207 RTS PC ;RETURN TO ACCESS NEXT TRACK
2843
2844 ;:*****
2845
2846 ;BOUNCE BETWEEN DECREASING ID AND INCREASING OD
2847 ; S = 5
2848
2849
2850 011610 105737 001201 SEQ5: TSTB ID ;TEST FIRST TIME BIT OF ID
2851 011614 100011 BPL 1$
2852 011616 005237 011356 INC TRKCNTR ;ADJUST FOR ONE EXTRA MOVEMENT TO TRK 00
2853 011622 142737 000200 001201 BICB #200,ID ;CLEAR FIRST TIME BIT OF ID
2854 011630 113737 011366 011360 MOVB XID,TARGET ;MOVE TO ID
2855 011636 000751 BR NEWTRK
2856 011640 105737 001200 1$: TSTB OD ;TEST FIRST TIME BIT OF OD
2857 011644 100007 BPL 2$
2858 011646 142737 000200 001200 BICB #200,OD ;CLEAR FIRST TIME BIT OF OD
2859 011654 113737 011364 011360 MOVB XOD,TARGET ;MOVE TO OD
2860 011662 000737 BR NEWTRK
2861 011664 132737 000001 011356 2$: BITB #1,TRKCNTR ;TEST COUNTER FOR ODD OR EVEN
2862 011672 001006 BNE 3$ ;ODD MOVE TO NEW ID
2863 011674 005237 011364 INC XOD ;EVEN ADD 1 TO OD
2864 011700 113737 011364 011360 MOVB XOD,TARGET
2865 011706 000405 BR LASTTRK ;SEE IF LAST TRACK IS BEING ACCESSED
2866 011710 005337 011366 3$: DEC XID ;SUBTRACT 1 FROM ID
2867 011714 113737 011366 011360 MOVB XID,TARGET
2868 011722 123727 011356 000001 LASTTRK: CMPB TRKCNTR,#1
2869 011730 001003 BNE XRETURN
2870 011732 113737 001200 011360 MOVB OD,TARGET ;THIS IS LAST TRACK RETURN TO OD
2871 011740 000710 XRETURN: BR NEWTRK
2872
2873 ;:*****
2874
2875 ;STROBE BETWEEN OD AND DECREASING ID
2876 ; S = 6
```

```

2877
2878 011742 105737 001200 SEQ6: TSTB OD ;CHECK FIRST TIME BIT
2879 011746 100007 BPL 1$ ;NOT FIRST TIME
2880 011750 042737 100200 001200 BIC #100200,OD ;WAS FIRST TIME,CLEAR THE BITS
2881 011756 005337 011356 DEC TRKCNT ;RESET COUNTER TO DOUBLE THE NUMBER OF TRACKS
2882 011762 006337 011356 ASL TRKCNT
2883 011766 123737 011362 001200 1$: CMPB PRESTRK,OD ;WAS OD JUST ACCESSED
2884 011774 001006 BNE ODNEXT ;NO GO TO OD
2885 011776 113737 011366 011360 MOVB XID,TARGET ;DECREASING ID IS NEXT TRACK
2886 012004 005337 011366 DEC XID
2887 012010 000664 BR NEWTRK
2888 012012 113737 001200 011360 ODNEXT: MOVB OD,TARGET ;OD IS NEXT TRACK
2889 012020 000664 BR NEWTRK
2890
2891
2892
2893 ;:*****
2894 ;RANDOM SEQUENCING OF TRACKS
2895 ;FOR PROPER USE OF RANDOM TRACK SEQUENCING, THE OD/ID LIMITS
2896 ;SHOULD NOT BE SET FOR LESS THAN HALF THE TRACKS.
2897 ; S = 0 OR 7
2898
2899 012022 042737 100200 001200 SEQ7: BIC #100200,OD ;CLEAR THE FIRST TIME BITS
2900 012030 004737 012122 JSR PC,RANGEN ;GET A RANDOM NUMBER
2901 012034 042737 177600 012214 BIC #177600,RANUM ;CLEAR ALL BUT LOW 7 BITS
2902 012042 123737 012214 011366 IDCOMP: CMPB RANUM,XID ;IS RANUM LARGER THAN ID ADDRESS
2903 012050 003404 BLE ODCOMP ;NO,RANUM IS LESS OR EQUAL TO ID
2904 012052 163737 011366 012214 SUB XID,RANUM ;YES,SUBTRACT ID FROM IT
2905 012060 000770 BR IDCOMP ;SEE IF RANUM IS NOW OK
2906 012062 123737 012214 011364 ODCOMP: CMPB RANUM,XUD ;IS RANUM SMALLER THAN OD ADDRESS
2907 012070 002004 BGE PRESCHK ;NO,RANUM IS GREATER OR EQUAL TO OD
2908 012072 063737 011366 012214 ADD XOD,RANUM ;YES ADD OD TO RANUM
2909 012100 000770 BR ODCOMP ;SEE IF RANUM IS NOW OK
2910 012102 123737 012214 011362 PRESCHK: CMPB RANUM,PRESTRK ;IF RANUM EQUALS PRESENT TRACK
2911 012110 001744 BEQ SEQ7 ;GET ANOTHER RANDOM NUMBER
2912 012112 013737 012214 011360 MOV RANUM,TARGET ;RANUM OK PUT IT IN TARGET TRACK
2913 012120 000664 BR NEWTRK
2914
2915 012122 012700 000001 RANGEN: MOV #1,R0
2916 012126 063700 012210 ADD RAN1,R0
2917 012132 063700 012212 ADD RAN2,R0
2918 012136 042700 170000 BIC #170000,R0
2919 012142 000241 CLC
2920 012144 006100 ROL R0
2921 012146 006100 ROL R0
2922 012150 010037 012210 MOV R0,RAN1
2923 012154 005000 CLR R0
2924 012156 013700 012212 MOV RAN2,R0
2925 012162 006000 ROR R0
2926 012164 006000 ROR R0
2927 012166 063700 012210 ADD RAN1,R0
2928 012172 042700 170000 BIC #170000,R0
2929 012176 010037 012212 MOV R0,RAN2
2930 012202 010037 012214 MOV R0,RANUM
2931 012206 000207 RTS PC
2932

```

2933 012210 000000
2934 012212 000000
2935 012214 000000

RAN1: 0
RAN2: 0
RANUM: 0

::*****

:SECTOR INITIALIZATION AND SELECTION

2941 012216 005737 001202

INITSECTOR: TST FIRST ;TEST FIRST AND LAST FOR 0
BNE 1\$;SECTORS SPECIFIED USE THEM
INC FIRST ;NONE SPECIFIED SET FIRST TO 1
MOVB #32, LAST ;SET LAST TO MAXIMUM
1\$: MOVB LAST, SECCNTR ;SET UP SECTOR COUNTER
SUB FIRST, SECCNTR
INC SECCNTR
CLRB SECCNTR+1
MOVB FIRST, TSECTOR ;PUT FIRST SECTOR IN TARGET SECTOR
SUB THREE, TSECTOR ;SUB 3 FROM TSECTOR AS FIRST TIME THROUGH
;IT GETS ADDED BACK ON.
MOV #1, INTLEAV ;SET INTERLEAVE OFFSET
RTS PC

2942 012222 001005

2943 012224 005237 001202

2944 012230 112737 00C032 001203

2945 012236 113737 001203 012306

2946 012244 163737 0012C2 012306

2947 012252 005237 012306

2948 012256 105037 012307

2949 012262 113737 001202 012310

2950 012270 163737 012314 012310

2951

2952 012276 012737 000001 012312

2953 012304 000207

2954

2955 012306 000000

2956 012310 000000

2957 012312 000000

2958 012314 000003

2959

2960 012316 063737 012314 012310

2961 012324 123737 001203 012310

2962 012332 002010

2963 012334 113737 001202 012310

2964 012342 063737 012312 012310

2965 012350 005237 012312

2966 012354 000207

2967

2968

2969

2970

2971

2972

2973

2974

2975

2976

2977

2978

2979

2980

2981

2982

2983

2984

2985

2986 012356 105737 012663

2987 012362 100002

2988 012364 000000

SECCNTR: 0
TSECTOR: 0
INTLEAV: 0
THREE: 3

GETSECTOR: ADD THREE, TSECTOR ;ADD 3 FOR INTERLEAVING
CMPB LAST, TSECTOR
BGE 1\$;NEW SECTOR IS WITHIN LIMITS
MOVB FIRST, TSECTOR ;RESET TARGET SECTOR TO INTERLEAVE
ADD INTLEAV, TSECTOR ;ADD ON INTERLEAVE OFFSET VALUE
INC INTLEAV ;UP DATE THE OFFSET VALUE
1\$: RTS PC

.SBTTL TYPE ROUTINE

::*****
:*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
:*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
:*NOTE1: \$NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
:*NOTE2: \$FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
:*NOTE3: \$FILLC CONTAINS THE CHARACTER TO FILL AFTER.
:*
:*CALL:
:*1) USING A TRAP INSTRUCTION
:* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
:*OR
:* TYPE
:* M\$SADR
:*

\$TYPE: TSTB \$TFPLG ;:IS THERE A TERMINAL?
BPL 1\$;:BR IF YES
HALT ;:HALT HERE IF NO TERMINAL

2989	012366	000407			BR	3\$:: LEAVE	
2990	012370	010046			1\$: MOV	R0, -(SP)	:: SAVE R0	
2991	012372	017600	000002		MOV	@2(SP), R0	:: GET ADDRESS OF ASCIZ STRING	
2992	012376	112046			2\$: MOV	(R0)+, -(SP)	:: PUSH CHARACTER TO BE TYPED ONTO STACK	
2993	012400	001005			BNE	4\$:: BR IF IT ISN'T THE TERMINATOR	
2994	012402	005726			TST	(SP)+	:: IF TERMINATOR POP IT OFF THE STACK	
2995	012404	012600			60\$: MOV	(SP)+, R0	:: RESTORE R0	
2996	012406	062716	000002		3\$: ADD	#2, (SP)	:: ADJUST RETURN PC	
2997	012412	000002			RTI		:: RETURN	
2998	012414	122716	000011		4\$: CMPB	#HT, (SP)	:: BRANCH IF <HT>	
2999	012420	001430			BEQ	8\$		
3000	012422	122716	000200		CMPB	#CRLF, (SP)	:: BRANCH IF NOT <CRLF>	
3001	012426	001006			BNE	5\$		
3002	012430	005726			TST	(SP)+	:: POP <CR><LF> EQUIV	
3003	012432	104401			TYPE		:: TYPE A CR AND LF	
3004	012434	012665			\$CRLF			
3005	012436	105037	012644		CLRB	\$CHARCNT	:: CLEAR CHARACTER COUNT	
3006	012442	000755			BR	2\$:: GET NEXT CHARACTER	
3007	012444	004737	012526		5\$: JSR	PC, \$TYPEC	:: GO TYPE THIS CHARACTER	
3008	012450	123726	012662		6\$: CMPB	\$FILLC, (SP)+	:: IS IT TIME FOR FILLER CHARS.?	
3009	012454	001350			BNE	2\$:: IF NO GO GET NEXT CHAR.	
3010	012456	013746	012660		MOV	\$NULL, -(SP)	:: GET # OF FILLER CHARS. NEEDED	
3011							:: AND THE NULL CHAR.	
3012	012462	105366	000001		7\$: DECB	1(SP)	:: DOES A NULL NEED TO BE TYPED?	
3013	012466	002770			BLT	6\$:: BR IF NO--GO POP THE NULL OFF OF STACK	
3014	012470	004737	012526		JSR	PC, \$TYPEC	:: GO TYPE A NULL	
3015	012474	105337	012644		DECB	\$CHARCNT	:: DO NOT COUNT AS A COUNT	
3016	012500	000770			BR	7\$:: LOOP	
3017								
3018								
3019								
3020	012502	112716	000040		8\$: MOV	#' , (SP)	:: REPLACE TAB WITH SPACE	
3021	012506	004737	012526		9\$: JSR	PC, \$TYPEC	:: TYPE A SPACE	
3022	012512	132737	000007	012644	BITB	#7, \$CHARCNT	:: BRANCH IF NOT AT	
3023	012520	001372			BNE	9\$:: TAB STOP	
3024	012522	005726			TST	(SP)+	:: POP SPACE OFF STACK	
3025	012524	000724			BR	2\$:: GET NEXT CHARACTER	
3026	012526				\$TYPEC:			
3027	012526	105777	000116		TSTB	@\$TKS	:: CHAR IN KYBD BUFFER?	:MJD001
3028	012532	100022			BPL	10\$:: BR IF NOT	:MJD001
3029	012534	017746	000112		MOV	@\$TKB, -(SP)	:: GET CHAR	:MJD001
3030	012540	042716	177600		BIC	#177600, (SP)	:: STRIP EXTRANEIOUS BITS	:MJD001
3031	012544	122716	000023		CMPB	#\$XOFF, (SP)	:: WAS CHAR XOFF	:MJD001
3032	012550	001012			BNE	102\$:: BR IF NOT	:MJD001
3033	012552				101\$:			:MJD001
3034	012552	105777	000072		TSTB	@\$TKS	:: WAIT FOR CHAR	:MJD001
3035	012556	100375			BPL	101\$:MJD001
3036	012560	117716	000066		MOV	@\$TKB, (SP)	:: GET CHAR	:MJD001
3037	012564	042716	177600		BIC	#177600, (SP)	:: STRIP IT	:MJD001
3038	012570	122716	000021		CMPB	#\$XON, (SP)	:: WAS IT XON?	:MJD001
3039	012574	001366			BNE	101\$:: BR IF NOT	:MJD001
3040	012576				102\$:			:MJD001
3041	012576	005726			TST	(SP)+	:: FIX STACK	:MJD001
3042	012600				10\$:			:MJD001
3043	012600	105777	000050		TSTB	@\$TPS	:: WAIT UNTIL PRINTER IS READY	
3044	012604	100375			BPL	10\$:MJD001


```

3101 012740 010446      MOV      R4,-(SP)      ::SAVE R4
3102 012742 010546      MOV      R5,-(SP)      ::SAVE R5
3103 012744 113704 013115  MOVVB    $OMODE+1,R4   ::GET THE NUMBER OF DIGITS TO TYPE
3104 012750 005404      NEG      R4
3105 012752 062704 000006  ADD      #6,R4         ::SUBTRACT IT FOR MAX. ALLOWED
3106 012756 110437 013114  MOVVB    R4,$OMODE     ::SAVE IT FOR USE
3107 012762 113704 013113  MOVVB    $OFILL,R4     ::GET THE ZERO FILL SWITCH
3108 012766 016605 000012  MOV      12(SP),R5     ::PICKUP THE INPUT NUMBER
3109 012772 005003      CLR      R3           ::CLEAR THE OUTPUT WORD
3110 012774 006105      1$:     ROL      R5           ::ROTATE MSB INTO 'C'
3111 012776 000404      BR       3$           ::GO DO MSB
3112 013000 006105      2$:     ROL      R5           ::FORM THIS DIGIT
3113 013002 006105      ROL      R5
3114 013004 006105      ROL      R5
3115 013006 010503      MOV      R5,R3
3116 013010 006103      3$:     ROL      R3           ::GET LSB OF THIS DIGIT
3117 013012 105337 013114  DECB     $OMODE        ::TYPE THIS DIGIT?
3118 013016 100016      BPL      7$           ::BR IF NO
3119 013020 042703 177770  BIC      #177770,R3    ::GET RID OF JUNK
3120 013024 001002      BNE      4$           ::TEST FOR 0
3121 013026 005704      TST      R4           ::SUPPRESS THIS 0?
3122 013030 001403      BEQ      5$           ::BR IF YES
3123 013032 005204      4$:     INC      R4           ::DON'T SUPPRESS ANYMORE 0'S
3124 013034 052703 000060  BIS      #'0,R3       ::MAKE THIS DIGIT ASCII
3125 013040 052703 000040  BIS      #' ,R3       ::MAKE ASCII IF NOT ALREADY
3126 013044 110337 013110  MOVVB    R3,8$        ::SAVE FOR TYPING
3127 013050 104401 013110  TYPE     ,8$          ::GO TYPE THIS DIGIT
3128 013054 105337 013112  7$:     DECB     $OCNT     ::COUNT BY 1
3129 013060 003347      BGT      2$           ::BR IF MORE TO DO
3130 013062 002402      BLT      6$           ::BR IF DONE
3131 013064 005204      INC      R4           ::INSURE LAST DIGIT ISN'T A BLANK
3132 013066 000744      BR       2$           ::GO DO THE LAST DIGIT
3133 013070 012605      6$:     MOV      (SP)+,R5    ::RESTORE R5
3134 013072 012604      MOV      (SP)+,R4    ::RESTORE R4
3135 013074 012603      MOV      (SP)+,R3    ::RESTORE R3
3136 013076 016666 000002 000004  MOV      2(SP),4(SP)  ::SET THE STACK FOR RETURNING
3137 013104 012616      MOV      (SP)+,(SP)
3138 013106 000002      RTI                    ::RETURN
3139 013110      000      8$:     .BYTE    0           ::STORAGE FOR ASCII DIGIT
3140 013111      000      .BYTE    0           ::TERMINATOR FOR TYPE ROUTINE
3141 013112      000      $OCNT:  .BYTE    0           ::OCTAL DIGIT COUNTER
3142 013113      000      $OFILL: .BYTE    0           ::ZERO FILL SWITCH
3143 013114 000000      $OMODE: .WORD    0           ::NUMBER OF DIGITS TO TYPE
3144      .SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
3145
3146      ::*****
3147      ::*SAVE R0-R5
3148      ::*CALL:
3149      ::* SAVREG
3150      ::*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE:
3151      ::*
3152      ::*TOP---(+16)
3153      ::* +2---(+18)
3154      ::* +4---R5
3155      ::* +6---R4
3156      ::* +8---R3
    
```

3157
3158
3159
3160
3161 013116
3162 013116 010046
3163 013120 010146
3164 013122 010246
3165 013124 010346
3166 013126 010446
3167 013130 010546
3168 013132 016646 000022
3169 013136 016646 000022
3170 013142 016646 000022
3171 013146 016646 000022
3172 013152 000002
3173
3174
3175
3176
3177 013154
3178 013154 012666 000022
3179 013160 012666 000022
3180 013164 012666 000022
3181 013170 012666 000022
3182 013174 012605
3183 013176 012604
3184 013200 012603
3185 013202 012602
3186 013204 012601
3187 013206 012600
3188 013210 000002
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198
3199 013212 022737 000176 001214
3200 013220 001074
3201 013222 105777 177422
3202 013226 100071
3203 013230 117746 177416
3204 013234 042716 177600
3205 013240 022726 000007
3206 013244 001062
3207 013246 123727 013776 000001
3208 013254 001456
3209
3210 013256 104401 013747
3211 013262 104401 013754
3212 013266 013746 000176

```

:++10---R2
:++12---R1
:++14---R0

$SAVREG:
MOV R0,-(SP)      ;;PUSH R0 ON STACK
MOV R1,-(SP)      ;;PUSH R1 ON STACK
MOV R2,-(SP)      ;;PUSH R2 ON STACK
MOV R3,-(SP)      ;;PUSH R3 ON STACK
MOV R4,-(SP)      ;;PUSH R4 ON STACK
MOV R5,-(SP)      ;;PUSH R5 ON STACK
MOV 22(SP),-(SP)  ;;SAVE PS OF MAIN FLOW
MOV 22(SP),-(SP)  ;;SAVE PC OF MAIN FLOW
MOV 22(SP),-(SP)  ;;SAVE PS OF CALL
MOV 22(SP),-(SP)  ;;SAVE PC OF CALL
RTI

:*RESTORE R0-R5
:*CALL:
:*
:* RESREG
$RESREG:
MOV (SP)+,22(SP)  ;;RESTORE PC OF CALL
MOV (SP)+,22(SP)  ;;RESTORE PS OF CALL
MOV (SP)+,22(SP)  ;;RESTORE PC OF MAIN FLOW
MOV (SP)+,22(SP)  ;;RESTORE PS OF MAIN FLOW
MOV (SP)+,R5      ;;POP STACK INTO R5
MOV (SP)+,R4      ;;POP STACK INTO R4
MOV (SP)+,R3      ;;POP STACK INTO R3
MOV (SP)+,R2      ;;POP STACK INTO R2
MOV (SP)+,R1      ;;POP STACK INTO R1
MOV (SP)+,R0      ;;POP STACK INTO R0
RTI

.SBTTL TTY INPUT ROUTINE

:*****
.ENABL LSB

:*****
:*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 FLAG MODE.
$CKSWR: CMP #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
BNE 15$ ;;BRANCH IF NO
TSTB @STKS ;;CHAR THERE?
BPL 15$ ;;IF NO, DON'T WAIT AROUND
MOVB @STKB,-(SP) ;;SAVE THE CHAR
BIC #^C177,(SP) ;;STRIP-OFF THE ASCII
CMP #7,(SP)+ ;;IS IT A CONTROL G?
BNE 15$ ;;NO, RETURN TO USER
CMPB $AUTOB,#1 ;;ARE WE RUNNING IN AUTO-MODE?
BEQ 15$ ;;BRANCH IF YES

$GTSWR: TYPE ,SCNTLG ;;ECHO THE CONTROL-G (^G)
TYPE ,SMSWR ;;TYPE CURRENT CONTENTS
MOV SWREG,-(SP) ;;SAVE SWREG FOR TYPEOUT

```

```

3213 013272 104402          TYP0C          ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
3214 013274 104401 013765  TYPE      ,SMNEW  ;;PROMPT FOR NEW SWR
3215 013300 005046          CLR      -(SP)   ;;CLEAR COUNTER
3216 013302 005046          CLR      -(SP)   ;;THE NEW SWR
3217 013304 105777 177340  TSTB     @STKS   ;;CHAR THERE?
3218 013310 100375          BPL      7$      ;;IF NOT TRY AGAIN
3219
3220 013312 117746 177334  MOVB     @STKB,-(SP) ;;PICK UP CHAR
3221 013316 042716 177600  BIC     #^C177,(SP) ;;MAKE IT 7-BIT ASCII
3222
3223
3224
3225 013322 021627 000025  9$:     CMP     (SP),#25  ;;IS IT A CONTROL-U?
3226 013326 001005          BNE     10$      ;;BRANCH IF NOT
3227 013330 104401 013742  TYPE     ,SCNTLU  ;;YES, ECHO CONTROL-U (^U)
3228 013334 062706 000006  20$:   ADD     #6,SP   ;;IGNORE PREVIOUS INPUT
3229 013340 000757          BR      19$      ;;LET'S TRY IT AGAIN
3230
3231
3232 013342 021627 000015  10$:   CMP     (SP),#15  ;;IS IT A <CR>?
3233 013346 001022          BNE     16$      ;;BRANCH IF NO
3234 013350 005766 000004  TST     4(SP)    ;;YES, IS IT THE FIRST CHAR?
3235 013354 001403          BEQ     11$      ;;BRANCH IF YES
3236 013356 016677 000002 165630  MOV     2(SP),@SWR ;;SAVE NEW SWR
3237 013364 062706 000006  11$:   ADD     #6,SP   ;;CLEAR UP STACK
3238 013370 104401 012665  14$:   TYPE     ,SCRLF ;;ECHO <CR> AND <LF>
3239 013374 123727 013777 000001  CMPB    $INTAG,#1 ;;RE-ENABLE TTY KBD INTERRUPTS?
3240 013402 001003          BNE     15$      ;;BRANCH IF NOT
3241 013404 012777 000100 177236  MOV     #100,@STKS ;;RE-ENABLE TTY KBD INTERRUPTS
3242 013412 000002          RTI     ;;RETURN
3243 013414 004737 012526  16$:   JSR     PC,$TYPEC ;;ECHO CHAR
3244 013420 021627 000060  CMP     (SP),#60 ;;CHAR < 0?
3245 013424 002420          BLT     18$      ;;BRANCH IF YES
3246 013426 021627 000067  CMP     (SP),#67 ;;CHAR . 7?
3247 013432 003015          BGT     18$      ;;BRANCH IF YES
3248 013434 042726 000060  BIC     #60,(SP)+ ;;STRIP-OFF ASCII
3249 013440 005766 000002  TST     2(SP)    ;;IS THIS THE FIRST CHAR
3250 013444 001403          BEQ     17$      ;;BRANCH IF YES
3251 013446 006316          ASL     (SP)    ;;NO, SHIFT PRESENT
3252 013450 006316          ASL     (SP)    ;;CHAR OVER TO MAKE
3253 013452 006316          ASL     (SP)    ;;ROOM FOR NEW ONE.
3254 013454 005266 000002  17$:   INC     2(SP)  ;;KEEP COUNT OF CHAR
3255 013460 056616 177776  BIS     -2(SP),(SP) ;;SET IN NEW CHAR
3256 013464 000707          BR      7$      ;;GET THE NEXT ONE
3257 013466 104401 012664  18$:   TYPE     ,SQUES ;;TYPE ?<CR><LF>
3258 013472 000720          BR      20$     ;;SIMULATE CONTROL-U
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268

```

:*****
:THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
:CALL:
:RDCHR ;;INPUT A SINGLE CHARACTER FROM THE TTY
:RETURN HERE ;;CHARACTER IS ON THE STACK
:WITH PARITY BIT STRIPPED OFF
:

```

3269
3270 013474 011646          $RDCHR: MOV      (SP),-(SP)      ;;PUSH DOWN THE PC
3271 013476 016666 000004 000002 MOV      4(SP),2(SP)      ;;SAVE THE PS
3272 013504 105777 177140 1$:  TSTB    @STKS          ;;WAIT FOR
3273 013510 100375          BPL      1$              ;;A CHARACTER
3274 013512 117766 177134 000004 MOVB    @STKB,4(SP)      ;;READ THE TTY
3275 013520 042766 177600 000004 BIC     #'^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
3276 013526 026627 000004 000023 CMP     4(SP),#23        ;;IS IT A CONTROL-S?
3277 013534 001013          BNE     3$              ;;BRANCH IF NO
3278 013536 105777 177106 2$:  TSTB    @STKS          ;;WAIT FOR A CHARACTER
3279 013542 100375          BPL     2$              ;;LOOP UNTIL ITS THERE
3280 013544 117746 177102          MOVB    @STKB,-(SP)      ;;GET CHARACTER
3281 013550 042716 177600          BIC     #'^C177,(SP)    ;;MAKE IT 7-BIT ASCII
3282 013554 022627 000021          CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
3283 013560 001366          BNE     2$              ;;IF NOT DISCARD IT
3284 013562 000750          BR      1$              ;;YES, RESUME
3285 013564 026627 000004 000021 3$:  CMP     4(SP),#$XON      ;;IS IT A RANDOM XON?
3286 013572 001744          BEQ     1$              ;;BRANCH IF YES
3287 013574 026627 000004 000140 CMP     4(SP),#140      ;;IS IT UPPER CASE?
3288 013602 002407          BLT     4$              ;;BRANCH IF YES
3289 013604 026627 000004 000175 CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
3290 013612 003003          BGT     4$              ;;BRANCH IF YES
3291 013614 042766 000040 000004 BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
3292 013622 000002          4$:  RTI              ;;GO BACK TO USER
3293
3294 *****
3295 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
3296 *CALL:
3297 *      RDLIN          ;;INPUT A STRING FROM THE TTY
3298 *      RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
3299 *                    ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
3300 $RDLIN: MOV      R3,-(SP)      ;;SAVE R3
3301 1$:  MOV      #$TTYIN,R3      ;;GET ADDRESS
3302 2$:  CMP      #$TTYIN+8.,R3    ;;BUFFER FULL?
3303     BLOS     4$              ;;BR IF YES
3304     RDCHR    ;;GO READ ONE CHARACTER FROM THE TTY
3305     MOVB    (SP)+,(R3)      ;;GET CHARACTER
3306 10$: CMPB    #177,(R3)      ;;IS IT A RUBOUT
3307     BNE     3$              ;;SKIP IF NOT
3308 4$:  TYPE    ,SQUES          ;;TYPE A '?'
3309     BR      1$              ;;CLEAR THE BUFFER AND LOOP
3310 3$:  MOVB    (R3),9$          ;;ECHO THE CHARACTER
3311     TYPE    ,9$
3312     CMPB    #15,(R3)+      ;;CHECK FOR RETURN
3313     BNE     2$              ;;LOOP IF NOT RETURN
3314     CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
3315     TYPE    ,LF            ;;TYPE A LINE FEED
3316     MOV     (SP)+,R3        ;;RESTORE R3
3317     MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
3318     MOV     4(SP),2(SP)    ;;FIRST ASCII CHARACTER ON IT
3319     MOV     #$TTYIN,4(SP)
3320     RTI
3321 9$:  .BYTE    0              ;;RETURN
3322     .BYTE    0              ;;STORAGE FOR ASCII CHAR. TO TYPE
3323     .BLKB   8              ;;TERMINATOR
3324 013732 000010 005015 000 $TTYIN: .BLKB   8              ;;RESERVE 8 BYTES FOR TTY INPUT
3324 013742 052536          $CNTLU: .ASCII  /^U/<15><12> ;;CONTROL 'U'

```

```

3325 013747 136 006507 000012 $CNTLG: .ASCIZ / ^G / <15> <12> ;:CONTROL 'G'
3326 013754 005015 053523 020122 $MSWR: .ASCIZ <15> <12> / SWR = /
3327 013762 020075 000
3328 013765 040 047040 053505 $MNEW: .ASCIZ / NEW = /
3329 013772 036440 000040
3330 013776 000
3331 013777 000
3332
3333
3334
3335 ;:*****
3336 ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
3337 ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
3338 ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
3339 ;*GO TO THAT ROUTINE.
3340 014000 010046 $TRAP: MOV R0, -(SP) ;:SAVE R0
3341 014002 016600 000002 MOV 2(SP), R0 ;:GET TRAP ADDRESS
3342 014006 005740 TST -(R0) ;:BACKUP BY 2
3343 014010 111000 MOVB (R0), R0 ;:GET RIGHT BYTE OF TRAP
3344 014012 006300 ASL R0 ;:POSITION FOR INDEXING
3345 014014 016000 014034 MOV $TRPAD(R0), R0 ;:INDEX TO TABLE
3346 014020 000200 RTS R0 ;:GO TO ROUTINE
3347
3348
3349 ;:THIS IS USE TO HANDLE THE 'GETPRI' MACRO
3350
3351 014022 011646 $TRAP2: MOV (SP), -(SP) ;:MOVE THE PC DOWN
3352 014024 016666 000004 000002 MOV 4(SP), 2(SP) ;:MOVE THE PSW DOWN
3353 014032 000002 RTI ;:RESTORE THE PSW
3354
3355 .SBTTL TRAP TABLE
3356
3357 ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
3358 ;*BY THE 'TRAP' INSTRUCTION.
3359
3360 : ROUTINE
3361 : -----
3362 014034 014022 $TRPAD: .WORD $TRAP2
3363 014036 012356 $TYPE ;:CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
3364 014040 012714 $TYPOC ;:CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
3365 014042 012670 $TYPOS ;:CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
3366 014044 012730 $TYPON ;:CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
3367
3368 014046 013262 $GTSWR ;:CALL=GTSWR TRAP+5(104405) GET SOFT-SWR SETTING
3369
3370 014050 013212 $CKSWR ;:CALL=CKSWR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SWR
3371 014052 013474 $RDCHR ;:CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
3372 014054 013624 $RDLIN ;:CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE
3373 014056 013116 $SAVREG ;:CALL=SAVREG TRAP+11(104411) SAVE R0-R5 ROUTINE
3374 014060 013154 $RESREG ;:CALL=RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE
3375
3376 .SBTTL POWER DOWN AND UP ROUTINES
3377
3378 ;:*****
3379 014062 012737 014226 000024 $PWRDN: MOV # $ILLUP, @#PWRVEC ;:SET FOR FAST UP
3380 014070 012737 000340 000026 MOV #340, @#PWRVEC+2 ;:PRIO:7
    
```

```

3381 014076 010046      MOV      R0,-(SP)      ;;PUSH R0 ON STACK
3382 014100 010146      MOV      R1,-(SP)      ;;PUSH R1 ON STACK
3383 014102 010246      MOV      R2,-(SP)      ;;PUSH R2 ON STACK
3384 014104 010346      MOV      R3,-(SP)      ;;PUSH R3 ON STACK
3385 014106 010446      MOV      R4,-(SP)      ;;PUSH R4 ON STACK
3386 014110 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
3387 014112 017746 165076  MOV      @SWR,-(SP)     ;;PUSH @SWR ON STACK
3388 014116 010637 014232  MOV      SP,$SAVR6     ;;SAVE SP
3389 014122 012737 014134 00C024  MOV      #SPWRUP,@#PWRVEC ;;SET UP VECTOR
3390 014130 000000      HALT
3391 014132 000776      BR       -2           ;;HANG UP
3392
3393
3394
3395 014134 012737 014226 000024  $PWRUP: MOV      #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
3396 014142 013706 014232      MOV      $SAVR6,SP     ;;GET SP
3397 014146 005037 014232      CLR      $SAVR6        ;;WAIT LOOP FOR THE TTY
3398 014152 005237 014232      1$: INC      $SAVR6     ;;WAIT FOR THE INC
3399 014156 001375      BNE      1$           ;;OF WORD
3400 014160 012677 165030      MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
3401 014164 012605      MOV      (SP)+,R5     ;;POP STACK INTO R5
3402 014166 012604      MOV      (SP)+,R4     ;;POP STACK INTO R4
3403 014170 012603      MOV      (SP)+,R3     ;;POP STACK INTO R3
3404 014172 012602      MOV      (SP)+,R2     ;;POP STACK INTO R2
3405 014174 012601      MOV      (SP)+,R1     ;;POP STACK INTO R1
3406 014176 012600      MOV      (SP)+,R0     ;;POP STACK INTO R0
3407 014200 012737 014062 000024  MOV      #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
3408 014206 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;PRIO:7
3409 014214 104401      TYPE
3410 014216 014234      $PWRMG: .WORD $POWER   ;;POWER FAIL MESSAGE POINTER
3411 014220 012716      MOV      (PC)+,(SP)   ;;RESTART AT RESTART
3412 014222 002700      $PWRAD: .WORD RESTART ;;RESTART ADDRESS
3413 014224 000002      RTI
3414 014226 000000      $SILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
3415 014230 000776      BR       -2           ;; BEFORE THE POWER DOWN WAS COMPLETE
3416 014232 000000      $SAVR6: 0            ;;PUT THE SP HERE
3417 014234 005015 047520 042527  $POWER: .ASCIZ <15><12>'POWER'
3418 014242 000122
3419
3420
3421
3422
3423
3424
3425
3426
3427
3428
3429
3430
3431 014244 016637 000002 014274  $SB2D: MOV      2(SP),1$     ;;SAVE BINARY NUMBER
3432 014252 012746 014274      MOV      #1$,-(SP)    ;;SET POINTER
3433 014256 004737 014300      JSR      PC,@#$SB2D   ;;CALL DOUBLE LENGTH CONVERT
3434 014262 062716 000005      ADD      #5,(SP)      ;;ONLY ALLOW FIVE CHARACTERS
3435 014266 012666 000002      MOV      (SP)+,2(SP)  ;;PICKUP POINTER
3436 014272 000207      RTS      PC           ;;RETURN

```

POWER UP ROUTINE

```

$PWRUP: MOV      #SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
MOV      $SAVR6,SP     ;;GET SP
CLR      $SAVR6        ;;WAIT LOOP FOR THE TTY
1$: INC      $SAVR6     ;;WAIT FOR THE INC
BNE      1$           ;;OF WORD
MOV      (SP)+,@SWR    ;;POP STACK INTO @SWR
MOV      (SP)+,R5     ;;POP STACK INTO R5
MOV      (SP)+,R4     ;;POP STACK INTO R4
MOV      (SP)+,R3     ;;POP STACK INTO R3
MOV      (SP)+,R2     ;;POP STACK INTO R2
MOV      (SP)+,R1     ;;POP STACK INTO R1
MOV      (SP)+,R0     ;;POP STACK INTO R0
MOV      #SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
MOV      #340,@#PWRVEC+2 ;;PRIO:7
TYPE
$PWRMG: .WORD $POWER   ;;POWER FAIL MESSAGE POINTER
MOV      (PC)+,(SP)   ;;RESTART AT RESTART
$PWRAD: .WORD RESTART ;;RESTART ADDRESS
RTI
$SILLUP: HALT        ;;THE POWER UP SEQUENCE WAS STARTED
BR       -2           ;; BEFORE THE POWER DOWN WAS COMPLETE
$SAVR6: 0            ;;PUT THE SP HERE
$POWER: .ASCIZ <15><12>'POWER'

```

.EVEN
.SBTTL SINGLE LENGTH BINARY TO DECIMAL ASCIZ ROUTINE

*THIS ROUTINE WILL CONVERT A 16-BIT UNSIGNED BINARY NUMBER TO AN
*UNSIGNED DECIMAL ASCIZ NUMBER.

```

*CALL
* MOV      NUMBER,-(SP) ;;PUT BINARY NUMBER ON THE STACK
* JSR      PC,@#$SB2D   ;;CALL
* RETURN   ;;ADDRESS OF THE 1ST ASCIZ CHAR.IS ON THE STACK

```


3437 014274 0C0000 0000CJ
3438
3439
3440
3441
3442
3443
3444
3445
3446
3447
3448
3449
3450
3451 014300 104411
3452 014302 016602 000002
3453 014306 012700 014460
3454 014312 010066 000002
3455 014316 012201
3456 014320 012202
3457 014322 012737 000012 014376
3458 014330 012704 014410
3459 014334 012705 014412
3460 014340 005003
3461 014342 161401
3462 014344 005602
3463 014346 161502
3464 014350 002402
3465 014352 005203
3466 014354 000772
3467 014356 062401
3468 014360 005502
3469 014362 062402
3470 014364 022525
3471 014366 052703 000060
3472 014372 110320
3473 014374 005327
3474 014376 000000
3475 014400 001357
3476 014402 105020
3477 014404 104412
3478 014406 000207
3479 014410 145000
3480 014412 035632
3481 014414 160400
3482 014416 002765
3483 014420 113200
3484 014422 000230
3485 014424 041100
3486 014426 000017
3487 014430 103240
3488 014432 000001
3489 014434 023420
3490 014436 000000
3491 014440 001750
3492 014442 000000

1\$: .WORD 0,0
.SBTTL DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

*THIS ROUTINE WILL CONVERT A 32-BIT BINARY NUMBER TO AN UNSIGNED
*DECIMAL (ASCII) NUMBER. THE SIGN OF THE BINARY NUMBER MUST BE
*POSITIVE.
*CALL
* MOV #PNTR,-(SP) ;; POINTER TO LOW WORD OF BINARY NUMBER
* JSR PC,@#\$DB2D
* RETURN ;; THE FIRST ADDRESS OF ASCIZ
;; IS ON THE STACK
\$DB2D: SAVREG ;; SAVE REGISTERS
MOV 2(SP),R2 ;; PICKUP THE DATA POINTER
MOV #SDECVL,R0 ;; GET ADDRESS OF '\$SDECVL' STRING
MOV R0,2(SP) ;; PUT ADDRESS OF ASCIZ STRING ON STACK
MOV (R2)+,R1 ;; PICKUP THE BINARY NUMBER
MOV (R2)+,R2
MOV #10,,4\$;; SET UP TO DO 10 CONVERSIONS
MOV #STNPWR,R4 ;; ADDRESS OF TEN POWER
MOV #STNPWR+2,R5
1\$: CLR R3 ;; CLEAR PARTIAL
2\$: SUB (R4),R1 ;; SUBTRACT TEN POWER
SBC R2
SUB (R5),R2
BLT 3\$;; BR IF TEN POWER TOO LARGE
INC R3 ;; ADD 1 TO PARTIAL
BR 2\$;; LOOP
3\$: ADD (R4)+,R1 ;; RESTORE SUBTRACTED VALUE
ADC R2
ADD (R4)+,R2
CMP (R5)+,(R5)+ ;; MOVE TO NEXT TEN POWER
BIS #0,R3 ;; CHANGE PARTIAL TO ASCII
MOVB R3,(R0)+ ;; SAVE IT
DEC (PC)+ ;; DONE?
4\$: .WORD 0
BNE 1\$;; BR IF NO
CLRB (R0)+ ;; TERMINATOR
RESREG ;; RESTORE REGISTERS
RTS PC ;; RETURN
\$STNPWR: 145000 ;; 1.0E09
35632
160400 ;; 1.0E08
2765
113200 ;; 1.0E07
230
041100 ;; 1.0E06
17
103240 ;; 1.0E05
1
23420 ;; 1.0E04
0
1750 ;; 1.0E03
0

3493	014444	000144	144	::1.0E02
3494	014446	000000	0	
3495	014450	000012	12	::1.0E01
3496	014452	000000	0	
3497	014454	000001	1	::1.0E00
3498	014456	000000	0	
3499	014460	000014		\$DECVL: .BLKB 12. ::RESERVE STORAGE FOR ASCII STRING

::*****

:TYPE NUMERICAL ASCII STRING,RIGHT JUSTIFIED
:REPLACING LEADING ZEROS WITH SPACES.

:FIRST ADDRESS OF ASCII STRING MUST BE ON TOP OF THE STACK

3508	014474	010046		RTJUST:	MOV R0,-(SP)	:SAVE R0
3509	014476	016600	000004		MOV 4(SP),R0	:PICK UP ADDRESS OF ASCII STRING
3510	014502	010037	014534		MOV R0,3\$:SAVE ADDRESS FOR TYPE OUT
3511	014506	105710		1\$:	TSTB (R0)	:IS THIS THE TERMINATOR
3512	014510	001406			BEQ 2\$:IF YES TYPE IT OUT
3513	014512	122710	000060		CMPB #'0,(R0)	:IS IT A ZERO
3514	014513	001005			BNE 4\$:IF NO GO PRINT IT
3515	014520	112720	000040		MOVB #' ,(R0)+	:IF YES REPLACE IT WITH A SPACE
3516	014524	000770			BR 1\$:TEST NEXT CHAR.
3517	014526	112740	000060	2\$:	MOVB #'0,-(R0)	:STRING OFF ALL ZEROS,PUT BACK THE LAST ONE
3518	014532	104401		4\$:	TYPE	:TYPE THE STRING
3519	014534	000000		3\$:	OPEN	
3520	014536	012600			MOV (SP)+,R0	:RESTORE R0
3521	014540	012616			MOV (SP)+,(SP)	:RESTORE THE STACK
3522	014542	000207			RTS PC	:RETURN

:TYPES 16 BIT WORD IN DECIMAL

3526	014544	012546		SGLDEC:	MOV (R5)+,-(SP)	:PUT NUMBER TO BE TYPED ON STACK
3527	014546	004737	014244		JSR PC,@#SB2D	:CONVERT NUMBER TO DECIMAL
3528	014552	004737	014474		JSR PC,RTJUST	:TYPE THE DECIMAL NUMBER
3529	014556	000205			RTS R5	

3532	014560	047125	054105	042520	MUNXDD:	.ASCII 'UNEXPECTED D D MARK'
------	--------	--------	--------	--------	---------	------------------------------

3533	014566	052103	042105	042040		
3534	014574	042040	046440	051101		

3535	014602	000113				
------	--------	--------	--	--	--	--

3536						
------	--	--	--	--	--	--

3537	014604	020104	020104	040515	MDDMIS:	.ASCII 'D D MARK MISSING'
------	--------	--------	--------	--------	---------	---------------------------

3538	014612	045522	046440	051511		
3539	014620	044523	043516	000		

3540						
------	--	--	--	--	--	--

3541						
------	--	--	--	--	--	--

3542	014625	104	052101	026101	MDERHDR:	.ASCII 'DATA, NO STATUS ERROR'
------	--------	-----	--------	--------	----------	--------------------------------

3543	014632	047040	020117	052123		
------	--------	--------	--------	--------	--	--

3544	014640	052101	051525	042440		
------	--------	--------	--------	--------	--	--

3545	014646	051122	051117	000		
------	--------	--------	--------	-----	--	--

3546						
------	--	--	--	--	--	--

3547	014653	040	047117	052040	MTRK:	.ASCII '' ON TRACK''
------	--------	-----	--------	--------	-------	----------------------

3548	014660	040522	045503	000		
------	--------	--------	--------	-----	--	--

```

3549
3550
3551 014665 040 043040 047522 MPRES: .ASCIZ '' FROM TRACK''
3552 014672 020115 051124 041501
3553 014700 000113
3554
3555 014702 027440 051440 041505 MSECT: .ASCIZ '' / SECTOR''
3556 014710 047524 000122
3557
3558 014714 005015 041040 052131 MCOLMUN: .ASCIZ <15><12>'' BYTE BAD GOOD''<15><12>
3559 014722 020105 041040 042101
3560 014730 020040 047507 042117
3561 014736 005015 000
3562
3563 014741 015 041412 052501 DLOAD: .ASCII <15><12>''CAUTION - IF YOU DESIRE TO TEST UNIT 0''
3564 014746 044524 047117 026440
3565 014754 044440 020106 047531
3566 014762 020125 042504 044523
3567 014770 042522 052040 020117
3568 014776 042524 052123 052440
3569 015004 044516 020124 060
3570 015011 015 051012 050105 .ASCII <15><12>''REPLACE LOAD MEDIUM WITH A SCRATCH DISKETTE''
3571 015016 040514 042503 046040
3572 015024 040517 020104 042515
3573 015032 044504 046525 053440
3574 015040 052111 020110 020101
3575 015046 041523 040522 041524
3576 015054 020110 044504 045523
3577 015062 052105 042524
3578 015066 005015 044124 047105 .ASCIZ <15><12>''THEN PRESS CONTINUE''
3579 015074 050040 042522 051523
3580 015102 041440 047117 044524
3581 015110 052516 000105
3582
3583 015114 047516 042040 044522 MNODRY: .ASCIZ ''NO DRIVES READY''<15><12>
3584 015122 042526 020123 042522
3585 015130 042101 006531 000012
3586
3587 015136 020040 020040 020040 TAB: .ASCIZ <40><40><40><40><40><40>
3588 015144 000
3589
3590 015145 040 000040 DBLSP: .ASCIZ <40><40>
3591
3592 015150 005015 000 MCRLF: .ASCIZ <15><12>
3593
3594 015153 104 000 MEOP: .ASCIZ ''D''
3595
3596 015155 055 000 MEREOP: .ASCIZ ''-''
3597
3598 015157 007 000 MABELL: .ASCIZ <07>
3599
3600 015161 105 051122 051117 MERHEADER: .ASCIZ ''ERROR CONDITIONS ''
3601 015166 041440 047117 044504
3602 015174 044524 047117 020123
3603 015202 000040
3604

```

C 6

MAINDEC-11-CZRXA-F MACY11 30(1046) 12-APR-82 14:30 PAGE 72
 CZRXAF.P11 09-APR-82 15:11 DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

3605	015204	047125	052111	030040	MUNIT0:	.ASCIZ 'UNIT 0 ''
3606	015212	000040				
3607						
3608	015214	047125	052111	030440	MUNIT1:	.ASCIZ 'UNIT 1 ''
3609	015222	000040				
3610						
3611	015224	040510	020123	042502	MDELET:	.ASCIZ 'HAS BEEN DELETED.'
3612	015232	047105	042040	046105		
3613	015240	052105	042105	000056		
3614						
3615	015246	044440	052116	053040	MINTVEC:	.ASCIZ '' INT VECTOR = ''
3616	015254	041505	047524	020122		
3617	015262	020075	000			
3618						
3619	015265	122	041530	020123	MRXCS:	.ASCIZ 'RXCS = ''
3620	015272	020075	000			
3621						
3622	015275	040	054122	041104	MRXDS.	.ASCIZ '' RXDB = ''
3623	015302	036440	000040			
3624						
3625	015306	052123	052101	051525	MASTAT:	.ASCIZ ''STATUS A = ''
3626	015314	040440	036440	000040		
3627						
3628	015322	052123	052101	051525	MBSTAT:	.ASCIZ ''STATUS B = ''
3629	015330	041040	036440	000040		
3630						
3631	015336	005015	000012		DBLLF:	.ASCIZ <15><12><12>
3632						
3633	015342	036520	000040		MPAT:	.ASCIZ 'P= ''
3634						
3635	015346	036524	000040		MTEST:	.ASCIZ 'T= ''
3636						
3637	015352	036523	000040		MSEQ:	.ASCIZ 'S= ''
3638						
3639	015356	047516	044440	052116	MINTER:	.ASCIZ 'NO INTERRUPT AT DONE ERROR''
3640	015364	051105	050125	020124		
3641	015372	052101	042040	047117		
3642	015400	020105	051105	047522		
3643	015406	000122				
3644						
3645	015410	047125	047113	053517	MUKNINT:	.ASCIZ 'UNKNOWN INTERRUPT''
3646	015416	020116	047111	042524		
3647	015424	052522	052120	000		
3648						
3649	015431	124	052117	046101	MERCT:	.ASCIZ ''TOTAL READ CHECK ERRORS = ''
3650	015436	051040	040505	020104		
3651	015444	044103	041505	020113		
3652	015452	051105	047522	051522		
3653	015460	036440	000040			
3654						
3655	015464	044506	046114	052502	MFIL:	.ASCIZ ''FILLBUFFER ''
3656	015472	043106	051105	000040		
3657						
3658	015500	046505	052120	041131	MEMPTY:	.ASCIZ 'EMPTYBUFFER ''
3659	015506	043125	042506	020122		
3660	015514	000				

SEQ 0067

3661						
3662	015515	104	044522	042526	MICON:	.ASCIZ 'DRIVE(S) ''
3663	015522	051450	020051	000040		
3664						
3665	015530	042524	052123	044040	MHUNG:	.ASCIZ ''TEST HUNG' '<15><12>
3666	015536	047125	006507	000012		
3667						
3668	015544	047516	051516	040524	MNONSTD:	.ASCIZ 'NONSTANDARD TRACK / SECTOR LIMITS''
3669	015552	042116	051101	020104		
3670	015560	051124	041501	020113		
3671	015566	020057	042523	052103		
3672	015574	051117	046040	046511		
3673	015602	052111	000123			
3674						
3675	015606	042117	000075		MOD:	.ASCIZ 'OD=''
3676						
3677	015612	042111	000075		MID:	.ASCIZ ''ID=''
3678						
3679	015616	044506	051522	036524	MFIRST:	.ASCIZ ''FIRST=''
3680	015624	000				
3681						
3682	015625	114	051501	036524	MLAST:	.ASCIZ ''LAST=''
3683	015632	000				
3684						
3685	015633	111	044516	027124	MINIT1:	.ASCIZ ''INIT.DONE NOT SET ERROR' '<15><12>
3686	015640	047504	042516	047040		
3687	015646	052117	051440	052105		
3688	015654	042440	051122	051117		
3689	015662	005015	000			
3690						
3691	015665	111	044516	027124	MINIT2:	.ASCIZ ''INIT.DONE SET ERROR' ' <15><12>
3692	015672	047504	042516	051440		
3693	015700	052105	042440	051122		
3694	015706	051117	005015	000		
3695						
3696	015713	104	042040	042440	MDDER:	.ASCIZ 'D D ERROR''
3697	015720	051122	051117	000		
3698						
3699	015725	122	041505	053117	MREC:	.ASCIZ ''RECOVERABLE ''
3700	015732	051105	041101	042514		
3701	015740	000040				
3702						
3703	015742	051103	020103	051105	MBADCRC:	.ASCIZ ''CRC ERROR NO DATA ERROR''
3704	015750	047522	020122	047516		
3705	015756	042040	052101	020101		
3706	015764	051105	047522	000122		
3707						
3708	015772	042522	042101	000040	MREAD:	.ASCIZ ''READ ''
3709						
3710	016000	040510	052114	032040	MHALT4:	.ASCIZ ''HALT 4' ' <15><12>
3711	016006	005015	000			
3712						
3713	016011	110	046101	020124	MHALT3:	.ASCIZ ''HALT 3' ' <15><12>
3714	016016	006463	000012			
3715						
3716	016022	040510	052114	030440	MHALT11:	.ASCIZ ''HALT 11' ' <15><12>

3717	016030	0C6461	000012		
3718					
3719	016034	040504	040524	041440	MCRC: .ASCIZ 'DATA CRC ERROR'
3720	016042	041522	042440	051122	
3721	016050	051117	000		
3722					
3723					
3724	016053	040	047125	042522	MUNREC: .ASCIZ '' UNRECOVERABLE ''
3725	016060	047503	042526	040522	
3726	016066	046102	020105	000	
3727					
3728	016073	123	042505	020113	MSEEK: .ASCIZ ''SEEK ERROR''
3729	016100	051105	047522	000122	
3730					
3731	016106	051127	052111	020105	MWRITE: .ASCIZ 'WRITE ''
3732	016114	000			
3733					
3734	016115	120	051101	052111	MPAR: .ASCIZ 'PARITY ERROR'
3735	016122	020131	051105	047522	
3736	016130	000122			
3737					
3738	016132	051105	047522	020122	MNOFLAG: .ASCIZ 'ERROR FLAG ERROR'
3739	016140	046106	043501	042440	
3740	016146	051122	051117	000	
3741					
3742	016153	122	030530	020061	MDUMP: .ASCIZ 'RX11 DUMP''
3743	016160	052504	050115	000	
3744					
3745	016165	105	051122	051117	MERS: .ASCIZ 'ERRORS''
3746	016172	000123			
3747					
3748	016174	044123	051117	020124	MSHORT: .ASCIZ ''SHORT DUMP''
3749	016202	052504	050115	000	
3750					
3751	016207	122	051505	040524	MRESTART: .ASCIZ 'RESTARTS = ''
3752	016214	052122	020123	020075	
3753	016222	000			
3754					
3755	016223	120	051501	020123	MNOPAS: .ASCIZ 'PASS ABORTED''
3756	016230	041101	051117	042524	
3757	016236	000104			
3758					
3759	016240	040520	051523	051505	MPASS: .ASCIZ 'PASSES = ''
3760	016246	036440	000040		
3761					
3762	016252	051127	052111	042524	MWRTEN: .ASCIZ 'WRITTEN ''
3763	016260	020116	000		
3764					
3765	016263	057	000040		MSLASH: .ASCIZ ''/ ''
3766					
3767	016266	000040			SPACE: .ASCIZ <40>
3768					
3769	016270	051105	047522	051522	MCODES: .ASCIZ 'ERRORS PER ERROR CODES''<15><12>
3770	016276	050040	051105	042440	
3771	016304	051122	051117	041440	
3772	016312	042117	051505	005015	

MAINDEC-11-CZRXA-F
CZRRAF.P11 09-APR-82

MACY11 30(1046)
15:11

12-APR-82 14:30 PAGE 75
DOUBLE LENGTH BINARY TO DECIMAL ASCII CONVERT ROUTINE

SEQ 0070

3773	016320	000				
3774						
3775	016321	124	040522	045503	MTKLOG:	.ASCIZ 'TRACK ACCESSED MOVED TO UNIT 0 UNIT 1 ERRORS'<15><12>
3776	016326	020040	041501	042503		
3777	016334	051523	042105	020040		
3778	016342	046440	053117	042105		
3779	016350	052040	020117	020040		
3780	016356	052440	044516	020124		
3781	016364	020060	052440	044516		
3782	016372	020124	020061	051105		
3783	016400	047522	051522	005015		
3784	016406	000				
3785						
3786	016407	050	047516	042516	MNONE:	.ASCIZ '(NONE)'<15><12>
3787	016414	006451	000012			
3788						
3789	016420	047507	042117	000	MGOOD:	.ASCIZ 'GOOD'
3790						
3791	016425	040	041440	042510	MSUM:	.ASCIZ ' CHECK SUM '
3792	016432	045503	051440	046525		
3793	016440	000040				
3794						
3795	016442	005015	054122	030461	MRX11:	.ASCIZ <15><12>'RX11 / RXV11'
3796	016450	027440	051040	053130		
3797	016456	030461	000			
3798						
3799	016461	015	052012	040522	OD2BIG:	.ASCII <15><12> 'TRACK LIMITS SELECTED OUT OF RANGE,'
3800	016466	045503	046040	046511		
3801	016474	052111	020123	042523		
3802	016502	042514	052103	042105		
3803	016510	047440	052125	047440		
3804	016516	020106	040522	043516		
3805	016524	026105				
3806	016526	051525	047111	020107		.ASCIZ 'USING OD=0, ID=114'<15><12>
3807	016534	042117	030075	020054		
3808	016542	042111	030475	032061		
3809	016550	005015	000			
3810						
3811	016553	015	051412	041505	S2BIG:	.ASCII <15><12> 'SECTOR LIMITS SELECTED OUT OF RANGE,'
3812	016560	047524	020122	044514		
3813	016566	044515	051524	051440		
3814	016574	046105	041505	042524		
3815	016602	020104	052517	020124		
3816	016610	043117	051040	047101		
3817	016616	042507	054			
3818	016621	125	044523	043516		.ASCIZ 'USING FIRST=1, LAST=32'<15><12>
3819	016626	043040	051111	052123		
3820	016634	030475	020054	040514		
3821	016642	052123	031475	006462		
3822	016650	000012				
3823						
3824	016652	005015	040515	047111	MREV:	.ASCIZ <15><12> 'MAINDEC-11-CZRXA-F' <15><12>
3825	016660	042504	026503	030461		
3826	016666	041455	051132	040530		
3827	016674	043055	005015	000		
3828						

3829 016702
 3830
 3831
 3832
 3833
 3834
 3835
 3836 016702 000200
 3837
 3838
 3839 017102 000012
 3840 017104 000012
 3841 017106 000012
 3842 017110 000012
 3843 017112 000012
 3844 017114 000012
 3845 017116 000012
 3846 017120 000012
 3847
 3848
 3849
 3850 017122 000000
 3851 017124 000000
 3852 017126 000000
 3853 017130 000000
 3854 017132 000000
 3855
 3856 017134 000000
 3857 017136 000000
 3858 017140 000000
 3859 017142 000000
 3860 017144 000000
 3861 017146 000000
 3862 017150 000000
 3863 017152 000000
 3864 017154 000000
 3865 017156 000000
 3866 017160 000000
 3867 017162 000000
 3868 017164 000000
 3869 017166 000000
 3870 017170 000000
 3871 017172 000000
 3872 017174 000000
 3873 017176 000000
 3874 017200 000000
 3875 017202 000000
 3876 017204 000000
 3877 017206 000000
 3878 017210 000000
 3879 017212 000000
 3880 017214 000000
 3881 017216 000000
 3882 017220 000000
 3883 017222 000000
 3884 017224 000000

.EVEN
 ;:*****
 ;THE FOLLOWING LOCATIONS ARE USED FOR DATA STORAGE,RETRY COUNTERS
 ;ACCESS COUNTERS ETC.
 BUFADR: .BLKB 200
 ;RETRY COUNTERS
 RDRETRY: 10. ;FOR SETUP PURPOSE RDRETRY MUST BE FIRST ON RETRY LIST
 WTRETRY: 10.
 DDRETRY: 10.
 DATARETRY: 10.
 P2RETRY: 10.
 PRETRY: 10.
 CRETRY: 10.
 SRETRY: 10. ;FOR SETUP PURPOSE SRETRY MUST BE LAST ON RETRY LIST
 ;GENERAL ERROR COUNTERS
 PARLOG: 0 ;FOR SETUP PURPOSE PARLOG MUST BE FIRST ON COUNTERS LIST
 HPARLOG: 0
 NOERLOG: 0
 UKNINT: 0
 INTER: 0
 ;DRIVE RELATED ERROR COUNTERS
 ZSEKLOG: 0 ;'Z' LOGS ARE FOR DRIVE UNIT 'Z'ERO
 SEKLOG: 0 ;NON'Z' LOGS ARE FOR DRIVE UNIT 1
 ZCRCLOG: 0
 CRCLOG: 0
 ZCRCBAD: 0
 CRCBAD: 0
 ZRDLOG: 0
 RDLOG: 0
 ZWRTLOG: 0
 WRTLOG: 0
 ZDATALOG: 0
 DATALOG: 0
 ZDDMIS: 0
 DDMIS: 0
 ZUNXDD: 0
 UNXDD: 0
 ZHSEKLOG: 0
 HSEKLOG: 0
 ZHCRCLOG: 0
 HCRCLOG: 0
 ZHCRCBAD: 0
 HCRCBAD: 0
 ZHRDLOG: 0
 HRDLOG: 0
 ZHWRTLOG: 0
 HWRTLOG: 0
 ZHDATALOG: 0
 HDATALOG: 0
 ZHDDLOG: 0

*** NOTE: ***
 ;ERROR CODES MUST NOT BE CHANGED
 ;FROM THIS ORDER. ERROR DUMP
 ;ASSUMES TAGS OF LOGS ARE IN ORDER SHOWN.


```

3885 017226 0C0000 HDDLOG: 0
3886
3887 :ERROR CODES
3888 017230 000021 ERCODE: .BLKW 17.
3889
3890 :THE FOLLOWING 2 BLOCKS OF WORDS ARE TRACK ACCESS COUNTER AND
3891 :HEAD MOVED TO TRACK COUNTERS. THEY ARE DOUBLE PRECISION
3892 :COUNTERS IN THE FOLLOWING FORMATT:
3893
3894 :LOC.X LOW ORDER WORD TRACK 0
3895 :LOC.X+2 HIGH ORDER WORD TRACK 0
3896 :LOC.X+4 LOW ORDER WORD TRACK 1
3897 :LOC.X+6 HIGH ORDER WORD TRACK 1
3898 :ETC.
3899
3900 :TOTAL ACCESS / TRACK
3901 017272 000232 TKACC: .BLKW 232
3902
3903 :TOTAL HEAD MOVEMENT TO TRACK
3904 017756 000232 HDMOVE: .BLKW 232
3905
3906 :THE FOLLOWING 2 BLOCKS OF COUNTERS ARE SINGLE PRECISION
3907
3908 :ERROR PER TRACK ON UNIT 0
3909 020442 000115 U0TRK: .BLKW 115
3910
3911 :ERROR PER TRACK ON UNIT 1
3912 020674 000115 U1TRK: .BLKW 115
3913
3914 :TOTAL WRITE AND READ FUNCTIONS DOUBLE PRECISION COUNTERS
3915 021126 000000 000000 WTCNTR: .WORD 0,0
3916 021132 000000 000000 RDCNTR: .WORD 0,0
3917
3918 :RESTART AND PASSES COMPLETED COUNTERS
3919 021136 000000 RESTCNTR: 0
3920 021140 000000 PASCNTR: 0 ;FOR SETUP PURPOSE PASCNTR MUST BE PLACED LAST
3921
3922 021142 032737 000020 010704 U1LOG: BIT #BIT4,UNITSEL ;TEST FOR UNIT IN OPERATION
3923 021150 001402 BEQ 1$ ;IF BIT 4 IS 0 RETURN AND INC. UNIT 0 COUNTERS
3924 021152 062703 000002 ADD #2,R3 ;ADJUST THE CONTENTS OF R3 FOR
3925 ;THE ADDRESS OF UNIT 1 ERROR LOG COUNTERS.
3926 021156 000207 1$: RTS PC
3927
3928 :STATISTICAL ERROR REPORT ROUTEEN
3929 :PRINTS ALL ERROR LOGS AND OPERATING CONDITIONS
3930
3931 ;:*****
3932
3933 021160 012706 001200 ERDUMP: MOV #STACK,SP
3934 021164 104401 015336 TYPE ,DBLLF
3935 021170 005737 022612 TST SHORTRPT ;IS SHORT REPORT REQUESTED
3936 021174 001405 BEQ 1$ ;NO,PRINT LONG REPORT
3937 021176 104401 015136 TYPE ,TAB
3938 021202 104401 016174 TYPE ,MSHORT ;YES,TYPE SHORT REPOR HEADER
3939 021206 000404 BR 2$
3940 021210 104401 015136 1$: TYPE ,TAB
  
```

3941	021214	104401	016153		TYPE ,MDUMP	:TYPE LONG REPORT HEADER
3942	021220	104401	016652	2\$:	TYPE ,MREV	:PRINT NAME AND REVISION OF PROGRAM
3943	021224	004737	002452		JSR PC,ICOND	:TYPE OUT SELECTED DRIVES AND TESTS
3944	021230	005737	021136		TST RESTCNTR	:HAVE THERE BEEN ANY RESTARTS
3945	021234	001412			BEQ 3\$:NO
3946	021236	104401	016207		TYPE ,MRESTART	:YES,PRINT OUT HOW MANY
3947	021242	013737	021136	021254	MOV RESTCNTR,12\$	
3948	021250	004537	014544		JSR R5,SGLDEC	
3949	021254	000000		12\$:	OPEN	
3950	021256	104401	015336		TYPE ,DBLLF	
3951	021262	005737	021140	3\$:	TST PASCNTR	:HAVE ANY PASSES BEEN COMPLETED
3952	021266	001005			BNE 4\$:YES,PRINT OUT HOW MANY
3953	021270	104401	016223		TYPE ,MNOPAS	:NO,TYPE PASS ABORTED
3954	021274	104401	015336		TYPE ,DBLLF	
3955	021300	000412			BR 5\$	
3956	021302	104401	016240	4\$:	TYPE ,MPASS	:TYPE OUT NUMBER OF PASSES
3957	021306	013737	021140	021320	MOV PASCNTR,7\$	
3958	021314	004537	014544		JSR R5,SGLDEC	
3959	021320	000000		7\$:	OPEN	
3960	021322	104401	015150		TYPE ,MCRLF	
3961	021326	005737	022612	5\$:	TST SHORTRPT	:IS THIS A SHORT REPORT
3962	021332	001030			BNE AERRS	:YES,DON'T PRINT WRITE/READ TOTALS
3963						
3964					:PRINTS TOTAL NUMBER OF SECTORS WRITTEN AND/OR	
3965					:READ FOR THE DURATION OF RUN TIME.	
3966	021334	104401	016252		TYPE ,MWRTEN	:PRINT WRT/RD MESSAGE
3967	021340	104401	016263		TYPE ,MSLASH	
3968	021344	104401	015772		TYPE ,MREAD	
3969	021350	104401	015145		TYPE ,DBLSP	
3970	021354	012746	021126		MOV #WTCNTR,-(SP)	:ADDRESS OF TOTAL WRITES ON STACK
3971	021360	004737	014300		JSR PC,@#\$DB2D	:TYPE TOTAL WRITES
3972	021364	004737	014474		JSR PC,RTJUST	
3973	021370	104401	016263		TYPE ,MSLASH	
3974	021374	012746	021132		MOV #RDCNTR,-(SP)	:ADDRESS OF READS ON STACK
3975	021400	004737	014300		JSR PC,@#\$DB2D	:TYPE TOTAL READS
3976	021404	004737	014474		JSR PC,RTJUST	
3977	021410	104401	015336		TYPE ,DBLLF	
3978						
3979					:TYPE OUT MICRO CPU RELATED ERRORS	
3980	021414	104401	015150	AERRS:	TYPE ,MCRLF	
3981	021420	005737	017122		TST PARLOG	:WERE THERE ANY PARITY ERRORS
3982	021424	001435			BEQ 1\$	
3983	021426	013737	017122	021440	MOV PARLOG,20\$:YES,TYPE OUT THE NUMBER
3984	021434	004537	014544		JSR R5,SGLDEC	
3985	021440	000000		20\$:	OPEN	
3986	021442	104401	015145		TYPE ,DBLSP	
3987	021446	104401	016115		TYPE ,MPAR	
3988	021452	104401	015150		TYPE ,MCRLF	
3989	021456	005737	017124		TST HPARLOG	:WERE THERE ANY HARD PARITY ERRORS
3990	021462	001416			BEQ 1\$	
3991	021464	013737	017124	021476	MOV HPARLOG,21\$:YES,TYPE OUT THE COUNT
3992	021472	004537	014544		JSR R5,SGLDEC	
3993	021476	000000		21\$:	OPEN	
3994	021500	104401	015145		TYPE ,DBLSP	
3995	021504	104401	016053		TYPE ,MUNREC	
3996	021510	104401	016115		TYPE ,MPAR	

3997	021514	104401	015150		TYPE ,MCRLF	
3998	021520	005737	017126	1\$:	TST NOERLOG	;ANY STATUS ERROR FLAG ERRORS
3999	021524	001414			BEQ 3\$;IF NONE CHECK NEXT ERROR
4000	021526	013737	017126	021540	MOV NOERLOG,24\$;YES,TYPE OUT COUNT
4001	021534	004537	014544		JSR R5,SGLDEC	
4002	021540	000000		24\$:	OPEN	
4003	021542	104401	015145		TYPE ,DBLSP	
4004	021546	104401	016132		TYPE ,MNOFLAG	
4005	021552	104401	015150		TYPE ,MCRLF	
4006	021556	005737	017132	3\$:	TST INTER	;ANY NO INTERRUPT ON DONE ERRORS
4007	021562	001414			BEQ 4\$;IF NONE CHECK NEXT ERROR GROUP
4008	021564	013737	017132	021576	MOV INTER,25\$;YES,PRINT OUT NUMBER
4009	021572	004537	014544		JSR R5,SGLDEC	
4010	021576	000000		25\$:	OPEN	
4011	021600	104401	015145		TYPE ,DBLSP	
4012	021604	104401	015356		TYPE ,MINTER	
4013	021610	104401	015150		TYPE ,MCRLF	
4014					;TYPES DRIVE RELATED ERRORS	
4015	021614	104401	015336	4\$:	TYPE ,DBLLF	
4016	021620	104401	015204		TYPE ,MUNIT0	;TYPE OUT COLUMN HEADINGS
4017	021624	104401	015214		TYPE ,MUNIT1	
4018	021630	104401	016266		TYPE ,SPACE	
4019	021634	104401	016165		TYPE ,MERS	
4020	021640	104401	015150		TYPE ,MCRLF	
4021	021644	005002			CLR R2	;R2 USED AS AN ERROR PRINTED FLAG
4022	021646	013746	022614		MOV ZERO,-(SP)	;PUT ADDR OF ZERO MARK ON STACK
4023	021652	012746	014560		MOV #MUNXDD,-(SP)	;PUT ADDRESSES OF ALL RECOVERABLE
4024	021656	012746	014604		MOV #MDDMIS,-(SP)	;ERROR MESSAGES ON THE STACK
4025	021662	012746	014625		MOV #MDERHDR,-(SP)	
4026	021666	012746	016106		MOV #MWRITE,-(SP)	
4027	021672	012746	015772		MOV #MREAD,-(SP)	
4028	021676	012746	015747		MOV #MBADCRC,-(SP)	
4029	021702	012746	016034		MOV #MCRC,-(SP)	
4030	021706	012746	016073		MOV #MSEEK,-(SP)	
4031	021712	012701	017134		MOV #ZSEKLOG,R1	;PUT FIRST ERROR CUONTER IN REGISTER
4032	021716	005716		7\$:	TST (SP)	;GET ADDRESS OF MESSAGE
4033	021720	001436			BEQ 5\$;IF 0 FINISHED SOFT ERRS DO HARD ERRS
4034	021722	005721			TST (R1)+	;IS DRIVE 0 COUNTER CLEAR
4035	021724	001005			BNE 6\$;NO,GO PRINT ERROR COUNTER
4036	021726	005711			TST (R1)	;IS DRIVE 1 COUNTER CLEAR
4037	021730	001003			BNE 6\$;NO,GO PRINT ERROR COUNTER
4038	021732	005721			TST (R1)+	;ADJUST R1 FOR NEXT UNIT 0 COUNTER
4039	021734	005726			TST (SP)+	;ADJUST STACK FOR NEXT ADDRESS
4040	021736	000767			BR 7\$;GO TEST NEXT PAIR OF COUNTERS
4041	021740	005202		6\$:	INC R2	;MAKE ERROR PRINTED FLAG NON ZERO
4042	021742	014137	021752		MOV -(R1),30\$;TYPE CONTENTS OF UNIT 0 COUNTER
4043	021746	004537	014544		JSR R5,SGLDEC	
4044	021752	000000		30\$:	OPEN	
4045	021754	104401	015145		TYPE ,DBLSP	
4046	021760	005721			TST (R1)+	;ADJUST R1 FOR UNIT 1 COUNTER
4047	021762	012137	021772		MOV (R1)+,31\$;TYPE COUNTER,GET ADDR OF NEXT IN R1
4048	021766	004537	014544		JSR R5,SGLDEC	
4049	021772	000000		31\$:	OPEN	
4050	021774	104401	015145		TYPE ,DBLSP	
4051	022000	012637	022006		MOV (SP)+,13\$	
4052	022004	104401			TYPE	;TYPE ERROR MESSAGE FROM STACK

```

4053 022006 0C0000      13$:      OPEN
4054 022010 104401 015150      TYPE ,MCRLF
4055 022014 000740      BR 7$      ;GET NEXT COUNTER
4056 022016 104401 015150      5$:      TYPE ,MCRLF
4057 022022 013746 022614      MOV ZERO,-(SP)      ;PUT ADDR OF ZERO MARK ON THE STACK
4058 022026 012746 015713      MOV #MDDER,-(SP)    ;PUT ADDRESS OF ALL HARD ERROR
4059 022032 012746 014625      MOV #MDERHDR,-(SP) ;MESSAGES ON THE STACK
4060 022036 012746 016106      MOV #MWRITE,-(SP)
4061 022042 012746 015772      MOV #MREAD,-(SP)
4062 022046 012746 015742      MOV #MBADCRC,-(SP)
4063 022052 012746 016034      MOV #MCRC,-(SP)
4064 022056 012746 016073      MOV #MSEEK,-(SP)
4065 022062 012701 017174      MOV #ZHSEKLOG,R1   ;PUT FIRST HARD ERROR COUNTER IN R1
4066 022066 005716      12$:      TST (SP)           ;GET ADDRESS OF MESSAGE FROM STACK
4067 022070 001437      BEQ BERRS         ;IF 0 FINISHED ERROR LOGS
4068 022072 005721      TST (R1)+        ;IS DRIVE 0 COUNTER CLEAR
4069 022074 001005      BNE 11$          ;NO,PRINT ERROR COUNTER AND MESSAGE
4070 022076 005711      TST (R1)         ;IS DRIVE 1 COUNTER CLEAR
4071 022100 001003      BNE 11$          ;NO,GO PRINT THE ERROR
4072 022102 005721      TST (R1)+        ;ADJUST R1 FOR NEXT UNIT 0 LOG
4073 022104 005726      TST (SP)+        ;ADJUST STACK FOR NEXT ADDRESS
4074 022106 000767      BR 12$
4075 022110 014137 022120      11$:      MOV -(R1),32$    ;TYPE CONTENTS OF UNIT 0 COUNTER
4076 022114 004537 014544      JSR R5,SGLDEC
4077 022120 000000      32$:      OPEN
4078 022122 104401 015145      TYPE ,DBLSP
4079 022126 005721      TST (R1)+        ;ADJUST R1 FOR UNIT 1 COUNTER
4080 022130 012137 022140      MOV (R1)+,33$    ;TYPE UNIT 1 LOG
4081 022134 004537 014544      JSR R5,SGLDEC
4082 022140 000000      33$:      OPEN
4083 022142 104401 015145      TYPE ,DBLSP
4084 022146 104401 016053      TYPE ,MUNREC     ;TYPE UNRECOVERABLE
4085 022152 012637 022160      MOV (SP)+,14$
4086 022156 104401      TYPE            ;ERROR MESSAGE FROM STACK
4087 022160 000000      14$:      OPEN
4088 022162 104401 015150      TYPE ,MCRLF
4089 022166 000737      BR 12$          ;GET NEXT LOG
4090      ;PRINTS ERRORS PER ERROR CODES
4091 022170 005702      BERRS:      TST R2           ;WAS AN ERROR PRINTED IN THE LAST GROUP
4092 022172 001004      BNE 5$         ;IF R2 IS NONZERO BRANCH
4093 022174 104401 015136      TYPE ,TAB
4094 022200 104401 016407      TYPE ,MNONE     ;TYPE NONE
4095 022204 005002      5$:      CLR R2         ;CLEAR ERROR PRINTED FLAG AGAIN
4096 022206 104401 015336      TYPE ,DBLLF
4097 022212 104401 016270      TYPE ,MCODES    ;TYPE ERROR PER ERROR CODE
4098 022216 012700 000001      MOV #1,R0       ;SET UP CODE COUNTER
4099 022222 012701 017230      MOV #ERCODE,R1  ;GET ADDR OF FIRST ERROR CODE
4100 022226 020027 000022      2$:      CMP R0,#22     ;IS THE LAST CODE PRINTED
4101 022232 001001      BNE 3$         ;NO,TEST THE NEXT ONE
4102 022234 000427      BR TYPTRK      ;YES,GO PRINT TRACK INFOMATION
4103 022236 005721      3$:      TST (R1)+      ;TEST FOR 0 WORD,GET NEXT ADDRESS
4104 022240 001002      BNE 1$         ;IF NOT 0 TYPE COUNT AND CODE NUMBER
4105 022242 005200      INC R0         ;INC CODE COUNTER
4106 022244 000770      BR 2$
4107 022246 014137 022256      1$:      MOV -(R1),4$    ;PUT CONTENTS OF COUNTER IN OPEN
4108 022252 004537 014544      JSR R5,SGLDEC   ;FOR TYPE OUT

```

4109	022256	000000		4\$:	OPEN	
4110	022260	104401	015136		TYPE ,TAB	
4111						
4112	022264	010002			MOV R0,R2	;PUT CODE NUMBER IN REGISTER TO
4113	022266	006302			ASL R2	;TYPE IT OUT IN THE FROM OF ERROR CODE
4114	022270	006302			ASL R2	
4115	022272	006302			ASL R2	
4116	022274	010246			MOV R2,-(SP)	::SAVE R2 FOR TYPEOUT
4117	022276	104403			TYPOS	::GO TYPE--OCTAL ASCII
4118	022300	004			.BYTE 4	::TYPE 4 DIGIT(S)
4119	022301	001			.BYTE 1	::TYPE LEADING ZEROS
4120	022302	104401	015150		TYPE ,MCRLF	
4121	022306	005200			INC R0	;SET UP FOR NEXT COUNTER
4122	022310	005721			TST (R1)+	;GET NEXT ERROR COUNTER
4123	022312	000745			BR 2\$	
4124						
4125	022314	005702		TYPTRK:	;PRINTS TRACK ACCESS,HEAD MOVEMENT,AND ERRORS PER TRACK	
4126	022316	001004			TST R2	;WAS AN ERROR PRINTED IN LAST GROUP
4127	022320	104401	015136		BNE 5\$;IF PRINTED ERROR FLAG NONZERO BRANCH
4128	022324	104401	016407		TYPE ,TAB	
4129	022330	104401	015336		TYPE ,MNONE	;TYPE NONE
4130	022334	005737	022612	5\$:	TYPE ,DBLLF	
4131	022340	001121			TST SHORTRPT	;IF SHORT REPORT DON'T TYPE TRACK ACCESS OR ERRO
4132	022342	104401	016321		BNE 6\$	
4133	022346	005037	022426		TYPE ,MTKLOG	;TYPE FORMAT OF ERROR PRINT OUT
4134	022352	012704	017266		CLR 2\$;TRACK ADDRESS COUNTER
4135	022356	012703	017752		MOV #TKACC-4,R4	;SET UP ADDRESS OF TRACK ACCESSED
4136	022362	012702	020442		MOV #HDMOVE-4,R3	;ADDRESS OF HEAD MOVEMENT COUNTER
4137	022366	012701	020674		MOV #U0TRK,R2	;UNIT 0 ERROR PER TRACK COUNTER
4138	022372	062704	000004		MOV #U1TRK,R1	;UNIT 1 ERROR PER TRACK COUNTER
4139	022376	005714		1\$:	ADD #4,R4	;ADJUST FOR ADDRESS OF NEXT COUNTER
4140	022400	001010			TST (R4)	;HAS THIS TRACK BEEN ACCESSED
4141	022402	005764	000002		BNE 10\$;YES,PRINT OUT THE COUNTERS
4142	022406	001005			TST 2(R4)	;TEST UPPER WORD OF ACCESS COUNTER
4143	022410	062703	000004		BNE 10\$	
4144	022414	005722			ADD #4,R3	;THIS TRACK HAS NOT BEEN USED. ADJUST
4145	022416	005721			TST(R2)+	;REGISTERS TO POINT TO NEXT TRACK COUNTERS
4146	022420	000463			TST (R1)+	
4147	022422	004537	014544	10\$:	BR 12\$;TEST FOR LAST TRACK
4148	022426	000000		2\$:	JSR R5,SGLDEC	;TYPE TRACK NUMBER
4149	022430	104401	016266		OPEN	;THIS LOCATION USED AS TRACK ADDR COUNTER
4150	022434	010446			TYPE ,SPACE	
4151	022436	004737	014300		MOV R4,-(SP)	;TYPE TRACK ACCESSED COUNT
4152	022442	004737	014474		JSR PC,@#\$DB2D	
4153	022446	104401	016266		JSR PC,RTJUST	
4154	022452	062703	000004		TYPE ,SPACE	
4155	022456	010346			ADD #4,R3	
4156	022460	004737	014300		MOV R3,-(SP)	;TYPE HEAD MOVED TO COUNT
4157	022464	004737	014474		JSR PC,@#\$DB2D	
4158	022470	104401	015145		JSR PC,RTJUST	
4159	022474	104401	016266		TYPE ,DBLSP	
4160	022500	005712			TYPE ,SPACE	
4161	022502	001421			TST (R2)	;ARE THERE ANY UNIT 0 ERRORS
4162	022504	012237	022514		BEQ 20\$;NO,CHECK UNIT 1
4163	022510	004537	014544		MOV (R2)+,3\$;TYPE UNIT 0 ERRORS
4164	022514	000000		3\$:	JSR R5,SGLDEC	
					OPEN	

4165	022516	005711			TST (R1)	;ARE THERE ANY UNIT 1 ERRORS
4166	022520	001420			BEQ 7\$;NO,GO ADJUST POINTERS
4167	022522	104401	016266		TYPE ,SPACE	
4168	022526	104401	015145	22\$:	TYPE ,DBLSP	
4169	022532	012137	022542		MOV (R1)+,4\$;TYPE UNIT 1 ERRORS
4170	022536	004537	014544		JSR R5,SGLDEC	
4171	022542	0000C0		4\$:	OPEN	
4172	022544	000407			BR 11\$;GO TO NEXT TRACK
4173						
4174	022546	005722		20\$:	TST (R2)+	;NO ERRORS,ADJUST REG.FOR NEXT COUNTER
4175	022550	005711			TST (R1)	;ARE THERE ANY UNIT 1 ERRORS
4176	022552	001403			BEQ 7\$;NO,GO TO NEXT TRACK
4177	022554	104401	015136		TYPE ,TAB	;YES,TYPE TAB FOR PLACEMENT
4178	022560	000762			BR 22\$;GO TYPE THE NUMBER
4179	022562	005721		7\$:	TST (R1)+	;NO ERRORS,ADJUST REG. FOR NEXT COUNTER
4180	022564	104401	015150	11\$:	TYPE ,MCRLF	
4181	022570	005237	022426	12\$:	INC 2\$	
4182	022574	023727	022426	000115	CMP 2\$,#115	;HAVE ALL TRACK ERRORS BEEN TYPED
4183	022602	001273			BNE 1\$;NO,TYPE THE NEXT ONE
4184	022604	005037	022612	6\$:	CLR SHORTRPT	;SET UP FOR LONG REPORT
4185	022610	000000		HLT17:	HALT	;YES DONE
4186						
4187	022612	000000		SHORTRPT:	0	
4188	022614	000000		ZERO:	0	
4189						
4190		000001			.END	

COMMEN	1008#														
ENDCOM	1008#														
ERROR	902#														
ESCAPE	1008#														
GETPRI	1008#														
GETSWR	862#	1008#	1304												
MULT	1008#														
NEWTST	1008#														
POP	1008#	3182	3400	3401											
PUSH	1008#	3162	3381	3387											
REPORT	1008#														
SCOPE	903#														
SETPRI	1008#														
SETTRA	3355#	3364	3365	3366	3368	3370	3371	3372	3373	3374					
SETUP	1008#														
SKIP	1008#														
SLASH	1008#														
SPACE	1008#														
STARS	862#	1008#	1049	1073	1105	1125	1143	1203	1268	1286	1534	1553	1585	1653	1763
	1781	1984	2002	2143	2161	2179	2201	2243	2310	2480	2502	2530	2539	2548	2567
	2575	2590	2599	2619	2632	2705	2728	2745	2774	2788	2801	2818	2831	2845	2873
	2893	2937	2971	3069	3146	3191	3194	3262	3293	3334	3377	3393	3422	3440	3501
	3831	3931													
SWRSU	1008#														
TRMTRP	3355#														
TYPBIN	1008#														
TYPDEC	1008#														
TYPNAM	1008#														
TYPNUM	1008#														
TYPOCS	1008#	1328	1333	1338	1411	2369	2416	2422	2428	2466	2472	4116			
TYPOCT	1008#	3212													
TYPTXT	1008#														
\$\$ESCA	1008#														
\$\$NEWT	1008#														
\$\$SET	3355#	3364	3365	3366	3368	3370	3371	3372	3373	3374					
\$\$SKIP	1008#														
.EQUAT	862#	898													
.HEADE	862#	865													
.SETUP	862#	1063													
.\$DB2D	862#	3438													
.\$POWE	862#	3375													
.\$READ	862#	3189													
.\$SAVE	862#	3144													
.\$SB2D	862#	3420													
.\$TRAP	862#	3332													
.\$TYPE	862#	2969													
.\$TYPO	862#	3067													

. ABS. 022616 000

ERRORS DETECTED: 0

CZRRAF,CZRRAF/SOL/CRF/NL:TOC=SYSMAC.SML/ML,CZRRAF.P11
 RUN-TIME: 18 9 1 SECONDS

MAINDEC-11-CZRXA-F MACY11 30(1046) 12-APR-82 14:30 PAGE 96 L 7
CZRRAF.P11 09-APR-82 15:11 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0089

RUN-TIME RATIO: 87/29=2.9
CORE USED: 19K (38 PAGES)