

PDP11

MOS/CORE 0-124K EXER.
CZKMAF0

AH-8851F-MC

COPYRIGHT 75-79
FICHE 1 OF 1

SEP 1979

digital
MADE IN USA

1 .REPT 0

IDENTIFICATION

PRODUCT CODE: AC-8850F-MC
PRODUCT NAME: CZKMAF0 MOS/CORE 0-124K EXER
DATE CREATED: MAR., 1979
MAINTAINER: DIAGNOSTIC GROUP

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

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITALS COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

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

COPYRIGHT (C) 1975,1979 DIGITAL EQUIPMENT CORPORATION

CONTENTS

49		
50		
51		
52		
53	1.0	ABSTRACT
54	1.1	GETTING STARTED
55		
56	2.0	REQUIREMENTS
57	2.1	EQUIPMENT
58	2.2	STORAGE
59		
60	3.0	LOADING PROCEDURE
61		
62	4.0	STARTING PROCEDURE
63	4.1	SWITCH SETTINGS
64	4.2	CONTROL-C OPTION
65	4.3	STARTING ADDRESS =200
66		RESTART ADDRESS =250
67	4.4	PROGRAM AND/OR OPERATOR ACTION
68	4.5	LONG GALLOP OPTION
69		
70	5.0	PROGRAM HALTS (NORMAL + ERROR)
71		
72	6.0	ERRORS
73	6.1	ERROR MESSAGE FORMAT.
74	6.2	ERROR DICTIONARY
75	6.3	ERROR HISTORY
76	6.4	ERROR RECOVERY
77		
78	7.0	RESTRICTIONS
79		
80	8.0	MISCELLANEOUS
81	8.1	ADDRESS/BANK RANGES IN OCTAL AND DECIMAL
82	8.2	EXECUTION TIME
83	8.3	PASS COUNT AND TEST NO. LOCATIONS
84	8.4	STACK POINTER
85	8.6	POWER FAIL
86		
87	9.0	PROGRAM DESCRIPTION
88	9.1	NARRATIVE FLOW CHART
89	9.2	TEST TITLES
90		TEST 0: TEST FOR PROPER BANK SELECTION
91		TEST 1: CHECK DAT1/DATO LINES
92		TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
93		TEST 3: DUAL ADDRESS TEST A
94		TEST 4: DUAL ADDRESS TEST B
95		TEST 5: MARCHING 1'S AND 0'S
96		TEST 6: CELLS' VOLATILITY TEST
97		TEST 7: SHIFTING DIAGONAL
98		TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
99		TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
100		TEST 12: WORST CASE TESTING FOR CORE MEMORY
101		TEST 13: WRITE RECOVERY TEST
102		
103	10.0	RXDP & ACT11 & APT OPERATION

104 [1.0] ABSTRACT

105
106 THIS DIAGNOSTIC WILL TEST 0 - 124K OF MOS OR CORE MEMORY
107 ON ANY PDP-11 FAMILY COMPUTER. SOME TESTS ARE WORST CASE
108 FOR MOS AND SOME FOR CORE, BUT ALL TESTS ARE ALWAYS RUN.
109 THE TESTS OCCUPIES LESS THAN 2K OF MEMORY SO IT CAN BE
110 USED TO TEST A SYSTEM WITH ONLY 4K OF MEMORY. IF ONLY 4K
111 EXISTS, HOWEVER, THE ABSOLUTE LOADER IS NOT SAVED.

112
113 THIS PROGRAM CAN BE RUN UNDER XXDP, APT AND ACT MONITORS.
114 ON PROCESSORS WITH NO HARDWARE SWITCH REGISTER, SOFTWARE
115 SWITCH REGISTER = LOCATION 176.

116
117 [1.1] GETTING STARTED

118
119 IF NO HARDWARE SWITCH REGISTER SET LOCATION 176 TO OBTAIN SWITCH
120 OPTIONS.

121
122 TO START:

- 123 -----
124
125 A. SET SWITCH REGISTER = 00000
126 B. START AT 200.
127 C. THE MEMORY LIMITS WILL BE PRINTED.
128 D. SEE SECTION 4.4 FOR REST OF PRINTOUTS EXPECTED.
129 E. 'PASS#01' WILL BE TYPED LAST, AND THE TEST WILL
130 RESTART.
131 F. TO HALT THE TEST, TYPE CONTROL-C, THIS WILL INSURE THE
132 PROGRAM IS RELOCATED BACK TO LOWER MEMORY.
133 BE PATIENT, THE CONTROL-C IS ONLY RECOGNIZED AT THE END
134 OF THE CURRENT SUBTEST.
135 G. IF AN UNEXPECTED HALT OCCURS SEE SECTION 6.0. IF AN
136 ERROR # IS TYPED SEE SECTION 6.2.

137
138 !CAUTION! BEFORE 'DIGGING' INTO THE LISTING READ
139 SECTION 9.

140
141 SWITCH SETTING SUMMARY (SEE SECTION 4.1 FOR DETAILS)

142 -----
143
144 BIT15(100000) HALT ON ERROR
145 BIT14(040000) LOOP IN SUBTEST DEFINED BY BITS <3:0>
146 BIT13(020000) INHIBIT ERROR PRINTOUTS
147 BIT12(010000) ENABLE TESTING ABOVE 28K (WITH MEMORY MANAGEMENT)
148 BIT11(004000) ENABLE PARITY TESTING
149 BIT10(002000) HALT AFTER EACH SUBTEST
150 BIT09(001000) INHIBIT PROGRAM RELOCATION
151 BIT08(000400) TYPE FIRST FAILING BIT ERROR PER 4K.
152 BIT07(000200) ENABLE LONG GALLOPING TEST
153 BIT06(000100) INHIBIT MEMORY SIZING
154 BIT05(000040) INHIBIT 'PASS#XX' PRINTOUTS
155 BIT04(000020) INHIBIT PRINTOUTS
156 BIT03-BIT00 BEGINNING TEST NUMBER.
157
158
159

160 [2.0] REQUIREMENTS

161
162 [2.1] EQUIPMENT

163
164 STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE
165 AND FROM 4K TO 124K OF MEMORY. PROGRAM WILL ALSO RUN ON THE
166 PDT-11 AND ON 30K LSI SYSTEMS.
167

168
169
170 [2.2] STORAGE

171
172 PROGRAM STORAGE - 0000 - 7744. PROGRAM EXPANDS FOR ERROR
173 HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR.
174 (SEE SECTION 9. FOR DETAILS)
175

176
177
178 [3.0] LOADING PROCEDURE

179
180 USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.
181

182
183
184 [4.0] STARTING PROCEDURE

185
186 [4.1] SWITCH SETTINGS

187
188 SOFTWARE SWITCH REGISTER = LOCATION 176
189

190 BIT15(100000) HALT ON ERROR
191

192 BIT14(040000) LOOP ON TEST DEFINED BY SWITCH REGISTER BITS <3:0>
193

194 BIT13(020000) INHIBIT ERROR PRINTOUTS
195

196 BIT12(010000) ENABLE MEMORY MANAGEMENT (TESTING ABOVE 28K, 30K SYSTEM DOES NOT
197 NEED KT SUPPORT)
198

199 BIT11(004000) ENABLE PARITY MODULES.
200 !'PAR' WILL BE TYPED
201

202 BIT10(002000) HALT AFTER EACH SUBTEST
203 !PRESS CONTINUE TO DO NEXT SUBTEST
204

205 BIT09(001000) INHIBIT PROGRAM RELOCATION
206 !IF SET LOCATIONS 430-7776 WILL NOT BE
207 !TESTED.
208

209 BIT08(000400) TYPE FIRST FAILING BIT IN EACH 4K BANK ONLY.
210 !THE TOTAL ERROR COUNT (UP TO 377) WILL
211 !BE SAVED IN THE ERROR HISTORY.
212

213 BIT07(000200) ENABLE LONG GALLOPING TEST.
214 !'GLP' WILL BE TYPED.
215 !CAUTION! INCREASES TEST TIME BY FACTOR OF 25.


```
216
217      BIT06(000100)  INHIBIT MEMORY SIZING.
218                      !THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:
219                      (VALUES TO TEST 0-8K ARE SHOWN)
220                      (LOWTWO=LOCATION 324)
221
222                      LOWTWO: 0          ;STORE BITS 17:16 OF LOW TEST ADDRESS
223                      LOWADD: 0         ;STORE REST OF LOW TEST ADDRESS
224                      ;DO NOT ATTEMPT TO SET THE LOWER LIMIT
225                      ;AT OR ABOVE 160000 ON A 30K LSI SYSTEM.
226                      ;THE PROGRAM WILL ASSUME MEMORY MANAGEMENT
227                      ;MUST BE USED.
228                      HIGHTWO: 0        ;STORE BITS 17:16 OF HIGH TEST ADDRESS
229                      HIGHADD: 37776    ;STORE REST OF HIGH TEST ADDRESS
230
231      BIT05(000040)  INHIBIT 'PASS#XX' PRINTOUTS
232
233      BIT04(000020)  A.  INHIBIT ERROR HISTORY PRINTOUTS.  THE
234                      ERROR HISTORY CAN STILL BE OBTAINED
235                      BY TYPING CONTROL-C.
236                      B.  INHIBIT PRINTOUTS 'PAR','GLP','TST13 BNK XX'.
237
238      BIT03-BIT00    NUMBER OF TEST (0-13) TO RUN FIRST.
239                      !NORMALLY USED WITH BIT14 (LOOP ON TEST)
240
241
242
243 [4.2]  CONTROL-C OPTION
244
245      CONTROL C [^C] AFTER COMPLETION OF THE CURRENT TEST.
246                      THE ERROR HISTORY (SEE SEC. 6.3) WILL BE
247                      TYPED. THE PROGRAM WILL HALT IN LOWER MEMORY.
248                      PRESSING CONTINUE WILL RESTART THE DIAGNOSTIC.
249
250 [4.3]  STARTING ADDRESS= 200
251        RESTART ADDRESS = 250 OR 200
252
253        RESTART AT 200 CLEARS PASS COUNT ($PASS) AND PRINTS 'CZKMAF' TITLE.
254
255
256
257
258
259 [4.4]  PROGRAM AND/OR OPERATOR ACTION
260
261        1)  LOAD PROGRAM INTO MEMORY USING ABSOLUTE LOADER.
262        2)  SET OPTIONS (SEE SEC. 4.1)
263        3)  START THE PROGRAM AT 200
264        4)  THE FOLLOWING IS AN EXAMPLE WITH EXPLANATIONS
265            OF THE PRINTOUTS EXPECTED.
266
267        'XXXXX-YYYYY'      ;ADDRESSES OF TEST BOUNDARIES.
268
269        'PAR'              ;IF PARITY OPTION SELECTED
270
271        'GLP'              ;IF LONG GALLOPING OPTION SELECTED.
```

272 :PRINTED AS TST11 IS ENTERED.
 273
 274 'TST13 BNK 00' :ENTERING BANK 00 IN TEST 13.
 275 'TST13 BNK 01' :AND BANK 1...
 276 ETC... :UNTIL ALL BANKS (UP TO 7) HAVE BEEN TESTED.
 277
 278 'REL' :THE DIAGNOSTIC RELOCATES TO HIGHEST
 279 :LOCATIONS UNDER TEST AND RUNS TST0-TST13 AGAIN.
 280
 281 'TST13 BNK 00' :TESTING BANK 00 IN TEST 13 (RELOCATED STATE.)
 282 :NOTE-ONLY BANK 00 IS TESTED IN THE RELOCATED STATE.
 283
 284 'PASS#XX' :WHERE 'XX' IS THE PASS NO.
 285
 286
 287 ADDITIONAL PRINTOUTS
 288 'NO PAR' :PRINTED IF PARITY SELECTED BUT NOT AVAILABLE.
 289
 290 'NO KT' :PRINTED IF SWR BIT 12 IS SET AND NO MEMORY
 291 :MANAGEMENT AVAILABLE.
 292
 293
 294

4.5 LONG GALLOP OPTION

NORMAL WORST CASE SR SETTING = 0000. FOR LONG GALLOP
 SR = 200. LONG GALLOP OPTION SHOULD ONLY BE USED IF AN
 MOS MEMORY PROBLEM IS SUSPECTED AND NO OTHER SUBTESTS
 WILL FAIL. THE TEST TIME IS INCREASED 25 TIMES.

[5.0] PROGRAM HALTS (NORMAL+ ERROR)

THIS IS A LIST OF EXPECTED HALTS. IF THE TEST HALTS
 IN A LOCATION NOT IN THIS LIST AND IT IS LESS THAN 776, IT
 MAY BE DUE TO A DEVICE INTERRUPTING.
 NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS
 MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND
 BY SUBTRACTING 500 FROM THE HALT PC AND ADDING THIS DIFFERENCE TO THE
 CONTENTS OF SAVR6 [LOC. 350].

PC	REASON	RECOVERY
--	-----	-----
112	TRAP TO LOC. 4	EXAMINE R6, IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED.
---	POWER FAIL	POWER UP WILL RECOVER IF IN CORE MEMORY. IF NOT CORE OPERATION IS UNDEFINED.
1714	HALT AT END OF TEST SWITCH SET.	PRESS CONTINUE TO GO TO NEXT SUBTEST.

328 6156 HALT ON ERROR PRESS CONTINUE.
329 SWITCH SET.
330
331 6240 CONTROL-C TYPED PRESS CONTINUE TO RE-
332 OR FATAL ERROR START TEST.
333 OCCURRED
334
335
336 [6.0] ERRORS
337
338 [6.1] ERROR MESSAGE FORMAT
339
340 THE ERROR PRINTOUT CONSISTS OF 6 OCTAL WORDS IN THE FOLLOWING
341 FORMAT:
342
343 'LOCATION GOOD BAD PC ERROR PASFLG'
344
345
346 'ADR ERR' WILL BE PRINTED PRIOR IF AN ADDRESSING ERROR IS SUSPECTED.
347 'PAR ERR' WILL BE PRINTED PRIOR IF A PARITY ERROR TRAP OCCURRED
348 !CAUTION! IF PARITY ERROR THE GOOD DATA PRINTOUT IS THE
349 PARITY MODULE UNIBUS ADDRESS THAT FAILED.
350
351
352 WHERE:
353
354 LOCATION= FAILING MEMORY LOCATION
355 GOOD = GOOD DATA [DATA THAT WAS EXPECTED]
356 BAD = BAD DATA [DATA THAT WAS FOUND]
357 PC = PROGRAM COUNTER AT ERROR CALL.
358 ERROR = FAILING ERROR NO. (SEE SEC 6.2 - ERROR DICTIONARY)
359 PASFLG = CONTENTS OF LOCATION PASFLG. THIS MAY NOT BE RELEVANT.
360 (SEE SEC. 6.2-ERROR DICTIONARY)
361
362
363 !THE TEST WILL CONTINUE AFTER THE ERROR PRINTOUT.
364 !'NO KT' WILL BE TYPED IF TESTING ABOVE 28K SELECTED AND NO MEMORY
365 !MANAGEMENT IS FOUND. (30K SYSTEM DOES NOT NEED KT SUPPORT)
366
367 !'NO PAR' WILL BE TYPED IF PARITY OPTION SELECTED
368 !AND NO PARITY MODULES WERE FOUND.
369
370 (FATAL ERRORS)
371
372 'ERR #XXXXXX' WILL BE TYPED WHERE 'XXXXXX' IS
373 THE ERROR NUMBER. THE DIAGNOSTIC WILL USUALLY HALT ON THIS TYPE
374 OF ERROR. SEE SEC. 6.2 -ERROR DICTIONARY - FOR DESCRIPTIONS
375 OF THE ERROR.
376
377
378
379 (APT MODE ERRORS)
380
381 ALL ERRORS ARE TREATED AS FATAL UNDER APT. WHEN AN
382 ERROR OCCURS UNDER APT A '1' IS STORED IN LOCATION
383 \$MSGTY AND THE PROGRAM HALTS AT FATHLT.

```
384
385          $FATAL CONTAINS THE ERROR NO. IN THE LOW BYTE AND
386          THE FAILING BANK NO. UNDER TEST IN THE HIGH BYTE.
387
388
389
390
391 [6.2]  ERROR DICTIONARY
392
393          THIS IS A LIST OF ERROR NUMBERS PRINTED AND POSSIBLE
394          CAUSES FOR THE ERROR.
395          THE ROUTINE NAME WHERE THE ERROR CALL ORIGINATED IS GIVEN IN
396          BRACKETS.
397          NOTE- 'BAKPAT' REFERS TO THE BACKGROUND PATTERN WRITTEN INTO MEMORY
398          FOR VARIOUS TESTS. IF PARITY SELECTED IT HAS A VALUE = 376 ,ELSE=377
399          'SWAPPED BAKPAT' = 77000 IF PARITY SELECTED, ELSE=77400
400
401 .ENDR
402
403
404 ;ERR # 0      ;[BUSER] BUS ERROR TRAP TO LOC. 4 OCCURRED
405               ;      THIS ERROR IS NOT PRINTED AND IS FOR 'APT' USE.
406
407 ;ERR # 1      ;[TSTTRP]FATAL DATA ERROR
408               ;LOCATIONS 0000-430 FAILED 1'S + 0'S TEST.
409               ;R0 = GOOD DATA
410               ;R1 = ADDRESS OF FAILING LOCATION.
411
412 ;ERR # 2      ;[APTSIZ] APT FATAL ERROR
413               ;APT MEMORY TABLES NOT SETUP CORRECTLY.
414               ;CHECK LOCATIONS $MAMS1 [430] TO $MADR4[446]
415               ;. FOR CORRECT MEMORY SIZE DATA.
416
417 ;ERR # 3      ;[TSTSIZ] OPERATOR FATAL ERROR
418               ;SELECTED MEMORY SIZE GREATER THAN 28K
419               ;(30K SYSTEM DOES NOT NEED KT SUPPORT), BUT
420               ;SR BIT12 (10000) NOT SET.
421               ;SET BIT12 AND RESTART AT 200.
422
423 ;ERR # 4      ;[TSTSIZ] OPERATOR FATAL ERROR
424               ;LOWEST SELECTED TEST LIMIT IS HIGHER THAN
425               ;HIGHEST TEST LIMIT.  SET LOCATIONS 'LOWTWO'[322]
426               ;TO 'HIGHADD' [330] CORRECTLY AND RESTART
427               ;AT 200.
428
429 ;ERR # 5      ;[TST0] TEST SEQUENCE ERROR
430               ;TST0 HAS BEEN ENTERED OUT OF SEQUENCE
431               ;TESTN SHOULD = 00
432               ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
433               ;IF POSSIBLE SELECT ANOTHER 4K BANK
434               ;BANK 0 AND RERUN THE TEST ON THE FAILING MEMORY.
435
436 ;ERR # 6      ;[TST0] DUAL ADDRESSING ERROR
437               ;FOR THIS ERROR THE GOOD DATA PRINTED IS AN
438               ;ADDRESS.  THIS IS THE ADDRESS SELECTED WHEN
439               ;THE SAME DATA WAS WRITTEN INTO THE FAILING
```


440 :LOCATION. CHECK BANK SELECT CIRCUITRY
441
442 :ERR # 7 :[TST0] ADDRESS AND DATA ERROR
443 :IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA
444 :WRITTEN INTO THE FAILING LOCATION WAS IN
445 :ERROR ALSO.
446
447 :ERR # 10 :[TST0] DATA ERROR
448 :IF BAD DATA = 0000 COULD BE AN ADDRESSING
449 :ERROR , ELSE COMPARE GOOD AND BAD DATA FOR FAILING BITS.
450
451 :ERR # 11 :[TST0] ADDRESSING ERROR
452 :THE FAILING ADDRESS RESPONDED BUT IS NON-
453 :EXISTENT. MAY BE A DUAL ADDRESSING PROBLEM.
454
455 :ERR # 12 :[TST1] TEST SEQUENCE ERROR
456 :\$TESTN [404] SHOULD = 01
457 : THE DIAGNOSTIC HAS BEEN CORRUPTED.
458
459 :ERR # 13 :[TST1] DATA ERROR
460 :COMPARE GOOD AND BAD PRINTED DATA, FAILING
461 :DATA BITS MAY SHORTED OR SWAPPED.
462
463 :ERR # 14 :[TST2] TEST SEQUENCE ERROR
464 :\$TESTN [404] SHOULD = 02
465 : THE DIAGNOSTIC HAS BEEN CORRUPTED.
466
467 :ERR # 15 :[TST2] ADDRESS OR DATA ERROR
468 :IF 'ADR ERR' NOT PRINTED THEN THE BYTE SELECT
469 :CIRCUITRY PROBABLY FAILED.
470
471 :ERR # 16 :[TST3] TEST SEQUENCE ERROR
472 :\$TESTN [404] SHOULD = 03
473 : THE DIAGNOSTIC HAS BEEN CORRUPTED.
474
475 :ERR # 17 :[TST3] DUAL ADDRESSING ERROR
476 :DUAL ADDRESSING PROBLEM FOR BITS THAT DIFFER
477 :IN GOOD AND BAD DATA PRINTOUT.
478
479 :ERR # 20 :[TST3] DUAL ADDRESSING ERROR
480 :FOR THIS ERROR THE DATA PRINTED IS AN ADDRESS.
481 :THIS IS THE ADDRESS THAT WAS SELECTED WHEN THE
482 :SAME DATA WAS WRITTEN INTO THE FAILING LOCATION.
483
484 :ERR # 21 :[TST3] DUAL ADDRESSING ERROR
485 :SAME AS ERROR #20 EXCEPT DIFFERENT DATA
486 : (SWAPPED BAKPAT) WAS WRITTEN.
487
488 :ERR # 22 :[TST4] TEST SEQUENCE ERROR
489 :\$TESTN [404] SHOULD = 04.
490 : THE DIAGNOSTIC HAS BEEN CORRUPTED.
491
492 :ERR # 23 :[TST4] DUAL ADDRESSING ERROR
493 :IF PASFLG = 0 THEN THE FAILING LOCATION
494 :AND FAILING DATA ARE DUAL ADDRESSES.
495

```
496 ;ERR # 24      ;[TST5] TEST SEQUENCE ERROR
497                ;$TESTN [404] SHOULD = 05
498                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
499
500 ;ERR # 25      ;[TST5] DATA ERROR
501                ;DATA WRITE OR READ ERROR.
502 ;ERR # 26      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
503                ;IF PASFLG=0 FAILED MARCHING 1'S + 0'S IN
504                ;                MAX TO MIN DIRECTION.
505                ;IF PASFLG=1 FAILED MARCHING 1'S + 0'S IN
506                ;                MIN TO MAX DIRECTION
507                ;IF PASFLG=3 FAILED MARCHING 0'S + 1'S IN
508                ;                MAX TO MIN DIRECTION.
509
510 ;ERR # 27      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
511                ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA IS
512                ;CHECKED IMMEDIATELY AFTER BEING WRITTEN.
513
514 ;ERR # 30      ;[TST6] TEST SEQUENCE ERROR
515                ;$TESTN SHOULD = 06
516                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
517
518 ;ERR # 31      ;[TST6] VOLATILITY/REFRESH TEST ERROR
519                ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
520                ;IF PASFLG=1 THE FAILING LOCATION CHANGED WHILE
521                ;                ANOTHER LOCATIONS WAS WRITTEN FOR
522                ;                2 MS. THE OTHER LOCATION IS SAVED
523                ;                IN SAVLOC [352]
524                ;IF PASFLG=2 SWAPPED BAKPAT (77400 OR 77000)
525                ;                WRITE OR READ ERROR.
526                ;IF PASFLG=3 SAME AS IF PASFLG=2 EXCEPT
527                ;                THE DATA IS SWAPPED B/XPAT.
528
529 ;ERR # 32      ;[TST7] TEST SEQUENCE ERROR
530                ;$TESTN SHOULD = 07
531                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
532
533 ;ERR # 33      ;[TST7] SHIFTING DIAGONAL DATA ERROR
534                ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
535                ;IF PASFLG=1 BAKPAT READ CHECK ERROR
536                ;IF PASFLG GREATER THAN 1 BUT EVEN VALUE THEN:
537                ;                THE FAILING LOCATION COULD NOT BE WRITTEN INTO.
538                ;IF PASFLG GREATER THAN 1 BUT ODD VALUE THEN:
539                ;                THE FAILING LOCATION WAS WRITTEN CORRECTLY
540                ;                BUT LOST THE DATA.
541
542 ;ERR # 34      ;[TST10] TEST SEQUENCE ERROR
543                ;$TESTN SHOULD = 10
544                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
545
546 ;ERR # 35      ;[TST10] BAKPAT DATA ERROR
547                ;BAKPAT WRITE OR READ ERROR INTO THE FAILING LOCATION.
548
549 ;ERR # 36      ;[TST10] READ RECOVERY DATA ERROR
550                ; THIS ERROR CAN BE REPORTED BY TST10 AND TST11.
551                ; (THEY SHARE CODE). SEE $TESTN [404] FOR WHICH TEST FAILED.
```



```

552      ;FOR BOTH TESTS COMPARE THE GOOD AND BAD DATA AT THE FAILING
553      ;LOCATION TO SEE WHICH BITS FAILED.
554
555 ;ERR # 37      ;[TST10] READ RECOVERY DATA ERROR
556      ;IDENTICAL TO THE PREVIOUS ERROR EXCEPT SWAPPED BAKPAT IS
557      ;USED AS WRITE AND READ DATA.
558
559 ;ERR # 40      ;[TST11] TEST SEQUENCE ERROR
560      ;$TESTN SHOULD = 11
561      ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
562
563 ;ERR # 41      ;[TST12] TEST SEQUENCE ERROR
564      ;$TESTN SHOULD = 12
565      ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
566
567 ;ERR # 42      ;[TST12] WORST CASE CORE TEST DATA ERROR
568      ;IF PASFLG=1 COMPARE GOOD AND BAD DATA FOR FAILING BITS.
569      ;IF PASFLG=2 THE FAILING LOCATION WAS WRITTEN AND READ
570      ;      WITH GOOD DATA,BUT FAILED READ CHECK
571      ;      READING IN THE MIN. TO MAX DIRECTION.
572      ;IF PASFLG=3 SAME CONDITIONS AS PASFLG=2 EXCEPT FAILED
573      ;      DOING THE READ CHECK FROM MAX TO MIN DIRECTION.
574
575 ;ERR # 43      ;[TST12] WORST CASE CORE TEST DATA ERROR
576      ;      IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA WRITTEN
577      ;AND READ IS COMPLEMENTED.
578
579 ;ERR # 44      ;[TST13] TEST SEQUENCE ERROR
580      ;$TESTN SHOULD = 13
581      ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
582
583 ;ERR # 45      ;[TST13] WRITE RECOVERY TEST DATA ERROR
584      ;IF PASFLG=0 COMPARE GOOD AND BAD DATA FOR FAILING BITS.
585      ;IF PASFLG=77400 DATA ERROR FOUND WHILE DOING A SECOND READ CHECK.
586      ;IF PASFLG=77402 DATA ERROR FOUND IN FAILING LOCATION AFTER
587      ;      SMALL TEST PROGRAM RUN IN FAILING BANK.
588
589 ;ERR # 46      ;[TST13] WRITE RECOVERY TEST DATA ERROR
590      ;      DATA ERROR FOUND JUST BEFORE THE SMALL TEST
591      ;WAS TO BE RUN IN THE FAILING BANK. TO AVOID 'BLOWING' UP
592      ;WHEN THE SMALL TEST IS RUN TST13 IS ABORTED.
593
594 ;ERR # 47      ;[TST13] WRITE RECOVERY TEST DATA ERROR
595      ;      IDENTICAL TO ERROR #XXX EXCEPT THE DATA WRITTEN
596      ;AND READ IS DIFFERENT.(177667).
597      ;177667 IS THE COMPLEMENT OF 'JMP (R0)' (110) WHICH IS
598      ;THE ESCAPE FROM THE SMALL TEST PROGRAM RUN IN THE BANK
599      ;UNDER TEST.
600
601 ;ERR # 50      ;[PARERR] PARITY TRAP ERROR
602      ;PARITY TRAP TO 114 OCCURRED.
603      ;FOR THIS ERROR PRINTOUT THE 'GOOD DATA' IS ACTUALLY
604      ;THE FAILING PARITY MODULE UNIBUS ADDRESS.
605      ;SAVLOC [352] CONTAINS THE PC WHERE THE TRAP OCCURRED.
606
607 ;ERR # 51      ;[PARITY] PARITY TRAP FATAL ERROR

```

608 ; A PARITY TRAP TO 114 OCCURRED, BUT NO PARITY MODULES COULD BE FOUND
609 ;WITH AN ERROR BIT (BIT15) SET.

610
611 ;ERR # 52 ;[NOMM] OPERATOR FATAL ERROR
612 ; TESTING ABOVE 28K WAS SELECTED, BUT NO MEMORY MANAGEMENT
613 ;OPTION WAS FOUND.(30K SYSTEM DOES NOT NEED KT)
614 ; RESET SWITCH OPTIONS AND RESTART AT 200.

615
616 ;ERR # 53 ;[PARITY] OPERATOR FATAL ERROR
617 ; PARITY TESTING WAS SELECTED BUT NO PARITY MODULES
618 ;WERE FOUND.
619 ; RESET SWITCH OPTIONS AND START AT 200.

620
621 .REPT 0

622
623
624 [6.3] ERROR HISTORY

625
626 LOCATIONS IN MEMORY ARE SET ASIDE TO COLLECT A HISTORY
627 OF THE FAILING BITS IN A PARTICULAR MEMORY BANK. THIS
628 DATA IS COLLECTED FOR EVERY ERROR REGARDLESS OF SWITCH
629 SETTINGS.

630
631 NORMALLY THE DATA IS OUTPUT AT THE END OF TESTING, BUT
632 IF CONTROL-C IS TYPED IT IS OUTPUT AT THE END OF THE
633 CURRENT TEST.

634
635 THE ERROR HISTORY IS INTENDED TO HIGHLIGHT IF THE ERRORS
636 ARE DUE TO 1 BIT FAILING OR ONLY ADDRESS ERRORS.

637
638
639
640 ERROR HISTORY FORMAT:

641
642
643 ERROR BANK COUNT
644 -----

645

646

647

648

649

650

651

652

653

654

655

656

657

658

659

660

661

662

663

[6.4] ERROR RECOVERY

IF THE PROGRAM IS HALTED AFTER REPORTING AN ERROR IT CAN EITHER
BE CONTINUED OR RESTARTED AT 200 OR 250 (SEE SEC 4.2). HOWEVER FOR
CPU'S THAT DESTROY CONTENTS OF REGISTERS AFTER COMING TO A HALT
THE PROGRAM SHOULD ONLY BE RESTARTED.

[7.0] RESTRICTIONS

MEMORY UNDER TEST SHOULD BE CONTIGUOUS. FOR SYSTEMS HAVING NON-CONTIGUOUS MEMORY THE MEMORY BOUNDARIES SHOULD BE DEFINED BY THE OPERATOR. (CONTIGUOUS MEMORY IS DEFINED AS A MEMORY THAT CAN BE BOTH READ AND WRITTEN IN CONSECUTIVE LOCATIONS.)

[8.0] MISCELLANEOUS

[8.1] ADDRESS/BANK RANGES IN OCTAL AND DECIMAL

THIS REFERENCE TABLE CROSS REFERENCES THE MEMORY BANK NO.S, THE RANGE AND THE PAR USED WHEN MEMORY MANAGEMENT IS ENABLED. IT IS ALSO USEFUL TO SHOW STARTING ADDRESSES IN A PARTICULAR 4K BANK.

BANK NO.	DECIMAL RANGE	OCTAL RANGE	[PAGE ADDRESS REGISTER]		UNIBUS ADDRESS
			USED/CONTENT		
0	0 - 4K	000000-017776	0	0000	772340
1	4K - 8K	020000-037776	NOT USED		
2	8K-12K	040000-057776	NOT USED		
3	12K-16K	060000-077776	NOT USED		
4	16K-20K	100000-117776	NOT USED		
5	20K-24K	120000-137776	NOT USED		
6	24K-28K	140000-157776	NOT USED		
7	28K-32K	160000-177776	NOT USED	ON 30K (LSI-11) SYSTEMS	
			1	1600	772342
8	32K-36K	200000-217776	2	2000	772344
9	36K-40K	220000-237776	3	2200	772346
10	40K-44K	240000-257776	4	2400	772350
11	44K-48K	260000-277776	5	2600	772352
12	48K-52K	300000-317776	6	3000	772354
13	52K-56K	320000-337776	1	3200	
14	56K-60K	340000-357776	2	3400	
15	60K-64K	360000-377776	3	3600	
16	64K-68K	400000-417776	4	4000	
17	68K-72K	420000-437776	5	4200	
18	72K-76K	440000-457776	6	4400	
19	76K-80K	460000-477776	1	4600	
20	80K-84K	500000-517776	2	5000	
21	84K-88K	520000-537776	3	5200	
22	88K-92K	540000-557776	4	5400	
23	92K-96K	560000-577776	5	5600	
24	96K-100K	600000-617776	6	6000	
25	100K-104K	620000-637776	1	6200	

720	26	104K-108K	640000-657776	2	6400	
721	27	108K-112K	660000-677776	3	6600	
722	28	112K-116K	700000-717776	4	7000	
723						
724	29	116K-120K	720000-737776	5	7200	
725	30	120K-124K	740000-757776	6	7400	
726	31	124K-128K	760000-777776	7	7600	772354
727						

NOTES:

1. THE PAR (PAGE ADDRESS REGISTER) CONTENTS ARE SHOWN IN A TEST THAT SELF SIZES.
IF THE LIMITS OF TESTING ARE SET BY THE OPERATOR AND
IF THE BANK IS ABOVE 28K PAR NO. 1 WILL BE SET TO THE
BEGINNING PAGE. FOR EXAMPLE IF THE TESTING WAS TO
BEGIN WITH BANK 8 PAR NO. 1 WOULD EQUAL 2000, PAR 2
WOULD EQUAL 2200 ETC.

[8.2] EXECUTION TIME

HERE ARE SOME TYPICAL EXECUTION TIMES.

LSI-11 AND 4K:= 100 SECS.
LSI-11 AND 8K:= 5 MINUTES.

[8.2] PASS COUNT AND TEST NO. LOCATIONS

\$PASS [406] = PASS COUNT - CLEARED BY START AT 200.

\$TESTN [404] = CURRENT TEST NO. AND RELOCATION, PARITY FLAGS.

WHERE:
LOW BYTE = TEST NO.
IF BIT15 = 1 TEST IS RELOCATED
IF BIT13 = 1 PARITY UNDER TEST.

[8.4] STACK POINTER

THE STACK STARTS AT 500 WHEN THE PROGRAM IS NOT RELOCATED.
SAVR6[350] CONTAINS THE STACK STARTING VALUE WHEN THE DIAGNOSTIC
IS RELOCATED.

SAVR6 ALSO CONTAINS THE STARTING ADDRESS OF THE PROGRAM WHEN
IT IS RELOCATED.

[8.5] POWER FAIL

THE DIAGNOSTIC CAN BE POWER FAILED WITH NO ERRORS. TO USE,
START THE TEST AS USUAL AND POWER DOWN THEN UP AT ANY TIME.
THE PROGRAM SHOULD TYPE 'P' AND CONTINUE TO RUN FROM TEST 0
IN THE SAME STATE [I.E. STATE OF RELOCATION] AS IT WAS BEFORE
THE POWER WAS INTERRUPTED, HOWEVER IF THE DIAGNOSTIC WAS IN
A MEMORY THAT CAN NOT HOLD DATA WITH THE POWER DOWN THEN THE
PROGRAM WILL NOT RECOVER FROM POWER FAIL AND ON POWER-UP
OPERATION IS UNDEFINED.

776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831

[9.0] PROGRAM DESCRIPTION

[9.1] NARRATIVE FLOW CHART

THE TEST IS LOADED INTO LOCATIONS 0000 - 7744 BUT
EXPANDS DEPENDING ON HOW MUCH MEMORY IS UNDER TEST.
SEE STEP 6. BELOW FOR A DETAILED EXPLANATION.

THE FOLLOWING NARRATIVE FLOW CHART DESCRIBES MAJOR
PROGRAM OPERATION. FOR THE PERSON WHO NEEDS DETAIL THE
TAG ASSOCIATED WITH THE OPERATION IS GIVEN IN BRACKETS.

FOR THIS DISCUSSION SWITCH SETTINGS ARE IGNORED AND EVERYTHING IS
ASSUMED ENABLED.

1. [START] PRINT 'CZKMAF' TITLE
2. [TSTRP] SAVE DATA FROM LOCATIONS 0-376
INTO 7744-10314.
3. [TSTRP] TEST LOCATIONS 0-376 BY WRITING AND
READING 1'S AND 0'S. NOTE THIS IS THE ONLY
EXPLICIT TESTING OF THESE LOCATIONS.
4. [SLFSIZ] SIZE MEMORY BY WRITING INTO SUCCEEDING
MEMORY LOCATIONS UNTIL TIMEOUT TRAP TO 4 OCCURS,
OR 30K BOUNDARY REACHED.
ENABLE MEMORY MANAGEMENT AND SIZE MEMORY ABOVE
28K.
NOTE: IF UNDER XXDP CHAIN MODE IN 30K SYSTEM, SYSTEM IS SIZED
TO 28K.
5. [TYPsiz] TYPE MEMORY TEST LIMITS.
6. [SETSTK] SPACE IS SAVED AT THE END OF THE TEST
FOR AN ERROR HISTORY. FOR EACH 4K BANK 18 BYTES ARE SAVED
IN THE FOLLOWING FORMAT:

!ADR	ERR!	PAR	ERR!
!BIT14	!BIT15	!	!
!BIT12	!BIT13	!	!
!BIT10	!BIT11	!	!
!BIT08	!BIT09	!	!
!BIT06	!BIT07	!	!
!BIT04	!BIT05	!	!
!BIT02	!BIT03	!	!
!BIT00	!BIT01	!	!

IF GREATER THAN 4K UNDER TEST THE ABSOLUTE LOADER
(300 ADDRESSES) IS APPENDED. IF GREATER THAN 4K
AND UNDER XXDP CHAIN MODE 5376 (OCTAL) ADDRESSES
ARE APPENDED TO THE TEST. THIS SAVES THE XXDP

- 832 MONITOR, AND ALLOWS THE LOCATIONS OCCUPIED BY XXDP
833 TO BE TESTED.
834
- 835 7. [CLRMEM] CALL 'PARITY' ROUTINE AND IF SELECTED,
836 ENABLE ALL PARITY MODULES. 'PARMAP' [LOC. 352]
837 CONTAINS A MAP OF PARITY MODULES FOUND. IF
838 MODULE 172336 BIT 15 IS SET, IF #172334 FOUND BIT 14
839 IS SET ETC..
840
- 841 8. [CLRMEM] CLEAR MEMORY CURRENTLY UNDER TEST
842
- 843 9. [CONT] DISPATCH TO TST0
844
- 845 10. [TST0] EXECUTE TEST 0. SEE SECTION 10 FOR TEST
846 DESCRIPTIONS.
847
- 848 11. [TSTSCP] COMES HERE AFTER EACH TEST AND IF
849 CNTRL-C TYPED THEN GO TO ERROR HISTORY PRINTOUT.
850 IF SR=2000 THEN HALT
851 IF SR=40000 THEN LOOP ON TEST DEFINED BY <3:0>
852 ELSE CONTINUE TO NEXT TEST.
853
- 854 12. [TST1-TST12] EXECUTE TST1-TST12 EACH TIME
855 GOING TO STEP 9.
856
- 857 13. [TST13] TEST 13 IS DIFFERENT FROM TESTS 0-12,
858 BECAUSE IT IS A SMALL PROGRAM ACTUALLY RUNNING
859 IN THE MEMORY UNDER TEST. BEFORE THIS SMALL
860 PROGRAM IS STARTED 'TST13 BNK XX' IS TYPED.
861 THIS IS DONE IN CASE THE PROGRAM FAILS. THE
862 USER CAN THEN AT LEAST TELL WHICH BANK OF MEMORY
863 FAILED.
864
- 865 14. [RELOC] THE PROGRAM RELOCATES TO HIGH MEMORY
866 TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG).
867 WHERE 'ENDPRG' IS THE CONTENTS OF ENDSTK[306].
868 I.E THE LAST PROGRAM ADDRESS. NOTE 'REL' IS
869 PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
870
- 871 15. TESTS 0-13 ARE RUN AS DESCRIBED ABOVE EXCEPT
872 ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED.
873
- 874 16. [RELOER] RELOCATE THE PROGRAM BACK TO LOWER
875 MEMORY.
876
- 877 17. [LOWER] IF CONTROL-C TYPED GO PRINT ERROR
878 HISTORY.
879
- 880 18. [TSTMM] IF MEMORY MANAGEMENT SELECTED AND AVAILABLE,
881 RUN TESTS 0-13 ON THE FIRST 24K SLICE ABOVE 28K.
882
- 883 19. [CONTMM] CALL 'UPMM' TO UPDATE MEMORY MANAGEMENT
884 PAR REGISTERS TO POINT TO THE NEXT 24K SLICE OF
885 UPPER MEMORY.
886
- 887 20. [MAXADR] REPEAT STEPS 18 + 19 UNTIL ALL


```
888          MEMORY ABOVE 28K IS TESTED.
889
890      21. [ENDPAS] PRINT ERROR HISTORY OF FAILING BITS
891
892      22. [$EOP] DISABLE PARITY MODULES.
893          PRINT 'PASS#XX'
894
895
896
897 [9.2] TEST TITLES
898
899     SEE THE TEST HEADINGS IN THE LISTING FOR DETAILS ON EACH TEST.
900
901         TEST 0: TEST FOR PROPER BANK SELECTION
902         TEST 1: CHECK DATI/DATO LINES
903         TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
904         TEST 3: DUAL ADDRESS TEST A
905         TEST 4: DUAL ADDRESS TEST B
906         TEST 5: MARCHING 1'S AND 0'S
907         TEST 6: CELLS' VOLATILITY TEST
908         TEST 7: SHIFTING DIAGONAL
909         TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
910         TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
911         TEST 12: WORST CASE TESTING FOR CORE MEMORY
912         TEST 13: WRITE RECOVERY TEST
913
914
915 [10.0] RXDP & ACT11 & APT OPERATION
916
917     RXDP CHAIN MODE
918     -----
919
920     OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
921
922     1. NO 'CZKMAF' TITLE IS PRINTED.
923     2. NO TEST 13 PRINTOUTS SUCH AS 'TST13 BNK 00'.
924     3. THE PROGRAM ALWAYS HALTS ON ERROR.
925     4. AT THE END OF TEST ($ENDAD) CONTROL IS RETURNED TO
926        THE RXDP CHAIN MONITOR VIA LOCATION 42.
927     5. IF 30K SYSTEM ONLY 28K WILL BE TESTED IN XXDP CHAIN MODE
928
929     ACT11
930     -----
931
932     OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
933
934     1. NO PRINTOUTS EXCEPT ERROR PRINTOUTS.
935     2. THE PROGRAM ALWAYS HALTS ON ERROR.
936     3. AT THE END OF TEST ($ENDAD) CONTROL IS RETURNED TO
937        THE ACT11 MONITOR VIA LOCATION 42.
938
939     APT
940     ---
941
942     OPERATION IS SIMILAR TO STAND ALONE EXCEPT:
943
```

944 1. THE SOFTWARE SWITCH REGISTER BECOMES LOCATION 422 (\$SWREG).
945 2. AUTO SIZING CAN BE INHIBITED BY SETTING BIT 7 OF BYTE
946 LOCATION 421 (\$ENVM).
947 3. ALL PRINTOUTS CAN BE INHIBITED BY SETTING BIT 5 OF
948 BYTE LOCATION 421 (\$ENVM).
949 4. ALL ERRORS CAUSE LOCATION 400 (\$MSGTY) TO BE SET =
950 0001 AND THE PROGRAM HALTS AT LOCATION 6240 (FATHLT).
951 LOCATION 402 (\$FATAL) CONTAINS THE ERROR NO. IN THE
952 LOW BYTE AND THE FAILING MEMORY BANK NO. IN THE HIGH
953 BYTE.
954 NOTE: THE ENVIRONMENTAL MODE BYTE SHOULD BE SET TO 240 WHILE THE SOFTWARE
955 ENVIRONMENTAL BYTE SHOULD BE SET TO 001.
956
957
958
959 .ENDR


```

960 .ENABL ABS
961 .NLIST MD,MC,CND
962
963 .LIST ME,BIN,SEQ,LOC
964 .TITLE CZKMA
965 .*COPYRIGHT (C) MARCH 1979
966 .*DIGITAL EQUIPMENT CORP.
967 .*MAYNARD, MASS. 01754
968 .*
969 .*PROGRAM BY DIAGNOSTIC ENGINEERING
970 .*
971 .*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
972 .*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
973 160000 $SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
974
975
976
977
978 ;;TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS
979
980
981
982
983 000240 SCOPE =NOP
984
985 000042 000042 .=42
986 000042 000000 .WORD 0 ;FOR ACT/XXDP
987
988 .SBTTL ACT11 HOOKS
989
990
991
992 000044
993 000046 $SVPC=. ;SAVE PC
994 000046 000156 .=46
995 000052 000052 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
996 000052 040000 .=52
997 000044 .WORD 40000 ;;2)SET LOC.52 TO 40000
998 .=$SVPC ;; RESTORE PC
999
1000 000070 000070 000136 000024 PWRDN: .=70
1001 000076 000000 MOV #PWRUP,@#24
1002 HALT

```

```

1003
1004
1005      000104      . = 104
1006      : GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
1007 000104 013727 000001 000400 BUSER: MOV @#1,$MSGTY ; TELL APT FATAL ERROR#000
1008 000112 000000      HALT ; *ERROR* TRAP TO LOC. 4 OCCURRED.
1009      : 114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS. SETUP IN
1010      : ROUTINE 'BEGIN'.
1011      . = 120
1012
1013
1014
1015      : * WRITE MEMORY BACKGROUND
1016      : * -----
1017      : *
1018      : * THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
1019      : * THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES
1020      : * THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE
1021      : * HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE
1022      : * SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
1023      : *
1024
1025 000120 010401      WRTMEM: MOV R4,R1 ; SET R1 TO LOWEST LOCATION UNDER TEST
1026 000122 013700 000316      MOV @#BAKPAT,R0 ; LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
1027 000126 010021      2$: MOV R0,(R1)+ ; STARTING FROM THE LOWEST LOCATION WRITE THE
1028 000130 020105      CMP R1,R5 ; MEMORY TO BACK GROUND PATTERN
1029 000132 103775      BLO 2$
1030 000134 000207      RTS PC ; RETURN FROM THE SUBROUTINE
1031
1032
1033 000136 013706 000350      PWRUP: MOV @#SAVR6,SP ; RESTORE STACK POINTER
1034 000142 012700 006112      MOV #PNTMES-BEGIN,R0
1035 000146 060600      ADD SP,R0 ; GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
1036      : RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE
1037 000150 004710      JSR PC,(R0) ; GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A 'P'
1038 000152 000120      .ASCIZ /P/
1039      .EVEN
1040
1041 000154 000411      BR START
1042
1043      : * SERVICE XXDP/ACT11
1044 000156 004710      $ENDAD: JSR PC,(R0) ; RETURN TO ACT11/XXDP MONITOR
1045 000160 000240      NOP ; IF QUICK VERIFY=RESET ELSE NOP
1046 000162 000240      NOP ; IF QUICK VERIFY=CLR #-1 ELSE INC #0
1047 000164 000240      NOP ; IF QUICK VERIFY=BR -4 ELSE NOP
1048 000166 000430      BR RESTR ; REPEAT TEST UNDER ACT11/XXDP
1049
1050      . = 176
1051 000176 000000      SWREG: .WORD 0
1052
1053
1054      : *****
1055      : SBTTL START AND RESTART ROUTINES
1056      : * RESTART AT 200 TO CLEAR APT TABLES
1057      : *****
1058 000200 013706 000350      START: MOV @#SAVR6,SP ; SETUP STACK POINTER.

```



```

1059 000204 012703 000412          MOV    #SUNIT,R3      ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1060 000210 005043          1$: CLR    -(R3)          ;CLEAR A MAILBOX LOCATION
1061 000212 022703 000400          CMP    #SMAIL,R3      ;DONE?
1062 000216 001374          BNE     1$              ;BRANCH IF NO
1063 000220 105737 000042          TSTB   @#42          ;ACT11 MODE?
1064 000224 001011          BNE     RESTRT          ;BRANCH IF YES
1065 000226 105737 000405          TSTB   @#$TESTN+1      ;ARE WE RELOCATED?
1066 000232 100406          BMI     RESTRT          ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
1067 000234 004767 006344          JSR     PC,TPCRLF      ;PRINT TITLE
1068 000240 055103 046513 043101  .ASCIZ  /CZKMAF0/
1069 000246 000060
1070
1071          .EVEN
1072 000250 012704 007744          RESTRT: MOV   #ENDPRG,R4      ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1073 000254 012703 000346          MOV   #SAVR5,R3      ;CAUSE R3 TO POINT TO THE LOCATION SAVR5
1074 000260 012305          MOV   (R3)+,R5      ;RESTORE R5
1075 000262 012306          MOV   (R3)+,SP      ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1076 000264 010600          MOV   SP,R0        ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1077 000266 012746 000340          MOV   #340,-(SP)     ;SET HIGH PRIORITY FOR RTI
1078 000272 010046          MOV   R0,-(SP)
1079 000274 000002          RTI
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089          .SBTTL  APT PARAMETER BLOCK
1090
1091          ;*****
1092          ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1093          ;*****
1094          .SX=.          ;;SAVE CURRENT LOCATION
1095          .=24          ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1096 000024 000200          200      ;;FOR APT START UP
1097          .=44          ;;POINT TO APT INDIRECT ADDRESS PNTR.
1098 000044 000276          $APTHDR ;;POINT TO APT HEADER BLOCK
1099          .=$X          ;;RESET LOCATION COUNTER
1100          ;*****
1101          ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1102          ;INTERFACE SPEC.
1103
1104 000276          $APTHD:
1105 000276 000000          $HIBTS: .WORD 0      ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1106 000300 000400          $MBADR: .WORD $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1107 000302 001440          $TSTM:  .WORD 800.    ;;RUN TIM OF LONGEST TEST
1108 000304 002260          $PASTM: .WORD 1200.   ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1109 000306 000000          $UNITM: .WORD      ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1110 000310 000024          .WORD  $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
1111
1112
1113          000405          REL=$TESTN+1
1114
;IT WILL BE 0 IF THE PROGRAM IS IN THE LOWER
;CORE. BIT 7 OF THE BYTE WILL BE SET IF THE

```

Line	Address	Label	Value	Comment
1115				;PROGRAM IS IN A RELOCATED STATE AND BIT 5
1116				;WILL BE SET IF PARITY BITS ARE BEING TESTED
1117	000276	MMAVA:	.= \$APTHD	
1118				;THIS BYTE IS USED TO DETERMINE IF MEMORY
1119				;MANAGEMENT IS AVAILABLE OR NOT
1120				
1121	000277	TYPENB:	.= MMAVA+1	
1122				;THIS BYTE IS USED TO DETERMINE IF THE
1123				;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
1124				
1125	000300	\$PRERR:	.= TYPENB+1	
1126				;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
1127				;A PARITY ERROR
1128				
1129	000301	\$ADERR:	.= \$PRERR+1	
1130				;THIS BYTE IS USED TO DETERMINE IF THE
1131				;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
1132				
1133	000302	STRTDI:	.= \$ADERR+1	
1134	000304		.= STRTDI+2	
1135		LOWBNK:		
1136	000306		.= LOWBNK+2	
1137		PASFLG:		
1138				;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
1139				;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
1140				;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
1141				
1142	000310		.= PASFLG+2	
1143	000312	ENDSTK:		
1144			.= ENDSTK+2	
1145	000312	PBNK:		;HOLDS BANK UNDER TEST FOR 'TST BNK XX' PRINTOUT.
1146	000312	DECWRD:		
1147	000314		.= DECWRD+2	
1148	000	TYPCNT:	.BYTE 0	;THIS BYTE DETERMINES THE NUMBER OF WORDS
1149				;TO BE TYPED
1150	000315	SAVKBB:	.BYTE 0	;THIS LOCATION IS USED TO SAVE THE CHARACTER
1151				;HIT BY THE OPERATOR
1152				
1153				
1154				
1155	177560	TKS=	177560	
1156	177562	\$KBB=	177562	
1157	177564	\$TPS=	177564	
1158	177566	\$TPB=	177566	
1159	177572	SRO=	177572	
1160	000316	BAKPAT:	.WORD 377	;BACKGROUND PATTERN WRITTEN TO MEMORY.
1161				
1162	000320	SWAPAT:	.WORD	
1163	000322	RELBOT:	BEGIN-50	;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.
1164				
1165				
1166				
1167	000324			
1168	000326			
1169				
1170	000330			

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 23
 CZKMAF.P11 05-MAR-79 09:02 APT PARAMETER BLOCK

SEQ 0023

```

1171 000332 037776      HIGHADD:      37776      ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1172                      ;*****
1173
1174 000334 000000      $HIMAX: 0      ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1175 000336 017776      $MAXM: 17776    ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1176
1177 000340 000000      MAXMEM: .WORD    ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1178
1179 000342 000000      SAVMAX: .WORD
1180 000344 000000      SAVR4:  .WORD
1181 000346 000000      SAVR5:  .WORD
1182
1183                      ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1184 000350 000500      SAVR6: .WORD BEGIN ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1185 000352 000000      PARMAP: 0      ;MAP OF PARITY MODULES UNDER TEST.
1186 000354 000000      SAVLOC: 0      ;TEST 6 STORES ERROR INFO HERE
1187 000356 000000      PARSP: 0      ;SAVE SP DURING PARITY ERROR TRAP.
1188 000360 000000      PARPS: 0      ;SAVE PSW DURING PARITY ERROR TRAP.
1189                      ;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
1190                      ;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
1191                      ;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE.
1192                      ;IN THIS CRUDE FASHION.
1193
1194
1195                      ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT

```

1196 000400

1197

1198

1199

1200

1201 000400

1202 000400 000000

1203 000402 000000

1204 000404 000000

1205 000406 000000

1206 000410 000000

1207 000412 000000

1208 000414 000000

1209 000416 000000

1210 000420

1211 000420 000

1212 000421 000

1213 000422 000000

1214 000424 000000

1215 000426 000000

1216

1217

1218

1219

1220

1221

1222 000430 000

1223 000431 000

1224

1225

1226

1227

1228 000432 000000

1229

1230 000434 000

1231 000435 000

1232 000436 000000

1233 000440 000

1234 000441 000

1235 000442 000000

1236 000444 000

1237 000445 000

1238 000446 000000

1239 000450

1240

1241

1242

1243

.=400
 .SBTTL APT MAILBOX-ETABLE

::*****

.EVEN

\$MAIL:

\$MSGTY: .WORD AMSGTY

\$FATAL: .WORD AFATAL

\$TESTN: .WORD ATESTN

\$PASS: .WORD APASS

\$DEVCT: .WORD ADEVCT

\$UNIT: .WORD AUNIT

\$MSGAD: .WORD AMSGAD

\$MSGLG: .WORD AMSGLG

\$ETABLE:

\$ENV: .BYTE AENV

\$ENVM: .BYTE AENVM

\$SWREG: .WORD ASWREG

\$USWR: .WORD AUSWR

\$CPUOP: .WORD ACPUOP

::*

::*

::*

::*

::*

::*

\$MAMS1: .BYTE AMAMS1

\$MTYP1: .BYTE AMTYP1

::*

::*

::*

::*

\$MADR1: .WORD AMADR1

::*

\$MAMS2: .BYTE AMAMS2

\$MTYP2: .BYTE AMTYP2

\$MADR2: .WORD AMADR2

\$MAMS3: .BYTE AMAMS3

\$MTYP3: .BYTE AMTYP3

\$MADR3: .WORD AMADR3

\$MAMS4: .BYTE AMAMS4

\$MTYP4: .BYTE AMTYP4

\$MADR4: .WORD AMADR4

\$ETEND:

.MEXIT

::APT MAILBOX

::MESSAGE TYPE CODE

::FATAL ERROR NUMBER

::TEST NUMBER

::PASS COUNT

::DEVICE COUNT

::I/O UNIT NUMBER

::MESSAGE ADDRESS

::MESSAGE LENGTH

::APT ENVIRONMENT TABLE

::ENVIRONMENT BYTE

::ENVIRONMENT MODE BITS

::APT SWITCH REGISTER

::USER SWITCHES

::CPU TYPE,OPTIONS

BITS 15-11=CPU TYPE

11/04=01,11/05=02,11/20=03,11/40=04,11/45=05

11/70=06,PDQ=07,Q=10

BIT 10=REAL TIME CLOCK

BIT 9=FLOATING POINT PROCESSOR

BIT 8=MEMORY MANAGEMENT

::HIGH ADDRESS,M.S. BYTE

::MEM. TYPE,BLK#1

MEM.TYPE BYTE -- (HIGH BYTE)

900 NSEC CORE=001

300 NSEC BIPOLAR=002

500 NSEC MOS=003

::HIGH ADDRESS,BLK#1

MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE' ABOVE

::HIGH ADDRESS,M.S. BYTE

::MEM. TYPE,BLK#2

::MEM.LAST ADDRESS,BLK#2

::HIGH ADDRESS,M.S. BYTE

::MEM. TYPE,BLK#3

::MEM.LAST ADDRESS,BLK#3

::HIGH ADDRESS,M.S. BYTE

::MEM. TYPE,BLK#4

::MEM.LAST ADDRESS,BLK#4

::*****

.SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.


```
1244      ;:*****
1245 000450 177570      SWR: 177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
1246
1247      ;=500
1248 000500 010706      BEGIN: MOV PC,SP ;SET UP STACK POINTER TO EQUAL BEGIN ADDRESS
1249
1250 000502 005746      TST -(SP)
1251 000504 010637 000350      MOV SP,@#SAVR6 ;SAVE SP FOR FUTURE USE
1252 000510 012737 000070 000024      MOV #PWRDN,@#24 ;PREPARE FOR ANY FUTURE POWER DOWN
1253 000516 005037 000300      CLR @#SPRERR
1254 000522 005037 000314      CLR @#TYPCNT
1255 000526 012700 000114      MOV #114,R0 ;PREPARE TO SETUP PARITY TRAP VECTOR
1256 000532 012710 005504      MOV #PARERR-,-6,(R0)
1257 000536 060720      ADD PC,(R0)+ ;TO PARERR
1258 000540 012710 000340      MOV #340,(R0) ;AND PSW OF 340
1259 000544 105737 000405      TSTB @#REL ;IS THIS CODE RELOCATED?
1260 000550 100002      BPL ONEPAS ;BRANCH IF NO
1261 000552 000167 000614      JMP TSTREL ;THIS CODE IS RELOCATED SO GET TEST SIZE.
1262
1263 000556 005737 000406      ONEPAS: TST @#SPASS ;IS THIS THE FIRST PASS?
1264 000562 001402      BEQ TSTRP ;BRANCH IF YES (TEST TRAP CATCHER ADDRESSES)
1265 000564 000167 000430      JMP SETSTK ;GET THE TEST SIZE
1266 000570 012704 007744      TSTRP: MOV #ENDPRG,R4 ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1267 000574 012700 000377      MOV #377,R0
1268 000600 010037 000316      MOV R0,@#BAKPAT
1269 000604 005001      CLR R1
1270 000606 012124      2$: MOV (R1)+,(R4)+ ;SAVE FROM 0000 TO BEGIN-30 AT END OF PROGRAM FOR NOW
1271 000610 020127 000400      CMP R1,#$MAIL
1272 000614 103774      BLO 2$
1273 000616 005741      3$: TST -(R1) ;PREPARE TO TEST THE TRAP VECTORS
1274 000620 010011      4$: MOV R0,(R1) ;CHECK THE TRAP VECTORS FOR THE CAPABILITY
1275 ;OF HOLDING 0'S & 1'S
1276 000622 020011      CMP R0,(R1) ;IS THE DATA OK?
1277 000624 001403      BEQ 6$ ;BRANCH IF YES
1278
1279 000626 004767 005334      JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1280 000632 000001      1 ;*****ERROR NUMBER 1*****
1281
1282 000634 000300      6$: SWAB R0
1283 000636 001370      BNE 4$
1284 000640 005701      TST R1 ;IF WE HAVE NOT REACHED THE LOWEST MEMORY LOCATION
1285 000642 001365      BNE 3$ ;THEN REPEAT FROM 3$
1286 000644 012701 000400      MOV #$MAIL,R1
1287 000650 014441      8$: MOV -(R4),-(R1) ;RESTORE TRAP CATCHER ETC.
1288 000652 005701      TST R1
1289 000654 001375      BNE 8$
1290 000656 012700 000006      SETSWR: MOV #6,R0
1291 000662 012710 000340      MOV #340,(R0) ;SET UP TIME OUT TRAP PSW
1292 000666 012740 000700      MOV #4$,-(R0) ;AND THE RETURN ADDRESS
1293 000672 005777 177552      2$: TST @SWR ;DOES THE SWITCH REGISTER POINTED BY SWR EXIST ?
1294 000676 000404      BR 5$ ;BRANCH IF YES
1295 000700 022626      4$: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
1296 000702 012737 000176 000450      MOV #SWREG,@#SWR ;AND PLACE THE ADDRESS OF THE SWITCH REGISTER
1297 ;DESIGNED FOR THE COMPUTERS NOT HAVING HARDWARE
1298 ;SWITCH REGISTER AND RUNNING STAND ALONE
1299 000710 105737 000420      5$: TSTB @#SENV ;RUNNING UNDER APT?
```

```

1300 000714 001403          BEQ  APTSIZ      ;BRANCH IF NO
1301 000716 012737 000422 000450  MOV  #SSWREG,@#SWR ;SET SWR EQUAL TO APT SWITCH REGISTER.
1302
1303
1304
1305
1306      ;APTSIZ- THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE AND WHEN
1307      ;A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO GIVEN HIGH ADDRESS.
1308      ; IF APT DEFINES SIZE THE LOW TEST ADDRESS MUST=00000.(DUE TO ETABLE FORMAT)
1309      ;FLOW:
1310      ; IF BLOCK 4 (OR 3,2,1) TYPE NON ZERO THEN GET APT HIGH ADDRESS AND EXIT.
1311      ; ELSE SEND ERROR #3
1312      ;NOTE: THE MEMORY TYPE IS IGNORED SINCE ALL TESTS ARE RUN REGARDLESS OF MEMORY TYPE.
1313
1314 000724 012703 000340  APTSIZ: MOV  #MAXMEM,R3      ;POINT R3 TO MAXMEM.
1315 000730 013737 000330 000334  MOV  @#HIGHTWO,@#$HIMAX ;IN CASE NO SELF SIZING DONE.
1316 000736 013737 000332 000336  MOV  @#HIGHADD,@#$MAXM ;IN CASE NO SELF SIZING DONE.
1317 000744 105737 000421      TSTB  @#SENVN      ;DOES APT ALLOW SELF SIZING?
1318 000750 10C021          BPL  TRYSR      ;BRANCH IF YES
1319
1320 000752 012701 000451      MOV  #SMTYP4+4,R1 ;POINT R1 TO BLOCK TYPE 4(+4)
1321 000756 162701 000004 1$: SUB  #4,R1 ;POINT R1 TO NEXT BLOCK TYPE.
1322 000762 105711          TSTB  (R1) ;IS THE BLOCK TYPE NON ZERO?
1323 000764 001006          BNE  2$ ;BRANCH IF YES (MEMORY EXISTS)
1324 000766 020127 000431      CMP  R1,#SMTYP1 ;ALL APT BLOCK TYPES BEEN CHECKED?
1325 000772 101371          BHI  1$ ;BRANCH IF NO
1326
1327 000774 004767 005166          JSR  PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1328 001000 000002          2: ;*****ERROR NUMBER 2*****
1329
1330 001002 004767 006324 2$: JSR  PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO $MAXM + $HIMAX
1331 001006 004767 006320      JSR  PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO HIGHADD+HIGHTWO
1332 001012 000464      BRTPSZ: BR  TYPISZ ; TYPE THE SIZE OF MEMORY UNDER TEST
1333
1334 001014 032777 000100 177426 TRYSR: BIT  #100,@SWR ;USER DEFINED MEMORY TEST BOUNDARIES??
1335 001022 001060          BNE  TYPISZ ;BRANCH IF YES (DON'T SIZE MEMORY)
1336
1337
1338
1339
1340
1341 001024 010401          SLFSIZ: MOV  R4,R1 ;SETUP R1 AND R4 TO THE LOWEST ADDRESS OF MEMORY
1342 001026 012710 001072      MOV  #4$,(R0) ;SET UP RETURN ADDRESS FROM TIME OUT TRAP TO 4$
1343 001032 011111 2$: MOV  (R1),(R1) ;WRITE A MEMORY LOCATION INTO ITSELF AND TRAP IF NXM
1344 001034 062701 000002      ADD  #2,R1 ;ADD 2 TO THE ADDRESS POINTER
1345 001040 022701 170000      CMP  #170000,R1 ;CHECK IF BEYOND 30K MEMORY BOUNDARY
1346 001044 101372          BHI  2$ ;KEEP ON SIZING UP THE MEMORY UNTIL NXM TRAP
1347      ;(TIME OUT TRAP) IS ENCOUNTERED
1348      ;OR 30K BOUNDARY REACHED
1349 001046 005737 000042      TST  @#42 ;IF NOT XXDP
1350 001052 001410          BEQ  5$ ; THEN CONTINUE
1351 001054 023737 000042 000046      CMP  @#42,@#46 ; ELSE IF NOT XXDP CHAINING
1352 001062 001404          BEQ  5$ ; THEN CONTINUE
1353 001064 162701 010000      SUB  #10000,R1 ; ELSE SET MAX. MEM. AT 28K
1354 001070 000401          BR  5$
1355

```


BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0027

1356	001072	022626		4\$:	CMP	(SP)+,(SP)+	:RESTORE THE STACK POINTER
1357	001074	004767	005764	5\$:	JSR	PC,MMNG	:SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE
1358							:AND IF IT HAS TO BE TESTED
1359	001100	105737	000276		TSTB	@#MMVA	:SEE IF MEMORY MANAGEMENT HAS TO BE TESTED
1360	001104	001416			BEQ	12\$	:IF NO MEM. MANG. THEN GO TO 12\$
1361	001106	012710	001120	6\$:	MOV	#8\$,(R0)	:SET UP THE RETURN ADDRESS FROM TRAP TO 8\$
1362	001112	012701	020000		MOV	#20000,R1	:BEGIN CHECKING MEMORY ABOVE 28K
1363	001116	000745			BR	2\$	
1364	001120	022626		8\$:	CMP	(SP)+,(SP)+	:RESTORE STACK PCINTER
1365	001122	022701	160000		CMP	#160000,R1	:IF R1 DID NOT READ ALL THE LOCATIONS POINTED BY
1366							:PAGE ADDRESS REGISTER 6 THEN IT HAS REACHED THE
1367							:MAXIMUM AVAILABLE MEMORY
1368	001126	001005			BNE	12\$	:IN WHICH CASE GO TO 12\$
1369	001130	013702	172352		MOV	@#172352,R2	:PREPARE TO UPDATE MEMORY MANAGEMENT REGISTERS
1370	001134	004767	005730		JSR	PC,MMREG	:OTHERWISE GO TO UPDATE MEM. MANG. REGISTERS
1371	001140	000762			BR	6\$	
1372	001142	024341		12\$:	CMP	-(R3),-(R1)	:CAUSE R3 TO POINT TO LOCATION \$MAXM AND R1
1373							:TO THE MAXIMUM AVAILABLE MEMORY
1374	001144	004767	006072		JSR	PC,PUTADR	:GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
1375							:AT LOCATIONS \$MAXM AND \$HIMAX
1376	001150	024343			CMP	-(R3),-(R3)	:MAKE R3 POINT TO HIGHADD
1377	001152	004767	006064		JSR	PC,PUTADR	:PLACE THE ADDRESS IN R1 AT LOCATIONS HIGHADD
1378							:AND HIGHTWO
1379	001156	005743			TST	-(R3)	
1380	001160	005043			CLR	-(R3)	:CLEAR THE LOCATION LOWADD
1381	001162	005043			CLR	-(R3)	:AND LOWTWO
1382	001164	012720	000104	TYPsiz:	MOV	#BUSER,(R0)+	:SET UP VECTOR FOR ANY FUTURE TRAP
1383	001170	010403			MOV	R4,R3	:SET R3 TO POINT TO THE LOWEST AVAILABLE MEMORY
1384							:LOCATION
1385	001172	012701	000324		MOV	#LCWTWO,R1	
1386	001176	004767	005370		JSR	PC,PCRLF	:TYPE CR/LF
1387	001202	004767	005536		JSR	PC,OCTTYP	:TYPE LOW TEST ADDRESSE (LOWTWO+LOWADD)
1388	001206	004767	005272	TYPMEM:	JSR	PC,\$TYPE	:TYPE '-'
1389	001212	000055			.ASCIZ	/-/'	
1390					.EVEN		
1391	001214	004767	005524		JSR	PC,OCTTYP	:TYPE HIGHEST TEST ADDRESS (HIGHTWO+HIGHADD)
1392	001220	012703	000330	SETSTK:	MOV	#HIGHTWO,R3	:MAKE R3 POINT TO THE HIGH ORDER BITS OF TOP ADDRESS
1393	001224	004767	006116		JSR	PC,\$GTSIZ	:GET THE BITS 13-17 OF THE TOP ADDRESS
1394							:PLACED IN BITS 0-4 OF R2
1395	001230	010401			MOV	R4,R1	:SET R1 TO LOWEST TEST ADDRESS
1396							
1397	001232	062704	000022	4\$:	ADD	#18.,R4	:APPEND THE ERROR STACK FOR THE MEMORY UNDER
1398							:TEST TO THE END OF THE PROGRAM
1399	001236	005302			DEC	R2	
1400	001240	002374			BGE	4\$	
1401	001242	022737	167776	000336	CMP	#167776,@#\$MAXM	:CHECK IF THIS IS A 30K SYSTEM
1402	001250	001002			BNE	5\$	:BRANCH IF NOT
1403	001252	062704	000022		ADD	#18.,R4	:SAVE ANOTHER BANKS WORTH OF ERROR STACK
1404	001256	010437	000310	5\$:	MOV	R4,@#ENDSTK	:SAVE THE ADDRESS OF THE END OF THE ERROR STACK
1405	001262	005021		6\$:	CLR	(R1)+	:CLEAR THE ERROR STACK
1406	001264	020104			CMP	R1,R4	
1407	001266	101775			BLOS	6\$	
1408	001270	012737	157776	000340	MOV	#157776,@#MAXMEM	:SET MAXMEM TO MAXIMUM VIRTUAL ADDRESS
1409	001276	005723			TST	(R3)+	:TESTING MEMORY MANAGEMENT?
1410	001300	001005			BNE	SAVLDR	:BRANCH IF YES (GO SAVE LOADERS AT TOP OF VIRTUAL MEMORY
1411	001302	021327	170000		CMP	(R3),#170000	:IS THE VIRTUAL ADDRESS ABOVE 167776?

```
1412 001306 103002          BHIS  SAVLDR      ;BRANCH IF YES (GO SAVE LOADERS)
1413 001310 011363 000002      MOV      (R3),2(R3) ;OTHERWISE MAKE THE CONTENTS OF LOCATION MAXMEM
1414                                     ;EQUAL TO THE MAXIMUM AVAILABLE MEMORY
1415                                     ;AND FALL INTO SAVE LOADERS.
1416
1417 001314 004767 006100      SAVLDR: JSR      PC,CLRMM      ; DISABLE THE MEMORY MANAGEMENT UNIT
1418 001320 005723              TST      (R3)+          ;MAKE R3 TO POINT TO THE LOCATION MAXMEM
1419 001322 011305              MOV      (R3),R5         ;R5 CONTAINS THE ADDRESS OF MAXIMUM AVAILABLE MEM.
1420
1421                          ;IF ONLY 4K BEING TESTED DON'T SAVE LOADERS
1422
1423 001324 020527 017776          CMP      R5,#17776      ;ONLY TESTING 4K MAX?
1424 001330 103416              BLO      4$              ;BRANCH IF YES (DON'T SAVE LOADERS)
1425
1426 001332 162705 000276      3$: SUB      #276,R5        ;PREPARE TO SAVE 300 BYTES OF THE LOADERS
1427 001336 005737 000042          TST      @#42          ;IF RUNNING UNDER XXDP
1428 001342 001406              BEQ      2$              ; THEN CONTINUE
1429 001344 023737 000042 000046      CMP      @#42,@#46      ; ELSE IF NOT UNDER XXDP CHAIN MODE
1430 001352 001402              BEQ      2$              ; THEN CONTINUE
1431 001354 162705 005376          SUB      #<1502.*2>-276,R5 ; ELSE SAVE 1500. WORDS FOR XXDP CHAIN MODE
1432 001360 012524      2$: MOV      (R5)+,(R4)+          ;SAVE LOADER
1433 001362 020513              CMP      R5,(R3)
1434 001364 101775              BLOS     2$
1435 001366 012323      4$: MOV      (R3)+,(R3)+          ;SAVE THE CONTENTS OF LOCATION MAXMEM IN SAVMAX
1436 001370 010423              MOV      R4,(R3)+          ;AND THE CONTENTS OF R4 AT SAVR4
1437
1438 001372 010537 000346      TSTREL: MOV     R5,@#SAVR5      ;SAVE HIGHEST VIRTUAL ADDRESS+2
1439 001376 004767 006016      TSTSIZ: JSR      PC,CLRMM      ;GO TO DISABLE MEMORY MANAGEMENT UNIT
1440 001402 005745              TST      -(R5)            ;SET R5 BACK TO HIGHEST VIRTUAL ADDRESS
1441 001404 012703 000324      1$: MOV      #LOWTWO,R3      ;PREPARE TO LOAD R4 AND R5 WITH THE MEMORY BOUNDRIES
1442 001410 005723              TST      (R3)+            ;IF THE BITS 16,17 OF THE LOWEST LOCATION UNDER
1443                                     ;TEST ARE NON ZERO
1444 001412 001003              BNE      2$              ;THEN GO TO 2$
1445 001414 021327 157776          CMP      (R3),#157776      ;IF THE LOWEST LOCATION UNDER TEST IS HIGHER THAN
1446                                     ;157776 THEN GO TO TEST MEMORY MANAGEMENT
1447 001420 103411              BLO      4$
1448 001422 032777 010000 177020 2$: BIT      #10000,@SWR      ;IS MEMORY MANAGEMENT SELECTED?
1449 001430 001003              BNE      3$              ;YES ALL IS WELL
1450 001432 004767 004530          JSR      PC,FATERR        ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1451 001436 000003              3$
1452                                     ;*****ERROR NUMBER 3*****
1453 001440 000167 003512      3$: JMP      TSTM
1454 001444 020423      4$: CMP      R4,(R3)+          ;GO TO TEST MEMORY MANAGEMENT
1455                                     ;COMPARE TOP OF PROGRAM (WITH SAVED LOADERS) TO
1456                                     ;LOWEST LOCATION UNDER TEST
1457 001446 103002          BHIS  6$
1458 001450 016304 177776          MOV      -2(R3),R4        ;ADJUST R4 TO POINT TO THE LOWEST LOCATION UNDER TEST
1459 001454 005723      6$: TST      (R3)+          ;IF BITS 16-17 OF HIGHEST LOCATION TO BE TESTED
1460 001456 001003              BNE      8$              ;ARE NON ZERO THEN GO TO 8$
1461 001460 021305              CMP      (R3),R5          ;OTHERWISE SEE IF THE HIGHEST LOCATION TO BE
1462                                     ;TESTED IS HIGHER THAN HIGHEST VIRTUAL ADDRESS
1463 001462 101001              BHI      8$              ;IF SO THEN GO TO 8$
1464 001464 011305              MOV      (R3),R5          ;MODIFY R5
1465 001466 105737 000405      8$: TSTB     @#REL          ;ARE WE RELOCATED.?
1466 001472 100014              BPL      10$            ;BRANCH IF NO
1467 001474 013704 000322          MOV      @#RELBOT,R4      ;SET BOTTOM TEST ADDRESS WHEN RELOCATED.
```



```

1468 001500 020527 017776      CMP      R5,#17776      ;ARE WE RELOCATED IN BANK 0?
1469 001504 103402              BLO       9$          ;BRANCH IF YES
1470 001506 012705 017776      MOV      #17776,R5      ;ELSE SET HIGH MEMORY UNDER TEST=4K
1471                               9$:      CMP      R4,R5      ;IS LOW LIMIT LOWER THAN HIGH LIMIT?
1472 001512 020405              BLO       10$         ;BRANCH IF YES
1473 001514 103403              JSR      PC,FATERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1474 001516 004767 004444      4          ;*****ERROR NUMBER 4*****
1475 001522 000004
1476
1477 001524 012703 000342      10$:     MOV      #SAVMAX,R3
1478 001530 011343              MOV      (R3),-(R3)      ;RESTORE THE CONTENTS OF MAXMEM
1479 001532 062713 000002      MEMTST: ADD     #2,(R3)      ;MAKE THE CONTENTS OF MAXMEM = MAXIMUM AVAILABLE
1480                               ;MEMORY +2
1481 001536 005725              TST      (R5)+          ;AND SET R5=MAX MEMORY+2
1482
1483                               ;CLEAR MEMORY UNDER TEST
1484
1485 001540 010500      CLRMEM: MOV      R5,R0      ;MOVE HIGH ADDRESS TO R0
1486 001542 005040      2$:      CLR      -(R0)        ;BEGIN CLEARING THE MEMORY FROM THE TOP
1487 001544 020004              CMP      R0,R4        ;UNTIL THE BOTTOM IS REACHED
1488 001546 101375              BHI       2$
1489 001550 012702 000001      MOV      #1,R2      ;SET R2 TO ENABLE PARITY MODULE CODE.
1490 001554 004767 005740      JSR      PC,PARITY      ;ENABLE PARITY IF WANTED AND AVAILABLE.
1491 001560 012702 000316      MOV      #BAKPAT,R2
1492 001564 012212              MOV      (R2)+,(R2)      ;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
1493 001566 000312              SWAB      (R2)
1494 001570 017702 176654      MOV      @SWR,R2      ;LOAD R2 WITH THE OPTIONS STORED AT $SWREG
1495 001574 042702 177760      BIC      #177760,R2      ;ONLY LEAVE THE LOWER 4 BITS OF $SWREG IN R2 TO GO TO
1496                               ;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
1497
1498
1499
1500                               ;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
1501
1502 001600 005037 000306      CONT:   CLR      @#PASFLG      ;INIT SUBTEST PASS FLAG.
1503 001604 110237 000404      MOV      R2,@#STESTN      ;SET UP $STESTN WITH THE TEST NUMBER GOING
1504                               ;TO BE EXECUTED
1505 001610 010401      LOOP:   MOV      R4,R1      ;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
1506 001612 010400      MOV      R4,R0      ;PLACE THE ADDRESS OF THE LOWEST LOCATION UNDER
1507                               ;TEST IN R0
1508 001614 010403              MOV      R4,R3      ;AND IN R3
1509 001616 006302              ASL      R2
1510 001620 060702              ADD      PC,R2
1511 001622 066207 000004      ADD      TBL-,(R2),PC      ;GO TO THE TEST #
1512                               ;STORED IN BITS 0-3 OF SWITCH REGISTER
1513
1514
1515 001626 000102      TBL:      TST0-TBL      ;RELATIVE ADDRESS OF TEST # 0
1516 001630 000340      TST1-TBL      ;RELATIVE ADDRESS OF TEST # 1
1517 001632 000440      TST2-TBL      ;RELATIVE ADDRESS OF TEST # 2
1518 001634 000550      TST3-TBL      ;RELATIVE ADDRESS OF TEST # 3
1519 001636 001016      TST4-TBL      ;RELATIVE ADDRESS OF TEST # 4
1520 001640 001126      TST5-TBL      ;RELATIVE ADDRESS OF TEST # 5
1521 001642 001274      TST6-TBL      ;RELATIVE ADDRESS OF TEST # 6
1522 001644 001430      TST7-TBL      ;RELATIVE ADDRESS OF TEST # 7
1523 001646 001654      TST10-TBL     ;RELATIVE ADDRESS OF TEST # 10

```

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 30

CZKMAF.P11 05-MAR-79 09:02

BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0030

1524 001650 002204
1525 001652 002256
1526 001654 002530
1527 001656 003156

TST11-TBL
TST12-TBL
TST13-TBL
RELOC-TBL

:RELATIVE ADDRESS OF TEST # 11
:RELATIVE ADDRESS OF TEST # 12
:RELATIVE ADDRESS OF TEST # 13
:RELATIVE ADDRESS OF ROUTINE 'RELOC'

1528
1529
1530
1531
1532

:R5 IS POINTING TO THE TOP OF THE MEMORY TO BE TESTED+2
:R4 & R0 ARE POINTING TO THE LOWEST ADDRESS OF MEMORY TO BE TESTED


```

1533          : *      SCOPE ROUTINE
1534          : *      -----
1535          : *
1536          : *
1537          : *      PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
1538          : *      IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE.
1539          : *      IF SR= 2000 (BIT10) THEN HALT
1540          : *      IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SR BITS<3:0>
1541          : *      ELSE CONTINUE TO NEXT TEST.
1542          : *
1543          : *
1544
1545 001660 105737 000420      TSTSCP: TSTB    @#$ENV      ;ARE WE RUNNING UNDER APT?
1546 001664 001002          BNE      CNTSCP      ;IF SO THEN GO TO CNTSCP
1547 001666 004767 006002          JSR      PC,CHECKC ;TEST FOR CONTROL-C AND IF TYPED GO
1548                                     ;PRINT ERROR HISTORY AND HALT AT FATHLT.
1549 001672 113702 000404      CNTSCP: MOVB    @#$TESTN,R2 ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
1550                                     ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
1551                                     ;OF R2 WILL BE 0
1552 001676 005237 000410          INC      @#$DEVCT ;TELL APT WE ARE STILL RUNNING OKAY
1553 001702 032777 002000 176540      BIT      #2000,@SWR ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
1554 001710 001401          BEQ      TSTGO      ;IF NOT THEN GO TO 2$
1555 001712 000000          SWHALT: HALT      ;HALT AT END OF TEST SWITCH SET.
1556
1557 001714 032777 040000 176526      TSTGO: BIT    #40000,@SWR ;IS THE PROGRAM GOING TO LOOP ON TEST
1558 001722 001332          BNE      LOOP      ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
1559 001724 105202          INCB    R2
1560 001726 000724          BR      CONT      ;GO TO CONT AND CONTINUE EXECUTING THE NEXT TEST
  
```

```

1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573 001730 105737 000404
1574 001734 001403
1575 001736 004767 004224
1576 001742 000005
1577
1578 001744 012703 177777
1579 001750 010401
1580 001752 010310
1581 001754 020001
1582 001756 001417
1583 001760 005711
1584 001762 001430
1585 001764 020311
1586
1587
1588 001766 001004
1589 001770 012767 000006 000042
1590
1591 001776 000403
1592 002000
1593 002000 012767 000007 000032
1594
1595 002006 010046
1596 002010 105237 000301
1597 002014 000407
1598 002016 020311
1599 002020 001411
1600 002022 012767 000010 000010
1601
1602 002030 010046
1603 002032 010300
1604 002034 004767 003570
1605 002040 000000
1606 002042 012600
1607
1608 002044 013706 000350
1609 002050 062701 020000
1610
1611
1612 002054 020105
1613
1614 002056 103736
1615
1616 002060 105737 000421

```

```

*****
:TEST 0 TEST FOR PROPER BANK SELECTION
:*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
: OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
: LOCATION UNDER TEST
:*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
: 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
: LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
: [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
:*(3) THIS TEST ALSO CHECKS TO SEE THAT NONE OF THE NON EXIST-
: ING BANK RESPOND WHEN THEY ARE ADDRESSED
*****
TST0: TSTB @#STESTN ;CHECK FOR PROPER TEST SEQUENCE
      BEQ .+10
      JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
      5 ;*****ERROR NUMBER 5*****

      MOV #177777,R3
1$: MOV R4,R1 ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
    MOV R3,(R0) ;SET ALL THE BITS AT (R0)
2$: CMP R0,R1 ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
    BEQ 4$ ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
    TST (R1) ;OTHERWISE CHECK (R1) FOR ALL 0'S
    BEQ 5$
    CMP R3,(R1) ;IF R1 IS NOT EQUAL TO R0 AND (R1)
                ;DOES NOT CONTAIN ALL 0'S THEN
                ;CHECK TO SEE IF (R0) = (R1)
    BNE 3$
    MOV #6,12$ ;*ERROR* SETUP ERROR NO. IN 12$
                ;*****ERROR NUMBER #6*****

    BR 10$
3$: MOV #7,12$ ;*ERROR* SETUP ERROR NO. IN 12$
    ;*****ERROR NUMBER #7*****

10$: MOV R0,-(SP) ;SAVE R0 ON STACK
    INCB @#$ADERR ;AN ADDRESSING ERROR IS SUSPECTED
    BR 11$
4$: CMP R3,(R1) ;CHECK (R1) FOR ALL 1'S
    BEQ 5$
    MOV #10,12$ ;*ERROR* SETUP ERROR NO. IN 12$
                ;*****ERROR NUMBER #10*****

    MOV R0,-(SP) ;SAVE R0 ON STACK
    MOV R3,R0
11$: JSR PC,ERROR ;GO TO THE ERROR SUBROUTINE
12$: .WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
    MOV (SP)+,R0 ;RESTORE R0

5$: MOV @#SAVR6,SP ;RESTORE THE STACK POINTER
    ADD #20000,R1 ;CAUSE R1 TO POINT TO THE SAME CHIP
                ;LOCATION IN THE NEXT 4K BANK OF MEMORY
                ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
    CMP R1,R5 ;COMPARE R1 WITH THE HIGHEST MEMORY
                ;LOCATION WHICH IS STORED IN R5
    BLO 2$ ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$

    TSTB @#SENVN ;HAS APT INHIBITED SIZING?

```


Address	Op Code	Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10	Op 11	Op 12	Op 13	Op 14	Op 15	Op 16	Op 17	Op 18	Op 19	Op 20	Op 21	Op 22	Op 23	Op 24	Op 25	Op 26	Op 27	Op 28	Op 29	Op 30	Op 31	Op 32	Op 33	Op 34	Op 35	Op 36	Op 37	Op 38	Op 39	Op 40	Op 41	Op 42	Op 43	Op 44	Op 45	Op 46	Op 47	Op 48	Op 49	Op 50	Op 51	Op 52	Op 53	Op 54	Op 55	Op 56	Op 57	Op 58	Op 59	Op 60	Op 61	Op 62	Op 63	Op 64	Op 65	Op 66	Op 67	Op 68	Op 69	Op 70	Op 71	Op 72	Op 73	Op 74	Op 75	Op 76	Op 77	Op 78	Op 79	Op 80	Op 81	Op 82	Op 83	Op 84	Op 85	Op 86	Op 87	Op 88	Op 89	Op 90	Op 91	Op 92	Op 93	Op 94	Op 95	Op 96	Op 97	Op 98	Op 99	Op 100	Op 101	Op 102	Op 103	Op 104	Op 105	Op 106	Op 107	Op 108	Op 109	Op 110	Op 111	Op 112	Op 113	Op 114	Op 115	Op 116	Op 117	Op 118	Op 119	Op 120	Op 121	Op 122	Op 123	Op 124	Op 125	Op 126	Op 127	Op 128	Op 129	Op 130	Op 131	Op 132	Op 133	Op 134	Op 135	Op 136	Op 137	Op 138	Op 139	Op 140	Op 141	Op 142	Op 143	Op 144	Op 145	Op 146	Op 147	Op 148	Op 149	Op 150	Op 151	Op 152	Op 153	Op 154	Op 155	Op 156	Op 157	Op 158	Op 159	Op 160	Op 161	Op 162	Op 163	Op 164	Op 165	Op 166	Op 167	Op 168	Op 169	Op 170	Op 171	Op 172	Op 173	Op 174	Op 175	Op 176	Op 177	Op 178	Op 179	Op 180	Op 181	Op 182	Op 183	Op 184	Op 185	Op 186	Op 187	Op 188	Op 189	Op 190	Op 191	Op 192	Op 193	Op 194	Op 195	Op 196	Op 197	Op 198	Op 199	Op 200	Op 201	Op 202	Op 203	Op 204	Op 205	Op 206	Op 207	Op 208	Op 209	Op 210	Op 211	Op 212	Op 213	Op 214	Op 215	Op 216	Op 217	Op 218	Op 219	Op 220	Op 221	Op 222	Op 223	Op 224	Op 225	Op 226	Op 227	Op 228	Op 229	Op 230	Op 231	Op 232	Op 233	Op 234	Op 235	Op 236	Op 237	Op 238	Op 239	Op 240	Op 241	Op 242	Op 243	Op 244	Op 245	Op 246	Op 247	Op 248	Op 249	Op 250	Op 251	Op 252	Op 253	Op 254	Op 255	Op 256	Op 257	Op 258	Op 259	Op 260	Op 261	Op 262	Op 263	Op 264	Op 265	Op 266	Op 267	Op 268	Op 269	Op 270	Op 271	Op 272	Op 273	Op 274	Op 275	Op 276	Op 277	Op 278	Op 279	Op 280	Op 281	Op 282	Op 283	Op 284	Op 285	Op 286	Op 287	Op 288	Op 289	Op 290	Op 291	Op 292	Op 293	Op 294	Op 295	Op 296	Op 297	Op 298	Op 299	Op 300	Op 301	Op 302	Op 303	Op 304	Op 305	Op 306	Op 307	Op 308	Op 309	Op 310	Op 311	Op 312	Op 313	Op 314	Op 315	Op 316	Op 317	Op 318	Op 319	Op 320	Op 321	Op 322	Op 323	Op 324	Op 325	Op 326	Op 327	Op 328	Op 329	Op 330	Op 331	Op 332	Op 333	Op 334	Op 335	Op 336	Op 337	Op 338	Op 339	Op 340	Op 341	Op 342	Op 343	Op 344	Op 345	Op 346	Op 347	Op 348	Op 349	Op 350	Op 351	Op 352	Op 353	Op 354	Op 355	Op 356	Op 357	Op 358	Op 359	Op 360	Op 361	Op 362	Op 363	Op 364	Op 365	Op 366	Op 367	Op 368	Op 369	Op 370	Op 371	Op 372	Op 373	Op 374	Op 375	Op 376	Op 377	Op 378	Op 379	Op 380</
---------	---------	------	------	------	------	------	------	------	------	------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	-------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	--------	----------

```

1650
1651
1652
1653
1654
1655 002166 122737 000001 000404 TST1: CMPB #1, @#TESTN ;CHECK FOR PROPER TEST SEQUENCE
1656
1657
1658 002174 001403 BEQ .+10
1659 002176 004767 003764 JSR PC, SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1660 002202 000012 12 *****ERROR NUMBER 12*****
1661
1662 002204 012700 000001 1$: MOV #1, R0
1663 002210 010002 MOV R0, R2 ;SET R2=1
1664 002212 010011 2$: MOV R0, (R1) ;MOV 1 AT LOCATION (R1)
1665 002214 020011 3$: CMP R0, (R1) ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
1666 002216 001403 BEQ 4$
1667 002220 004767 003404 JSR PC, ERROR ;*ERROR* REPORT ERROR MESSAGE
1668 002224 000013 13 *****ERROR NUMBER 13*****
1669
1670
1671 002226 005702 4$: TST R2 ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
1672 002230 001406 BEQ 5$ ;IF SO THEN GO TO 5$
1673 002232 006300 ASL R0 ;SHIFT THE 1 BROUGHT IN AT 1$ IN
1674 ;DATA DIRECTION
1675 002234 103366 BCC 2$ ;IF THE 1 HAS NOT BEEN SHIFTED THRU
1676 ;THE 16 DATA BITS THEN REPEAT FROM 2$
1677 002236 005002 CLR R2 ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
1678 002240 012700 177776 MOV #177776, R0
1679 002244 000762 BR 2$
1680
1681 002246 000261 5$: SEC ;SET C BIT
1682 002250 006100 ROL R0 ;SHIFT A 0 16 TIMES IN DATA DIRECTION
1683 002252 103757 BCS 2$ ;IF THE 0 HAS NOT BEEN SHIFTED THRU
1684 ;THE 16 DATA BITS THEN REPEAT FROM 2$
1685 002254 062701 020000 ADD #20000, R1 ;OTHERWISE GO TO THE NEXT BANK OF
1686 ;4K MEMORY AND REPEAT THE TEST
1687 002260 020105 CMP R1, R5
1688 002262 103750 BLO 1$
1689 002264 000737 END1: BR END0
1690

```



```

1691
1692
1693
1694
1695
1696
1697
1698
1699 002266 122737 000002 000404 TST2: CMPB #2,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
1700
1701 002274 001403 BEQ .+10
1702 002276 004767 003664 JSR PC,SEQERR ;*ERROR*: REPORT ERROR MESSAGE AND HALT AT FATHLT
1703 002302 000014 14 ;*****ERROR NUMBER 14*****
1704
1705 002304 013700 000316 1$: MOV @#BAKPAT,R0
1706 002310 110021 MOVB R0,(R1)+
1707 002312 113721 000317 MOVB @#BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
1708 002316 020105 CMP R1,R5
1709 002320 103771 BLO 1$
1710
1711 002322 020041 2$: CMP R0,-(R1) ;TEST THE MEMORY TO SEE IF IT CONTAINS
1712 ;THE WORD STORED IN BAKPAT
1713 002324 001416 BEQ 8$
1714 002326 062701 000002 ADD #2,R1
1715 002332 123741 000317 CMPB @#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
1716 002336 001402 BEQ 4$
1717 002340 120041 CMPB R0,-(R1) ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
1718 002342 001002 BNE 6$
1719 002344 105237 000301 4$: INCB @#$ADERR ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
1720 002350 042701 000001 6$: BIC #1,R1 ;MAKE THE ADDRESS IN R1 EVEN
1721 002354 004767 003250 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1722 002360 000015 15 ;*****ERROR NUMBER 15*****
1723
1724 002362 020104 8$: CMP R1,R4 ;KEEP ON TESTING THE MEMORY UNTIL
1725 002364 101356 BHI 2$ ;R1 EQUALS THE LOWEST ADDRESS
1726 002366 000337 000316 SWAB @#BAKPAT ;CHANGE THE DATA PATTERN
1727 002372 001744 BEQ 1$ ;IF THE DATA PATTERN DOES NOT HAVE LOW
1728 ;BYTE =0 THEN FALL THRU
1729 002374 000733 END2: BR END1
1730
1731 ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
1732

```

```

1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744 002376 122737 000003 000404 TST3: CMPB #3, @#STESTN ;CHECK FOR PROPER TEST SEQUENCE
1745 002404 001403 BEQ .+10
1746 002406 004767 003554 JSR PC, SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1747 002412 000016 16 ;*****ERROR NUMBER 16*****
1748
1749 002414 005003 CLR R3
1750 002416 004737 000120 2$: JSR PC, @#WRTMEM ; WRITE MEMORY WITH THE BACKGROUND STORED
1751 ;AT LOCATION BAKPAT
1752 002422 005002 4$: CLR R2
1753 002424 050302 6$: BIS R3, R2 ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
1754 002426 020204 CMP R2, R4 ;IF R2 IS LESS THAN R4
1755 002430 103465 BLO 16$ ;THEN DO NOTHING
1756 002432 020205 CMP R2, R5 ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
1757 002434 103077 BHIS 20$ ;TESTED THEN EXIT THE TEST
1758 002436 000312 SWAB (R2) ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
1759 ;THE LOCATION POINTED BY R2
1760 002440 005001 CLR R1
1761 002442 050301 7$: BIS R3, R1
1762 002444 020104 CMP R1, R4 ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
1763 002446 103445 BLO 12$ ;THEN GO TO 12$
1764 002450 020105 CMP R1, R5
1765 002452 103053 BHIS 15$
1766 002454 020102 CMP R1, R2 ;CHECK THE MEMORY FOR CORRECT DATA
1767 002456 001431 BEQ 10$
1768 002460 020011 CMP R0, (R1) ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
1769 ;THE SAME WORD AS BAKPAT
1770 002462 001437 BEQ 12$ ;IN WHICH CASE GO BACK TO 12$
1771 002464 012767 000017 000032 MOV #17, 22$ ;*ERROR* SETUP ERROR NO. IN 22$
1772 ;*****ERROR NUMBER #17*****
1773 002472 010046 8$: MOV R0, -(SP) ;PLACE R0 ON THE STACK
1774 002474 000316 SWAB (SP)
1775 002476 022611 CMP (SP)+, (R1) ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
1776 ;AS A SWAPPED R0
1777 002500 001003 BNE 9$ ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
1778 ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
1779 ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
1780 ;ENTIRE WORD
1781 002502 012767 000020 000014 MOV #20, 22$ ;*ERROR* SETUP ERROR NO. IN 22$
1782 ;*****ERROR NUMBER #20*****
1783 002510 105237 000301 9$: INCB @#SADERR ;ADDRESSING PROBLEM IS DETECTED
1784 002514 010046 MOV R0, -(SP) ;SAVE R0
1785 002516 010200 MOV R2, R0 ;SET R0=GOOD ADDRESS FOR ERROR REPORT
1786 002520 004767 003104 JSR PC, ERROR ;GO TO THE ERROR SUBROUTINE
1787 002524 000000 22$: .WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1788 002526 012600 MOV (SP)+, R0 ;RESTORE R0
  
```


1789	002530	010011		MOV	R0,(R1)	:RESTORE (R1)
1790	002532	020037	000316	CMP	R0,@#BAKPAT	:IF THE CONTROL CAME HERE FROM 15\$-2 THEN
1791	002536	001411		BEQ	12\$	
1792	002540	000407		BR	11\$	:RETURN TO 11\$
1793	002542	000300		10\$: SWAB	R0	:MAKE R0 SAME AS SWAPPED BAKPAT
1794	002544	020011		CMP	R0,(R1)	:IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
1795						:EQUAL TO SWAPPED R0
1796	002546	001404		BEQ	11\$	:IN WHICH CASE GO BACK TO 11\$
1797	002550	012767	000021 177746	MOV	#21,22\$	:*ERROR* SETUP ERROR NO. IN 22\$
1798						:*****ERROR NUMBER #21*****
1799	002556	000745		BR	8\$	:AND GO TO 8\$
1800	002560	000300		11\$: SWAB	R0	:RESTORE R0 TO BAKPAT
1801	002562	040301		12\$: BIC	R3,R1	:TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
1802	002564	005701		TST	R1	:IF R1 IS 0 THEN PLACE A 1 IN R1
1803	002566	001001		BNE	13\$	:OTHERWISE GO TO 13\$
1804	002570	005201		INC	R1	
1805	002572	006101		13\$: ROL	R1	
1806	002574	020127	020000	CMP	R1,#20000	:IF R1 IS LESS THAN A 4K BOUNDARY
1807	002600	103720		BLO	7\$	:THEN REPEAT FROM 7\$
1808	002602	000312		15\$: SWAB	(R2)	:RESTORE (R2) TO BAKPAT
1809	002604	040302		16\$: BIC	R3,R2	:TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
1810						:STORED IN R2
1811	002606	005702		TST	R2	:IF R2 = 0 THEN MOVE A 1 TO R2
1812	002610	001001		BNE	18\$	:OTHERWISE GO TO 18\$
1813	002612	005202		INC	R2	
1814	002614	006102		18\$: ROL	R2	:SHIFT A ONE IN THE ADDRESS WORD
1815	002616	020227	020000	CMP	R2,#20000	:IS THE ADDRESS IN R2 MORE THAN THE BOUNDARY
1816						:OF 4K
1817	002622	103700		BLO	6\$	:IF NOT THEN GO TO 6\$
1818	002624	060203		ADD	R2,R3	:OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
1819	002626	020337	000340	CMP	R3,@#MAXMEM	:IF R3 IS POINTING TO A BANK THAT IS LOWER
1820						:THAN MAXMEM
1821	002632	103673		BLO	4\$	:THEN REPEAT FROM 4\$
1822	002634	000337	000316	20\$: SWAB	@#BAKPAT	
1823	002640	001656		BEQ	TST3	:REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
1824						:THE LOWER BYTE OF BAKPAT IS 0
1825	002642	000654		END3: BR	END2	

```

1826
1827
1828
1829
1830
1831
1832 002644 122737 000004 000404
1833 002652 001403
1834 002654 004767 003306
1835 002660 000022
1836
1837 002662 005003
1838 002664 010100
1839 002666 005703
1840 002670 001401
1841 002672 005100
1842 002674 010021
1843 002676 020105
1844 002700 103771
1845
1846 002702 020041
1847 002704 001405
1848 002706 105237 000301
1849
1850 002712 004767 002712
1851 002716 000023
1852
1853 002720 010100
1854 002722 162700 000002
1855 002726 005703
1856 002730 001401
1857 002732 005100
1858 002734 020104
1859 002736 101361
1860 002740 112737 000001 000306
1861 002746 005103
1862 002750 001345
1863
1864 002752 000733
1865

*****
*TEST 4 DUAL ADDRESS TEST B
* (1) THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
* AND READING THE ADDRESS IN THE LOCATION AND THEN
* WRITING AND READING ADDRESS COMPLEMENT
*****
TST4: CMPB #4,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
      BEQ .+10
      JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
      22 ;*****ERROR NUMBER 22*****

1$: CLR R3
   MOV R1,R0
   TST R3 ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
   BEQ 2$ ;IN THE LOCATION
   COM R0 ;OTHERWISE STORE COMPLEMENT
   MOV R0,(R1)+ ;OF THE ADDRESS
   CMP R1,R5 ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
   BLO 1$

3$: CMP R0,-(R1) ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
   BEQ 4$
   INCB @#SADERR ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
   ;BIT PROBLEM
   JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
   23 ;*****ERROR NUMBER 23*****

4$: MOV R1,R0
   SUB #2,R0 ;CHECK THAT THE ADDRESS IS STORED AT
   TST R3 ;LOCATION IF R3 IS NOT 0
   BEQ 5$ ;OTHERWISE CHECK FOR
   COM R0 ;ADDRESS COMPLEMENT
   CMP R1,R4
   BHI 3$
   MOVB #1,@#PASFLG ;SET PASFLG FOR ERROR REPORT.
   COM R3 ;COMPLEMENT THE CONTENTS OF R3
   BNE 1$ ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
   ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU

END4: BR END3

```



```

1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880 002754 122737 000005 000404 TST5: CMPB #5,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
1881
1882 002762 001403 BEQ .+10
1883 002764 004767 003176 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1884 002770 000024 24 ;*****ERROR NUMBER 24*****
1885
1886 002772 004737 000120 1$: JSR PC,@#WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1887 ;WORD STORED IN BAKPAT
1888 002776 020041 2$: CMP R0,-(R1) ;READ THE CONTENTS OF LOCATION POINTED BY R1
1889 003000 001403 BEQ 3$ ;TO SEE IF IT HAS THE SAME VALUE AS R0
1890 003002 004767 002622 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1891 003006 000025 25 ;*****ERROR NUMBER 25*****
1892
1893 003010 000300 3$: SWAB R0
1894 003012 010011 MOV R0,(R1) ;SWAP THE BYTES AT (R1)
1895 003014 021100 CMP (R1),R0 ;READ (R1) FOR CORRECT VALUE
1896 003016 001403 BEQ 4$
1897 003020 004767 002604 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
1898 003024 000026 26 ;*****ERROR NUMBER 26*****
1899
1900
1901 003026 000300 4$: SWAB R0 ;SWAP THE BYTES OF THE REGISTER
1902 ;CONTAINING BACKGROUND PATTERN
1903 003030 001023 BNE 9$ ;IF THE LOWER BYTE OF THE REGISTER
1904 ;IS NOT 0 THEN THE PROGRAM IS READING
1905 ;THE MEMORY TO CONTAIN A BACK GROUND OF
1906 ;BAKPAT AND WRITING THE SWAPPED WORD
1907 ;IN WHICH CASE GO TO 9$
1908
1909
1910
1911 003032 005703 5$: TST R3 ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1912 ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
1913 ;IF R3 EQUAL 0 THEN THE PROGRAM IS
1914 ;READING/WRITING MIN. TO MAX. OTHERWISE
1915 ;IT IS GOING IN MAX. TO MIN. DIRECTION
1916 003034 001023 BNE 10$ ;IF R3 IS NOT CLEAR THEN GO TO 10$
1917 003036 062701 000002 6$: ADD #2,R1 ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1918 003042 020105 CMP R1,R5 ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
1919 ;BE TESTED
1920 003044 103006 7$: BHS 8$ ;IF R1>R5 THEN GO TO 8$ OTHERWISE
1921 003046 020011 CMP R0,(R1) ;READ (R1) FOR THE CORRECT DATA
  
```

1922	003050	001757		BEQ	3\$		: WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1923							: AND REPEAT UNTIL R1 > R5
1924	003052	004767	002552	JSR	PC,ERROR		: *ERROR* REPORT ERROR MESSAGE
1925	003056	000027		27			: *****ERROR NUMBER 27*****
1926							
1927	003060	000753		BR	3\$		
1928	003062	105237	000306 :	INCB	@#PASFLG		
1929	003066	000300		SWAB	R0		
1930	003070	001742		BEQ	2\$		: IF THE LOWER BYTE OF R0 IS ALL 0'S
1931							: THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1932							: AND READING BAKPAT GOING FROM MAX. TO MIN.[PASFLG=4]
1933	003072	005103		COM	R3		: OTHERWISE CLEAR R0
1934	003074	010401		MOV	R4,R1		: PUT THE LOWEST TESTING ADDRESS IN R1
1935	003076	000763		BR	7\$		: AND BEGIN READING 0'S, WRITING 1'S AND
1936							: READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]
1937							
1938	003100	005703		TST	R3		: IF R3 IS NON 0, I.E. PASFLG=3
1939	003102	001353	9\$:	BNE	5\$		: THEN READ BAKPAT, WRITE
1940							: SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1941							: IN MIN. TO MAX. DIRECTION
1942	003104	020104		CMP	R1,R4		: OTHERWISE TEST IS PROCEEDING IN MAX. TO
1943							: MIN. DIRECTION.
1944	003106	101333		BHI	2\$		: KEEP ON LOOPING UNTIL R1=R4
1945	003110	105237	000306	INCB	@#PASFLG		
1946	003114	000300		SWAB	R0		
1947	003116	001753		BEQ	7\$		: IF R0 SWAPPED HAS LOWER BYTE=0
1948							: THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1949							: AND READ BAKPAT GOING FROM MIN. TO MAX.
1950	003120	000714	END5:	BR	END4		
1951							


```
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
```

```
*****
;*TEST 6      CELLS' VOLATILITY TEST
*****
;* (1)  THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
;* (2)  WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
;*      AND THEN INCREMENTS PASFLG
;* (3)  IT THEN READS/SWAPS BYTES/WRITES A LOCATION X FOR
;*      OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
;* (4)  REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
;* (5)  IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
;*      BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
;*      SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
;* (6)  REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
;*      BAKPAT INSTEAD OF BAKPAT.
*****
```

003122	122737	000006	000404	TST6:	CMPB	#6,@#TESTN	;CHECK FOR PROPER TEST SEQUENCE
					BEQ	.+10	
					JSR	PC,SEQERR	;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
					30		*****ERROR NUMBER 30*****
003130	001403						
003132	004767	003030					
003136	000030						
003140	004737	000120		RPT6:	JSR	PC,@#WRTMEM	;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
							;WORD STORED AT LOCATION BAKPAT
003144	005037	000306			CLR	@#PASFLG	
003150	010403			1\$:	MOV	R4,R3	;SET R3
003152	010401			2\$:	MOV	R4,R1	;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
003154	020011			3\$:	CMP	R0,(R1)	;CHECK (R1) FOR CORRECT DATA
003156	001403				BEQ	4\$	
003160	004767	002444			JSR	PC,ERROR	;*ERROR* REPORT ERROR MESSAGE
003164	000031				31		*****ERROR NUMBER 31*****
003166	062701	000002		4\$:	ADD	#2,R1	;INCREMENT R1 BY 2
003172	020105				CMP	R1,R5	;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
003174	103767				BLO	3\$	
003176	132737	000001	000306		BITB	#1,@#PASFLG	;CHECK TO SEE IF PASFLG=0 OR 2
003204	001002				BNE	5\$	
003206	105237	000306			INCB	@#PASFLG	;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
003212	020305			5\$:	CMP	R3,R5	;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
003214	103012				BHIS	7\$	
003216	012702	037776			MOV	#37776,R2	;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
003222	000313			6\$:	SWAB	(R3)	
003224	005302				DEC	R2	
003226	001375				BNE	6\$	
003230	010337	000354			MOV	R3,@#SAVLOC	;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
003234	062703	020000			ADD	#20000,R3	;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
							;R3 TO POINT TO A LOCATION IN THE NEXT
							;4K BANK OF MEMORY
003240	000744				BR	2\$	
003242	105237	000306		7\$:	INCB	@#PASFLG	;MAKE PASFLG=2
003246	000337	000316			SWAB	@#BAKPAT	;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
003252	001732				BEQ	RPT6	;THEN GO BACK TO THE LOCATION RPT6
003254	000721			END6:	BR	END5	

```

2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018 003256 122737 000007 000404 TST7: CMPB #7,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2019
2020 003264 001403 BEQ .+10
2021 003266 004767 002674 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2022 003272 000032 32 ;*****ERROR NUMBER 32*****
2023
2024 003274 005037 000306 2$: CLR @#PASFLG
2025 003300 010337 000304 MOV R3,@#LOWBNK ;LOWBNK CONTAINS ADDRESS OF THE LOWEST LOCATION
2026 ;IN THE 4K BANK THAT CAN BE TESTED
2027 003304 010302 MOV R3,R2
2028 003306 052702 017777 BIS #17777,R2 ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK
2029 003312 005202 INC R2 ;ADD 1 TO POINT IT TO NEXT BANK
2030 003314 001402 BEQ 3$ ;BRANCH IF ZERO (IT MUST BE A 30K SYSTEM)
2031 003316 020502 CMP R5,R2
2032 003320 103001 BHIS 4$ ;IF R2 IS GREATER THAN R5 THEN GO TO 4$
2033 003322 010502 3$: MOV R5,R2 ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION
2034 ;THAT CAN BE TESTED
2035 003324 010337 000302 4$: MOV R3,@#STRTDI ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE
2036 ;DIAGONAL
2037 003330 013701 000304 MOV @#LOWBNK,R1 ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K
2038 ;BANK
2039 003334 013700 000316 6$: MOV @#BAKPAT,R0 ;STORE THE CONTENTS OF BAKPAT IN R0
2040 003340 020103 CMP R1,R3 ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?
2041 003342 001010 BNE 10$ ;IF NOT THEN GO TO 10$
2042 003344 062703 000002 ADD #2,R3 ;THE FOLLOWING CODE IS USED TO PLACE THE
2043 003350 032703 000176 BIT #176,R3 ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL
2044 003354 001402 BEQ 8$ ;IN R3
2045 003356 062703 000200 ADD #200,R3
2046 003362 000300 8$: SWAB R0 ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN
2047 003364 132737 000001 000306 10$: BITB #1,@#PASFLG ;CONTENTS OF LOCATION PASFLG WILL BE EVEN IF THE
2048 ;MEMORY IS BEING WRITTEN AND IT WILL BE ODD
2049 ;IF IT IS ONLY BEING READ
2050 003372 001001 BNE 12$ ;IF IT IS BEING READ ONLY THEN GO TO 12$
2051 003374 010011 MOV R0,(R1) ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS
2052 ;OF R0
2053 003376 020011 12$: CMP R0,(R1) ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN
2054 ;PROPER DATA
2055 003400 001403 BEQ 14$ ;IF IT IS OK THEN GO TO 14$
2056 003402 004767 002222 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
2057 003406 000033 33 ;*****ERROR NUMBER 33*****
2058
2059 003410 062701 000002 14$: ADD #2,R1 ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION
2060 003414 020102 CMP R1,R2 ;IS IT THE END OF THE BANK ?
2061 003416 103746 BLO 6$ ;IF NOT THEN GO TO 6$
2062 003420 005237 000410 16$: INC @#DEVCT ;TELL APT WE ARE STIL RUNNING OKAY
  
```


2063	003424	105237	000306		INCB	@#PASFLG	
2064	003430	013703	000302		MOV	@#STRTDI,R3	:LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL
2065	003434	132737	000001	000306	BITB	#1,@#PASFLG	:HAS THE READ OF THE MEMORY BEEN DONE ?
2066	003442	001330			BNE	4\$	:IF NOT THEN GO TO 4\$
2067	003444	005723			TST	(R3)+	:ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL
2068	003446	020302			CMP	R3,R2	:AND UNLESS THE END OF THE BANK IS REACHED
2069	003450	103003			BHIS	18\$	:
2070	003452	105737	000306		TSTB	@#PASFLG	:OR THE DIAGONAL HAS BEEN ROTATED 64 TIMES
2071	003456	100322			BPL	4\$	:REPEAT FROM 4\$
2072	003460	013703	000304	18\$:	MOV	@#LOWBNK,R3	:MAKE R3 POINT TO THE LOWEST LOCATION IN THE
2073							:IN THE BANK UNDER TEST
2074	003464	000337	000316		SWAB	@#BAKPAT	
2075	003470	001715			BEQ	4\$	:AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE
2076							:SWAPPED BACK GROUND PATTERN THEN GO TO 4\$
2077	003472	010203			MOV	R2,R3	:MAKE THE PRESENT HIGH BOUNDARY AS THE NEXT
2078							:LOW BOUNDARY
2079	003474	020205			CMP	R2,R5	:UNLESS THE PRESENT HIGH BOUNDARY IS ALSO THE
2080							:HIGH BOUNDARY FOR THE MEMORY UNDER TEST
2081	003476	103676			BLO	2\$	
2082	003500	000665		END7:	BR	END6	

```

2083      ;*****
2084      ;*TEST 10      READ RECOVERY GALLOPING TEST/EVERY 64TH CELL
2085
2086      ;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
2087      ;*      STORED AT LOCATION BAKPAT
2088      ;*(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
2089      ;*      (LETS NAME IT 'A')
2090      ;*(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
2091      ;*(4) SWAPS BYTES FOR LOCATION 'A'.
2092      ;*(5) READS 'A', READS 'B'
2093      ;*(6) 'B' = 'B'+200 (MAKES 'B'=64TH CELL I.E. 200TH OCTAL
2094      ;*      LOCATION FROM THE PRESENT LOCATION OF 'B')
2095      ;*(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
2096      ;*      END OF THE 4K BANK OF THE MEMORY IN WHICH 'A' IS RESIDING
2097      ;*(8) A = A+2
2098      ;*(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
2099      ;*(10) GOES TO THE NEXT 4K BANK OF MEMORY AND REPEATS STEPS
2100      ;*      3-9 UNTIL THE END OF THE MEMORY
2101      ;*(11) AFTER EXECUTING THE TEST BYTES ARE SWAPPED AT
2102      ;*      LOCATION BAKPAT AND STEPS 1-10 ARE REPEATED
2103      ;*(12) IN THIS TEST R0 IS POINTING TO LOCATION 'A', R1 TO
2104      ;*      LOCATION 'B', R2 TO THE END OF THE 4K BANK IN WHICH THE
2105      ;*      TEST IS TAKING PLACE AND R3 TO THE LOWEST LOCATION IN THE
2106      ;*      COLUMN/ROW CONTAINING 'A' AND 'B'
2107      ;*(13) MOST OF THE CODE USED BY THIS TEST IS ALSO USED BY TEST 11
2108
2109      ;*****
2110 003502 122737 000010 000404 TST10: CMPB    #10,@#TESTIN ;CHECK FOR PROPER TEST SEQUENCE
2111
2112 003510 001403          BEQ      .+10
2113 003512 004767 002450      JSR      PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2114 003516 000034          34 ;*****ERROR NUMBER 34*****
2115
2116 003520 010402          MOV      R4,R2
2117 003522 052702 017776 RPT10: BIS      #17776,R2 ;SET R2 TO THE LOWEST MEMORY UNDER TEST
2118 ;*MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K
2119 003526 062702 000002 GALLOP: ADD      #2,R2 ;BANK FOR WHICH GALLOPING WILL BE PERFORMED
2120 003532 001402          BEQ      1$ ;INCREMENT R2 BY 2
2121 003534 020205          CMP      R2,R5 ;BR IF IT WENT TO 0 (IT MUST BE A 30K SYSTEM)
2122 003536 101401          BLOS     2$ ;IF THE HIGH BOUNDRY OF THE TEST IS HIGHER THAN
2123 003540 010502          1$: MOV      R5,R2 ;THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2
2124 003542 005046          2$: CLR      -(SP)
2125 003544 010200          MOV      R2,R0
2126 003546 013740 000316 4$: MOV      @#BAKPAT,-(R0) ;WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF
2127 ;BAKPAT
2128 003552 020003          CMP      R0,R3
2129 003554 101374          BHI      4$
2130 003556 010301          6$: MOV      R3,R1 ;R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT
2131 ;CAN BE TESTED IN THIS BLOCK
2132 003560 023710 000316      CMP      @#BAKPAT,(R0) ;BEFORE STARTING THE GALLOPING TEST FOR LOCATION
2133 ;(R0) CHECK IT
2134 003564 001410          BEQ      8$ ;CONTINUE IF OK
2135 003566 010001          MOV      R0,R1 ;OTHERWISE PREPARE TO REPORT THE ERROR
2136 003570 013700 000316      MOV      @#BAKPAT,R0
2137 003574 004767 002030      JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
2138 003600 000035          35 ;*****ERROR NUMBER 35*****

```


2139									
2140	003602	010011				MOV	R0,(R1)		;RESTORE THE CONTENTS OF (R1)
2141	003604	010100				MOV	R1,R0		;RESTORE R0
2142									
2143	003606	000310			8\$:	SWAB	(R0)		
2144	003610	031011			10\$:	BIT	(R0),(R1)		;CHECK TO SEE THAT NONE OF THE BITS SET
2145									;IN (R0) ARE SET IN (R1) AND VICE VERSA
2146	003612	020001				CMP	R0,R1		;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1
2147									
2148	003614	001412				BEQ	12\$		
2149	003616	021137	000316			CMP	(R1),@#BAKPAT		;CHECK THAT (R1) HAS BAKPAT IN IT
2150	003622	001407				BEQ	12\$		
2151	003624	010046				MOV	R0, -(SP)		;SAVE R0 ON STACK
2152	003626	013700	000316			MOV	@#BAKPAT,R0		;PLACE THE PATTERN WORD IN R0
2153	003632	004767	001772			JSR	PC,ERROR		;*ERROR* REPORT ERROR MESSAGE
2154	003636	000036					36		;*****ERROR NUMBER 36*****
2155									
2156	003640	012600				MOV	(SP)+,R0		;RESTORE R0
2157	003642	021037	000320		12\$:	CMP	(R0),@#SWAPAT		;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT
2158	003646	001412				BEQ	14\$		
2159	003650	010146				MOV	R1, -(SP)		;SAVE R1 ON THE STACK
2160	003652	010001				MOV	R0,R1		;MAKE R1 POINT TO THE FAILING LOCATION
2161	003654	013700	000320			MOV	@#SWAPAT,R0		;LOAD R0 WITH THE EXPECTED RESULT IN (R1)
2162	003660	004767	001744			JSR	PC,ERROR		;*ERROR* REPORT ERROR MESSAGE
2163	003664	000037					37		;*****ERROR NUMBER 37*****
2164									
2165	003666	010011				MOV	R0,(R1)		;RECOVER (R1) FROM THE ERROR
2166	003670	010100				MOV	R1,R0		;RESTORE R0
2167	003672	012601				MOV	(SP)+,R1		;AND RESTORE R1
2168	003674	122737	000011	000404	14\$:	CMPB	#11,@#TESTN		;IS THE PROGRAM EXECUTING TEST # 11 ?
2169	003702	001402				BEQ	16\$		;IF SO THEN GO TO 16\$
2170	003704	062701	000176			ADD	#176,R1		
2171	003710	062701	000002		16\$:	ADD	#2,R1		;MAKE R1 POINT TO THE NEXT ADJACENT CELL
2172	003714	020102				CMP	R1,R2		;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDARY
2173	003716	103734				BLO	10\$		;THEN REPEAT FROM 10\$
2174	003720	000320				SWAB	(R0)+		;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST
2175									;WAS BEING PERFORMED
2176	003722	122737	000011	000404		CMPB	#11,@#TESTN		;IS IT TEST 11 ?
2177	003730	001407				BEQ	17\$		;IF SO THEN GO TO 17\$
2178	003732	005723				TST	(R3)+		;OTHERWISE INCREMENT R3 BY 2
2179	003734	062716	000002			ADD	#2,(SP)		;FOR EVERY ROW/COLUMN TESTED ADD 2
2180	003740	105716				TSTB	(SP)		
2181	003742	100002				BPL	17\$		;UNTIL (SP) IS 200
2182	003744	161603				SUB	(SP),R3		;SUBTRACT 200 FROM R3
2183	003746	005016				CLR	(SP)		
2184	003750	032700	000177		17\$:	BIT	#177,R0		;AT A 64TH CALL BOUNDARY?
2185	003754	001002				BNE	18\$		;BRANCH IF NO
2186	003756	005237	000410			INC	@#DEVCT		;TELL APT WE ARE STILL RUNNING
2187	003762	020002			18\$:	CMP	R0,R2		;IF R0 HAS NOT REACHED THE END OF THE BOUNDARY
2188	003764	103674				BLO	6\$		;THEN REPEAT FROM 6\$
2189	003766	162603				SUB	(SP)+,R3		;RESTORE SP AND R3
2190	003770	000337	000320			SWAB	@#SWAPAT		
2191	003774	000337	000316			SWAB	@#BAKPAT		
2192	004000	001660				BEQ	2\$		;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 2\$
2193	004002	010203				MOV	R2,R3		;OTHERWISE MAKE THE PRESENT HIGH BOUNDARY AS THE
2194									;NEXT LOW BOUNDARY

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 46

CZKMAF.P11 05-MAR-79 09:02

T10

READ RECOVERY GALLOPING TEST/EVERY 64TH CELL

SEQ 0046

2195	004004	020205			CMP	R2,R5	
2196	004006	001410			BEQ	END10	:IF PREVIOUS HIGH BOUNDARY WAS THE END OF THE
2197							:TEST BOUNDARY THEN EXIT THE TEST
2198	004010	032702	017776		B!T	#17776,R2	:WAS IT A 4K BOUNDARY ?
2199	004014	001025			BNE	RPT11	:IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG
2200							:GALLOPING TEST DISABLED
2201	004016	122737	000011	000404	CMPB	#11,0#STESTN	:IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11
2202	004024	001421			BEQ	RPT11	
2203	004026	000635			BR	RPT10	
2204	004030	000623			END10: BR	END7	:OTHERWISE REPEAT TEST 10
2205							
2206							
2207							


```

2208 ::*****
2209 :*TEST 11 READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
2210
2211 :*(1) THIS TEST WRITES MEMORY WITH BAKPAT
2212 :*(2) THE TEST BEGINS AT THE LOWEST LOCATION BEING TESTED
2213 :* (LETS NAME IT 'B')
2214 :*(3) 'A' 'B' [MOVE THE ADDRESS OF 'B' TO THE POINTER FOR LOCATION 'A']
2215 :*(4) SWAPS BYTES FOR LOCATION 'A'
2216 :*(5) READS 'A', READS 'B'
2217 :*(6) 'B'='B'+2
2218 :*(7) IF GALLOPING OPTION BIT AT $SWREG IS HIGH THEN STEPS 4 AND 5
2219 :* ARE REPEATED UNTIL 'B' REACHES THE HIGHEST MEMORY LOCATION
2220 :* OF THE 4K BANK IN WHICH 'A' IS RESIDING, THEN 'A' IS
2221 :* DECREMENTED BY 2 AND AFTER MAKING 'B' TO POINT TO THE LOWEST
2222 :* LOCATION OF THE 4K MEMORY BANK CONTAINING 'A' STEPS 3,4,5 AND
2223 :* 6 ARE REPEATED UNTIL 'A' EQUALS THE END OF THE ENTIRE MEMORY
2224 :*(8) IF GALLOPING OPTION BIT IS NOT HIGH THEN STEPS 4 AND 5 ARE
2225 :* REPEATED UNTIL 'B' IS POINTING TO A CELL IN THE NEXT COLUMN
2226 :* IF SEQUENTIAL CELLS LIE ALONG THE ROW, OR THE NEXT ROW
2227 :* IF SEQUENTIAL CELLS LIE ALONG THE COLUMN, AT WHICH TIME
2228 :* STEPS 2,3,4,5 AND 7 ARE REPEATED UNTIL THE END OF THE MEMORY
2229 :*(9) TEST IS REPEATED FOR THE OPPOSITE BACKGROUND DATA
2230 :*(10) IN THIS TEST R0 POINTS TO LOCATION 'A', R1 TO LOCATION
2231 :* 'B', R2 TO THE HIGHEST LOCATION AND R3 TO THE LOWEST
2232 :* LOCATION IN A 64/4K CELL BOUNDARY
2233 :*(11) MOST OF THE CODE USED BY TEST 10 IS ALSO USED BY THIS TEST
2234
2235 ::*****
2236 004032 122737 000011 000404 TST11: CMPB #11,@$TESTN ;CHECK FOR PROPER TEST SEQUENCE
2237
2238 004040 001403 BEQ .+10
2239 004042 004767 002120 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2240 004046 000040 40 ;*****ERROR NUMBER 40*****
2241
2242 004050 010402 MOV R4,R2 ;MAKE R2 TO POINT TO THE LOWEST LOCATION
2243 ;UNDER TEST
2244 004052 105777 174372 TSTB @SWR ;LONG GALLOP ENABLED?
2245 004056 100004 BPL RPT11 ;BRANCH IF NO
2246 004060 004767 002526 JSR PC,PNTMES ;TYPE 'GLP'
2247 004064 046107 000120 .ASCIZ /GLP/
2248 004070 105777 174354 RPT11: TSTB @SWR ;LONG GALLOPING ENABLED?
2249 004074 100612 BMI RPT10 ;BRANCH IF YES
2250 ;TO RPT10
2251 004076 052702 000176 BIS #176,R2 ;OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
2252 ;TO GET THE HIGH BOUNDARY
2253
2254 004102 000611 BR GALLOP ; PERFORM GALLOPING TEST

```

```

2255 ::*****
2256 ;*TEST 12 WORST CASE TESTING FOR CORE MEMORY
2257 ;*(1) STARTING FROM THE LOWEST LOCATION UNDER TEST THE MEMORY
2258 ;* IS WRITTEN WITH A BACKGROUND OF BAKPAT, HOWEVER LOCATIONS
2259 ;* HAVING ADDRESS SUCH THAT EXCLUSIVE OR OF ADDRESS BITS 1 &
2260 ;* 8 = 1 ARE WRITTEN TO A VALUE OF SWAPPED BAKPAT
2261 ;*(2) STARTING FROM THE LOWEST LOCATION THE MEMORY IS CHECKED
2262 ;* TO CONTAIN THE CORRECT DATA AS EXPLAINED IN STEPS 3 & 4,
2263 ;* UNTILL THE HIGHEST LOCATION UNDER TEST IS REACHED
2264 ;*(3) READ EACH LOCATION FOR THE CORRECT CONTENT
2265 ;*(4) COMPLEMENT THE LOCATION AND READ IT; COMPLEMENT THE LOCATION
2266 ;* BACK TO ITS ORIGINAL VALUE AND READ IT AGAIN
2267 ;*(5) STARTING FROM THE HIGHEST LOCATION UNDER TEST REPEAT STEPS
2268 ;* 3 & 4 UNTIL THE LOWEST LOCATION UNDER TEST IS REACHED
2269 ;*(6) REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
2270 ;* OF ADDRESS BITS 8 & 13 =1 ARE WRITTEN TO SWAPPED BAKPAT
2271 ;*(7) REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
2272 ;* OF ADDRESS BITS 3 & 9 =1 ARE WRITTEN TO SWAPPED BAKPAT
2273 ;*(8) REPEAT STEPS 1-7 WITH A BACKGROUND OF SWAPPED BAKPAT AND
2274 ;* THE LOCATIONS TO BE WRITTEN TO SWAPPED BAKPAT WRITTEN TO
2275 ;* BAKPAT.
2276 ::*****
2277 004104 122737 000012 000404 TST12: CMPB #12,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
2278 004112 001403 BEQ .+10
2279 004114 004767 002046 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2280 004120 000041 41 *****ERROR NUMBER 41*****
2281
2282
2283 004122 012702 000002 MOV #2,R2 ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1
2284 004126 012703 000400 MOV #400,R3 ;AND 8
2285 004132 112737 000001 000306 1$: MOVB #1,@#PASFLG ;INITIALIZE THE COUNTER FOR THE SUBTEST
2286 004140 010401 2$: MOV R4,R1 ;PLACE THE STARTING ADDRESS OF MEMORY UNDER
2287 ;TEST IN R1
2288 004142 013700 000316 4$: MOV @#BAKPAT,R0
2289 004146 030201 BIT R2,R1 ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET
2290 004150 001004 BNE 8$ ;IF IT IS SET THEN GO TO 8$
2291 004152 030301 BIT R3,R1 ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET
2292 004154 001404 BEQ 12$ ;IF IT IS NOT SET THEN GO TO 12$
2293 004156 005100 6$: COM R0 ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS
2294 ;POINTED BY R2 & POINTED BY R3 = 1 IN WHICH
2295 ;CASE PREPARE TO WRITE THE LOCATION
2296 ;WITH A COMPLEMENT OF LOCATIONS NOT MEETING
2297 ;THIS CONDITION
2298 004160 000402 BR 12$
2299 004162 030301 8$: BIT R3,R1 ;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND

```


CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 49
CZKMAF.P11 05-MAR-79 09:02 T12 WORST CASE TESTING FOR CORE MEMORY

SEQ 0049

2300

2301 004164 001774

BEQ 6\$

;CHECK ADDRESS BIT POINTED BY R3
;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 6\$

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 50
CZKMAF.P11 05-MAR-79 09:02 T12 WORST CASE TESTING FOR CORE MEMORY

SEQ 0050

2302	004166	132737	000002	000306	12\$:	BITB	#2,@#PASFLG	:IS IT 2ND OR 3RD PASS OF THE SUBTEST ?
2303	004174	001001				BNE	14\$	:IF SO THEN READ THE MEMORY

2304	004176	010011				MOV	R0,(R1)	:OTHERWISE WRITE THE MEMORY BFORE READING IT
2305	004200	020011			14\$:	CMP	R0,(R1)	:READ THE MEMORY FOR CORRECT CONTENTS
2306	004202	001403				BEQ	16\$	
2307	004204	004767	001420			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
2308	004210	000042				42		:*****ERROR NUMBER 42*****
2309								
2310	004212	012746	000002		16\$:	MOV	#2,-(SP)	
2311	004216	005100			18\$:	COM	R0	
2312	004220	005111				COM	(R1)	
2313	004222	020011				CMP	R0,(R1)	:READ THE MEMORY AGAIN
2314	004224	001404				BEQ	19\$	
2315	004226	004767	001376			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
2316	004232	000043				43		:*****ERROR NUMBER 43*****
2317								
2318	004234	010011				MOV	R0,(R1)	:RESTORE THE LOCATION (R1)
2319	004236	005316			19\$:	DEC	(SP)	
2320	004240	001366				BNE	18\$	:EXECUTE THE CODE FROM 18\$ TWICE
2321	004242	005726				TST	(SP)+	:RESTORE THE STACK POINTER
2322	004244	122737	000003	000306		CMPB	#3,@#PASFLG	:IS IT THE 3RD PASS OF THE SUBTEST ?
2323	004252	001412				BEQ	20\$	:IF SO THEN GO TO 20\$
2324	004254	062701	000002			ADD	#2,R1	:IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN
2325								:MIN. TO MAX. DIRECTION
2326	004260	020105				CMP	R1,R5	:HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?
2327	004262	103727				BLO	4\$	:IF NOT THEN REPEAT FROM 4\$
2328	004264	105237	000306			INCB	@#PASFLG	
2329	004270	122737	000002	000306		CMPB	#2,@#PASFLG	:IF IT IS THE 2ND PASS OF THE SUBTEST
2330	004276	001720				BEQ	2\$	:THEN REPEAT FROM 2\$
2331	004300	162701	000002		20\$:	SUB	#2,R1	:OTHERWISE EXECUTE THE TEST IN MAX. TO MIN.
2332								:DIRECTION
2333	004304	020104				CMP	R1,R4	:HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?
2334	004306	103315				BHIS	4\$	:IF NOT THEN REPEAT FROM 4\$
2335	004310	012702	020000			MOV	#20000,R2	:PREPARE TO CHECK THE MEMORY WITH THE XOR OF
2336								:ADDRESS BITS 8 AND 13
2337	004314	105237	000307			INCB	@#PASFLG+1	:THE SUB TEST HAS CHECKED THE XOR ONE KIND
2338	004320	123727	000307	000002		CMPB	@#PASFLG+1,#2	:HAS TWO XOR COMBINATIONS BEEN CHECKED ?
2339	004326	103701				BLO	1\$	:IF NOT THEN GO TO 1\$
2340	004330	101004				BHI	22\$	:IF ALL THREE HAVE BEEN CHECKED THEN GO TO 22\$
2341	004332	012702	000010			MOV	#10,R2	:IF IT IS THE 2ND XOR COMBINATION THEN CHECK
2342	004336	006303				ASL	R3	:FOR ADDRESS BITS 3 & 8
2343	004340	000674				BR	1\$	
2344	004342	005137	000316		22\$:	COM	@#BAKPAT	:IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED
2345	004346	105737	000316			TSTB	@#BAKPAT	
2346	004352	001654				BEQ	TST12	:BAKPAT THEN RE-EXECUTE THE TEST
2347	004354	000625			END12:	BR	END10	

```

2348 *****
2349 *TEST 13 WRITE RECOVERY TEST
2350 * THIS TEST DIFFERS FROM 0-12 IN THAT IT CONSISTS OF A SMALL TEST PROGRAM
2351 * ACTUALLY RUNNING IN THE 4K BANK UNDER TEST.
2352 * THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
2353 * TO AID IN THE DEBUG, BEFORE A BANK IS ENTERED 'TST13 BNK XX'
2354 * IS TYPED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
2355 * BANK FAILED.
2356 * THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH 'MOV R2,-(PC)'
2357 * AND THE OTHER 1/2 CONTAINING '177667'. '177667' IS THE COMPLEMENT
2358 * OF 'JMP (R0)' INSTRUCTION.
2359 * R2 CONTAINS 'COM -(R1)' INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS
2360 * THE HIGHEST TEST ADDRESS IN THAT BANK. THE HIGHEST TEST ADDRESS IS
2361 * USUALLY ON 4K BOUNDARIES. WHEN TESTING BANK 0 RELOCATED, HOWEVER
2362 * R1 CONTAINS THE FIRST FREE TEST ADDRESS BELOW THE DIAGNOSTIC.
2363 * IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
2364 * THE TEST EXECUTION IS AS FOLLOWS:
2365 * 1. THE 'MOV R2,-(PC)' INSTRUCTION EXECUTES STORING
2366 * THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC)).
2367 * 2. SINCE R2 CONTAINS A 'COM -(R1)' INSTRUCTION IT COMPLEMENTS
2368 * THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
2369 * '177667' SO AFTER THE COM -(R1) IT EQUALS 110
2370 * CLEVERLY THIS IS THE 'JMP (R0)' INSTRUCTION.
2371 * 3. THIS SEQUENCE CONTINUES UNTIL THE 'MOV R2,-(PC)' INSTRUCTIONS
2372 * REACH THE MIDDLE OF THE TEST BANK. THEN THE 'JMP (R0)' INSTRUCTION IS
2373 * AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
2374 * TO TEST 13.
2375 * 4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.
2376 *
2377 *****
2378 004356 122737 000013 000404 TST13: CMPB #13,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2379 004364 001403 BEQ .+10
2380 004366 004767 001574 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2381 004372 000044 44 *****ERROR NUMBER 44*****
2382
2383 004374 012702 010247 1$: MOV #10247,R2 ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
2384 ;IN R2.
2385 004400 012700 177667 MOV #177667,R0 ;PLACE THE COMPLEMENT OF THE INSTRUCTION
2386 ;JMP (R0) IN R0
2387 ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
2388 ;SINCE THE TEST STORES 'MOV R2,-(PC)' IN 1/2 AND 177667 IN THE OTHER 1/2
2389
2390 004404 010546 2$: MOV R5,-(SP) ;SAVE R5
2391 004406 010446 MOV R4,-(SP) ;STORE LOWEST ADDRESS ON STACK
2392 004410 000241 29$: CLC
2393 004412 006005 ROR R5 ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
2394 004414 006004 ROR R4 ;DO SAME FOR LOWEST ADDRESS
2395 004416 160405 SUB R4,R5 ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
2396 004420 006005 ROR R5 ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
2397 004422 103002 BCC 30$ ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS.
2398 004424 062716 000002 ADD #2,(SP) ;INCREASE LOWEST TEST ADDRESS BY 2
2399 004430 012604 30$: MOV (SP)+,R4 ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
2400 004432 012605 MOV (SP)+,R5 ;RESTORE HIGHEST TEST ADDRESS
2401 004434 010403 MOV R4,R3 ;PLACE THE LOWEST LOCATION UNDER TEST
2402 ;IN R3
2403 004436 000406 BR 28$ ;LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0

```


2404	004440	042703	017776	3\$:	BIC	#17776,R3	:CAUSE R3 TO POINT TO THE LOWEST LOCATION
2405							:IN THE 4K BANK UNDER TEST
2406	004444	001507			BEQ	14\$	:IF ADDRESS WENT TO 0, IT MUST BE A 30K SYSTEM
2407	004446	105737	000405		TSTB	@#REL	:ARE WE RELOCATED?
2408	004452	100504			BMI	14\$	:BRANCH IF YES-TEST BANK0 ONLY-
2409	004454	020305		28\$:	CMP	R3,R5	:IF R3 IS HIGHER THAN THE HIGHEST LOCATION
2410	004456	103102			BHIS	14\$	:UNDER TEST THEN EXIT
2411							:IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0
2412	004460	020527	020000		CMP	R5,#20000	:IS HIGHEST TEST ADDRESS BELOW 4K?
2413	004464	103002			BHIS	31\$	:BRANCH IF NO
2414	004466	010501			MOV	R5,R1	:SET R1 TO HIGHEST TEST ADDRESS IN BANK0
2415	004470	000407			BR	32\$	
2416							
2417	004472	010301		31\$:	MOV	R3,R1	:SET R1 TO LOWEST CURRENT TEST ADDRESS
2418	004474	052701	017777		BIS	#17777,R1	:SET LOW ORDER ADDRESS BITS
2419	004500	005201			INC	R1	:CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
2420							:OF THE 4K BANK BEING POINTED BY R3
2421	004502	001402			BEQ	32\$	:BRANCH IF R1 WENT TO 0 (WHICH MIGHT
2422							:HAVE HAPPENED IF TESTING A 30K LSI SYSTEM)
2423	004504	020105			CMP	R1,R5	:COMPARE R1 TO HIGHEST ADDRESS UNDER TEST
2424	004506	101401			BLOS	33\$	:BRANCH IF WITHIN RANGE
2425	004510	010501		32\$:	MOV	R5,R1	:SET R1 TO THE MAXIMUM AVAILABLE MEMORY
2426							
2427	004512	132737	000001 000306	33\$:	BITB	#1,@#PASFLG	:IS THE LOWEST BIT OF LOCATION PASFLG
2428	004520	001101			BNE	16\$	:SET? IN WHICH CASE BACK GROUND HAS
2429							:ALREADY BEEN WRITTEN AND WRITE RECOVERY
2430							:TEST IS BEING PERFORMED
2431							
2432	004522	020304		4\$:	CMP	R3,R4	:OTHERWISE WRITE THE BACKGROUND
2433	004524	103430			BLO	8\$	:DEFINED AT STEP 3.
2434	004526	105737	000307		TSTB	@#PASFLG+1	:IS THE TEST JUST DOING READ, I.E.
2435	004532	001002			BNE	6\$	:IS THE PASFLG+1 LOCATION NON ZERO? IF SO
2436							:THEN GO TO 6\$
2437	004534	012713	010247		MOV	#10247,(R3)	:WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,-(PC)
2438	004540	020213		6\$:	CMP	R2,(R3)	:READ (R3) TO CONTAIN CORRECT DATA
2439	004542	001421			BEQ	8\$	
2440	004544	010046			MOV	R0,-(SP)	:SAVE R0
2441	004546	010146			MOV	R1,-(SP)	:AND R1 ON THE STACK
2442	004550	010301			MOV	R3,R1	
2443	004552	010200			MOV	R2,R0	:SET R0= GOOD DATA FOR ERROR PRINTOUT
2444	004554	004767	001050		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
2445	004560	000045			45		:*****ERROR NUMBER 45*****
2446							
2447	004562	012601			MOV	(SP)+,R1	:RESTORE R1
2448	004564	012600			MOV	(SP)+,R0	:AND R0
2449	004566	105737	000306		TSTB	@#PASFLG	:IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
2450							:THE PROPER DATA THEN WE DON'T WANT TO GO AND
2451							:EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
2452							:TEST
2453	004572	001005			BNE	8\$	:BRANCH IF PASFLG NOT=0
2454							
2455	004574	010200			MOV	R2,R0	:SAVE FOR ERROR REPORT
2456	004576	004767	001026		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
2457	004602	000046			46		:*****ERROR NUMBER 46*****
2458							
2459	004604	000663			BR	END12	:ABORT TST 13.

2460							
2461	004606	062703	000002		8\$:	ADD #2,R3	:INCREMENT R3 BY 2
2462	004612	162701	000002			SUB #2,R1	:DECREMENT R1 BY 2
2463	004616	020105				CMP R1,R5	:WRITE THE BACKGROUND DEFINED AT STEP 4.
2464	004620	103014				BHIS 12\$	
2465	004622	020103				CMP R1,R3	:HAS STORING THE 177667 REACHED WHERE 'MOV R2,-(PC) IS?
2466	004624	103405				BLO 10\$	:BRANCH IF YES DON'T DESTROY THE MOV R2,-(PC) IS.
2467	004626	105737	000307			TSTB @#PASFLG+1	:IS THE THE READ ONLY CHECK PASS?
2468	004632	001002				BNE 10\$	:BRANCH IF YES
2469	004634	012711	177667			MOV #177667,(R1)	:WRITE THE LOCATION WITH THE COMPLEMENT OF THE
2470							:OP CODE JMP (R0)
2471	004640	020011			10\$:	CMP R0,(R1)	:READ R1 TO CONTAIN CORRECT DATA
2472	004642	001403				BEQ 12\$	
2473	004644	004767	000760			JSR PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
2474	004650	000047				47	:*****ERROR NUMBER 47*****
2475							
2476	004652	020301			12\$:	CMP R3,R1	:IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
2477	004654	103722				BLO 4\$	:THEN REPEAT FROM 4\$
2478							
2479							:RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
2480							
2481	004656	062703	020000		13\$:	ADD #20000,R3	:OTHERWISE GO TO THE NEXT 4K BANK
2482	004662	000666				BR 3\$	
2483							
2484	004664	122737	000001	000306	14\$:	CMPB #1,@#PASFLG	:THE PROGRAM CONTROL COMES HERE AS FOLLOWS
2485							:1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
2486							:WRITE/READ CYCLE FOR THE BACK GROUND
2487							:AND WANTS TO BEGIN THE WRITE RECOVERY TEST
2488	004672	001437				BEQ 24\$	:2-PASFLG=1, PROGRAM HAS JUST COMPLETED
2489							:THE WRITE RECOVERY TEST AND WANTS TO
2490							:READ MEMORY FOR CORRECT DATA
2491	004674	103627				BLO END12	:3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
2492							:MEMORY AND WANTS TO GO THE NEXT TEST.
2493							
2494	004676	105137	000307			COMB @#PASFLG+1	:ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
2495							:ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
2496							:ENTRY DISABLE READ ONLY
2497	004702	001240				BNE 2\$	
2498	004704	012702	005141			MOV #5141,R2	:PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
2499							:IN R2
2500	004710	012700	177740			MOV #13\$-.-6,R0	:PLACE THE RETURN ADDRESS IN R0 AS 13\$
2501	004714	060700				ADD PC,R0	:THUS WHEN THE READ RECOVERY TEST REACHES
2502							:THE MIDDLE OF THE 4K MEMORY THEN THE
2503							:INSTRUCTION EXECUTED WILL BE JMP (R0)
2504							:BRANCHING THE PROGRAM TO 13\$
2505	004716	105237	000306		15\$:	INCB @#PASFLG	:INCREMENT PASFLG BY 1.
2506	004722	000630				BR 2\$	
2507							
2508	004724	032777	000020	173516	16\$:	BIT #20,@SWR	:HAS THE PRINTOUTS BEEN SUPRESSED ?
2509	004732	001016				BNE 18\$	:IF SO THEN GO TO 18\$
2510	004734	105737	000042			TSTB @#42	:IS THE PROGRAM RUNNING UNDER ACT?
2511	004740	001013				BNE 18\$	:BRANCH IF YES
2512	004742	004767	001644			JSR PC,PNTMES	:TYPE THE BANK UNDER TEST
2513	004746	051524	030524	020063		.ASCIIZ /TST13 BNK/	
2514	004754	047102	000113				
2515						.EVEN	

Address	Op Code	Op 1	Op 2	Op 3	Op 4	Comments	
2516	004760	004767	002454	JSR	PC,GETBNK	:GET BANK NO. UNDER TEST INTO DECWRD FOR PRINT.	
2517	004764	004767	001650	JSR	PC,\$TPDEC	:TYPE BANK NO. UNDER TEST	
2518							
2519	004770	000113		18\$:	JMP	(R3)	:BEGIN EXECUTING MOV R2,-(PC) ,COM -(R1) SEQUENCE IN TES
2520							
2521							
2522	004772	105137	000307	24\$:	COMB	@#PASFLG+1	
2523	004776	012700	000110		MOV	#110,R0	:PLACE THE OP CODE FOR JMP (R0) IN R0
2524	005002	000745			BR	15\$	: READ THE MEMORY FOR CORRECT DATA AFTER
2525							:INCREMMENTING PASFLG TO 2
2526							
2527							
2528							:TST13 EXITS VIA END12.

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 56
 CZKMAF.P11 05-MAR-79 09:02 PARITY OR RELOCATE?

SEQ 0056

```

2529 005004 012737 000377 000316 RELOC: MOV #377,@#BAKPAT
2530 005012 105737 000276 TSTB @#MMAVA ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2531 005016 001065 BNE CONTMM ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2532 ;MEMORY MANAGEMENT
2533 005020 032777 001000 173422 BIT #1000,@SWR ;RELOCATION WANTED?
2534 005026 001046 BNE CKDONE ;BRANCH IF NO
2535 005030 105737 000405 TSTB @#REL ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2536 005034 100420 BMI RELOER ; PLACE THE PROGRAM BACK IN LOWER CORE
2537 005036 112737 000200 000405 MOVB #200,@#REL ;OTHERWISE PREPARE TO RELOCATE
2538
2539 ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
2540
2541
2542 005044 004767 001542 JSR PC,PNTMES ;TYPE 'REL'
2543 005050 042522 047514 000103 .ASCIZ /RELOC/
2544 .EVEN
2545 005056 013705 000340 MOV @#MAXMEM,R5 ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
2546 ;AVAILABLE MEMORY
2547 005062 014445 2$: MOV -(R4),-(R5) ;RELOCATE THE PROGRAM
2548 005064 020427 000430 CMP R4,#BEGIN-50 ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN-50
2549 005070 101374 BHI 2$
2550 005072 000165 000050 JMP 50(R5)
2551
2552 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2553
2554
2555 005076 013705 000346 RELOER: MOV @#SAVR5,R5 ;RESTORE R5
2556 005102 105737 000405 TSTB @#REL ;IS DIAGNOSTIC IN RELOCATED STATE?
2557 005106 100016 BPL CKDONE ;BRANCH IF NO
2558
2559 005110 012704 000430 2$: MOV #BEGIN-50,R4 ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2560 005114 012524 MOV (R5)+,(R4)+
2561 005116 020537 000340 CMP R5,@#MAXMEM
2562 005122 103774 BLO 2$
2563 005124 105037 000405 CLRB @#REL
2564 005130 010537 000346 MOV R5,@#SAVR5 ;SAVE R5
2565 005134 012706 000500 MOV #BEGIN,SP ;RESET STACK TO LOWER MEMORY
2566 005140 010637 000350 MOV SP,@#SAVR6 ;'BEGIN' USES THIS TO RESET THE STACK.
2567 005144 000137 005150 CKDONE: JMP @#LOWER ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2568
2569
2570
2571 005150 105737 000315 LOWER: TSTB @#SAVKBB ;HERE DUE TO ^C TYPED?
2572 005154 001073 BNE $TPSTK ;BRANCH IF YES (TYPE ERROR STACK)
2573 005156 004767 001702 TSTMM: JSR PC,MMWNG ;SET THE REGISTERS IF THE MEMORY MANAGEMENT
2574 ;IS AVAILABLE
2575 005162 105737 000276 TSTB @#MMAVA ;IS MEM. MANAG. AVAILABLE ?
2576 005166 001462 BEQ ENDPAS ;BRANCH IF NO
2577 005170 000402 BR $CNTMM ;BEGIN TESTING ABOVE 28K
2578 005172 004767 002036 CONTMM: JSR PC,UPMM ;GO TO UPDATE MEM. MANAG. REGISTERS
2579 005176 012703 000324 $CNTMM: MOV #LOWTWO,R3 ;MAKE R3 POINT TO THE LOCATION LOWTWO
2580 005202 004767 002142 JSR PC,GETSIZ ;LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2581 ;OF THE LOWEST ADDRESS UNDER TEST
2582 005206 012704 020000 MOV #20000,R4 ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2583 ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
2584 005212 020237 172342 CMP R2,@#172342 ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF

```


Address	Op-Code	Register	Value	Comment
2585				:PAR1 ?
2586	005216	103405		:IF SO THEN GO 2\$
2587	005220	050104		:SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2588				:OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2589	005222	162702	000200	
2590	005226	004767	001636	
2591	005232	004767	002112	2\$: JSR PC,GETSIZ
2592				: SET MEM. MANAG. REGISTERS
2593				:PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2594	005236	004767	000020	:IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2595	005242	010005		:OF LOCATION HIGHADD IN R1
2596	005244	004767	002100	: GET THE ADDRESS OF MAX. MEM. UNDER TEST
2597	005250	004767	000006	
2598	005254	010013		: PREPARE TO SET UP LOCATION MAXMEM
2599	005256	000167	174256	: GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2600				:AND STORE INTO 'MAXMEM'
2601				:GO TEST A 24K SLICE ABOVE 28K.
2602				
2603				:MAXADR - SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K.
2604				:REGISTERS:
2605				:R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
2606				:R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
2607				:R2= PAR BLOCK NO. CURRENTLY UNDER TEST.
2608	005262	010046		MAXADR: MOV R0,-(SP) ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
2609	005264	012700	172356	MOV #172356,R0 ;R0=PAR7 UNIBUS ADDRESS
2610				:**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2611	005270	162716	020000	2\$: SUB #20000,(SP) ;DECREMENT VIRTUAL ADDRESS BY 4K
2612	005274	050116		BIS R1,(SP) ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
2613	005276	020240		CMP R2,-(R0) ;DOES CURRENT PAR= TEST BLOCK NO.?
2614	005300	001410		BEQ 3\$;BRANCH IF YES
2615	005302	020027	172340	CMP R0,#172340 ;ARE WE AT PAR0?
2616	005306	101370		BHI 2\$;NO KEEP TRYING
2617				:**END LOOP TO FIND PAR ADDRESS UNDER TEST
2618	005310	005720		TST (R0)+ ;SET TO PAR CURRENT
2619	005312	021002		CMP (R0),R2 ;IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
2620	005314	003006		BGT 4\$;BRANCH IF YES (FALL INTO ENDPAS)
2621	005316	012716	157776	MOV #157776,(SP) ;EXIT WITH MAXADR= 28K SEGMENT TEST SIZE
2622	005322	012600		3\$: MOV (SP)+,R0 ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2623	005324	062700	000002	ADD #2,R0 ;MAKE MAXIMUM MEMORY+2
2624	005330	000207		RTS PC ;AND EXIT MAXADR ROUTINE
2625				
2626	005332	022626		4\$: CMP (SP)+,(SP)+ ;FIXUP STACK
2627				;AND FALL THRU TO ENDPAS.

```

2628      ;* TYPE ROUTINE FOR ERROR STACK
2629      ;* -----
2630      ;*
2631      ;* THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
2632      ;* FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
2633
2634
2635
2636 005334 032777 000020 173106 ENDPAS: BIT    #20,@SWR      ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
2637 005342 001055          BNE    $EOP                ;IF NOT THEN GO TO $EOP
2638 005344 012746 177777      $TPSTK: MOV    #-1,-(SP)    ;THE PROGRAM HAS REACHED THE END AND ERROR
2639                                     ;STACK AND END OF PASS WILL BE TYPED OUT
2640 005350 012701 007744          MOV    #ENDPRG,R1      ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
2641                                     ;FOR 0 TO 4K MEMORY IN R1
2642 005354 012703 000376      TYPSTK: MOV    #376,R3
2643 005360 005216          INC    (SP)                ;IF WE HAVE GONE THRU THE ENTIRE
2644 005362 020137 000310          CMP    R1,@#ENDSTK    ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
2645 005366 103043          BHS    $EOP                ;THEN GO TO TYPE END OF PASS
2646 005370 112702 000022      MOV    #18.,R2
2647 005374 105302          RETSTK: DECB   R2           ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED.
2648 005376 002766          BLT    TYPSTK              ;BEEN CHECKED FOR ERROR THEN SEE IF THERE
2649                                     ;IS ANY MORE 4K MEMORY BANK
2650 005400 105721          TSTB   (R1)+              ;OTHERWISE CHECK THE BYTE STORED AT (R1)
2651 005402 001774          BEQ    RETSTK              ;IF IT IS 0 WE WILL NOT TYPE IT
2652 005404 020227 000020          CMP    R2,#16.      ;IS THE POINTER POINTING TO ERROR STACK BYTE
2653                                     ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
2654                                     ;THE SPECIFIC MEMORY BANK
2655 005410 103404          BLO    2$                  ;IF NOT THEN GO TO TYPE BIT NUMBER
2656 005412 101026          BHI    PARFL              ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
2657                                     ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
2658 005414 004767 001000          JSR    PC,TPADER      ;OTHERWISE TYPE 'ADDRESS ERROR'
2659 005420 000404          BR     FAILNM
2660 005422 010237 000312      2$:  MOV    R2,@#DECWRD   ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
2661                                     ;IN DECIMAL
2662 005426 004767 001202          JSR    PC,TYPDEC      ;GO TO TYPE THE BIT NUMBER IN DECIMAL
2663 005432 011637 000312      FAILNM: MOV    (SP),@#DECWRD ;PREPARE TO TYPE THE PAGE NUMBER
2664 005436 004767 001176          JSR    PC,$TPDEC    ;IN DECIMAL
2665 005442 005043          CLR    -(R3)
2666 005444 114113          MOV    -(R1),(R3)          ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
2667                                     ;FAILURE OCCURED
2668 005446 105021          CLRB   (R1)+              ;CLEAR THE ERROR STACK
2669 005450 005043          CLR    -(R3)
2670 005452 105237 000314          INCB   @#TYPCNT
2671 005456 004767 001316          JSR    PC,RPTOCT    ;ENABLE THE TYPE OUT OF 1 WORDS
2672                                     ;TYPE THE 4K BANK AND THE NUMBER OF TIMES
2673 005462 012703 000376          MOV    #376,R3      ;THIS FAILURE WAS SEEN
2674 005466 000742          BR     RETSTK              ;RESET SCRATCH STACK FOR EACH BIT PRINTED.
2675 005470 004767 000750      PARFL: JSR    PC,TPPRER
2676 005474 000756          BR     FAILNM              ; TYPE 'PAR ERR'

```



```

2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688 005476 005002
2689 005500 004767 002014
2690 005504 105737 000315
2691 005510 001043
2692 005512 005237 000406
2693 005515 032777 000040 172724
2694 005524 001012
2695 005526 004767 001052
2696 005532 040520 051523 000043
2697
2698 005540 013737 000406 000312
2699 005546 004767 001066
2700 005552 013700 000042
2701 005556 001405
2702 005560 004767 000012
2703 005564 000005
2704
2705
2706
2707 005566 000137 000156
2708
2709 005572 000137 000250
2710
2711 005576 004767 001616
2712 005602 013704 000344
2713 005606 014445
2714 005610 020437 000310
2715 005614 101374
2716 005616 000207
2717
2718
2719
2720 005620 004767 177752
2721 005624 000167 000400
2722
2723

; * END OF PASS
; * -----
; *
; * TYPE 'PASS#' AND DISABLE PARITY.
; * ALSO SERVICE ACT11.
; * AND EVERY CONSECUTIVE PASSES UNLESS BIT 4 OF $SWREG IS HIGH
; *

$EOP: CLR R2 ;SET R2= PARITY MODULE DISABLE CODE
      JSR PC,PARITY ;GO DISABLE PARITY MODULES IF SELECTED.
      TSTB @#SAVKBB ;CONTROL-C TYPED?
      BNE CTLC ;BRANCH IF YES-RESTORE LOADERS AND HALT-
      INC @#$PASS ;INCREMENT PASS COUNT
      BIT #4C,@SWR ;'PASS#XX' PRINTOUT WANTED?
      BNE ACT11 ;BRANCH IF NO
      JSR PC,TPCRLF ; TYPE CR, LF, AND 'PASS#'
      .ASCIZ /PASS#/
      .EVEN

ACT11: MOV @#$PASS,@#DECWRD ;GET PASS COUNT
      JSR PC,$TPDEC ;TYPE IT
      MOV @#42,R0 ;GET THE MONITOR ADDRESS
      BEQ $DOAGN ;IF NONE
      JSR PC,RLODER ;RESTORE XXDP MONITOR
      RESET ;RETURN TO ACT11 MONITOR.

; * SERVICE XXDP/ACT11
      JMP @#$ENDAD ;JUMP TO ACT SERVICE

$DOAGN: JMP @#RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP

RLODER: JSR PC,CLRMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
      MOV @#SAVR4,R4 ;RESTORE R4 WITH SAVR4
4$: MOV -(R4),-(R5) ;RESTORE LOADERS
      CMP R4,@#ENDSTK
      BHI 4$
      RTS PC ;RETURN FROM RLODER CALL

;CONTROL C HANDLER
CTLC: JSR PC,RLODER ;RESTORE ABS LOADER
      JMP APTHLT ;IF NOT APT HALT AT FATHLT

```

```

2724      ;* ERROR HANDLING ROUTINE
2725      ;* -----
2726      ;*
2727      ;*
2728      ;* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
2729      ;* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
2730
2731 005630 017637 000000 000402 ERROR: MOV @ (SP), @# $FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2732 005636 010346 1$: MOV R3, -(SP) ;SAVE R3
2733 005640 010046 MOV R0, -(SP) ;AND R0 ON THE STACK
2734
2735 ;SETUP BANK NO. IN FATAL FOR APT
2736
2737 005642 010103 MOV R1, R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
2738 005644 004767 001570 JSR PC, GETBNK ;GET BANK NO. UNDER TEST INTO PBNK
2739 005650 013703 000312 MOV @# PBNK, R3 ;GET BANK UNDER TEST
2740 005654 110337 000403 MOV R3, @# $FATAL+1 ;STORE FAILING BANK NO. FOR APT
2741
2742 ;
2743
2744 005660 010346 MOV R3, -(SP) ;TEMPORARILY STORE R3
2745 005662 012703 000376 MOV #376, R3 ;MAKE R3 AS THE STACK POINTER
2746 005666 013743 000306 MOV @# PASFLG, -(R3) ;OUTPUT THE WORD STORED AT
2747 005672 005043 2$: CLR -(R3)
2748 005674 113713 000402 MOV @# $FATAL, (R3) ;PUT ERROR NO. ON ERROR STACK
2749 005700 016643 000006 MOV 6(SP), -(R3) ;PLACE THE RETURN PC AT (R3)
2750 005704 011143 MOV (R1), -(R3) ;PLACE BAD DATA,
2751 005706 010043 MOV R0, -(R3) ;AND GOOD DATA ON THE STACK
2752 005710 005043 CLR -(R3)
2753 005712 016313 000004 MOV 4(R3), (R3) ;TAKE THE
2754 005716 040013 BIC R0, (R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
2755 005720 046300 000004 BIC 4(R3), R0 ;TO FIND THE BITS THAT FAILED
2756 005724 050013 BIS R0, (R3) ;AND PLACE IT ON THE STACK
2757 005726 012700 001766 MOV #ENDPRG--24., R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
2758 005732 060700 6$: ADD PC, R0 ;OF THE STARTING OF THE ERROR STACK
2759 005734 062700 000022 ADD #18., R0 ;FOR THE SPECIFIC 4K BANK
2760 005740 005316 DEC (SP)
2761 005742 002374 BGE 6$
2762 005744 005726 TST (SP)+ ;RESTORE THE STACK POINTER
2763
2764 005746 105037 000277 ERRTYP: CLRB @# TYPENB ;DISABLE ANY TYPE OUT
2765 005752 105737 000300 1$: TSTB @# SPRERR ;IF THIS IS PARITY PROBLEM
2766 005756 001007 BNE 3$ ;THEN GO TO 3$
2767 005760 105720 TSTB (R0)+ ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
2768 005762 105737 000301 TSTB @# $ADERR ;IF THIS IS ADDRESSING PROBLEM
2769 005766 001003 BNE 3$ ;THEN GO TO 3$
2770 005770 105720 TSTB (R0)+ ;INCREMENT THE POINTER R0 BY 1
2771 005772 005713 2$: TST (R3) ;IS BIT 15 OF (R3) SET?
2772 005774 100015 BPL 4$ ;IF NOT THEN GO TO 4$
2773 005776 122710 000377 3$: CMPB #377, (R0) ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
2774 006002 001401 BEQ 5$ ;IF SO DON'T BUMP ERRCR COUNT
2775 006004 105210 INCB (R0) ;INCREMENT THE ERROR COUNTER BY 1
2776 006006 122710 000001 5$: CMPB #1, (R0) ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
2777 006012 001404 BEQ 7$ ;BRANCH IF NO
2778 006014 032777 000400 172426 BIT #400, @SWR ;STOP ERROR PRINTOUT AFTER 1 WANTED?
2779 006022 001002 BNE 4$ ;BRANCH IF YES (DON'T TYPE ERROR)

```


CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 61
 CZKMAF.P11 05-MAR-79 09:02 ERROR HANDLING ROUTINE

SEQ 0061

```

2780 006024 105237 000277      7$:   INCB   @#TYPENB      ;ENABLE THE TYPE OUT ROUTINE
2781 006030 105737 000300      4$:   TSTB   @#$PRERR      ;PARITY ERROR?
2782 006034 001403              BEQ     9$              ;BRANCH IF NO
2783 006036 004767 000402      JSR     PC,TPPRER      ;ELSE TYPE 'PAR ERR'
2784 006042 000411              BR      8$              ;AND DON'T TEST INDIVIDUAL BIT FAILURES.
2785 006044 105737 000301      9$:   TSTB   @#$ADERR      ;ADDRESS ERROR?
2786 006050 001403              BEQ     6$              ;BRANCH IF NO
2787 006052 004767 000342      JSR     PC,TPADERR      ;PRINT 'ADR ERR'
2788 006056 000403              BR      8$
2789 006060 105720              6$:   TSTB   (R0)+        ;POINT TO NEXT ENTRY IN ERROR STACK
2790 006062 006313              ASL     (R3)        ;IS THERE STILL AN ERROR BIT SET IN ERROR.
2791 006064 001342              BNE     2$              ;BR IF YES - KEEP FILLING ERROR STACK
2792 006066 112737 000006 000314 8$:   MOVB   #6,@#TYPCNT ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
2793                                     ;THE STACK POINTED BY R3
2794 006074 004767 001142              JSR     PC,PUTADR      ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
2795                                     ;AT LOCATIONS (R3) AND (R3-2)
2796 006100 004767 000616              JSR     PC,TYPERR      ;TYPE ERROR STACK (7 WORDS)
2797
2798 006104 005037 000300      10$:  CLR     @#$PRERR      ;CLEAR ADDRESS/PARITY ERROR FLAGS
2799 006110 012600              MOV     (SP)+,R0      ;RESTORE R0
2800 006112 012603              MOV     (SP)+,R3      ;AND R3
2801 006114 105737 000420      FNDERR: TSTB   @#$ENV      ;ARE WE RUNNING UNDER APT?
2802 006120 001404              BEQ     2$              ;IF NOT THEN TEST FOR HALT
2803 006122 012737 000001 000400      MOV     #1,@#$MSGTY ;OTHERWISE INFORM THE APT
2804 006130 000442              BR      FATHLT      ;GOTO FATHLT AND WAIT FOR APT.
2805
2806 006132 010246              2$:   MOV     R2,-(SP)      ;SAVE R2 TEMP
2807 006134 005777 172310              TST     @SWR      ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
2808                                     ;ON ERROR
2809 006140 100405              BMI     4$              ;IF SO THEN HALT ON ERROR
2810                                     ;CHECK FOR CONTROL-C KEY
2811
2812 006142 004767 001526              JSR     PC,CHECKC      ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
2813                                     ;AND HALT AT FATHLT.
2814 006146 105737 000042      7$:   TSTB   @#42        ;ARE WE RUNNING UNDER ACT?
2815 006152 001401              BEQ     6$              ;BRANCH IF NO
2816
2817 006154 000000              4$:   HALT              ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2818                                     ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2819                                     ;THE WORD STORED IN R0
2820 006156 012602              6$:   MOV     (SP)+,R2      ;RESTORE R2
2821 006160 062716 000002      ADD     #2,(SP)      ;RESTORE THE RETURN ADDRESS
2822 006164 000207      RTS     PC              ;RETURN FROM THE SUBROUTINE
2823
2824
2825
2826 006166              FATERR:
2827 006166 004767 000412 000043  SEQERR: JSR     PC,TPCRLF      ;TYPE 'ERR #'
2828 006172 051105 020122              .ASCIIZ /ERR #/
2829                                     .EVEN
2830
2831 006200 017637 000000 000402      MOV     @ (SP),@#$FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2832 006206 105237 000314              INCB   @#TYPCNT      ;TELL $TPNUM TO TYPE 1 WORD
2833 006212 012703 000376              MOV     #376,R3      ;$TPNUM USES R3 AS STACK
2834 006216 013743 000402              MOV     @#$FATAL,-(R3) ;PUT ERROR NO. ON STACK
2835 006222 005743              TST     -(R3)      ;$TPNUM REQUIRES THIS

```

ERROR HANDLING ROUTINE

SEQ 0062

2836	006224	004767	000560		JSR	PC,FATYP	;TYPE ERROR NO.
2837	006230	105737	000420	APTHLT:	TSTB	@#SENV	;RUNNING UNDER APT?
2838	006234	001327			BNE	FNDERR	;BRANCH IF YES
2839	006236	000000		FATHLT:	HALT		;FATAL ERROR OR ^C HALT.
2840	006240	000137	000250		JMP	@#RESTR	;RESTART TST BUT DON'T CLEAR PASS COUNT
2841							;IN CASE ^C RESTART.
2842							
2843							
2844							
2845							
2846							
2847							
2848							
2849							
2850							
2851							
2852							
2853							
2854							
2855							
2856	006244	012637	000356	PARERR:	MOV	(SP)+,@#PARSP	;SET PARSP TO RETURN ADDRESS
2857	006250	011637	000360		MOV	(SP),@#PARPS	;SAVE PSW FOR RETURN
2858	006254	013706	000350		MOV	@#SAVR6,SP	;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
2859							;TO COMPLETE THE ERROR SERVICE ROUTINE.
2860	006260	010067	000130		MOV	R0,SAVR0	;SAVE R0 DURING PARITY SERVICE
2861	006264	010167	000126		MOV	R1,SAVR1	;SAVE R1 DURING PARITY SERVICE
2862	006270	013701	000352		MOV	@#PARMAP,R1	;GET PARITY AVAILABLE MAP
2863	006274	012700	172100		MOV	#172100,R0	;R0= FIRST PARITY ADDRESS.
2864							
2865	006300	005701			TST	R1	;ANY PARITY MODULES AVAILABLE?
2866	006302	001441			BEQ	4\$	;BR IF NO -FATAL ERROR-
2867	006304	006001		1\$:	ROR	R1	;SHIFT PARITY MAP BIT INTO C BIT.
2868	006306	103005			BCC	2\$	;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
2869	006310	005710			TST	(R0)	;PARITY MODULE ERROR BIT SET?
2870	006312	100406			BMI	3\$	;BRANCH IF YES -CALL 'ERROR' ROUTINE
2871	006314	020027	172136		CMP	R0,#172136	;DONE ALL PARITY MODULES?
2872	006320	002032			BGE	4\$	;BR IF YES- GO TO FATAL ERROR CALL-
2873	006322	062700	000002	2\$:	ADD	#2,R0	;POINT TO NEXT PARITY ADDRESS
2874	006326	000766			BR	1\$	;AND KEEP TRYING
2875	006330	042710	100000	3\$:	BIC	#100000,(R0)	;CLEAR PARITY ERROR BIT.
2876	006334	011001			MOV	(R0),R1	;GET PARITY MODULE CSR
2877	006336	006101			ROL	R1	;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
2878	006340	006101			ROL	R1	
2879	006342	006101			ROL	R1	
2880	006344	006101			ROL	R1	
2881	006346	042701	000777		BIC	#777,R1	;SAVE ERROR ADDRESS ONLY
2882	006352	105237	000300		INCB	@#SPRERR	;TELL 'ERROR' PARITY ERROR CALL.
2883	006356	004767	177246		JSR	PC,ERROR	;*ERROR* REPORT ERROR MESSAGE
2884	006362	000050			50		;*****ERROR NUMBER 50*****
2885							
2886	006364	016700	000024		MOV	SAVR0,R0	;RESTORE R0
2887	006370	016701	000022		MOV	SAVR1,R1	;RESTORE R1
2888	006374	013746	000360		MOV	@#PARPS,-(SP)	;SET RETURN PSW ON STACK
2889	006400	013746	000356		MOV	@#PARSP,-(SP)	;AND SET RETURN ADDRESS ON STACK
2890	006404	000002			RTI		;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
2891							


```

2892 ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
2893 4$:
2894 006406 004767 177554 JSR PC,FATER? ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2895 006412 000051 51 ;*****ERROR NUMBER 51*****
2896
2897
2898 ;R0+R1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
2899 ;STACK SPACE BETWEEN 500-450.
2900 006414 000000 SAVR0: 0 ;SAVE R0 DURING PARITY TRAP SERVICE
2901 006416 000000 SAVR1: 0 ;SAVE R1 DURING PARITY TRAP SERVICE
2902
2903
2904 006420 105737 000277 TPADER: TSTB @#TYPENB ;TYPE ERROR?
2905 006424 001406 BEQ 1$ ;BRANCH IF NO
2906 006426 004767 000160 JSR PC,PNTMES ; TYPE CR, LF AND 'ADR ER'
2907 006432 042101 020122 051105 .ASCIZ /ADR ERR/
2908 006440 000122
2909
2910 006442 000207 1$: .EVEN
2911 RTS PC
2912
2912 006444 105737 000277 TPPER: TSTB @#TYPENB ;ERROR PRINTOUTS ALLOWED?
2913 006450 001406 BEQ 1$ ;BRANCH IF NO
2914 006452 004767 000134 JSR PC,PNTMES ;GO TO TYPE CR, LF AND 'PAR ERR'
2915 006456 040520 020122 051105 .ASCIZ /PAR ERR/
2916 006464 000122
2917
2918 006466 000207 1$: .EVEN
RTS PC
  
```

```

2919
2920
2921          ;* TYPE OUT ROUTINE
2922          ;* -----
2923          ;*
2924          ;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
2925          ;*
2926
2927 006470 010146          NOTYP: MOV      R1,-(SP)
2928 006472 016601 000002  4$:      MOV      2(SP),R1
2929 006476 105721          BNE      4$          ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
2930 006500 001376          BR       RETTYP      ;PREPARE TO RETURN
2931 006502 000412          $TYPE: MOV      R1,-(SP)          ;SAVE R1
2932 006504 010146          MOV      R0,-(SP)          ;AND R0 ON THE STACK
2933 006506 010046          MOV      4(SP),R1          ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
2934 006510 016601 000004  2$:      MOVB    (R1)+,R0          ;PLACE THE BYTE TO BE TYPED IN R0
2935 006514 112100          BEQ      4$          ;IF IT IS END OF MESSAGE THEN GO TO 4$
2936 006516 001403          JSR      PC,$TPCHR          ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
2937 006520 004767 000022  BR       2$
2938 006524 000773          4$:      MOV      (SP)+,R0          ;RESTORE R0
2939 006526 012600          RETTYP: INC      R1          ;CAUSE R1 TO
2940 006530 005201          BIC      #1,R1          ;POINT TO EVEN ADDRESS
2941 006532 042701 000001  MOV      R1,2(SP)          ;MODIFY THE RETURN ADDRESS
2942 006536 010166 000002  MOV      (SP)+,R1          ;RESTORE R1
2943 006542 012601          BR       EXTYP          ;AND RETURN VIA RTS PC
2944 006544 000416
2945
2946 006546 132737 000040 000421 $TPCHR: BITB    #40,@#$ENVN          ;HAVE TYPE OUTS BEEN DISABLED?
2947 006554 001005          BNE      4$          ;IF SO THEN RETURN FROM THE SUBROUTINE
2948 006556 105737 177564  2$:      TSTB    @#$TPS          ;WAIT HERE
2949 006562 100375          BPL      2$          ;UNTIL THE PRINTER IS READY
2950 006564 110037 177566  4$:      MOVB    R0,@#$TPB          ;LOAD DATA TO BE TYPED INTO DATA REG.
2951 006570 000404          BR       EXTYP          ;RETURN
2952
2953 006572 004767 177706          PCRLF: JSR      PC,$TYPE          ;CR/LF
2954 006576 005015 000          .ASCIZ  <15><12>
2955          .EVEN
2956 006602 000207          EXTYP:  RTS      PC          ;RETURN
2957
2958 006604 004767 177762          TPCRLF: JSR      PC,PCRLF
2959 006610 000735          BR       $TYPE          ;TYPE CR/LF
2960          ;NOW GO TO TYPE THE REST OF THE MESSAGE
2961
2962 006612 032777 000020 171630 PNTMES: BIT      #20,@$SWR          ;PRINTOUTS ALLOWED?
2963 006620 001323          BNE      NOTYP          ;BRANCH IF NO
2964 006622 123737 000042 000046  CMPB    @#42,@#46          ;RUNNING UNDER ACT 11?
2965 006630 001717          BEQ      NOTYP          ;BRANCH IF YES -NOT PRINTOUT-
2966 006632 000764          BR       TPCRLF          ;SEND CR/LF AND TYPE MESSAGE.

```



```

2967
2968
2969
2970
2971
2972
2973
2974
2975 006634 004767 177732      TYPDEC: JSR      PC,PCRLF      ;TYPE CR/LF
2976
2977 006640 005046      $TPDEC: CLR      -(SP)
2978 006642 013746 000312      MOV      @#DECWRD,-(SP) ;GET THE WORD THAT HAS TO BE CONVERTED TO A
2979
2980 006646 162716 000012      2$:      SUB      #10.,(SP) ;DECIMAL NUMBER
2981 006652 002403      BLT      4$      ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
2982
2983 006654 005266 000002      INC      2(SP) ;GO TO 4$
2984 006660 000772      BR      2$      ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
2985 006662 062716 000012      4$:      ADD      #10.,(SP) ;AND RETURN TO 2$
2986 006666 052716 000060      BIS      #60,(SP) ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2987 006672 112667 000020      MOVB     (SP)+,6$-2 ;PLACE THE 1'S DIGIT TO BE TYPED
2988 006676 052716 000060      BIS      #60,(SP) ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2989 006702 112667 000007      MOVB     (SP)+,6$-3 ;PLACE THE 10'S DIGIT TO BE TYPED
2990 006706 004767 177572      JSR      PC,$TYPE ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
2991
2992 006712 020040 030040 000060      .ASCIZ   / 00/ ;3 SPACES
2993
2994 006720 000207      6$:      .EVEN
RTS      PC      ;RETURN FROM THE SUBROUTINE

```

```

2995
2996
2997          ;* OCTAL TYPE OUT ROUTINE
2998          ;* -----
2999          ;*
3000          ;* THIS ROUTINE IS USED TO TYPE OUT THE OCTAL VALUES
3001          ;* CONTROL SHOULD COME TO THIS ROUTINE WITH R3 POINTING TO
3002          ;* THE LOW ORDER BITS (I.E. BITS 0-15) OF THE ADDRESS TO
3003          ;* BE TYPED WHERE AS R3-2 SHOULD CONTAIN THE HIGH ORDER BITS
3004          ;* (I.E. BITS 16 & 17). CONTENTS OF LOCATION R3-1 AND R0 ARE
3005          ;* DESTROYED BY THIS SUBROUTINE
3006          ;* BYTE TYPCNT SHOULD BE SET TO THE NUMBER OF WORDS THAT HAVE
3007          ;* TO BE TYPED.
3008          ;*
3009
3010 006722 032777 020000 171520 TYPERR: BIT    #20000,@SWR    ;ERROR PRINTOUT WANTED?
3011 006730 001054                BNE    OCTXT              ;BRANCH IF NO
3012 006732 004767 177634        JSR    PC,PCRLF            ;TYPE CR/LF
3013 006736 004767 000012        JSR    PC,TYPOCT           ;TYPE OCTAL NO.
3014 006742 000447                BR     OCTXT              ;RETURN VIA RTS PC
3015 006744 012123                OCTTYP: MOV   (R1)+,(R3)+    ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
3016                                ;BY R3
3017 006746 012113                MOV   (R1)+,(R3)            ;AND NOW PLACE THE LOW ORDER BITS
3018 006750 105237 000314        INCB   @#TYPCNT             ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3019 006754 052743 000004        TYPOCT: BIS   #4,-(R3)
3020 006760 106113 2$:          ROLB   (R3)
3021 006762 103376                BCC    2$
3022 006764 005000                CLR    R0
3023 006766 106113                ROLB   (R3)                ;GET BITS 17 & 16 INTO R0
3024 006770 006100                ROL    R0
3025 006772 106113                ROLB   (R3)
3026 006774 006100                ROL    R0
3027 006776 000405                BR     $TPNUM
3028 007000 004767 177500        RPTOCT: JSR   PC,$TYPE      ; TYPE 3 SPACES
3029 007004 020040 000040        .ASCIZ  / /
3030                                .EVEN
3031 007010 005000                FATYP:  CLR    R0
3032 007012 012723 000006        $TPNUM: MOV   #6,(R3)+      ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3033 007016 000241 4$:          CLC
3034 007020 006113                ROL    (R3)
3035 007022 006100                ROL    R0
3036 007024 052700 000060        BIS   #60,R0              ;PLACE THE CARRY FROM (R3) IN R0
3037 007030 004767 177512        JSR    PC,$TPCHR           ;OR THE CONTENTS OF R0 WITH AN ASCII 0
3038 007034 005000                CLR    R0
3039 007036 006113                ROL    (R3)
3040 007040 006100                ROL    R0
3041 007042 006113                ROL    (R3)
3042 007044 006100                ROL    R0
3043 007046 105363 177776        DECB   -2(R3)             ;PLACE THE CARRY FROM (R3) IN R0
3044 007052 001361                BNE    4$                ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3045 007054 105337 000314        DECB   @#TYPCNT           ;THEN REPEAT FROM 4$
3046 007060 001347                BNE    RPTOCT             ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
3047 007062 000207                OCTXT: RTS    PC           ;TYPED THEN REPEAT FROM RPTOCT

```



```

3048
3049
3050
3051
3052
3053
3054
3055
3056
3057 007064 012702 001400 MEMMNG: MOV #1400,R2
3058 007070 105037 000276 MMREG: CLRB @MMMAVA ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
3059
3060 007074 032777 010000 171346 BIT #10000,@SWR ;THAT MEM. MANAG. IS AVAILABLE FOR TESTING
3061 007102 001441 BEQ RETMM ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
3062 007104 012700 000004 MOV #4,R0 ;IF NOT THEN RETURN FROM THE SUBROUTINE
3063 007110 012720 007210 MOV #NOMM,(R0)+ ;PREPARE TO SETUP TIME OUT VECTOR
3064 007114 012710 000340 MOV #340,(R0) ;RETURN ADDRESS TO NOMM
3065 007120 005037 177572 CLR @MSRO ;AND WITH A PSW OF 340
3066 007124 105237 000276 INCB @MMMAVA ;TRY TO REACH MEM. MANAG. SRO
3067
3068 007130 012701 172340 MOV #172340,R1 ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
3069 007134 005021 CLR (R1)+ ;BYTE
3070 007136 062702 000200 2$: ADD #200,R2 ;R1 IS POINTING TO PAR0
3071 007142 010221 MOV R2,(R1)+ ;PAR0 WILL POINT TO BANK 0
3072 007144 020127 172356 CMP R1,#172356 ;SETUP PAR1-PAR6
3073 007150 103772 BLO 2$
3074 007152 012711 007600 MOV #7600,(R1) ;PAR7 IS POINTING TO THE I/O PAGE
3075 007156 012701 172300 MOV #172300,R1
3076 007162 012721 077406 4$: MOV #77406,(R1)+ ;SETUP PDR0-PDR7
3077 007166 020127 172316 CMP R1,#172316
3078 007172 101773 BLOS 4$
3079 007174 005237 177572 INC @MSRO ;ENABLE MEM. MANAG.
3080 007200 005010 $RETMM: CLR (R0) ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
3081 007202 012740 000104 MOV #BUSER,-(R0)
3082 007206 000207 RETMM: RTS PC
3083
3084 007210 022626 NOMM: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
3085 007212 004767 177366 JSR PC,TPCRLF ;TYPE 'NO MEMORY MANAGEMENT MESSAGE
3086 007216 047516 045440 000124 .ASCIZ /NO KT/
3087 .EVEN
3088 007224 004767 176736 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
3089 007230 000052 52 *****ERROR NUMBER 52*****
3090
3091 007232 000762 BR $RETMM ; RESTORE TIME OUT TRAP VECTOR
3092
3093 007234 013702 172354 UPMM: MOV @#172354,R2 ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
3094 007240 000713 BR MMREG
  
```

```

3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106 007242 005063 177776
3107 007246 010113
3108 007250 105737 000276
3109 007254 001425
3110 007256 010146
3111 007260 042701 017777
3112 007264 040113
3113 007266 052701 004000
3114 007272 006001
3115 007274 103376
3116 007276 062701 172340
3117 007302 011101
3118 007304 052701 010000
3119 007310 006101
3120 007312 103376
3121 007314 006101
3122 007316 006143
3123 007320 006101
3124 007322 006123
3125 007324 050113
3126 007326 012601
3127 007330 000207
3128
3129
3130
3131
3132
3133
3134
3135
3136
3137
3138
3139
3140
3141 007332 016143 000001
3142 007336 005043
3143
3144 007340 116113 177777
3145 007344 000207

; * 18 BIT ADDRESS GENERATOR
; * -----
; *
; * THIS SUBROUTINE IS USED TO PLACE THE ADDRESS STORED IN R1
; * IN THE LOCATION POINTED BY R3. THE ADDRESS IN R1 IS CONVERTED
; * TO AN 18 BIT ADDRESS ONLY IF MEM. MANAG. IS AVILABLE IN WHICH
; * CASE THE HIGH ORDER BITS OF THE ADDRESS ARE PLACED IN LOCATION
; * POINTED BY R3-2
; *
PUTADR: CLR -2(R3)
MOV R1,(R3) ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
TSTB @#MMVA ;IS THE MEM. MANAG. AVILABLE ?
BEQ 6$ ;IF NOT THEN RETURN FROM THE SUBROUTINE
MOV R1,-(SP) ;SAVE R1
BIC #17777,R1 ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
BIC R1,(R3) ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
BIS #4000,R1 ;PREPARE TO SHIFT R1 BY 12 PLACES
2$: ROR R1
BCC 2$ ;GET THE NUMBER OF PAR IN R1
ADD #172340,R1 ;GET THE ADDRESS OF PAR IN R1
MOV (R1),R1 ;LOAD R1 WITH THE CONTENTS OF PAR
BIS #10000,R1
4$: ROL R1
BCC 4$ ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
ROL R1
ROL -(R3) ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
ROL R1
ROL (R3)+ ;PLACE BIT 16 OF THE ADDRESS
BIS R1,(R3) ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
MOV (SP)+,R1 ;RESTORE R1
6$: RTS PC ;RETURN FROM THE SUBROUTINE

; * GET ADDRESS FROM THE APT MAILBOX
; * -----
; *
; * THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
; * PLACE IT IN THE LOCATION USED BY THE PROGRAM TO DEFINE THE
; * MEMORY BOUNDRIES.
; * PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
; * ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
; * THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
; * TO BE PLACED
; *
GETADR: MOV 1(R1),-(R3) ;PLACE THE LOW ORDER BITS OF THE ADDRESS
CLR -(R3) ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
;HAVE TO BE PLACED
MOVB -1(R1),(R3) ;PLACE BITS 16 & 17
2$: RTS PC ;RETURN FROM THE SUBROUTINE

```



```

3146
3147
3148
3149
3150
3151
3152
3153
3154
3155
3156 007346 005046          $GTSIZ: CLR      -(SP)          ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3157                                     ;0-4 OF R2
3158
3159 007350 012301          GETSIZ: MOV      (R3)+,R1
3160 007352 011302          MOV      (R3),R2          ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3161 007354 042702 017777          BIC      #17777,R2          ;CLEAR ADDRESS BITS 0-12
3162 007360 052702 000040          2$: BIS      #40,R2
3163 007364 006001          4$: ROR      R1
3164 007366 006002          ROR      R2          ;ROTATE R1 AND R2 7 TIMES
3165 007370 103375          BCC      4$
3166 007372 005716          TST      (SP)          ;IF RETURN PC IS ZERO THEN IT MUST BE THE
3167 007374 001004          BNE      6$          ;FLAG THAT WAS SET AT $GTSIZ
3168 007376 005726          TST      (SP)+          ;POP THE FLAG OFF STACK
3169 007400 052702 000100          BIS      #100,R2          ;KEEP ROTATING
3170 007404 000767          BR       4$
3171 007406 012301          6$: MOV      (R3)+,R1          ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3172 007410 012700 160000          MOV      #160000,R0
3173 007414 040001          BIC      R0,R1          ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3174 007416 000207          RTS      PC          ;RETURN FROM THE SUBROUNE
3175
3176
3177
3178
3179
3180
3181
3182
3183
3184 007420 105737 000276          CLRMM: TSTB     @#MMAVA          ;WAS THE MEMORY MANAGEMENT ENABLED ?
3185 007424 001404          BEQ      1$          ;IF NOT THEN GO TO 1$
3186 007426 005037 177572          CLR      @#SRO          ;DISABLE THE MEMORY MANAGEMENT
3187 007432 105037 000276          CLRB     @#MMAVA          ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
3188 007436 000207          1$: RTS      PC          ;RETURN FROM THE SUBROUTINE
3189
3190
3191
3192
3193
3194
3195
3196
3197
3198 007440 010046          ;* GET BANK NO. UNDER TEST
3199 007442 010346          ; CALLED BY ERRTP AND TST13 TO GET BANK NO. UNDER TEST INTO PBNK.
3200 007444 042703 017777          ;REGISTERS
3201 007450 052703 010000          ;R0=POINTER TO PAR UNDER TEST
          ;R3=VIRTUAL ADDRESS ON ENTRY
          ;R0+R3 ARE RESTORED ON EXIT.
          GETBNK: MOV      R0,-(SP)          ;SAVE R0
          MOV      R3,-(SP)          ;SAVE R3
          BIC      #17777,R3          ;SAVE ONLY VIRTUAL BANK BITS
          BIS      #10000,R3          ;SETUP R3 SHIFT BIT

```

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 70

CZKMAF.P11 05-MAR-79 09:02

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

SEQ 0070

```

3202 007454 000241          CLC
3203 007456 006003      1$: ROR      R3          ;SHIFT A BANK BIT
3204 007460 103376          BCC      1$          ;UNTIL IN BITS <2:0> OF R3
3205 007462 105737 000276   TSTB     @#MMAVA      ;MEMORY MANAGEMENT UNDER TEST?
3206 007466 001407          BEQ      2$          ;NO EXIT
3207
3208          ;GET PAR ADDRESS AND PHYSICAL BANK NO.
3209 007470 006303          ASL      R3          ;MAKE R3 PAR ADDRESS OFFSET.
3210 007472 062703 172340   ADD      #172340,R3    ;MAKE FULL PAR ADDRESS.
3211 007476 011300          MOV      (R3),R0      ;GET PAR CONTENTS
3212 007500 006300          ASL      R0
3213 007502 000300          SWAB     R0          ;SHIFT BANK BITS TO BITS <7:0>
3214 007504 110003          MOVB     R0,R3        ;SET R3 TO PHYSICAL BANK NO.
3215 007506 010337 000312   2$: MOV      R3,@#PBK    ;STORE PHYSICAL BANK NO.
3216 007512 012603          MOV      (SP)+,R3    ;RESTORE R3
3217 007514 012600          MOV      (SP)+,R0    ;RESTORE R0
3218 007516 000207          RTS      PC          ;RETURN TO CALLER
3219
3220
3221          ; PARITY ENABLE/DISABLE ROUTINE
3222          :
3223          :
3224          : THIS ROUTINE ENABLES OR DISABLES PARITY MODULES AND PRINTS ASSOCIATED MEESSAGES.
3225          : IF PARITY AVAILABLE THEN BIT13 OF 'REL' IS SET AND 'PAR'ITY IS PRINTED.
3226          : ALSO THE BACKGROUND TEST PATTERN (LOC. BAKPAT) IS SET=376
3227          :
3228          : REGISTER USAGE.
3229          : R0= POINTS TO BUS TIMEOUT TRAP VECTOR (LOC. 4)
3230          : R1= HOLDS PARITY MODULE UNIBUS ADDRESS.
3231          : R2= ON ENTRY HOLDS ENABLE/DISABLE CODE .
3232          : IF R2=0 THEN DISABLE
3233          : IF R2=1 THEN ENABLE
3234          : R3= SCRATCH TO SETUP LOC. PARMAP WITH A MAP OF PARITY MODULES PRESENT.
3235          :
3236          : CALL IS
3237          :
3238          : MOV      #1,R2 ;ENABLE CODE
3239          : JSR      PC,PARITY
3240
3241 007520 032777 004000 170722 PARITY: BIT      #4000,@SWR    ;PARITY TEST WANTED?
3242 007526 001456          BEQ      6$          ;BRANCH IF NO
3243
3244 007530 012700 000004          MOV      #4,R0      ;POINT R0 TO BUS TIMEOUT ADDRESS.
3245 007534 012710 000116          MOV      #5$-,-6,(R0) ;SET RETURN FROM TIMEOUT TRAP TO 5$
3246 007540 060710          ADD      PC,(R0)      ;IN THE CURRENT BANK.
3247 007542 005037 000352      1$: CLR      @#PARMAP    ;CLEAR PARITY MAP HOLDER.
3248 007546 012701 172140          MOV      #172140,R1    ;SET R1 TO LAST PARITY MODULE ADDRESS+2
3249 007552 012703 100000          MOV      #100000,R3    ;SET R3 TO PARMAP AVAILABLE CODE BEGIN.
3250 007556 010241          2$: MOV      R2,-(R1)    ;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE.
3251 007560 050337 000352          EIS      R3,@#PARMAP ;NO TRAP TO 5$, SO SET PARITY AVAILABLE.
3252 007564 000241          CLC
3253 007566 006003          3$: ROR      R3          ;SETUP NEXT PARMAP BIT
3254 007570 103372          BCC      2$          ;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
3255 007572 012710 000104          MOV      #BUSER,(R0) ;RESET BUS TIMEOUT TRAP VECTOR
3256 007576 005702          TST      R2          ;IS THIS A DISABLE CALL?
3257 007600 001431          BEQ      6$          ;BRANCH IF YES (EXIT)

```


CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 71

CZKMAF.P11 05-MAR-79 09:02

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

SEQ 0071

```

3258 007602 005737 000352      TST    @#PARMAP      ;WERE ANY PARITY MODULES FOUND?
3259 007606 001011              BNE     4$           ;BRANCH IF YES
3260 007610 004767 176770      JSR     PC,TPCRLF      ;PRINT 'NO PAR'
3261 007614 047516 050040 051101 .ASCIZ  /NO PAR/
3262 007622      000              .EVEN
3263      007624              JSR     PC,FATERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
3264 007624 004767 176336      JSR     53          ;*****ERROR NUMBER 53*****
3265 007630 000053
3266
3267
3268 007632 152737 000040 000405 4$: BISB    #40,@#REL      ;SET PARITY UNDER TEST FLAG
3269 007640 012737 000376 000316      MOV     #376,@#BAKPAT ;SET BACKGROUND PATTERN TO
3270                                ;WORST CASE PARITY CODE.
3271 007646 004767 176732      JSR     PC,TPCRLF      ;PRINT 'TST PARITY'
3272 007652 040520 000122      .ASCIZ  /PAR/
3273      .EVEN
3274 007656 000405      BR      EXITC          ;AND EXIT VIA RTS PC
3275
3276                                ;GET HERE IF PARITY ADDRESS TIMED OUT TO LOC. 4
3277

```

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

SEQ 0072

```

3278 007660 022626      5$:    CMP      (SP)+,(SP)+      ;RESET STACK FROM TRAP
3279 007662 000741      BR      3$                      ;KEEP TRYING PARITY ADDRESSES.
3280
3281 007664 142737 000040 000405 6$:    BICB      #40,@#REL      ;CLEAR PARITY TESTING FLAG
3282 007672      EXITC:
3283 007672 000207      7$:    RTS      PC              ;RETURN TO CALLER
3284
3285
3286
3287
3288
3289      ;CHECKC
3290      ; THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
3291      ; TEST OR IN THE ERROR TYPE ROUTINE.
3292      ; IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
3293      ; RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
3294      ; FINALLY IT HALTS AT FATHLT.
3295 007674 105037 000315      CHECKC: CLRB      @#SAVKBB      ;INIT CONTROL-C FLAG.
3296 007700 105737 177560      TSTB      @#TKS          ;ANY CHAR. TYPED?
3297 007704 100372      BPL      EXITC          ;BR IF NO-EXIT VIA RTS PC-
3298 007706 113702 177562      MOVB      @#$KBB,R2      ;GET THE CHAR TYPED.
3299 007712 042702 000200      BIC      #200,R2 ;CLEAR THE PARITY BIT.
3300 007716 122702 000003      CMPB      #3,R2          ;IS IT CONTROL-C?
3301 007722 001363      BNE      EXITC          ;BRANCH IF NO -EXIT VIA RTS PC-
3302 007724 110237 000315      MOVB      R2,@#SAVKBB      ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
3303 007730 004767 176650      JSR      PC,TPCRLF      ;PRINT '^C'
3304 007734 041536      .ASCIZ  /^C/
3305      .EVEN
3306 007740 000167 175132      JMP      RELOER          ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
3307
3308      .=7744
3309 007744 000000      ENDPRG: 0
3310
3311
3312
3313
3314      000001      .END
  
```

;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
 ;STACK.FOR EACH 4K BANK 18. BYTES ARE SAVED.
 ;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
 ;AFTER THE ERROR STACK.
 ;FOR 4K MEMORY SIZE THEN PROGRAM=7744+22=7776

CHECKC	007674	1547	2812	3295#															
CKDONE	005144	2534	2557	2567#															
CLRMEM	001540	1485#	2599																
CLRMM	007420	1417	1439	2711	3184#														
CNTSCP	001672	1546	1549#																
CONT	001600	1502#	1560																
CONTMM	005172	2531	2578#																
CTLC	005620	2691	2720#																
DECWRD	000312	1146#	1147	2660*	2663*	2698*	2978												
ENDPAS	005334	2576	2636#																
ENDPRG	007744	1072	1266	2640	2757	3309#													
ENDSTK	000310	1143#	1144	1404*	2644	2714													
END0	002164	1647#	1689																
END1	002264	1689#	1729																
END10	004030	2196	2204#	2347															
END12	004354	2347#	2459	2491															
END2	002374	1729#	1825																
END3	002642	1825#	1864																
END4	002752	1864#	1950																
END5	003120	1950#	2006																
END6	003254	2006#	2082																
END7	003500	2082#	2204																
ERROR	005630	1604	1667	1721	1786	1850	1890	1897	1924	1982	2056	2137	2153	2162					
		2307	2315	2444	2456	2473	2731#	2883											
ERRTYP	005746	2764#																	
EXITC	007672	3274	3282#	3297	3301														
EXTYP	006602	2944	2951	2956#															
FAILNM	005432	2659	2663#	2676															
FATERR	006166	1279	1327	1450	1474	1632	2826#	2894	3088	3264									
FATHLT	006236	2804	2839#																
FATYP	007010	2836	3031#																
FNDERR	006114	2801#	2838																
GALLOP	003526	2119#	2254																
GETADR	007332	1330	1331	3141#															
GETBNK	007440	2516	2738	3198#															
GETSIZ	007350	2580	2591	2596	3159#														
HIGHAD	000332	1171#	1316																
HIGHTW	000330	1170#	1315	1392															
LOOP	001610	1505#	1558																
LOWADD	000326	1168#																	
LOWBNK	000304	1136#	1137	2025*	2037	2072													
LOWER	005150	2567	2571#																
LOWTWO	000324	1167#	1385	1441	2579														
M =	000200	982#	2473																
MAXADR	005262	2594	2597	2608#															
MAXMEM	000340	1177#	1314	1408*	1624	1819	2545	2561											
MEMMNG	007064	1357	2573	3057#															
MEMTST	001532	1479#																	
MMAVA	000276	1118#	1121	1359	2530	2575	3058*	3066*	3108	3184	3187*	3205							
MMREG	007070	1370	2590	3058#	3094														
N =	000054	982#	1279	1282#	1327	1330#	1450	1453#	1474	1477#	1575	1578#	1589	1591#					
		1592	1595#	1600	1602#	1632	1635#	1659	1662#	1667	1670#	1702	1705#	1721					
		1724#	1746	1749#	1771	1773#	1781	1783#	1797	1799#	1834	1837#	1850	1853#					
		1883	1886#	1890	1893#	1897	1900#	1924	1927#	1972	1975#	1982	1985#	2021					
		2024#	2056	2059#	2113	2116#	2137	2140#	2153	2156#	2162	2165#	2239	2242#					
		2279	2282#	2307	2310#	2315	2318#	2380	2383#	2444	2447#	2456	2459#	2473					

[illegible]

SW11	=	004000	982#												
TBL		001626	1511	1515#	1516	1517	1518	1519	1520	1521	1522	1523	1524	1525	1526
			1527												
TKS	=	177560	1155#	3296											
TPADER		006420	2658	2787	2904#										
TPCRLF		006604	1067	2695	2827	2958#	2966	3085	3260	3271	3303				
TPPRER		006444	2675	2783	2912#										
TRYSR		001014	1318	1334#											
TSTGO		001714	1554	1557#											
TSTMM		005156	1453	2573#											
TSTREL		001372	1261	1438#											
TSTRP		000570	1264	1266#											
TSTSCP		0C1660	1545#	1647											
TSTSI2		001376	1439#												
TST0		001730	1515	1573#											
TST1		002166	1516	1655#											
TST10		003502	1523	2110#											
TST11		004032	1524	2236#											
TST12		004104	1525	2277#	2346										
TST13		004356	1526	2378#											
TST2		002266	1517	1699#											
TST3		002376	1518	1744#	1823										
TST4		002644	1519	1832#											
TST5		002754	1520	1880#											
TST6		003122	1521	1968#											
TST7		003256	1522	2018#											
TYPCNT		000314	1148#	1254*	2670*	2792*	2832*	3018*	3045*						
TYPDEC		006634	2662	2975#											
TYPENB		0C0277	1122#	1125	2764*	2780*	2904	2912							
TYPEOP		005526	2695#												
TYPERR		006722	2796	3010#											
TYPMEM		001206	1388#												
TYPOCT		006754	3013	3019#											
TYPSI2		001164	1332	1335	1382#										
TYPSTK		005354	2642#	2648											
UPMM		007234	2578	3093#											
WRTMEM		000120	1025#	1750	1886	1975									
\$A	=	000001	402#	404	405#	407	408#	412	413#	417	418#	423	424#	429	430#
			436	437#	442	443#	447	448#	451	452#	455	456#	459	460#	463
			464#	467	468#	471	472#	475	476#	479	480#	484	485#	488	489#
			492	493#	496	497#	500	501#	502	503#	510	511#	514	515#	518
			519#	529	530#	533	534#	542	543#	546	547#	549	550#	555	556#
			559	560#	563	564#	567	568#	575	576#	579	580#	583	584#	589
			590#	594	595#	601	602#	607	608#	611	612#	616	617#	620#	
\$ADERR		000301	1130#	1133	1596*	1719*	1783*	1848*	2768	2785					
\$APTHD		000276	1098	1104#	1117										
\$CNTMM		005176	2577	2579#											
\$CPUOP		000426	1215#												
\$DEVCT		000410	1206#	1552*	2062*	2186*									
\$DJAGJ		005572	2701	2709#											
\$ENDAD		000156	994	1044#	2707										
\$ENV		000420	1211#	1299	1545	2801	2837								
\$ENVM		000421	1212#	1317	1616	2946									
\$EOP		005476	2637	2645	2688#										
\$ETABL		000420	1210#												
\$ETEND		000450	1110	1239#											

\$FATAL	000402	1203#	2731*	2740*	2748	2831*	2834										
\$GTSIZ	007346	1393	3156#														
\$HD	= 000002	973															
\$HIBTS	000276	1105#															
\$HIMAX	000334	1174#	1315*														
\$KBB	= 177562	1156#	3298														
\$MADR1	000432	1228#															
\$MADR2	000436	1232#															
\$MADR3	000442	1235#															
\$MADR4	000446	1238#															
\$MAIL	000400	1061	1106	1110	1201#	1271	1286										
\$MAMS1	000430	1222#															
\$MAMS2	000434	1230#															
\$MAMS3	000440	1233#															
\$MAMS4	000444	1236#															
\$MAXM	000336	1175#	1316*	1401													
\$MBADR	000300	1106#															
\$MSGAD	000414	1208#															
\$MSGLG	000416	1209#															
\$MSGTY	000400	1007*	1202#	2803*													
\$MTYP1	000431	1223#	1324														
\$MTYP2	000435	1231#															
\$MTYP3	000441	1234#															
\$MTYP4	000445	1237#	1320														
\$NWTST	= 000001	1561#	1563	1650#	1652	1691#	1693	1733#	1735	1826#	1828	1866#	1868	1952#			
		1954	2007#	2009	2083#	2085	2208#	2210	2255#	2257	2348#	2350					
\$PASS	000406	1205#	1263	2692*	2698												
\$PASTM	000304	1108#															
\$PRERR	000300	1126#	1129	1253*	2765	2781	2798*	2882*									
\$RETM	007200	3080#	3091														
\$SVPC	= 000044	992#	997														
\$SWR	= 000000	973#	982#	1574	1656	1700	1745	1833	1881	1969	2019	2111	2237	2278			
		2379															
\$SWREG	000422	1213#	1301														
\$TESTN	000404	1065	1113	1204#	1503*	1549	1573	1655	1699	1744	1832	1880	1968	2018			
		2110	2168	2176	2201	2236	2277	2378									
\$TN	= 000014	963#	973	1561	1574#	1650	1656#	1691	1700#	1733	1745#	1826	1833#	1866			
		1881#	1952	1969#	2007	2019#	2083	2111#	2208	2237#	2255	2278#	2348	2379#			
\$TPB	= 177566	1158#	2950*														
\$TPCHR	006546	2937	2946#	3037													
\$TPDEC	006640	2517	2664	2699	2977#												
\$TPNUM	007012	3027	3032#														
\$TPS	= 177564	1157#	2948														
\$TPSTK	005344	2572	2638#														
\$TSTM	000302	1107#															
\$TYPE	006504	1388	2932#	2953	2959	2990	3028										
\$UNIT	000412	1059	1207#														
\$UNITM	000306	1109#															
\$USWR	000424	1214#															
\$Z	= 000362	1193#															
\$ZZ	= 007744	3308#															
\$SM	= 000200	2473#															
.	= 007746	980#	985#	992	993#	995#	997#	999#	1005#	1011#	1050#	1094	1095#	1097#			
		1099#	1117#	1121#	1125#	1129#	1133#	1135#	1137#	1142#	1144#	1147#	1193	1196#			
		1247#	1256	1511	1574	1629	1658	1701	1745	1833	1882	1971	2020	2112			
		2238	2278	2379	2500	2757	2955#	3245	3263#	3305#	3308#						

CZKMA MACY11 30A(1052) 05-MAR-79 09:02 PAGE 79

N 6

CZKMAF.P11 05-MAR-79 09:02

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0078

.\$X = 000276 1094# 1099

. ABS. 007746 000

ERRORS DETECTED: 0

DSKW:CZKMAF,CZKMAF/SOL/CRF/NL:TOC=CZKMAF.P11

RUN-TIME: 10 10 .6 SECONDS

RUN-TIME RATIO: 111/21=5.1

CORE USED: 11K (21 PAGES)