

MSV11-D

MOS/CORE 0-124K EXR
CNKMAAO

AH-T472A-MC
FICHE 1 OF 1

MAY 1983
COPYRIGHT 82-83
MADE IN USA



IDENTIFICATION

PRODUCT NAME: CNKMA40 MOS/CORE 0-124K EXER
PRODUCT CODE: AC-T471A-MC
PRODUCT DATE: DECEMBER, 1982
MAINTAINER: DIAGNOSTICS SERVICES/ISS
AUTHOR: DIAG/ISS

COPYRIGHT (C) 1982, 1983
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

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 DOCUMENT.

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

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


```
6101
6105
6106 ;ERR # 0      ;[BUSER] BUS ERROR TRAP TO LOC. 4 OCCURRED
6107                ;THIS ERROR IS NOT PRINTED AND IS FOR 'APT' USE.
6108
6109 ;ERR # 1      ;[TSTTRP] FATAL DATA ERROR
6110                ;LOCATIONS 0000-430 FAILED 1'S + 0'S TEST.
6111                ;R0 = GOOD DATA
6112                ;R1 = ADDRESS OF FAILING LOCATION.
6113
6114 ;ERR # 2      ;[LPTSIZ] APT FATAL ERROR
6115                ;APT MEMORY TABLES NOT SETUP CORRECTLY.
6116                ;CHECK LOCATIONS $MAMS1 [430] TO $MADR4[446]
6117                ;, FOR CORRECT MEMORY SIZE DATA.
6118
6119 ;ERR # 3      ;[TSTSIZ] OPERATOR FATAL ERROR
6120                ;SELECTED MEMORY SIZE GREATER THAN 28K
6121                ;(30K SYSTEM DOES NOT NEED KT SUPPORT), BUT
6122                ;SR BIT12 (10000) NOT SET.
6123                ;SET BIT12 AND RESTART AT 200.
6124
6125 ;ERR # 4      ;[TSTSIZ] OPERATOR FATAL ERROR
6126                ;LOWEST SELECTED TEST LIMIT IS HIGHER THAN
6127                ;HIGHEST TEST LIMIT. SET LOCATIONS 'LOWTWO'[322]
6128                ;TO 'HIGHADD' [330] CORRECTLY AND RESTART
6129                ;AT 200.
6130
6131 ;ERR # 5      ;[TST0] TEST SEQUENCE ERROR
6132                ;TST0 HAS BEEN ENTERED OUT OF SEQUENCE
6133                ;TESTN SHOULD = 00
6134                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6135                ;IF POSSIBLE SELECT ANOTHER 4K BANK
6136                ;BANK 0 AND RERUN THE TEST ON THE FAILING MEMORY.
6137
6138 ;ERR # 6      ;[TST0] DUAL ADDRESSING ERROR
6139                ;FOR THIS ERROR THE GOOD DATA PRINTED IS AN
6140                ;ADDRESS. THIS IS THE ADDRESS SELECTED WHEN
6141                ;THE SAME DATA WAS WRITTEN INTO THE FAILING
6142                ;LOCATION. CHECK BANK SELECT CIRCUITRY
6143
6144 ;ERR # 7      ;[TST0] ADDRESS AND DATA ERROR
6145                ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA
6146                ;WRITTEN INTO THE FAILING LOCATION WAS IN
6147                ;ERROR ALSO.
6148
6149 ;ERR # 10     ;[TST0] DATA ERROR
6150                ;IF BAD DATA = 0000 COULD BE AN ADDRESSING
6151                ;ERROR , ELSE COMPARE GOOD AND BAD DATA FOR FAILING BITS.
6152
6153 ;ERR # 11     ;[TST0] ADDRESSING ERROR
6154                ;THE FAILING ADDRESS RESPONDED BUT IS NON-
6155                ;EXISTENT. MAY BE A DUAL ADDRESSING PROBLEM.
6156
6157 ;ERR # 12     ;[TST1] TEST SEQUENCE ERROR
6158                ;$TESTN [404] SHOULD = 01
6159                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
```

```
6160
6161 ;ERR # 13      ;[TST1] DATA ERROR
6162                ;COMPARE GOOD AND BAD PRINTED DATA, FAILING
6163                ;DATA BITS MAY SHORTED OR SWAPPED.
6164
6165 ;ERR # 14      ;[TST2] TEST SEQUENCE ERROR
6166                ;$TESTN [404] SHOULD = 02
6167                ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
6168
6169 ;ERR # 15      ;[TST2] ADDRESS OR DATA ERROR
6170                ;IF 'ADR ERR' NOT PRINTED THEN THE BYTE SELECT
6171                ;CIRCUITRY PROBABLY FAILED.
6172
6173 ;ERR # 16      ;[TST3] TEST SEQUENCE ERROR
6174                ;$TESTN [404] SHOULD = 03
6175                ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
6176
6177 ;ERR # 17      ;[TST3] DUAL ADDRESSING ERROR
6178                ;DUAL ADDRESSING PROBLEM FOR BITS THAT DIFFER
6179                ;IN GOOD AND BAD DATA PRINTOUT.
6180
6181 ;ERR # 20      ;[TST3] DUAL ADDRESSING ERROR
6182                ;FOR THIS ERROR THE DATA PRINTED IS AN ADDRESS.
6183                ;THIS IS THE ADDRESS THAT WAS SELECTED WHEN THE
6184                ;SAME DATA WAS WRITTEN INTO THE FAILING LOCATION.
6185
6186 ;ERR # 21      ;[TST3] DUAL ADDRESSING ERROR
6187                ;SAME AS ERROR #20 EXCEPT DIFFERENT DATA
6188                ; (SWAPPED BAKPAT) WAS WRITTEN.
6189
6190 ;ERR # 22      ;[TST4] TEST SEQUENCE ERROR
6191                ;$TESTN [404] SHOULD = 04.
6192                ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
6193
6194 ;ERR # 23      ;[TST4] DUAL ADDRESSING ERROR
6195                ;IF PASFLG = 0 THEN THE FAILING LOCATION
6196                ;AND FAILING DATA ARE DUAL ADDRESSES.
6197
6198 ;ERR # 24      ;[TST5] TEST SEQUENCE ERROR
6199                ;$TESTN [404] SHOULD = 05
6200                ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.
6201
6202 ;ERR # 25      ;[TST5] DATA ERROR
6203                ;DATA WRITE OR READ ERROR.
6204 ;ERR # 26      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
6205                ;IF PASFLG=0 FAILED MARCHING 1'S + 0'S IN
6206                ;      MAX TO MIN DIRECTION.
6207                ;IF PASFLG=1 FAILED MARCHING 1'S + 0'S IN
6208                ;      MIN TO MAX DIRECTION
6209                ;IF PASFLG=3 FAILED MARCHING 0'S + 1'S IN
6210                ;      MAX TO MIN DIRECTION.
6211
6212 ;ERR # 27      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
6213                ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA IS
6214                ;CHECKED IMMEDIATELY AFTER BEING WRITTEN.
6215
```

```

6216 ;ERR # 30      ;[TST6] TEST SEQUENCE ERROR
6217                ;$TESTN SHOULD = 06
6218                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6219
6220 ;ERR # 31      ;[TST6] VOLATILITY/REFRESH TEST ERROR
6221                ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
6222                ;IF PASFLG=1 THE FAILING LOCATION CHANGED WHILE
6223                ;              ANOTHER LOCATIONS WAS WRITTEN FOR
6224                ;              2 MS. THE OTHER LOCATION IS SAVED
6225                ;              IN SAVLOC [352]
6226                ;IF PASFLG=2 SWAPPED BAKPAT (77400 OR 77000)
6227                ;              WRITE OR READ ERROR.
6228                ;IF PASFLG=3 SAME AS IF PASFLG=2 EXCEPT
6229                ;              THE DATA IS SWAPPED BAKPAT.
6230
6231 ;ERR # 32      ;[TST7] TEST SEQUENCE ERROR
6232                ;$TESTN SHOULD = 07
6233                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6234
6235 ;ERR # 33      ;[TST7] SHIFTING DIAGONAL DATA ERROR
6236                ;IF PASFLG=0 BAKPAT WRITE OR READ ERROR.
6237                ;IF PASFLG=1 BAKPAT READ CHECK ERROR
6238                ;IF PASFLG GREATER THAN 1 BUT EVEN VALUE THEN:
6239                ;              THE FAILING LOCATION COULD NOT BE WRITTEN INTO.
6240                ;IF PASFLG GREATER THAN 1 BUT ODD VALUE THEN:
6241                ;              THE FAILING LOCATION WAS WRITTEN CORRECTLY
6242                ;              BUT LOST THE DATA.
6243
6244 ;ERR # 34      ;[TST10] TEST SEQUENCE ERROR
6245                ;$TESTN SHOULD = 10
6246                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6247
6248 ;ERR # 35      ;[TST10] BAKPAT DATA ERROR
6249                ;BAKPAT WRITE OR READ ERROR INTO THE FAILING LOCATION.
6250
6251 ;ERR # 36      ;[TST10] READ RECOVERY DATA ERROR
6252                ;              THIS ERROR CAN BE REPORTED BY TST10 AND TST11.
6253                ;              (THEY SHARE CODE). SEE $TESTN [404] FOR WHICH TEST FAILED.
6254                ;FOR BOTH TESTS COMPARE THE GOOD AND BAD DATA AT THE FAILING
6255                ;LOCATION TO SEE WHICH BITS FAILED.
6256
6257 ;ERR # 37      ;[TST10] READ RECOVERY DATA ERROR
6258                ;IDENTICAL TO THE PREVIOUS ERROR EXCEPT SWAPPED BAKPAT IS
6259                ;USED AS WRITE AND READ DATA.
6260
6261 ;ERR # 40      ;[TST11] TEST SEQUENCE ERROR
6262                ;$TESTN SHOULD = 11
6263                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6264
6265 ;ERR # 41      ;[TST12] TEST SEQUENCE ERROR
6266                ;$TESTN SHOULD = 12
6267                ;THE DIAGNOSTIC HAS BEEN CORRUPTED.
6268
6269 ;ERR # 42      ;[TST12] WORST CASE CORE TEST DATA ERROR
6270                ;IF PASFLG=1 COMPARE GOOD AND BAD DATA FOR FAILING BITS.
6271                ;IF PASFLG=2 THE FAILING LOCATION WAS WRITTEN AND READ

```

```
6272      :           WITH GOOD DATA,BUT FAILED READ CHECK
6273      :           READING IN THE MIN. TO MAX DIRECTION.
6274      :IF PASFLG=3  SAME CONDITIONS AS PASFLG=2 EXCEPT FAILED
6275      :           DOING THE READ CHECK FROM MAX TO MIN DIRECTION.
6276
6277 ;ERR # 43      :[TST12] WORST CASE CORE TEST DATA ERROR
6278      :           IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA WRITTEN
6279      :AND READ IS COMPLEMENTED.
6280
6281 ;ERR # 44      :[TST13] TEST SEQUENCE ERROR
6282      :$TESTN SHOULD = 13
6283      :           THE DIAGNOSTIC HAS BEEN CORRUPTED.
6284
6285 ;ERR # 45      :[TST13] WRITE RECOVERY TEST DATA ERROR
6286      :IF PASFLG=0  COMPARE GOOD AND BAD DATA FOR FAILING BITS.
6287      :IF PASFLG=77400 DATA ERROR FOUND WHILE DOING A SECOND READ CHECK.
6288      :IF PASFLG=77402 DATA ERROR FOUND IN FAILING LOCATION AFTER
6289      :           SMALL TEST PROGRAM RUN IN FAILING BANK.
6290
6291 ;ERR # 46      :[TST13] WRITE RECOVERY TEST DATA ERROR
6292      :           DATA ERROR FOUND JUST BEFORE THE SMALL TEST
6293      :WAS TO BE RUN IN THE FAILING BANK. TO AVOID 'BLOWING' UP
6294      :WHEN THE SMALL TEST IS RUN TST13 IS ABORTED.
6295
6296 ;ERR # 47      :[TST13] WRITE RECOVERY TEST DATA ERROR
6297      :           IDENTICAL TO ERROR #XXX EXCEPT THE DATA WRITTEN
6298      :AND READ IS DIFFERENT.(177667).
6299      :177667 IS THE COMPLEMENT OF 'JMP (R0)' (110) WHICH IS
6300      :THE ESCAPE FROM THE SMALL TEST PROGRAM RUN IN THE BANK
6301      :UNDER TEST.
6302
6303 ;ERR # 50      :[PARERR] PARITY TRAP ERROR
6304      :           PARITY TRAP TO 114 OCCURRED.
6305      :FOR THIS ERROR PRINTOUT THE 'GOOD DATA' IS ACTUALLY
6306      :THE FAILING PARITY MODULE UNIBUS ADDRESS.
6307      : SAVLOC [352] CONTAINS THE PC WHERE THE TRAP OCCURRED.
6308
6309 ;ERR # 51      :[PARITY] PARITY TRAP FATAL ERROR
6310      :           A PARITY TRAP TO 114 OCCURRED, BUT NO PARITY MODULES COULD BE FOUND
6311      :WITH AN ERROR BIT (BIT15) SET.
6312
6313 ;ERR # 52      :[NOMM] OPERATOR FATAL ERROR
6314      :           TESTING ABOVE 28K WAS SELECTED, BUT NO MEMORY MANAGEMENT
6315      :OPTION WAS FOUND.(30K SYSTEM DOES NOT NEED KT)
6316      :           RESET SWITCH OPTIONS AND RESTART AT 200.
6317
6318 ;ERR # 53      :[PARITY] OPERATOR FATAL ERROR
6319      :           PARITY TESTING WAS SELECTED BUT NO PARITY MODULES
6320      :WERE FOUND.
6321      :           RESET SWITCH OPTIONS AND START AT 200.
6322
6329 .REPT 0
6330
6331
6332 [6.3]  ERROR HISTORY
6333
6334
```

6335 LOCATIONS IN MEMORY ARE SET ASIDE TO COLLECT A HISTORY
6336 OF THE FAILING BITS IN A PARTICULAR MEMORY BANK. THIS
6337 DATA IS COLLECTED FOR EVERY ERROR REGARDLESS OF SWITCH
6338 SETTINGS.

6339
6340 NORMALLY THE DATA IS OUTPUT AT THE END OF TESTING, BUT
6341 IF CONTROL-C IS TYPED IT IS OUTPUT AT THE END OF THE
6342 CURRENT TEST.

6343
6344 THE ERROR HISTORY IS INTENDED TO HIGHLIGHT IF THE ERRORS
6345 ARE DUE TO 1 BIT FAILING OR ONLY ADDRESS ERRORS.

6346
6347
6348
6349 ERROR HISTORY FORMAT:

6350
6351
6352 ERROR BANK COUNT
6353 -----
6354

6355
6356 WHERE:

6357
6358 ERROR = BIT THAT FAILED [NUMBER OF THE FAILING BIT IN DECIMAL I.E.
6359 0-15 WILL BE TYPED OUT OR THE WORDS "ADR ERR" OR "PAR ERR" WILL
6360 BE TYPED OUT IF ADDRESS ERROR OR PARITY ERROR WAS SEEN
6361 IN THE SPECIFIC BANK OF MEMORY
6362 BANK = 4K MEMORY BANK IN WHICH THIS FAILURE WAS SEEN
6363 A 0 FOR 0 TO 4k, A 1 FOR 4 TO 8k AND SO ON
6364 COUNT = NUMBER OF TIMES THIS MEMORY BANK FAILED.
6365 (377 IS MAXIMUM FAILURE COUNT RECORDED.)

6366 [6.4] ERROR RECOVERY

6367
6368 IF THE PROGRAM IS HALTED AFTER REPORTING AN ERROR IT CAN EITHER
6369 BE CONTINUED OR RESTARTED AT 200 OR 250 (SEE SEC 4.2). HOWEVER FOR
6370 CPU'S THAT DESTROY CONTENTS OF REGISTERS AFTER COMING TO A HALT
6371 THE PROGRAM SHOULD ONLY BE RESTARTED.

6372
6373 [7.0] RESTRICTIONS

6374
6375
6376 MEMORY UNDER TEST SHOULD BE CONTIGUOUS. FOR SYSTEMS HAVING NON-
6377 CONTIGUOUS MEMORY THE MEMORY BOUNDARIES SHOULD BE DEFINED BY THE
6378 OPERATOR. (CONTIGUOUS MEMORY IS DEFINED AS A MEMORY THAT CAN BE
6379 BOTH READ AND WRITTEN IN CONSECUTIVE LOCATIONS.)

6380
6381
6382
6383 [8.0] MISCELLANEOUS

6384
6385
6386 [8.1] ADDRESS/BANK RANGES IN OCTAL AND DECIMAL

6387
6388 THIS REFERENCE TABLE CROSS REFERENCES THE MEMORY BANK NO.S,
6389 THE RANGE AND THE PAR USED WHEN MEMORY MANAGEMENT IS ENABLED.
6390

IT IS ALSO USEFUL TO SHOW STARTING ADDRESSES IN A PARTICULAR 4K BANK.

BANK NO.	DECIMAL RANGE	OCTAL RANGE	[PAGE ADDRESS REGISTER] USED/CONTENT		UNIBUS ADDRESS
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	
26	104K-108K	640000-657776	2	6400	
27	108K-112K	660000-677776	3	6600	
28	112K-116K	700000-717776	4	7000	
29	116K-120K	720000-737776	5	7200	
30	120K-124K	740000-757776	6	7400	
31	124K-128K	760000-777776	7	7600	772354

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.

6447 [8.2] EXECUTION TIME

6448
6449 HERE ARE SOME TYPICAL EXECUTION TIMES.

6450
6451 LSI-11 AND 4K:= 100 SECS.
6452 LSI-11 AND 8K:= 5 MINUTES.
6453

6454
6455
6456 [8.2] PASS COUNT AND TEST NO. LOCATIONS

6457
6458 \$PASS [406] = PASS COUNT - CLEARED BY START AT 200.

6459
6460 \$TESTN [404] = CURRENT TEST NO. AND RELOCATION, PARITY FLAGS.

6461 WHERE:
6462 LOW BYTE = TEST NO.
6463 IF BIT15 = 1 TEST IS RELOCATED
6464 IF BIT13 = 1 PARITY UNDER TEST.
6465

6466
6467 [8.4] STACK POINTER

6468
6469 THE STACK STARTS AT 500 WHEN THE PROGRAM IS NOT RELOCATED.
6470 SAVR6[350] CONTAINS THE STACK STARTING VALUE WHEN THE DIAGNOSTIC
6471 IS RELOCATED.
6472 SAVR6 ALSO CONTAINS THE STARTING ADDRESS OF THE PROGRAM WHEN
6473 IT IS RELOCATED.
6474

6475 [8.5] POWER FAIL

6476
6477 THE DIAGNOSTIC CAN BE POWER FAILED WITH NO ERRORS. TO USE,
6478 START THE TEST AS USUAL AND POWER DOWN THEN UP AT ANY TIME.
6479 THE PROGRAM SHOULD TYPE 'P' AND CONTINUE TO RUN FROM TEST 0
6480 IN THE SAME STATE [I.E. STATE OF RELOCATION] AS IT WAS BEFORE
6481 THE POWER WAS INTERRUPTED, HOWEVER IF THE DIAGNOSTIC WAS IN
6482 A MEMORY THAT CAN NOT HOLD DATA WITH THE POWER DOWN THEN THE
6483 PROGRAM WILL NOT RECOVER FROM POWER FAIL AND ON POWER-UP
6484 OPERATION IS UNDEFINED.
6485

6486
6487
6488 [9.0] PROGRAM DESCRIPTION

6489
6490
6491 [9.1] NARRATIVE FLOW CHART

6492
6493 THE TEST IS LOADED INTO LOCATIONS 0000 - 7744 BUT
6494 EXPANDS DEPENDING ON HOW MUCH MEMORY IS UNDER TEST.
6495 SEE STEP 6. BELOW FOR A DETAILED EXPLANATION.
6496

6497 THE FOLLOWING NARRATIVE FLOW CHART DESCRIBES MAJOR
6498 PROGRAM OPERATION. FOR THE PERSON WHO NEEDS DETAIL THE
6499 TAG ASSOCIATED WITH THE OPERATION IS GIVEN IN BRACKETS.

6500
6501 FOR THIS DISCUSSION SWITCH SETTINGS ARE IGNORED AND EVERYTHING IS
6502 ASSUMED ENABLED.

6503
6504
6505
6506
6507
6508
6509
6510
6511
6512
6513
6514
6515
6516
6517
6518
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
6552
6553
6554
6555
6556
6557
6558

1. [START] PRINT 'CNKMAA' 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 MONITOR, AND ALLOWS THE LOCATIONS OCCUPIED BY XXDP TO BE TESTED.
7. [CLRMEM] CALL 'PARITY' ROUTINE AND IF SELECTED, ENABLE ALL PARITY MODULES. 'PARMAP' [LOC. 352] CONTAINS A MAP OF PARITY MODULES FOUND. IF MODULE 172336 BIT 15 IS SET, IF #172334 FOUND BIT 14 IS SET ETC..
8. [CLRMEM] CLEAR MEMORY CURRENTLY UNDER TEST
9. [CONT] DISPATCH TO TST0
10. [TST0] EXECUTE TEST 0. SEE SECTION 10 FOR TEST DESCRIPTIONS.
11. [TSTSCP] COMES HERE AFTER EACH TEST AND IF CNTRL-C TYPED THEN GO TO ERROR HISTORY PRINTOUT.

```
6559         IF SR=2000 THEN HALT
6560         IF SR=40000 THEN LOOP ON TEST DEFINED BY <3:0>
6561         ELSE CONTINUE TO NEXT TEST.
6562
6563 12. [TST1-TST12] EXECUTE TST1-TST12 EACH TIME
6564     GOING TO STEP 9.
6565
6566 13. [TST13] TEST 13 IS DIFFERENT FROM TESTS 0-12,
6567     BECAUSE IT IS A SMALL PROGRAM ACTUALLY RUNNING
6568     IN THE MEMORY UNDER TEST. BEFORE THIS SMALL
6569     PROGRAM IS STARTED 'TST13 BNK XX' IS TYPED.
6570     THIS IS DONE IN CASE THE PROGRAM FAILS. THE
6571     USER CAN THEN AT LEAST TELL WHICH BANK OF MEMORY
6572     FAILED.
6573
6574 14. [RELOC] THE PROGRAM RELOCATES TO HIGH MEMORY
6575     TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG).
6576     WHERE 'ENDPRG' IS THE CONTENTS OF ENDSTK[306].
6577     I.E THE LAST PROGRAM ADDRESS. NOTE 'REL' IS
6578     PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
6579
6580 15. TESTS 0-13 ARE RUN AS DESCRIBED ABOVE EXCEPT
6581     ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED.
6582
6583 16. [RELOER] RELOCATE THE PROGRAM BACK TO LOWER
6584     MEMORY.
6585
6586 17. [LOWER] IF CONTROL-C TYPED GO PRINT ERROR
6587     HISTORY.
6588
6589 18. [TSTMM] IF MEMORY MANAGEMENT SELECTED AND AVAILABLE,
6590     RUN TESTS 0-13 ON THE FIRST 24K SLICE ABOVE 28K.
6591
6592 19. [CONTMM] CALL 'UPMM' TO UPDATE MEMORY MANAGEMENT
6593     PAR REGISTERS TO POINT TO THE NEXT 24K SLICE OF
6594     UPPER MEMORY.
6595
6596 20. [MAXADR] REPEAT STEPS 18 + 19 UNTIL ALL
6597     MEMORY ABOVE 28K IS TESTED.
6598
6599 21. [ENDPAS] PRINT ERROR HISTORY OF FAILING BITS
6600
6601 22. [$EOP] DISABLE PARITY MODULES.
6602     PRINT 'PASS#XX'
6603
6604 [9.2] TEST TITLES
6605
6606     SEE THE TEST HEADINGS IN THE LISTING FOR DETAILS ON EACH TEST.
6607
6608     TEST 0: TEST FOR PROPER BANK SELECTION
6609     TEST 1: CHECK DAT1/DAT0 LINES
6610     TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
6611     TEST 3: DUAL ADDRESS TEST A
6612     TEST 4: DUAL ADDRESS TEST B
6613
6614
```

6615 TEST 5: MARCHING 1'S AND 0'S
6616 TEST 6: CELLS' VOLATILITY TEST
6617 TEST 7: SHIFTING DIAGONAL
6618 TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
6619 TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
6620 TEST 12: WORST CASE TESTING FOR CORE MEMORY
6621 TEST 13: WRITE RECOVERY TEST
6622
6623

6624 [10.0] RXDP & ACT11 & APT OPERATION

6625 RXDP CHAIN MODE
6626 -----
6627

6628 OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
6629

- 6630 1. NO "CNKMAA" TITLE IS PRINTED.
6631 2. NO TEST 13 PRINTOUTS SUCH AS "TST13 BNK 00".
6632 3. THE PROGRAM ALWAYS HALTS ON ERROR.
6633 4. AT THE END OF TEST (\$ENDAD) CONTROL IS RETURNED TO
6634 THE RXDP CHAIN MONITOR VIA LOCATION 42.
6635 5. IF 30K SYSTEM ONLY 28K WILL BE TESTED IN XXDP CHAIN MODE
6636

6637 ACT11
6638 -----
6639

6640 OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:
6641

- 6642 1. NO PRINTOUTS EXCEPT ERROR PRINTOUTS.
6643 2. THE PROGRAM ALWAYS HALTS ON ERROR.
6644 3. AT THE END OF TEST (\$ENDAD) CONTROL IS RETURNED TO
6645 THE ACT11 MONITOR VIA LOCATION 42.
6646

6647 APT
6648 ---
6649

6650 OPERATION IS SIMILAR TO STAND ALONE EXCEPT:
6651

- 6652 1. THE SOFTWARE SWITCH REGISTER BECOMES LOCATION 422 (\$SWREG).
6653 2. AUTO SIZING CAN BE INHIBITED BY SETTING BIT 7 OF BYTE
6654 LOCATION 421 (\$ENVM).
6655 3. ALL PRINTOUTS CAN BE INHIBITED BY SETTING BIT 5 OF
6656 BYTE LOCATION 421 (\$ENVM).
6657 4. ALL ERRORS CAUSE LOCATION 400 (\$MSGTY) TO BE SET =
6658 0001 AND THE PROGRAM HALTS AT LOCATION 6240 (FATHLT).
6659 LOCATION 402 (\$FATAL) CONTAINS THE ERROR NO. IN THE
6660 LOW BYTE AND THE FAILING MEMORY BANK NO. IN THE HIGH
6661 BYTE.
6662

6663 NOTE: THE ENVIRONMENTAL MODE BYTE SHOULD BE SET TO 240 WHILE THE SOFTWARE
6664 ENVIRONMENTAL BYTE SHOULD BE SET TO 001.
6665
6666
6667

6668 .ENDR

```

6670
6671 .ENABL ABS
6672 .NLIST MD,MC,CND
6673 .LIST ME,BIN,SEQ,LOC
6678 .TITLE CNKMA
(1) ;*COPYRIGHT (C) MARCH 1982
(1) ;*DIGITAL EQUIPMENT CORP.
(1) ;*MAYNARD, MASS. 01754
(1) ;*
(1) ;*PROGRAM BY DIAGNOSTIC ENGINEERING
(1) ;*
(1) ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
(1) ;*
(1) 160000 $SWR=160000 ;:HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
6679
6709
6710
6711
6743 ;:TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS
6744
6752
6753
6760
6761 000240 SCOPE =NOP
6762
6763 000042 .=42
6764 000000 .WORD 0 ;FOR ACT/XXDP
6765
6766 .SBTTL ACT11 HOOKS
(1) ;:*****
(2) ;HOOKS REQUIRED BY ACT11
(1) $SVPC=. ;SAVE PC
(1) .=46
(1) 000046 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
(1) 000052 .=52
(1) 000052 .WORD 40000 ;:2)SET LOC.52 TO 40000
(1) 000044 .=$SVPC ;: RESTORE PC
6767
6768 000070 .=70
6769 000070 012737 000144 000024 PWRDN: MOV #PWRUP,@#24
6770 000076 000000 HALT
6771

```



```

6773      000102      000102      . = 102
6774      000102      000002      .WORD 2 ;DISMISS BEVNT;VER:1
6775
6776      000104      000104      . = 104
6777      000104      013727      000001 000400 ; GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
6778      000112      000000      BUSER: MOV @#1,$MSGTY ;TELL APT FATAL ERROR#000
6779      000112      000000      HALT ;*ERROR* TRAP TO LOC. 4 OCCURRED.
6780      ;114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS. SETUP IN
6781      ;ROUTINE 'BEGIN'.
6782      000120      . = 120
6783
6784
6788
6789      ;* WRITE MEMORY BACKGROUND
6790      ;* -----
6791      ;*
6792      ;* THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
6793      ;* THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES
6794      ;* THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE
6795      ;* HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE
6796      ;* SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
6797      ;*
6798
6799      000120      010401      WRTMEM: MOV R4,R1 ;SET R1 TO LOWEST LOCATION UNDER TEST
6800      000122      013700      000316      MOV @#BAKPAT,R0 ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
6801      000126      010021      2$: MOV R0,(R1)+ ;STARTING FROM THE LOWEST LOCATION WRITE THE
6802      000130      020105      CMP R1,R5 ;MEMORY TO BACK GROUND PATTERN
6803      000132      103775      BLO 2$
6804      000134      000207      RTS PC ;RETURN FROM THE SUBROUTINE
6805
6806      ;PWRUP IS LOADED AT LOC.144 AND VECTOR 140 INITIALIZED TO BE SPECIFIC
6807      ;TO 11/21 PROCESSOR.
6808      . = 140
6809      000140      170000      .WORD 170000 ;VER:1 START ADDRESS OF ODT
6810      000142      000300      .WORD 300 ;VER:1 PRIORITY 6
6811
6812      000144      013706      000350      PWRUP: MOV @#SAVR6,SP ;RESTORE STACK POINTER
6813      000150      012700      006112      MOV #PNTMES-BEGIN,R0
6814      000154      060600      ADD SP,R0 ;GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
6815      ;RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE.
6816      000156      004710      JSR PC,(R0) ;GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A 'P'
6817      000160      000120      .ASCIIZ /P/
6818      .EVEN
6819
6820      000162      000406      BR START
6821
6822      ;* SERVICE XXDP/ACT11
6823      000164      004710      $ENDAD: JSR PC,(R0) ;RETURN TO ACT11/XXDP MONITOR
6824      000166      000240      NOP ;IF QUICK VERIFY=RESET ELSE NOP
6825      000170      000240      NOP ;IF QUICK VERIFY=CLR #-1 ELSE INC #0
6826      000172      000240      NOP ;IF QUICK VERIFY=BR -4 ELSE NOP
6827      000174      000425      BR RESTRT ;REPEAT TEST UNDER ACT11/XXDP
6828
6829      000176      000176      . = 176
6830      000176      000000      SWREG: .WORD 0
6831

```

```

6832
6833
6834
6835
6836
6837 000200 013706 000350
6838 000204 012703 000412
6839 000210 005043
6840 000212 022703 000400
6841 000216 001374
6842 000220 105737 000042
6843 000224 001011
6844 000226 105737 000405
6845 000232 100406
6846 000234 004767 006344
6847 000240 047103 046513 040501
      000246 000060

6848
6849
6850 000250 012704 007744
6851 000254 012703 000346
6852 000260 012305
6853 000262 012306
6854 000264 010600
6855 000266 012746 000300
6856 000272 010046
6857 000274 000002
6858
6859
6860
6861
6862
6863
6864
6865
6866
6867
(1)
(2)
(1)
(2)
(1) 000276
(1) 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 000276
(1) 000276
(2)
(1)
(1)
(1)
(1) 000276
(1) 000276 000000
(1) 000300 000400
(1) 000302 001440
(1) 000304 002260

;*****
;SBTTL START AND RESTART ROUTINES
;* RESTART AT 200 TO CLEAR APT TABLES
;*****
START: MOV @SAVR6,SP ;SETUP STACK POINTER.
      MOV #SUNIT,R3 ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1$: CLR -(R3) ;CLEAR A MAILBOX LOCATION
      CMP #SMAIL,R3 ;DONE?
      BNE 1$ ;BRANCH IF NO
      TSTB @#42 ;ACT11 MODE?
      BNE RESTRT ;BRANCH IF YES
      TSTB @#$TESTN+1 ;ARE WE RELOCATED?
      BMI RESTRT ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
      JSR PC,TPCRLF ;PRINT TITLE
      .ASCIZ /CNKMAA0/

      .EVEN

RESTRT: MOV #ENDPRG,R4 ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
      MOV #SAVR5,R3 ;CAUSE R3 TO POINT TO THE LOCATION SAVR5
      MOV (R3)+,R5 ;RESTORE R5
      MOV (R3)+,SP ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
      MOV SP,R0 ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
      MOV #300,-(SP) ;SET HIGH PRIORITY FOR RTI;VER:1
      MOV R0,-(SP)
      RTI ;GO TO 'START'-MAY BE RELOCATED.
           ;IF RELOCATED SEE LOCATION SAVR6 FOR START.

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

$APTHD:
$SHIBTS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
$STSM: .WORD 800. ;RUN TIM OF LONGEST TEST
$PASTM: .WORD 1200. ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)

```

(1)	000306	000000	\$UNITM: .WORD	;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1)	000310	000024	.WORD	\$ETEND-\$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
6868				
6869				
6870		000405	REL=\$TESTN+1	;IT WILL BE 0 IF THE PROGRAM IS IN THE LOWER
6871				;CORE. BIT 7 OF THE BYTE WILL BE SET IF THE
6872				;PROGRAM IS IN A RELOCATED STATE AND BIT 5
6873				;WILL BE SET IF PARITY BITS ARE BEING TESTED
6874		000276	MMAVA: .=\$APTHD	
6875	000276			;THIS BYTE IS USED TO DETERMINE IF MEMORY
6876				;MANAGEMENT IS AVAILABLE OR NOT
6877				
6878		000277	TYPENB: .=\$MMAVA+1	
6879	000277			;THIS BYTE IS USED TO DETERMINE IF THE
6880				;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
6881				
6882		000300	\$PRERR: .=\$TYPENB+1	
6883	000300			;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
6884				;A PARITY ERROR
6885				
6886		000301	\$ADERR: .=\$PRERR+1	
6887	000301			;THIS BYTE IS USED TO DETERMINE IF THE
6888				;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
6889				
6890		000302	STRTDI: .=\$ADERR+1	
6891	000302			
6892		000304	LOWBNK: .=\$STRTDI+2	
6893	000304			
6894		000306	PASFLG: .=\$LOWBNK+2	
6895	000306			;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
6896				;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
6897				;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
6898				
6899		000310	ENDSTK: .=\$PASFLG+2	
6900	000310			
6901		000312	PBNK: .=\$ENDSTK+2	
6902	000312			;HOLDS BANK UNDER TEST FOR 'TST BNK XX' PRINTOUT.
6903	000312		DECWRD: .=\$PBNK	
6904		000314	TYPCNT: .=\$DECWRD+2	
6905	000314	000	.BYTE 0	;THIS BYTE DETERMINES THE NUMBER OF WORDS
6906				;TO BE TYPED
6907	000315	000	SAVKBB: .BYTE 0	;THIS LOCATION IS USED TO SAVE THE CHARACTER
6908				;HIT BY THE OPERATOR
6909			.EVEN	
6910				
6911				
6912		177560	TKS= 177560	
6913		177562	\$KBB= 177562	
6914		177564	\$TPS= 177564	
6915		177566	\$TPB= 177566	
6916		177572	SRO= 177572	
6917	000316	000377	BAKPAT: .WORD 377	;BACKGROUND PATTERN WRITTEN TO MEMORY.
6918				
6919	000320	000000	SWAPAT: .WORD	
6920	000322	000430	RELBOT: BEGIN-50	;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.
6921				

```

6922      ;*****
6923      ;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR
6924 000324 000000      LOWTWO: 0      ;HOLDS BITS 17:16 OF LOW TEST ADDRESS
6925 000326 000000      LOWADD: 0      ;HOLDS BITS 15:0 OF LOW TEST ADDRESS
6926
6927 000330 000000      HIGHTWO: 0      ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
6928 000332 037776      HIGHADD: 37776      ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
6929      ;*****
6930
6931 000334 000000      $HIMAX: 0      ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
6932 000336 017776      $MAXM: 17776      ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
6933
6934 000340 000000      MAXMEM: .WORD      ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
6935
6936 000342 000000      SAVMAX: .WORD
6937 000344 000000      SAVR4: .WORD
6938 000346 000000      SAVR5: .WORD
6939
6940      ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
6941 000350 000500      SAVR6: .WORD BEGIN      ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
6942 000352 000000      PARMAP: 0      ;MAP OF PARITY MODULES UNDER TEST.
6943 000354 000000      SAVLOC: 0      ;TEST 6 STORES ERROR INFO HERE
6944 000356 000000      PARSP: 0      ;SAVE SP DURING PARITY ERROR TRAP.
6945 000360 000000      PARPS: 0      ;SAVE PSW DURING PARITY ERROR TRAP.
6946      ;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
6947      ;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
6948      ;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE.
6949      ;IN THIS CRUDE FASHION.
6950
6951
6952
6953
6954
6955
6956
6957
6958      ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
  
```

6960 000400
 6961
 (1)
 (2)
 (1)
 (1) 000400
 (1) 000400 000000
 (1) 000402 000000
 (1) 000404 000000
 (1) 000406 000000
 (1) 000410 000000
 (1) 000412 000000
 (1) 000414 000000
 (1) 000416 000000
 (1) 000420
 (1) 000420 000
 (1) 000421 000
 (1)
 (1) 000422 000000
 (1) 000424 000000
 (1) 000426 000000
 (1)
 (1)
 (1)
 (1)
 (1)
 (1)
 (1) 000430 000
 (1) 000431 000
 (1)
 (1)
 (1)
 (1)
 (1) 000432 000000
 (1)
 (1) 000434 000
 (1) 000435 000
 (1) 000436 000000
 (1) 000440 000
 (1) 000441 000
 (1) 000442 000000
 (1) 000444 000
 (1) 000445 000
 (1) 000446 000000
 (1) 000450
 (1)
 6962
 6963
 6964

.=400
 .SBTTL APT MAILBOX-ETABLE
 ::*****
 .EVEN
 \$MAIL: ::APT MAILBOX
 \$MSGTY: .WORD AMSGTY ::MESSAGE TYPE CODE
 \$FATAL: .WORD AFATAL ::FATAL ERROR NUMBER
 \$TESTN: .WORD ATESTN ::TEST NUMBER
 \$PASS: .WORD APASS ::PASS COUNT
 \$DEVCT: .WORD ADEVCT ::DEVICE COUNT
 \$UNIT: .WORD AUNIT ::I/O UNIT NUMBER
 \$MSGAD: .WORD AMSGAD ::MESSAGE ADDRESS
 \$MSGLG: .WORD AMSGLG ::MESSAGE LENGTH
 \$ETABLE: ::APT ENVIRONMENT TABLE
 \$ENV: .BYTE AENV ::ENVIRONMENT BYTE
 \$ENVN: .BYTE AENVN
 ::ENVIRONMENT MODE BITS
 \$SWREG: .WORD ASWREG ::APT SWITCH REGISTER
 \$USWR: .WORD AUSWR ::USER SWITCHES
 \$CPUOP: .WORD ACPUOP ::CPU TYPE, OPTIONS
 BIT 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
 \$MAMS1: .BYTE AMAMS1 ::HIGH ADDRESS, M.S. BYTE
 \$MTYP1: .BYTE AMTYP1 ::MEM. TYPE, BLK#1
 MEM. TYPE BYTE -- (HIGH BYTE)
 900 NSEC CORE=001
 300 NSEC BIPOLAR=002
 500 NSEC MOS=003
 \$MADR1: .WORD AMADR1 ::HIGH ADDRESS, BLK#1
 MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
 \$MAMS2: .BYTE AMAMS2 ::HIGH ADDRESS, M.S. BYTE
 \$MTYP2: .BYTE AMTYP2 ::MEM. TYPE, BLK#2
 \$MADR2: .WORD AMADR2 ::MEM. LAST ADDRESS, BLK#2
 \$MAMS3: .BYTE AMAMS3 ::HIGH ADDRESS, M.S. BYTE
 \$MTYP3: .BYTE AMTYP3 ::MEM. TYPE, BLK#3
 \$MADR3: .WORD AMADR3 ::MEM. LAST ADDRESS, BLK#3
 \$MAMS4: .BYTE AMAMS4 ::HIGH ADDRESS, M.S. BYTE
 \$MTYP4: .BYTE AMTYP4 ::MEM. TYPE, BLK#4
 \$MADR4: .WORD AMADR4 ::MEM. LAST ADDRESS, BLK#4
 \$ETEND:
 .MEXIT
 ::*****
 .SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.


```

6966
6967
6968 000450 177570
6969
6970 000500 000500
6971 000500 010706
6972
6973 000502 005746
6974 000504 010637 000350
6975 000510 012737 000070 000024
6976 000516 005037 000300
6977 000522 005037 000314
6978 000526 012700 000114
6979 000532 012710 005504
6980 000536 060720
6981 000540 012710 000300
6982 000544 105737 000405
6983 000550 100002
6984 000552 000167 000614
6985
6986 000556 005737 000406
6987 000562 001402
6988 000564 000167 000430
6989 000570 012704 007744
6990 000574 012700 000377
6991 000600 010037 000316
6992 000604 005001
6993 000606 012124
6994 000610 020127 000400
6995 000614 103774
6996 000616 005741
6997 000620 010011
6998
6999 000622 020011
7000 000624 001403
7001
7002 000626 004767 005334
(1) 000632 000001
(1)
7003 000634 000300
7004 000636 001370
7005 000640 005701
7006 000642 001365
7007 000644 012701 000400
7008 000650 014441
7009 000652 005701
7010 000654 001375
7011 000656 012700 000006
7012 000662 012710 000300
7013 000666 012740 000700
7014 000672 005777 177552
7015 000676 000404
7016 000700 022626
7017 000702 012737 000176 000450
7018
7019

```

```

*****
SWR: 177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER

BEGIN: MOV PC,SP ;SET UP STACK POINTER TO EQUAL BEGIN ADDRESS
        TST -(SP)
        MOV SP,@#SAVR6 ;SAVE SP FOR FUTURE USE
        MOV #PWRDN,@#24 ;PREPARE FOR ANY FUTURE POWER DOWN
        CLR @#SPRERR
        CLR @#TYPCNT
        MOV #114,R0 ;PREPARE TO SETUP PARITY TRAP VECTOR
        MOV #PARERR-,-6,(R0)
        ADD PC,(R0)+ ;TO PARERR
        MOV #300,(R0) ;AND PSW OF 300;VER:1
        TSTB @#REL ;IS THIS CODE RELOCATED?
        BPL ONEPAS ;BRANCH IF NO
        JMP TSTREL ;THIS CODE IS RELOCATED SO GET TEST SIZE.

ONEPAS: TST @#SPASS ;IS THIS THE FIRST PASS?
        BEQ TSTRP ;BRANCH IF YES (TEST TRAP CATCHER ADDRESSES)
        JMP SETSTK ;GET THE TEST SIZE
TSTRP: MOV #ENDPRG,R4 ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
        MOV #377,R0
        MOV R0,@#BAKPAT
        CLR R1
2$: MOV (R1)+,(R4)+ ;SAVE FROM 0000 TO BEGIN-30 AT END OF PROGRAM FOR NOW
        CMP R1,#$MAIL
        BLO 2$
3$: TST -(R1) ;PREPARE TO TEST THE TRAP VECTORS
4$: MOV R0,(R1) ;CHECK THE TRAP VECTORS FOR THE CAPABILITY
        ;OF HOLDING 0'S & 1'S
        CMP R0,(R1) ;IS THE DATA OK?
        BEQ 6$ ;BRANCH IF YES

        JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
        1 ;*****ERROR NUMBER 1*****

6$: SWAB R0
        BNE 4$
        TST R1 ;IF WE HAVE NOT REACHED THE LOWEST MEMORY LOCATION
        BNE 3$ ;THEN REPEAT FROM 3$
        MOV #MAIL,R1
8$: MOV -(R4),-(R1) ;RESTORE TRAP CATCHER ETC.
        TST R1
        BNE 8$
SETSWR: MOV #6,R0
        MOV #300,(R0) ;SET UP TIME OUT TRAP PSW;VER:1
        MOV #4$,-(R0) ;AND THE RETURN ADDRESS
2$: TST @SWR ;DOES THE SWITCH REGISTER POINTED BY SWR EXIST ?
        BR 5$ ;BRANCH IF YES
4$: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
        MOV #SWREG,@#SWR ;AND PLACE THE ADDRESS OF THE SWITCH REGISTER
        ;DESIGNED FOR THE COMPUTERS NOT HAVING HARDWARE
        ;SWITCH REGISTER AND RUNNING STAND ALONE

```

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-1
 CNKMAA.P11 27-DEC-82 13:15

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

SEQ 0019

```

7020 000710 105737 000420 5$: TSTB @#SENV ;RUNNING UNDER APT?
7021 000714 001403 BEQ APTSIZ ;BRANCH IF NO
7022 000716 012737 000422 000450 MOV #SSWREG,@#SWR ;SET SWR EQUAL TO APT SWITCH REGISTER.
7023
7024
7025
7026
7027 ;APTSIZ- THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE AND WHEN
7028 ;A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO GIVEN HIGH ADDRESS.
7029 ; IF APT DEFINES SIZE THE LOW TEST ADDRESS MUST=00000.(DUE TO ETABLE FORMAT)
7030 ;FLOW;
7031 ; IF BLOCK 4 (OR 3,2,1) TYPE NON ZERO THEN GET APT HIGH ADDRESS AND EXIT.
7032 ; ELSE SEND ERROR #3
7033 ;NOTE: THE MEMORY TYPE IS IGNORED SINCE ALL TESTS ARE RUN REGARDLESS OF MEMORY TYPE.
7034
7035 000724 012703 000340 APTSIZ: MOV #MAXMEM,R3 ;POINT R3 TO MAXMEM.
7036 000730 013737 000330 000334 MOV @#HIGHTWO,@#$HIMAX ;IN CASE NO SELF SIZING DONE.
7037 000736 013737 000332 000336 MOV @#HIGHADD,@#$MAXM ;IN CASE NO SELF SIZING DONE.
7038 000744 105737 000421 TSTB @#SENV ;DOES APT ALLOW SELF SIZING?
7039 000750 100021 BPL TRYSR ;BRANCH IF YES
7040
7041 000752 012701 000451 1$: MOV #SMTYP4+4,R1 ;POINT R1 TO BLOCK TYPE 4(+4)
7042 000756 162701 000004 SUB #4,R1 ;POINT R1 TO NEXT BLOCK TYPE.
7043 000762 105711 TSTB (R1) ;IS THE BLOCK TYPE NON ZERO?
7044 000764 001006 BNE 2$ ;BRANCH IF YES (MEMORY EXISTS)
7045 000766 020127 000431 CMP R1,#SMTYP1 ;ALL APT BLOCK TYPES BEEN CHECKED?
7046 000772 101371 BHI 1$ ;BRANCH IF NO
7047
7048 000774 004767 005166 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
7049 (1) 001000 000002 2 JSR 2 ;*****ERROR NUMBER 2*****
7050
7051 001002 004767 006324 2$: JSR PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO $MAXM + $HIMAX
7052 001006 004767 006320 JSR PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO HIGHADD+HIGHTWO
7053 001012 000464 BRTPSZ: BR TYPISZ ; TYPE THE SIZE OF MEMORY UNDER TEST
7054
7055 001014 032777 000100 177426 TRYSR: BIT #100,@SWR ;USER DEFINED MEMORY TEST BOUNDARIES??
7056 001022 001060 BNE TYPISZ ;BRANCH IF YES (DON'T SIZE MEMORY)
7057
7058
7059
7060 001024 010401 SLFSIZ: MOV R4,R1 ;SETUP R1 AND R4 TO THE LOWEST ADDRESS OF MEMORY
7061 001026 012710 001072 MOV #4$,(R0) ;SET UP RETURN ADDRESS FROM TIME OUT TRAP TO 4$
7062 001032 011111 2$: MOV (R1),(R1) ;WRITE A MEMORY LOCATION INTO ITSELF AND TRAP IF NXM
7063 001034 062701 000002 ADD #2,R1 ;ADD 2 TO THE ADDRESS POINTER
7064 001040 022701 170000 CMP #170000,R1 ;CHECK IF BEYOND 30K MEMORY BOUNDARY
7065 001044 101372 BHI 2$ ;KEEP ON SIZING UP THE MEMORY UNTIL NXM TRAP
7066 ;(TIME OUT TRAP) IS ENCOUNTERED
7067 ;OR 30K BOUNDARY REACHED
7068 001046 005737 000042 TST @#42 ;IF NOT XXDP
7069 001052 001410 BEQ 5$ ; THEN CONTINUE
7070 001054 023737 000042 000046 CMP @#42,@#46 ; ELSE IF NOT XXDP CHAINING
7071 001062 001404 BEQ 5$ ; THEN CONTINUE
7072 001064 162701 010000 SUB #10000,R1 ; ELSE SET MAX. MEM. AT 28K
7073 001070 000401 BR 5$

```

7074									
7075	001072	022626		4\$:	CMP	(SP)+,(SP)+		:RESTORE THE STACK POINTER	
7076	001074	004767	005764	5\$:	JSR	PC,MMNG		:SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE	
7077								:AND IF IT HAS TO BE TESTED	
7078	001100	105737	000276		TSTB	@#MMAVA		:SEE IF MEMORY MANAGEMENT HAS TO BE TESTED	
7079	001104	001416			BEQ	12\$:IF NO MEM. MANG. THEN GO TO 12\$	
7080	001106	012710	001120	6\$:	MOV	#8\$,(R0)		:SET UP THE RETURN ADDRESS FROM TRAP TO 8\$	
7081	001112	012701	020000		MOV	#20000,R1		:BEGIN CHECKING MEMORY ABOVE 28K	
7082	001116	000745			BR	2\$			
7083	001120	022626		8\$:	CMP	(SP)+,(SP)+		:RESTORE STACK POINTER	
7084	001122	022701	160000		CMP	#160000,R1		:IF R1 DID NOT READ ALL THE LOCATIONS POINTED BY	
7085								:PAGE ADDRESS REGISTER 6 THEN IT HAS REACHED THE	
7086								:MAXIMUM AVAILABLE MEMORY	
7087	001126	001005			BNE	12\$:IN WHICH CASE GO TO 12\$	
7088	001130	013702	172352		MOV	@#172352,R2		:PREPARE TO UPDATE MEMORY MANAGEMENT REGISTERS	
7089	001134	004767	005730		JSR	PC,MMREG		:OTHERWISE GO TO UPDATE MEM. MANG. REGISTERS	
7090	001140	000762			BR	6\$			
7091	001142	024341		12\$:	CMP	-(R3),-(R1)		:CAUSE R3 TO POINT TO LOCATION \$MAXM AND R1	
7092								:TO THE MAXIMUM AVAILABLE MEMORY	
7093	001144	004767	006072		JSR	PC,PUTADR		:GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1	
7094								:AT LOCATIONS \$MAXM AND \$HIMAX	
7095	001150	024343			CMP	-(R3),-(R3)		:MAKE R3 POINT TO HIGHADD	
7096	001152	004767	006064		JSR	PC,PUTADR		:PLACE THE ADDRESS IN R1 AT LOCATIONS HIGHADD	
7097								:AND HIGHTWO	
7098	001156	005743			TST	-(R3)			
7099	001160	005043			CLR	-(R3)		:CLEAR THE LOCATION LOWADD	
7100	001162	005043			CLR	-(R3)		:AND LOWTWO	
7101	001164	012720	000104	TYPSIZ:	MOV	#BUSER,(R0)+		:SET UP VECTOR FOR ANY FUTURE TRAP	
7102	001170	010403			MOV	R4,R3		:SET R3 TO POINT TO THE LOWEST AVAILABLE MEMORY	
7103								:LOCATION	
7104	001172	012701	000324		MOV	#LOWTWO,R1			
7105	001176	004767	005370		JSR	PC,PCRLF		:TYPE CR/LF	
7106	001202	004767	005536		JSR	PC,OCTTYP		:TYPE LOW TEST ADDRESSE (LOWTWO+LOWADD)	
7107	001206	004767	005272	TYPMEM:	JSR	PC,\$TYPE		:TYPE '-'	
7108	001212	000055			.ASCIZ	/-/			
7109					.EVEN				
7110	001214	004767	005524		JSR	PC,OCTTYP		:TYPE HIGHEST TEST ADDRESS (HIGHTWO+HIGHADD)	
7111	001220	012703	000330	SETSTK:	MOV	#HIGHTWO,R3		:MAKE R3 POINT TO THE HIGH ORDER BITS OF TOP ADDRESS	
7112	001224	004767	006116		JSR	PC,\$GTSIZ		:GET THE BITS 13-17 OF THE TOP ADDRESS	
7113								:PLACED IN BITS 0-4 OF R2	
7114	001230	010401			MOV	R4,R1		:SET R1 TO LOWEST TEST ADDRESS	
7115									
7116	001232	062704	000022	4\$:	ADD	#18.,R4		:APPEND THE ERROR STACK FOR THE MEMORY UNDER	
7117								:TEST TO THE END OF THE PROGRAM	
7118	001236	005302			DEC	R2			
7119	001240	002374			BGE	4\$			
7120	001242	022737	167776	000336	CMP	#167776,@#\$MAXM		:CHECK IF THIS IS A 30K SYSTEM	
7121	001250	001002			BNE	5\$:BRANCH IF NOT	
7122	001252	062704	000022		ADD	#18.,R4		:SAVE ANOTHER BANKS WORTH OF ERROR STACK	
7123	001256	010437	000310	5\$:	MOV	R4,@#ENDSTK		:SAVE THE ADDRESS OF THE END OF THE ERROR STACK	
7124	001262	005021		6\$:	CLR	(R1)+		:CLEAR THE ERROR STACK	
7125	001264	020104			CMP	R1,R4			
7126	001266	101775			BLOS	6\$			
7127	001270	012737	157776	000340	MOV	#157776,@#MAXMEM		:SET MAXMEM TO MAXIMUM VIRTUAL ADDRESS	
7128	001276	005723			TST	(R3)+		:TESTING MEMORY MANAGEMENT?	
7129	001300	001005			BNE	SAVLD		:BRANCH IF YES (GO SAVE LOADERS AT TOP OF VIRTUAL MEMORY	

7130	001302	021327	170000		CMP	(R3),#170000	: IS THE VIRTUAL ADDRESS ABOVE 167776?
7131	001306	103002			BHIS	SAVLDR	: BRANCH IF YES (GO SAVE LOADERS)
7132	001310	011363	000002		MOV	(R3),2(R3)	: OTHERWISE MAKE THE CONTENTS OF LOCATION MAXMEM
7133							: EQUAL TO THE MAXIMUM AVAILABLE MEMORY
7134							: AND FALL INTO SAVE LOADERS.
7135							
7136	001314	004767	006100		SAVLDR: JSP	PC,CLRMM	: DISABLE THE MEMORY MANAGEMENT UNIT
7137	001320	005723			TST	(R3)+	: MAKE R3 TO POINT TO THE LOCATION MAXMEM
7138	001322	011305			MOV	(R3),R5	: R5 CONTAINS THE ADDRESS OF MAXIMUM AVAILABLE MEM.
7139							
7140							: IF ONLY 4K BEING TESTED DON'T SAVE LOADERS
7141							
7142	001324	020527	017776		CMP	R5,#17776	: ONLY TESTING 4K MAX?
7143	001330	103416			BLO	4\$: BRANCH IF YES (DON'T SAVE LOADERS)
7144							
7145	001332	162705	000276	3\$:	SUB	#276,R5	: PREPARE TO SAVE 300 BYTES OF THE LOADERS
7146	001336	005737	000042		TST	@#42	: IF RUNNING UNDER XXDP
7147	001342	001406			BEQ	2\$: THEN CONTINUE
7148	001344	023737	000042	000046	CMP	@#42,@#46	: ELSE IF NOT UNDER XXDP CHAIN MODE
7149	001352	001402			BEQ	2\$: THEN CONTINUE
7150	001354	162705	005376		SUB	#<1502.*2>-276,R5	: ELSE SAVE 1500. WORDS FOR XXDP CHAIN MODE
7151	001360	012524		2\$:	MOV	(R5)+,(R4)+	: SAVE LOADER
7152	001362	020513			CMP	R5,(R3)	
7153	001364	01775			BLOS	2\$	
7154	001366	012323		4\$:	MOV	(R3)+,(R3)+	: SAVE THE CONTENTS OF LOCATION MAXMEM IN SAVMAX
7155	001370	010423			MOV	R4,(R3)+	: AND THE CONTENTS OF R4 AT SAVR4
7156							
7157	001372	010537	000346		TSTREL: MOV	R5,@#SAVR5	: SAVE HIGHEST VIRTUAL ADDRESS+2
7158	001376	004767	006016		TSTSIZ: JSR	PC,CLRMM	: GO TO DISABLE MEMORY MANAGEMENT UNIT
7159	001402	005745			TST	-(R5)	: SET R5 BACK TO HIGHEST VIRTUAL ADDRESS
7160	001404	012703	000324	1\$:	MOV	#LOWTWO,R3	: PREPARE TO LOAD R4 AND R5 WITH THE MEMORY BOUNDRIES
7161	001410	005723			TST	(R3)+	: IF THE BITS 16,17 OF THE LOWEST LOCATION UNDER
7162							: TEST ARE NON ZERO
7163	001412	001003			BNE	2\$: THEN GO TO 2\$
7164	001414	021327	157776		CMP	(R3),#157776	: IF THE LOWEST LOCATION UNDER TEST IS HIGHER THAN
7165							: 157776 THEN GO TO TEST MEMORY MANAGEMENT
7166	001420	103411			BLO	4\$	
7167	001422	032777	010000	177020	2\$:	BIT	#10000,@SWR
7168	001430	001003			BNE	3\$: IS MEMORY MANAGEMENT SELECTED?
7169	001432	004767	004530		JSR	PC,FATERR	: YES ALL IS WELL
(1)	001436	000003			3		: *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1)							: *****ERROR NUMBER 3*****
7170	001440	000167	003512	3\$:	JMP	TSTMM	: GO TO TEST MEMORY MANAGEMENT
7171	001444	020423		4\$:	CMP	R4,(R3)+	: COMPARE TOP OF PROGRAM (WITH SAVED LOADERS) TO
7172							: LOWEST LOCATION UNDER TEST
7173							
7174	001446	103002			BHIS	6\$	
7175	001450	016304	177776		MOV	-2(R3),R4	: ADJUST R4 TO POINT TO THE LOWEST LOCATION UNDER TEST
7176	001454	005723		6\$:	TST	(R3)+	: IF BITS 16-17 OF HIGHEST LOCATION TO BE TESTED
7177	001456	001003			BNE	8\$: ARE NON ZERO THEN GO TO 8\$
7178	001460	021305			CMP	(R3),R5	: OTHERWISE SEE IF THE HIGHEST LOCATION TO BE
7179							: TESTED IS HIGHER THAN HIGHEST VIRTUAL ADDRESS
7180	001462	101001			BHI	8\$: IF SO THEN GO TO 8\$
7181	001464	011305			MOV	(R3),R5	: MODIFY R5
7182	001466	105737	000405	8\$:	TSTB	@#REL	: ARE WE RELOCATED.?
7183	001472	100014			BPL	10\$: BRANCH IF NO

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-4
 CNKMAA.F11 27-DEC-82 13:15 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0022

7184	001474	013704	000322	MOV	@#RELBOT,R4	;SET BOTTOM TEST ADDRESS WHEN RELOCATED.
7185	001500	020527	017776	CMP	R5,#17776	;ARE WE RELOCATED IN BANK 0?
7186	001504	103402		BLO	9\$;BRANCH IF YES
7187	001506	012705	017776	MOV	#17776,R5	;ELSE SET HIGH MEMORY UNDER TEST=4K
7188						
7189	001512	020405		9\$: CMP	R4,R5	;IS LOW LIMIT LOWER THAN HIGH LIMIT?
7190	001514	103403		BLO	10\$;BRANCH IF YES
7191	001516	004767	004444	JSR	PC,FATERR	;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1)	001522	000004		4		;*****ERROR NUMBER 4*****
(1)						
7192	001524	012703	000342	10\$: MOV	#SAVMAX,R3	
7193	001530	011343		MOV	(R3),-(R3)	;RESTORE THE CONTENTS OF MAXMEM
7194	001532	062713	000002	MEMTST: ADD	#2,(R3)	;MAKE THE CONTENTS OF MAXMEM = MAXIMUM AVAILABLE
7195						;MEMORY +2
7196	001536	005725		TST	(R5)+	;AND SET R5=MAX MEMORY+2
7197						
7198						;CLEAR MEMORY UNDER TEST
7199						
7200	001540	010500		CLRMEM: MOV	R5,R0	;MOVE HIGH ADDRESS TO R0
7201	001542	005040		2\$: CLR	-(R0)	;BEGIN CLEARING THE MEMORY FROM THE TOP
7202	001544	020004		CMP	R0,R4	;UNTIL THE BOTTOM IS REACHED
7203	001546	101375		BHI	2\$	
7204	001550	012702	000001	MOV	#1,R2	;SET R2 TO ENABLE PARITY MODULE CODE.
7205	001554	004767	005740	JSR	PC,PARITY	;ENABLE PARITY IF WANTED AND AVAILABLE.
7206	001560	012702	000316	MOV	#BAKPAT,R2	
7207	001564	012212		MOV	(R2)+,(R2)	;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
7208	001566	000312		SWAB	(R2)	
7209	001570	017702	176654	MOV	@SWR,R2	;LOAD R2 WITH THE OPTIONS STORED AT \$SWREG
7210	001574	042702	177760	BIC	#177760,R2	;ONLY LEAVE THE LOWER 4 BITS OF \$SWREG IN R2 TO GO TO
7211						;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
7212						
7213						
7214						
7215						;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
7216						
7217	001600	005037	000306	CONT: CLR	@#PASFLG	;INIT SUBTEST PASS FLAG.
7218	001604	110237	000404	MOVB	R2,@#STESTN	;SET UP \$STESTN WITH THE TEST NUMBER GOING
7219						;TO BE EXECUTED
7220	001610	010401		LOOP: MOV	R4,R1	;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
7221	001612	010400		MOV	R4,R0	;PLACE THE ADDRESS OF THE LOWEST LOCATION UNDER
7222						;TEST IN R0
7223	001614	010403		MOV	R4,R3	;AND IN R3
7224	001616	006302		ASL	R2	
7225	001620	060702		ADD	PC,R2	
7226	001622	066207	000004	ADD	TBL-,(R2),PC	;GO TO THE TEST #
7227						;STORED IN BITS 0-3 OF SWITCH REGISTER
7228						
7229						
7230	001626	000102		TBL: TST0-TBL		;RELATIVE ADDRESS OF TEST # 0
7231	001630	000340		TST1-TBL		;RELATIVE ADDRESS OF TEST # 1
7232	001632	000440		TST2-TBL		;RELATIVE ADDRESS OF TEST # 2
7233	001634	000550		TST3-TBL		;RELATIVE ADDRESS OF TEST # 3
7234	001636	001016		TST4-TBL		;RELATIVE ADDRESS OF TEST # 4
7235	001640	001126		TST5-TBL		;RELATIVE ADDRESS OF TEST # 5
7236	001642	001274		TST6-TBL		;RELATIVE ADDRESS OF TEST # 6
7237	001644	001430		TST7-TBL		;RELATIVE ADDRESS OF TEST # 7

(NKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-5
(NKMAA.P11 27-DEC-82 13:15 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0023

7238	001646	001654	TST10-TBL	;RELATIVE ADDRESS OF TEST # 10
7239	001650	002204	TST11-TBL	;RELATIVE ADDRESS OF TEST # 11
7240	001652	002256	TST12-TBL	;RELATIVE ADDRESS OF TEST # 12
7241	001654	002530	TST13-TBL	;RELATIVE ADDRESS OF TEST # 13
7242	001656	003156	RELOC-TBL	;RELATIVE ADDRESS OF ROUTINE 'RELOC'

7243

7244

7245

7246

7247

;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

```

7252          ;*      SCOPE ROUTINE
7253          ;*      -----
7254          ;*
7255          ;*
7256          ;*      PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
7257          ;*      IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE.
7258          ;*      IF SR= 2000 (BIT10) THEN HALT
7259          ;*      IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SR BITS<3:0>
7260          ;*      ELSE CONTINUE TO NEXT TEST.
7261          ;*
7262
7263
7264 001660 105737 000420      TSTSCP: TSTB      @#$ENV      ;ARE WE RUNNING UNDER APT?
7265 001664 -001002          BNE      CNTSCP      ;IF SO THEN GO TO CNTSCP
7266 001666 004767 006002          JSR      PC,CHECKC ;TEST FOR CONTROL-C AND IF TYPED GO
7267                                ;PRINT ERROR HISTORY AND HALT AT FATHLT.
7268 001672 113702 000404      CNTSCP: MOVB      @#$TESTN,R2 ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
7269                                ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
7270                                ;OF R2 WILL BE 0
7271 001676 005237 000410          INC      @#$DEVCT ;TELL APT WE ARE STILL RUNNING OKAY
7272 001702 032777 002000 176540      BIT      #2000,@SWR ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
7273 001710 001401          BEQ      TSTGO      ;IF NOT THEN GO TO 2$
7274 001712 000000          SWHALT: HALT      ;HALT AT END OF TEST SWITCH SET.
7275
7276 001714 032777 040000 176526      TSTGO: BIT      #40000,@SWR ;IS THE PROGRAM GOING TO LOOP ON TEST
7277 001722 001332          BNE      LOOP      ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
7278 001724 105202          INCB     R2
7279 001726 000724          BR      CONT      ;GO TO CONT AND CONTINUE EXECUTING THE NEXT TEST
  
```

```

7292      ;*****
(3)      ;*TEST 0      TEST FOR PROPER BANK SELECTION
(4)      ;*(1)      THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
(4)      ;*      OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
(4)      ;*      LOCATION UNDER TEST
(4)      ;*(2)      IT CHECKS FOR PROPER BANK SELECTION BY WRITING
(4)      ;*      1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
(4)      ;*      LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
(4)      ;*      [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
(4)      ;*(3)      THIS TEST ALSO CHECKS TO SEE THAT NONE OF THE :JUN EXIST-
(4)      ;*      ING BANK RESPOND WHEN THEY ARE ADDRESSED
(3)      ;*****
(2) 001730 105737 000404      TST0: TSTB @#$TESTN      ;CHECK FOR PROPER TEST SEQUENCE
(2)
7293 001734 001403      BEQ      .+10
7294 001736 004767 004224      JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 001742 000005      5      ;*****ERROR NUMBER >*****
(1)
7295 001744 012703 177777      MOV      #177777,R3
7296 001750 010401      1$:      MOV      R4,R1      ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
7297 001752 010310      MOV      R3,(R0)      ;SET ALL THE BITS AT (R0)
7298 001754 020001      2$:      CMP      R0,R1      ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
7299 001756 001417      BEQ      4$      ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
7300 001760 005711      TST      (R1)      ;OTHERWISE CHECK (R1) FOR ALL 0'S
7301 001762 001430      BEQ      5$
7302 001764 020311      CMP      R3,(R1)      ;IF R1 IS NOT EQUAL TO R0 AND (R1)
7303      ;DOES NOT CONTAIN ALL 0'S THEN
7304      ;CHECK TO SEE IF (R0) = (R1)
7305 001766 001004      BNE      3$
7306 001770 012767 000006 000042      MOV      #6,12$      ;*ERROR* SETUP ERROR NO. IN 12$
(1)      ;*****ERROR NUMBER #6*****
7307 001776 000403      BR      10$
7308 002000      3$:      BR
(1) 002000 012767 000007 000032      MOV      #7,12$      ;*ERROR* SETUP ERROR NO. IN 12$
(1)      ;*****ERROR NUMBER #7*****
7309 002006 010046      10$:      MOV      R0,-(SP)      ;SAVE R0 ON STACK
7310 002010 105237 000301      INCB      @#$ADERR      ;AN ADDRESSING ERROR IS SUSPECTED
7311 002014 000407      BR      11$
7312 002016 020311      4$:      CMP      R3,(R1)      ;CHECK (R1) FOR ALL 1'S
7313 002020 001411      BEQ      5$
7314 002022 012767 000010 000010      MOV      #10,12$      ;*ERROR* SETUP ERROR NO. IN 12$
(1)      ;*****ERROR NUMBER #10*****
7315 002030 010046      MOV      R0,-(SP)      ;SAVE R0 ON STACK
7316 002032 010300      MOV      R3,R0
7317 002034 004767 003570      11$:      JSR      PC,ERROR      ;GO TO THE ERROR SUBROUTINE
7318 002040 000000      12$:      .WORD      ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
7319 002042 012600      MOV      (SP)+,R0      ;RESTORE R0
7320
7321 002044 013706 000350      5$:      MOV      @#SAVR6,SP      ;RESTORE THE STACK POINTER
7322 002050 062701 020000      ADD      #20000,R1      ;CAUSE R1 TO POINT TO THE SAME CHIP
7323      ;LOCATION IN THE NEXT 4K BANK OF MEMORY
7324      ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
7325 002054 020105      CMP      R1,R5      ;COMPARE R1 WITH THE HIGHEST MEMORY
7326      ;LOCATION WHICH IS STORED IN R5
7327 002056 103736      BLO      2$      ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$
7328

```

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-8
 CNKMAA.P11 27-DEC-82 13:15 TO TEST FOR PROPER BANK SELECTION

SEQ 0026

```

7329 002060 105737 000421      TSTB   @#SENVH      ;HAS APT INHIBITED SIZING?
7330 002064 100432              BMI     8$           ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
7331 002066 032777 000100 176354 BIT     #100,@SWR      ;HAS USER INHIBITED SIZING?
7332 002074 001026              BNE     8$           ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
7333
7334 002076 020127 157776      CMP     R1,#157776      ;SEE IF R1 HAS CROSSED 28K BOUNDRY OF VIRTUAL ADDRESS
7335 002102 101016              BHI     6$           ;IN WHICH CASE GO TO 6$
7336
7337 002104 020137 000340      CMP     R1,@#MAXMEM      ;SHOULD BE LEFT AS IS FOR 30K SYSTEMS (WHICH USE 16K CHI
7338
7339 002110 103755              BLO     5$           ;IS R1 LOWER THAN THE MAXIMUM AVAILABLE
7340 002112 012702 000006      MOV     #6,R2          ;MEMORY ?
7341 002116 012712 000300      MOV     #300,(R2)      ;IF SO THEN GO TO 5$
7342 002122 012742 177714      MOV     #5$-,-6,-(R2)  ;MAKE R2 POINT TO TRAP VECTOR+2 FOR NXM
7343 002126 060712              ADD     PC,(R2)        ;SET PSW TO 300;VER:1
7344 002130 011111              MOV     (R1),(R1)      ;SET UP RETURN ADDRESS FROM TRAP TO 5$
7345 002132 004767 004030      JSR     PC,FATERR      ;TRY TO WRITE TO NON-EXISTENT MEMORY (SHOULD TRAP)
7346 (1) 002136 000011              11          ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
7347
7347 002140 012702 000004      6$: MOV     #4,R2          ;*****ERROR NUMBER 11*****
7348 002144 012722 000006      MOV     #6,(R2)+      ;
7349 002150 005012              CLR     (R2)          ;RESTORE TRAP VECTOR
7350 002152 005010      8$: CLR     (R0)          ;
7351
7352 002154 062700 020000      ADD     #20000,R0      ;CAUSE R0 TO POINT TO THE SAME CHIP
7353
7354
7355 002160 020005              CMP     R0,R5          ;LOCATION IN THE NEXT 4K MEMORY BANK
7356
7357 002162 103672              BLO     1$           ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R0
7358 002164 000635      END0: BR      TSTSCP      ;COMPARE R0 WITH THE HIGHEST MEMORY
7359
7360

```

```

7366      ;*****
(3)      ;*TEST 1      CHECK DI/DO LINES
(4)      ;*(1)      THIS TEST CHECKS THE DATI/DATO LINES BY SHIFTING
(4)      ;*      A 1 IN THE WORD DIRECTION
(3)      ;*****
(2) 002166 122737 000001 000404 TST1: CMPB #1,0#TESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7367
7368
7369 002174 001403      BEQ      .+10
7370 002176 004767 003764 JSR      PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002202 000012      12      ;*****ERROR NUMBER 12*****
(1)
7371 002204 012700 000001 1$: MOV      #1,R0
7372 002210 010002      MOV      R0,R2 ;SET R2=1
7373 002212 010011      MOV      R0,(R1) ;MOV 1 AT LOCATION (R1)
7374 002214 020011      2$: CMP      R0,(R1) ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
7375 002216 001403      3$: BEQ      4$
7376 002220 004767 003404 JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 002224 000013      13      ;*****ERROR NUMBER 13*****
(1)
7377
7378 002226 005702      4$: TST      R2 ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
7379 002230 001406      BEQ      5$ ;IF SO THEN GO TO 5$
7380 002232 006300      ASL      R0 ;SHIFT THE 1 BROUGHT IN AT 1$ IN
7381 ;DATA DIRECTION
7382 002234 103366      BCC      2$ ;IF THE 1 HAS NOT BEEN SHIFTED THRU
7383 ;THE 16 DATA BITS THEN REPEAT FROM 2$
7384 002236 005002      CLR      R2 ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
7385 002240 012700 177776 MOV      #177776,R0
7386 002244 000762      BR       2$
7387
7388 002246 000261      5$: SEC ;SET C BIT
7389 002250 006100      ROL      R0 ;SHIFT A 0 16 TIMES IN DATA DIRECTION
7390 002252 103757      BCS      2$ ;IF THE 0 HAS NOT BEEN SHIFTED THRU
7391 ;THE 16 DATA BITS THEN REPEAT FROM 2$
7392 002254 062701 020000 ADD      #20000,R1 ;OTHERWISE GO TO THE NEXT BANK OF
7393 ;4K MEMORY AND REPEAT THE TEST
7394 002260 020105      CMP      R1,R5
7395 002262 103750      BLO      1$
7396 002264 000737      END1: BR      END0
7397

```



```

7406      ;*****
(3)      ;*TEST 2      TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
(4)      ;*(1)      THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
(4)      ;*      OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
(4)      ;*      OF BAKPAT AND READING IT
(4)      ;*(2)      MEMORY IS WRITTEN USING A BYTE AT A TIME
(4)      ;*(3)      STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
(3)      ;*****
(2) 002266 122737 000002 000404 TST2: CMPB #2,@$TESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7407
7408 002274 001403      BEQ      .+10
7409 002276 004767 003664 JSR      PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002302 000014      14      ;*****ERROR NUMBER 14*****
(1)
7410 002304 013700 000316 1$: MOV      @#BAKPAT,R0
7411 002310 110021      MOV      R0,(R1)+
7412 002312 113721 000317 MOV      @#BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
7413 002316 020105      CMP      R1,R5
7414 002320 103771      BLO      1$
7415
7416 002322 020041      2$: CMP      R0,-(R1) ;TEST THE MEMORY TO SEE IF IT CONTAINS
7417 ;THE WORD STORED IN BAKPAT
7418 002324 001416      BEQ      8$
7419 002326 062701 000002 ADD      #2,R1
7420 002332 123741 000317 CMPB      @#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
7421 002336 001402      BEQ      4$
7422 002340 120041      CMPB      R0,-(R1) ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
7423 002342 001002      BNE      6$
7424 002344 105237 000301 4$: INCB      @#$ADERR ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
7425 002350 042701 000001 6$: BIC      #1,R1 ;MAKE THE ADDRESS IN R1 EVEN
7426 002354 004767 003250 JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 002360 000015      15      ;*****ERROR NUMBER 15*****
(1)
7427 002362 020104      8$: CMP      R1,R4 ;KEEP ON TESTING THE MEMORY UNTIL
7428 002364 101356      BHI      2$ ;R1 EQUALS THE LOWEST ADDRESS
7429 002366 000337 000316 SWAB      @#BAKPAT ;CHANGE THE DATA PATTERN
7430 002372 001744      BEQ      1$ ;IF THE DATA PATTERN DOES NOT HAVE LOW
7431 ;BYTE =0 THEN FALL THRU
7432 002374 000733      END2: BR      END1
7433
7434 ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
7435

```

```

7447      ;*****
(3)      ;*TEST 3      DUAL ADDRESS TEST A
(4)
(4)      ;*(1) THIS TEST CHECKS FOR DUAL ADDRESSING PROBLEMS BY WRITING A
(4)      ;*      BACK GROUND OF BAKPAT.
(4)      ;*(2) STARTING FROM THE LOWEST LOCATION IN THE BANK THE TEST WRITES A
(4)      ;*      LOCATION WITH SWAPPED BAKPAT
(4)      ;*(3) READS THE MEMORY FOR PROPER CONTENTS
(4)      ;*(4) SHIFTS A 1 ALONG THE ADDRESS DIRECTION AND REPEATS STEPS 1-3
(4)      ;*(5) REPEATS STEP 1-4 FOR EACH 4K BANK
(3)      ;*****
(2) 002376 122737 000003 000404 TST3: CMPB #3,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7448 002404 001403      BEQ      .+10
7449 002406 004767 003554      JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002412 000016      16      ;*****ERROR NUMBER 16*****
(1)
7450 002414 005003      CLR      R3
7451 002416 004737 000120      2$: JSR      PC,@#WRTMEM      ; WRITE MEMORY WITH THE BACKGROUND STORED
7452      ;AT LOCATION BAKPAT
7453 002422 005002      4$: CLR      R2
7454 002424 050302      6$: BIS      R3,R2      ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
7455 002426 020204      CMP      R2,R4      ;IF R2 IS LESS THAN R4
7456 002430 103465      BLO      16$      ;THEN DO NOTHING
7457 002432 020205      CMP      R2,R5      ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
7458 002434 103077      BHIS     20$      ;TESTED THEN EXIT THE TEST
7459 002436 000312      SWAB     (R2)      ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
7460      ;THE LOCATION POINTED BY R2
7461 002440 005001      CLR      R1
7462 002442 050301      7$: BIS      R3,R1
7463 002444 020104      CMP      R1,R4      ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
7464 002446 103445      BLO      12$      ;THEN GO TO 12$
7465 002450 020105      CMP      R1,R5
7466 002452 103053      BHIS     15$
7467 002454 020102      CMP      R1,R2
7468 002456 001431      BEQ      10$
7469 002460 020011      CMP      R0,(R1)      ;CHECK THE MEMORY FOR CORRECT DATA
7470      ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
7471 002462 001437      BEQ      12$      ;THE SAME WORD AS BAKPAT
7472 002464 012767 000017 000032      MOV      #17,22$      ;IN WHICH CASE GO BACK TO 12$
(1)      ;*ERROR* SETUP ERROR NO. IN 22$
7473 002472 010046      8$: MOV      R0,-(SP)      ;*****ERROR NUMBER #17*****
7474 002474 000316      SWAB     (SP)      ;PLACE R0 ON THE STACK
7475 002476 022611      CMP      (SP)+,(R1)
7476      ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
7477 002500 001003      BNE      9$      ;AS A SWAPPED R0
7478      ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
7479      ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
7480      ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
7481 002502 012767 000020 000014      MOV      #20,22$      ;ENTIRE WORD
(1)      ;*ERROR* SETUP ERROR NO. IN 22$
7482 002510 105237 000301      9$: INCB     @#$ADERR      ;*****ERROR NUMBER #20*****
7483 002514 010046      MOV      R0,-(SP)      ;ADDRESSING PROBLEM IS DETECTED
7484 002516 010200      MOV      R2,R0      ;SAVE R0
7485 002520 004767 003104      JSR      PC,ERROR      ;SET R0=GOOD ADDRESS FOR ERROR REPORT
7486 002524 000000      22$: .WORD      ;GO TO THE ERROR SUBROUTINE
      ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
  
```

7487	002526	012600		MOV	(SP)+,R0	:RESTORE R0
7488	002530	010011		MCV	R0,(R1)	:RESTORE (R1)
7489	002532	020037	000*16	CMP	R0,@#BAKPAT	:IF THE CONTROL CAME HERE FROM 15\$-2 THEN
7490	002536	001411		BEQ	12\$	
7491	002540	000407		BR	11\$:RETURN TO 11\$
7492	002542	000300		SWAB	R0	:MAKE R0 SAME AS SWAPPED BAKPAT
7493	002544	020011		CMP	R0,(R1)	:IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
7494						:EQUAL TO SWAPPED R0
7495	002546	001404		BEQ	11\$:IN WHICH CASE GO BACK TO 11\$
7496	002550	012767	000021 177746	MOV	#21,22\$:*ERROR* SETUP ERROR NO. IN 22\$
(1)						:*****ERROR NUMBER #21*****
7497	002556	000745		BR	8\$:AND GO TO 8\$
7498	002560	000300		SWAB	R0	:RESTORE R0 TO BAKPAT
7499	002562	040301		BIC	R3,R1	:TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
7500	002564	005701		TST	R1	:IF R1 IS 0 THEN PLACE A 1 IN R1
7501	002566	001001		BNE	13\$:OTHERWISE GO TO 13\$
7502	002570	005201		INC	R1	
7503	002572	006101		ROL	R1	
7504	002574	020127	020000	CMP	R1,#20000	:IF R1 IS LESS THAN A 4K BOUNDARY
7505	002600	103720		BLO	7\$:THEN REPEAT FROM 7\$
7506	002602	000312		SWAB	(R2)	:RESTORE (R2) TO BAKPAT
7507	002604	040302		BIC	R3,R2	:TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
7508						:STORED IN R2
7509	002606	005702		TST	R2	:IF R2 = 0 THEN MOVE A 1 TO R2
7510	002610	001001		BNE	18\$:OTHERWISE GO TO 18\$
7511	002612	005202		INC	R2	
7512	002614	006102		ROL	R2	:SHIFT A ONE IN THE ADDRESS WORD
7513	002616	020227	020000	CMP	R2,#20000	:IS THE ADDRESS IN R2 MORE THAN THE BOUNDARY
7514						:OF 4K
7515	002622	103700		BLO	6\$:IF NOT THEN GO TO 6\$
7516	002624	060203		ADD	R2,R3	:OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
7517	002626	020337	000340	CMP	R3,@#MAXMEM	:IF R3 IS POINTING TO A BANK THAT IS LOWER
7518						:THAN MAXMEM
7519	002632	103673		BLO	4\$:THEN REPEAT FROM 4\$
7520	002634	000337	000316	SWAB	@#BAKPAT	
7521	002640	001656		BEQ	TST3	:REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
7522						:THE LOWER BYTE OF BAKPAT IS 0
7523	002642	000654		END3: BR	END2	

```

7530      ;*****
      (3)  ;*TEST 4      DUAL ADDRESS TEST B
      (4)      ;*(1)    THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
      (4)      ;*      AND READING THE ADDRESS IN THE LOCATION AND THEN
      (4)      ;*      WRITING AND READING ADDRESS COMPLEMENT
      (3)      ;*****
      (2) 002644 122737 000004 000404 TST4:  CMPB  #4,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
      (2)
7531 002652 001403      BEQ  .+10
7532 002654 004767 003306      JSR  PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
      (1) 002660 000022      22 ;*****ERROR NUMBER 22*****
      (1)
7533 002662 005003      CLR  R3
7534 002664 010100      1$:  MOV  R1,R0
7535 002666 005703      TST  R3 ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
7536 002670 001401      BEQ  2$ ;IN THE LOCATION
7537 002672 005100      COM  R0 ;OTHERWISE STORE COMPLEMENT
7538 002674 010021      2$:  MOV  R0,(R1)+ ;OF THE ADDRESS
7539 002676 020105      CMP  R1,R5 ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
7540 002700 103771      BLO  1$
7541
7542 002702 020041      3$:  CMP  R0,-(R1) ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
7543 002704 001405      BEQ  4$
7544 002706 105237 000301      INCB @#SADERR ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
7545 ;BIT PROBLEM
7546 002712 004767 002712      JSR  PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
      (1) 002716 000023      23 ;*****ERROR NUMBER 23*****
      (1)
7547 002720 010100      4$:  MOV  R1,R0
7548 002722 162700 000002      SUB  #2,R0 ;CHECK THAT THE ADDRESS IS STORED AT
7549 002726 005703      TST  R3 ;LOCATION IF R3 IS NOT 0
7550 002730 001401      BEQ  5$ ;OTHERWISE CHECK FOR
7551 002732 005100      COM  R0 ;ADDRESS COMPLEMENT
7552 002734 020104      5$:  CMP  R1,R4
7553 002736 101361      BHI  3$
7554 002740 112737 000001 000306      MOVB #1,@#PASFLG ;SET PASFLG FOR ERROR REPORT.
7555 002746 005103      COM  R3 ;COMPLEMENT THE CONTENTS OF R3
7556 002750 001345      BNE  1$ ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
7557 ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU
7558 002752 000733      END4: BR  END3
7559

```

```

7574      ;*****
(3)      ;*TEST 5      MARCHING 1'S AND 0'S
(4)      ;*(1)      THIS TEST WRITES A BACK GROUND OF THE WORD STORED
(4)      ;*      AT BAKPAT.
(4)      ;*(2)      READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
(4)      ;*      AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
(4)      ;*      DIRECTION OF MEMORY LOCATIONS.
(4)      ;*(3)      READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
(4)      ;*      WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
(4)      ;*      IN MIN. TO MAX. DIRECTION
(4)      ;*(4)      REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
(4)      ;*(5)      REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION
(4)
(3)      ;*****
(2) 002754 122737 000005 000404 TST5:  CMPB  #5,2#TESTN      ;CHECK FOR PROPER TEST SEQUENCE
(2)
7575
7576 002762 001403      BEQ      .+10
7577 002764 004767 003176 JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002770 000024      24      ;*****ERROR NUMBER 24*****
(1)
7578 002772 004737 000120 1$:      JSR      PC,2#WRTMEM      ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
7579      ;WORD STORED IN BAKPAT
7580 002776 020041      2$:      CMP      R0,-(R1)      ;READ THE CONTENTS OF LOCATION POINTED BY R1
7581 003000 001403      BEQ      3$      ;TO SEE IF IT HAS THE SAME VALUE AS R0
7582 003002 004767 002622 JSR      PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 003006 000025      25      ;*****ERROR NUMBER 25*****
(1)
7583 003010 000300      3$:      SWAB      R0
7584 003012 010011      MOV      R0,(R1)      ;SWAP THE BYTES AT (R1)
7585 003014 021100      CMP      (R1),R0      ;READ (R1) FOR CORRECT VALUE
7586 003016 001403      BEQ      4$
7587 003020 004767 002604 JSR      PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 003024 000026      26      ;*****ERROR NUMBER 26*****
(1)
7588
7589 003026 000300      4$:      SWAB      R0      ;SWAP THE BYTES OF THE REGISTER
7590      ;CONTAINING BACKGROUND PATTERN
7591 003030 001023      BNE      9$      ;IF THE LOWER BYTE OF THE REGISTER
7592      ;IS NOT 0 THEN THE PROGRAM IS READING
7593      ;THE MEMORY TO CONTAIN A BACK GROUND OF
7594      ;BAKPAT AND WRITING THE SWAPPED WORD
7595      ;IN WHICH CASE GO TO 9$
7596
7597
7598
7599 003032 005703      5$:      TST      R3      ;R3 WAS 0 WHEN THE PROGRAM ENTERED
7600      ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
7601      ;IF R3 EQUAL 0 THEN THE PROGRAM IS
7602      ;READING/WRITING MIN. TO MAX. OTHERWISE
7603      ;IT IS GOING IN MAX. TO MIN. DIRECTION
7604 003034 001023      BNE      10$      ;IF R3 IS NOT CLEAR THEN GO TO 10$
7605 003036 062701 000002 6$:      ADD      #2,R1      ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
7606 003042 020105      CMP      R1,R5      ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
7607      ;BE TESTED
7608 003044 103006      BHS      8$      ;IF R1>R5 THEN GO TO 8$ OTHERWISE
  
```

NKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-15
 NKMAA.P11 27-DEC-82 13:15 TS MARCHING 1'S AND 0'S

SEQ 0033

7609	003046	020011		7\$:	CMP	R0,(R1)	;READ (R1) FOR THE CORREC. DATA
7610	003050	001757			BEQ	3\$; WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
7611							;AND REPEAT UNTIL R1 > R5
7612	003052	004767	002552		JSR	PC,ERROR	;*ERROR* REPORT ERROR MESSAGE
(1)	003056	000027			27		;*****ERROR NUMBER 27*****
(1)							
7613	003060	000753			BR	3\$	
7614	003062	105237	000306	8\$:	INCB	2#PASFLG	
7615	003066	000300			SWAB	R0	
7616	003070	001742			BEQ	2\$;IF THE LOWER BYTE OF R0 IS ALL 0'S
7617							;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
7618							;AND READING BAKPAT GOING FROM MAX. TO MIN.[PASFLG=4]
7619	003072	005103			COM	R3	;OTHERWISE CLEAR R0
7620	003074	010401			MOV	R4,R1	;PUT THE LOWEST TESTING ADDRESS IN R1
7621	003076	000763			BR	7\$;AND BEGIN READING 0'S, WRITING 1'S AND
7622							;READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]
7623							
7624	003100	005703		9\$:	TST	R3	;IF R3 IS NON 0, I.E. PASFLG=3
7625	003102	001353			BNE	5\$;THEN READ BAKPAT, WRITE
7626							;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
7627							;IN MIN. TO MAX. DIRECTION
7628	003104	020104		10\$:	CMP	R1,R4	;OTHERWISE TEST IS PROCEEDING IN MAX. TO
7629							;MIN. DIRECTION.
7630	003106	101333			BHI	2\$;KEEP ON LOOPING UNTIL R1=R4
7631	003110	105237	000306		INCB	2#PASFLG	
7632	003114	000300			SWAB	R0	
7633	003116	001753			BEQ	7\$;IF R0 SWAPPED HAS LOWER BYTE=0
7634							;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
7635							;AND READ BAKPAT GOING FROM MIN. TO MAX.
7636	003120	000714		END5:	BR	END4	
7637							

```

7654 *****
(3) ;*TEST 6 CELLS' VOLATILITY TEST
(4)
(4) ;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
(4) ;*(2) WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
(4) ;* AND THEN INCREMENTS PASFLG
(4) ;*(3) IT THEN READS/SWAPS BYTES/WRITES A LOCATION X FOR
(4) ;* OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
(4) ;*(4) REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
(4) ;*(5) IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
(4) ;* BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
(4) ;* SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
(4) ;*(6) REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
(4) ;* BAKPAT INSTEAD OF BAKPAT.
(3) *****
(2) 003122 122737 000006 000404 TST6: CMPB #6,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7655
7656
7657 003130 001403 BEQ .+10
7658 003132 004767 003030 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003136 000030 30 ;*****ERROR NUMBER 30*****
(1)
7659 003140 004737 000120 RPT6: JSR PC,@#WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
7660 ;WORD STORED AT LOCATION BAKPAT
7661 003144 005037 000306 1$: CLR @#PASFLG ;SET R3
7662 003150 010403 2$: MOV R4,R3 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
7663 003152 010401 3$: MOV R4,R1 ;CHECK (R1) FOR CORRECT DATA
7664 003154 020011 3$: CMP R0,(R1)
7665 003156 001403 BEQ 4$ ;*ERROR* REPORT ERROR MESSAGE
7666 003160 004767 002444 JSR PC,ERROR ;*****ERROR NUMBER 31*****
(1) 003164 000031 31
(1)
7667 003166 062701 000002 4$: ADD #2,R1 ;INCREMENT R1 BY 2
7668 003172 020105 CMP R1,R5 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
7669 003174 103767 BLO 3$
7670 003176 132737 000001 000306 BITB #1,@#PASFLG ;CHECK TO SEE IF PASFLG=0 OR 2
7671 003204 001002 BNE 5$
7672 003206 105237 000306 INCB @#PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
7673
7674 003212 020305 5$: CMP R3,R5 ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
7675 003214 103012 BHIS 7$
7676 003216 012702 037776 6$: MOV #37776,R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
7677 003222 000313 SWAB (R3)
7678 003224 005302 DEC R2
7679 003226 001375 BNE 6$
7680 003230 010337 000354 MOV R3,@#SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
7681 003234 062703 020000 ADD #20000,R3 ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
7682 ;R3 TO POINT TO A LOCATION IN THE NEXT
7683 ;4K BANK OF MEMORY
7684 003240 000744 BR 2$
7685 003242 105237 000306 7$: INCB @#PASFLG ;MAKE PASFLG=2
7686 003246 000337 000316 SWAB @#BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
7687 003252 001732 BEQ RPT6 ;THEN GO BACK TO THE LOCATION RPT6
7688 003254 000721 END6: BR ENDS

```

```

7700      ;*****
(3)      ;*TEST 7      SHIFTING DIAGONAL
(4)
(4)      ;*(1) THIS TEST WRITES THE MEMORY WITH A BACKGROUND OF BAKPAT
(4)      ;*(2) IT WRITES A DIAGONAL OF SWAPPED BAKPAT THROUGH EACH MEMORY BANK
(4)      ;*(3) READS THE MEMORY FOR CORRECT DATA
(4)      ;*(4) SHIFTS THE DIAGONAL AND REPEATS STEP 3 UNTIL THE
(4)      ;*      DIAGONAL HAS BEEN SHIFTED 64 TIMES
(4)      ;*(5) WRITES A BACKGROUND OF SWAPPED BAKPAT, A DIAGONAL OF
(4)      ;*      BAKPAT AND REPEATS FROM STEP 3
(3)      ;*****
(2) 003256 122737 000007 000404 TST7: CMPB #7,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7701
7702 003264 001403      BEQ      .+10
7703 003266 004767 002674 JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003272 000032      32      ;*****ERROR NUMBER 32*****
(1)
7704 003274 005037 000306 2$: CLR      @#PASFLG
7705 003300 010337 000304      MOV      R3,@#LOWBNK      ;LOWBNK CONTAINS ADDRESS OF THE LOWEST LOCATION
7706      ;IN THE 4K BANK THAT CAN BE TESTED
7707 003304 010302      MOV      R3,R2
7708 003306 052702 017777 BIS      #17777,R2      ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK
7709 003312 005202      INC      R2      ;ADD 1 TO POINT IT TO NEXT BANK
7710 003314 001402      BEQ      3$      ;BRANCH IF ZERO (IT MUST BE A 30K SYSTEM)
7711 003316 020502      CMP      R5,R2
7712 003320 103001      BHS      4$      ;IF R2 IS GREATER THAN R5 THEN GO TO 4$
7713 003322 010502      3$: MOV      R5,R2      ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION
7714      ;THAT CAN BE TESTED
7715 003324 010337 000302 4$: MOV      R3,@#STRTDI      ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE
7716      ;DIAGONAL
7717 003330 013701 000304      MOV      @#LOWBNK,R1      ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K
7718      ;BANK
7719 003334 013700 000316 6$: MOV      @#BAKPAT,R0      ;STORE THE CONTENTS OF BAKPAT IN R0
7720 003340 020103      CMP      R1,R3      ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?
7721 003342 001010      BNE      10$      ;IF NOT THEN GO TO 10$
7722 003344 062703 000002 ADD      #2,R3      ;THE FOLLOWING CODE IS USED TO PLACE THE
7723 003350 032703 000176 BIT      #176,R3      ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL
7724 003354 001402      BEQ      8$      ;IN R3
7725 003356 062703 000200 ADD      #200,R3
7726 003362 000300 8$: SWAB      R0      ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN
7727 003364 132737 000001 000306 10$: BITB      #1,@#PASFLG      ;CONTENTS OF LOCATION PASFLG WILL BE EVEN IF THE
7728      ;MEMORY IS BEING WRITTEN AND IT WILL BE ODD
7729      ;IF IT IS ONLY BEING READ
7730      ;IF IT IS BEING READ ONLY THEN GO TO 12$
7731 003372 001001      BNE      12$
7732 003374 010011      MOV      R0,(R1)      ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS
7733      ;OF R0
7734 003376 020011 12$: CMP      R0,(R1)      ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN
7735      ;PROPER DATA
7736 003400 001403      BEQ      14$      ;IF IT IS OK THEN GO TO 14$
7737 003402 004767 002222 JSR      PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 003406 000033      33      ;*****ERROR NUMBER 33*****
(1)
7737 003410 062701 000002 14$: ADD      #2,R1      ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION
7738 003414 020102      CMP      R1,R2      ;IS IT THE END OF THE BANK ?
7739 003416 103746      BLO      6$      ;IF NOT THEN GO TO 6$
  
```


CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 62-18
 CNKMAA.P11 27-DEC-82 13:15 T7 SHIFTING DIAGONAL

SEQ 0036

7740	003420	005237	000410	16\$:	INC	@#SDEVCT
7741	003424	105237	000306		INCB	@#PASFLG
7742	003430	013703	000302		MOV	@#STRDI,R3
7743	003434	132737	000001 000306		BITB	#1,@#PASFLG
7744	003442	001330			BNE	4\$
7745	003444	005723			TST	(R3)+
7746	003446	020302			CMP	R3,R2
7747	003450	103003			BHIS	18\$
7748	003452	105737	000306		TSTB	@#PASFLG
7749	003456	100322			BPL	4\$
7750	003460	013703	000304	18\$:	MOV	@#LOWBNK,R3
7751						
7752	003464	000337	000316		SWAB	@#BAKPAT
7753	003470	001715			BEQ	4\$
7754						
7755	003472	010203			MOV	R2,R3
7756						
7757	003474	020205			CMP	R2,R5
7758						
7759	003476	103676			BLO	2\$
7760	003500	000645		END7:	BR	END6

;TELL APT WE ARE STIL RUNNING OKAY

;LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL
 ;HAS THE READ OF THE MEMORY BEEN DONE ?

;IF NOT THEN GO TO 4\$
 ;ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL
 ;AND UNLESS THE END OF THE BANK IS REACHED

;OR THE DIAGONAL HAS BEEN ROTATED 64 TIMES
 ;REPEAT FROM 4\$

;MAKE R3 POINT TO THE LOWEST LOCATION IN THE
 ;IN THE BANK UNDER TEST

;AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE
 ;SWAPPED BACK GROUND PATTERN THEN GO TO 4\$

;MAKE THE PRESENT HIGH BOUNDARY AS THE NEXT
 ;LOW BOUNDARY

;UNLESS THE PRESENT HIGH BOUNDARY IS ALSO THE
 ;HIGH BOUNDARY FOR THE MEMORY UNDER TEST

```

7788 *****
(3) *TEST 10 READ RECOVERY GALLOPING TEST/EVERY 64TH CELL
(4)
(4) *(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
(4) * STORED AT LOCATION BAKPAT
(4) *(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED
(4) * (LETS NAME IT 'A')
(4) *(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
(4) *(4) SWAPS BYTES FOR LOCATION 'A'.
(4) *(5) READS 'A', READS 'B'
(4) *(6) 'B' = 'B'+200 (MAKES 'B'=64TH CELL I.E. 200TH OCTAL
(4) * LOCATION FROM THE PRESENT LOCATION OF 'B')
(4) *(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
(4) * END OF THE 4K BANK OF THE MEMORY IN WHICH 'A' IS RESIDING
(4) *(8) A = A+2
(4) *(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
(4) *(10) GOES TO THE NEXT 4K BANK OF MEMORY AND REPEATS STEPS
(4) * 3-9 UNTIL THE END OF THE MEMORY
(4) *(11) AFTER EXECUTING THE TEST BYTES ARE SWAPPED AT
(4) * LOCATION BAKPAT AND STEPS 1-10 ARE REPEATED
(4) *(12) IN THIS TEST R0 IS POINTING TO LOCATION 'A', R1 TO
(4) * LOCATION 'B', R2 TO THE END OF THE 4K BANK IN WHICH THE
(4) * TEST IS TAKING PLACE AND R3 TO THE LOWEST LOCATION IN THE
(4) * COLUMN/ROW CONTAINING 'A' AND 'B'
(4) *(13) MOST OF THE CODE USED BY THIS TEST IS ALSO USED BY TEST 11
(3) *****
(2) 003502 122737 000010 000404 TST10: CMPB #10,@#STESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7789
7790 003510 001403 BEQ .+10
7791 003512 004767 002450 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003516 000034 34 ;*****ERROR NUMBER 34*****
(1)
7792 003520 010402 RPT10: MOV R4,R2 ;SET R2 TO THE LOWEST MEMORY UNDER TEST
7793 003522 052702 017776 BIS #17776,R2 ;MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K
7794 ;BANK FOR WHICH GALLOPING WILL BE PERFORMED
7795 003526 062702 000002 GALLOP: ADD #2,R2 ;INCREMENT R2 BY 2
7796 003532 001402 BEQ 1$ ;BR IF IT WENT TO 0 (IT MUST BE A 30K SYSTEM)
7797 003534 020205 CMP R2,R5 ;IF THE HIGH BOUNDRY OF THE TEST IS HIGHER THAN
7798 003536 101401 BLOS 2$ ;THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2
7799 003540 010502 1$: MOV R5,R2
7800 003542 005046 2$: CLR -(SP)
7801 003544 010200 MOV R2,R0
7802 003546 013740 000316 4$: MOV @#BAKPAT,-(R0) ;WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF
7803 ;BAKPAT
7804 003552 020003 CMP R0,R3
7805 003554 101374 BHI 4$
7806 003556 010301 6$: MOV R3,R1 ;R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT
7807 ;CAN BE TESTED IN THIS BLOCK
7808 003560 023710 000316 CMP @#BAKPAT,(R0) ;BEFORE STARTING THE GALLOPING TEST FOR LOCATION
7809 ;(R0) CHECK IT
7810 003564 001410 BEQ 8$ ;CONTINUE IF OK
7811 003566 010001 MOV R0,R1 ;OTHERWISE PREPARE TO REPORT THE ERROR
7812 003570 013700 000316 MOV @#BAKPAT,R0
7813 003574 004767 002030 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE

```

```

(1) 003600 000035 35 ;*****ERROR NUMBER 35*****
(1)
7814 003602 010011 MOV R0,(R1) ;RESTORE THE CONTENTS OF (R1)
7815 003604 010100 MOV R1,R0 ;RESTORE R0
7816
7817 003606 000310 8$: SWAB (R0)
7818 003610 031011 10$: BIT (R0),(R1) ;CHECK TO SEE THAT NONE OF THE BITS SET
7819 ;IN (R0) ARE SET IN (R1) AND VICE VERSA
7820 003612 020001 CMP R0,R1 ;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1
7821
7822 003614 001412 BEQ 12$
7823 003616 021137 000316 CMP (R1),@#BAKPAT ;CHECK THAT (R1) HAS BAKPAT IN IT
7824 003622 001407 BEQ 12$
7825 003624 010046 MOV R0,-(SP) ;SAVE R0 ON STACK
7826 003626 013700 000316 MOV @#BAKPAT,R0 ;PLACE THE PATTERN WORD IN R0
7827 003632 004767 001772 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003636 000036 36 ;*****ERROR NUMBER 36*****
(1)
7828 003640 012600 MOV (SP)+,R0 ;RESTORE R0
7829 003642 021037 000320 12$: CMP (R0),@#SWAPAT ;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT
7830 003646 001412 BEQ 14$
7831 003650 010146 MOV R1,-(SP) ;SAVE R1 ON THE STACK
7832 003652 010001 MOV R0,R1 ;MAKE R1 POINT TO THE FAILING LOCATION
7833 003654 013700 000320 MOV @#SWAPAT,R0 ;LOAD R0 WITH THE EXPECTED RESULT IN (R1)
7834 003660 004767 001744 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003664 000037 37 ;*****ERROR NUMBER 37*****
(1)
7835 003666 010011 MOV R0,(R1) ;RECOVER (R1) FROM THE ERROR
7836 003670 010100 MOV R1,R0 ;RESTORE R0
7837 003672 012601 MOV (SP)+,R1 ;AND RESTORE R1
7838 003674 122737 000011 000404 14$: CMPB #11,@#$TESTN ;IS THE PROGRAM EXECUTING TEST # 11 ?
7839 003702 001402 BEQ 16$ ;IF SO THEN GO TO 16$
7840 003704 062701 000176 ADD #176,R1
7841 003710 062701 000002 16$: ADD #2,R1 ;MAKE R1 POINT TO THE NEXT ADJACENT CELL
7842 003714 020102 CMP R1,R2 ;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDRY
7843 003716 103734 BLO 10$ ;THEN REPEAT FROM 10$
7844 003720 000320 SWAB (R0)+ ;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST
7845 ;WAS BEING PERFORMED
7846 003722 122737 000011 000404 CMPB #11,@#$TESTN ;IS IT TEST 11 ?
7847 003730 001407 BEQ 17$ ;IF SO THEN GO TO 17$
7848 003732 005723 TST (R3)+ ;OTHERWISE INCREMENT R3 BY 2
7849 003734 062716 000002 ADD #2,(SP) ;FOR EVERY ROW/COLUMN TESTED ADD 2
7850 003740 105716 TSTB (SP)
7851 003742 100002 BPL 17$ ;UNTIL (SP) IS 200
7852 003744 161603 SUB (SP),R3 ;SUBTRACT 200 FROM R3
7853 003746 005016 CLR (SP)
7854 003750 032700 000177 17$: BIT #177,R0 ;AT A 64TH CALL BOUNDARY?
7855 003754 001002 BNE 18$ ;BRANCH IF NO
7856 003756 005237 000410 INC @#$DEVCT ;TELL APT WE ARE STILL RUNNING
7857 003762 020002 18$: CMP R0,R2 ;IF R0 HAS NOT REACHED THE END OF THE BOUNDRY
7858 003764 103674 BLO 6$ ;THEN REPEAT FROM 6$
7859 003766 162603 SUB (SP)+,R3 ;RESTORE SP AND R3
7860 003770 000337 000320 SWAB @#SWAPAT
7861 003774 000337 000316 SWAB @#BAKPAT
7862 004000 001660 BEQ 2$ ;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 2$
7863 004002 010203 MOV R2,R3 ;OTHERWISE MAKE THE PRESENT HIGH BOUNDARY AS THE

```

```
7864                                ;NEXT LOW BOUNDRY  
7865    004004    020205                CMP      R2,R5  
7866    004006    001410                BEQ      END10  
7867                                           ;IF PREVIOUS HIGH BOUNDRY WAS THE END OF THE  
7868    004010    032702    017776        BIT      #17776,R2          ;TEST BOUNDRY THEN EXIT THE TEST  
7869    004014    001025                    BNE      RPT11             ;WAS IT A 4K BOUNDRY ?  
7870                                           ;IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG  
7871    004016    122737    000011    000404   CMPEB     #11,#$TESTN       ;GALLOPING TEST DISABLED  
7872    004024    001421                    BEQ      RPT11             ;IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11  
7873    004026    000635                    BR       RPT10  
7874    004030    000623                   END10:  BR       END7         ;OTHERWISE REPEAT TEST 10  
7875  
7876  
7877
```

```

7906      ::*****
(3)      :*TEST 11      READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
(4)
(4)      :*(1)      THIS TEST WRITES MEMORY WITH BAKPAT
(4)      :*(2)      THE TEST BEGINS AT THE LOWEST LOCATION BEING TESTED
(4)      :*      (LETS NAME IT 'B')
(4)      :*(3)      'A' 'B' [MOVE THE ADDRESS OF 'B' TO THE POINTER FOR LOCATION 'A']
(4)      :*(4)      SWAPS BYTES FOR LOCATION 'A'
(4)      :*(5)      READS 'A', READS 'B'
(4)      :*(6)      'B'='B'+2
(4)      :*(7)      IF GALLOPING OPTION BIT AT $SWREG IS HIGH THEN STEPS 4 AND 5
(4)      :*      ARE REPEATED UNTIL 'B' REACHES THE HIGHEST MEMORY LOCATION
(4)      :*      OF THE 4K BANK IN WHICH 'A' IS RESIDING, THEN 'A' IS
(4)      :*      DECREMENTED BY 2 AND AFTER MAKING 'B' TO POINT TO THE LOWEST
(4)      :*      LOCATION OF THE 4K MEMORY BANK CONTAINING 'A' STEPS 3,4,5 AND
(4)      :*      6 ARE REPEATED UNTIL 'A' EQUALS THE END OF THE ENTIRE MEMORY
(4)      :*(8)      IF GALLOPING OPTION BIT IS NOT HIGH THEN STEPS 4 AND 5 ARE
(4)      :*      REPEATED UNTIL 'B' IS POINTING TO A CELL IN THE NEXT COLUMN
(4)      :*      IF SEQUENTIAL CELLS LIE ALONG THE ROW, OR THE NEXT ROW
(4)      :*      IF SEQUENTIAL CELLS LIE ALONG THE COLUMN, AT WHICH TIME
(4)      :*      STEPS 2,3,4,5 AND 7 ARE REPEATED UNTIL THE END OF THE MEMORY
(4)      :*(9)      TEST IS REPEATED FOR THE OPPOSITE BACKGROUND DATA
(4)      :*(10)     IN THIS TEST R0 POINTS TO LOCATION 'A', R1 TO LOCATION
(4)      :*      'B', R2 TO THE HIGHEST LOCATION AND R3 TO THE LOWEST
(4)      :*      LOCATION IN A 64/4K CELL BOUNDRY
(4)      :*(11)     MOST OF THE CODE USED BY TEST 10 IS ALSO USED BY THIS TEST
(3)
(2) 004032 122737 000011 000404 :*****
(2)      :*T11: CMPB    #11,#$TESTN      ;CHECK FOR PROPER TEST SEQUENCE
7907
7908      004040 001403      BEQ      .+10
7909      004042 004767 002120 JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004046 000040      40      ;*****ERROR NUMBER 40*****
(1)
7910      004050 010402      MOV      R4,R2      ;MAKE R2 TO POINT TO THE LOWEST LOCATION
7911      ;UNDER TEST
7912      004052 105777 174372 TSTB     @SWR      ;LONG GALLOP ENABLED?
7913      004056 100004      BPL      RPT11     ;BRANCH IF NO
7914      004060 004767 002526 JSR      PC,PNTMES ;TYPE 'GLP'
7915      004064 046107 000120 .ASCIZ    /GLP/
7916      004070 105777 174354 RPT11: TSTB     @SWR      ;LONG GALLOPING ENABLED?
7917      004074 100612      BMI      RPT10     ;BRANCH IF YES
7918      ;TO RPT10
7919      004076 052702 000176 BIS      #176,R2    ;OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
7920      ;TO GET THE HIGH BOUNDRY
7921
7922      004102 000611      BR       GALLOP      ; PERFORM GALLOPING TEST

```

```

7945      ;*****
(3)      ;*TEST 12  WORST CASE TESTING FOR CORE MEMORY
(4)      ;*(1)  STARTING FROM THE LOWEST LOCATION UNDER TEST THE MEMORY
(4)      ;*      IS WRITTEN WITH A BACKGROUND OF BAKPAT, HOWEVER LOCATIONS
(4)      ;*      HAVING ADDRESS SUCH THAT EXCLUSIVE OR OF ADDRESS BITS 1 &
(4)      ;*      8 = 1 ARE WRITTEN TO A VALUE OF SWAPPED BAKPAT
(4)      ;*(2)  STARTING FROM THE LOWEST LOCATION THE MEMORY IS CHECKED
(4)      ;*      TO CONTAIN THE CORRECT DATA AS EXPLAINED IN STEPS 3 & 4,
(4)      ;*      UNTILL THE HIGHEST LOCATION UNDER TEST IS REACHED
(4)      ;*(3)  READ EACH LOCATION FOR THE CORRECT CONTENT
(4)      ;*(4)  COMPLEMENT THE LOCATION AND READ IT; COMPLEMENT THE LOCATION
(4)      ;*      BACK TO ITS ORIGINAL VALUE AND READ IT AGAIN
(4)      ;*(5)  STARTING FROM THE HIGHEST LOCATION UNDER TEST REPEAT STEPS
(4)      ;*      3 & 4 UNTIL THE LOWEST LOCATION UNDER TEST IS REACHED
(4)      ;*(6)  REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
(4)      ;*      OF ADDRESS BITS 8 & 13 =1 ARE WRITTEN TO SWAPPED BAKPAT
(4)      ;*(7)  REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
(4)      ;*      OF ADDRESS BITS 3 & 9 =1 ARE WRITTEN TO SWAPPED BAKPAT
(4)      ;*(8)  REPEAT STEPS 1-7 WITH A BACKGROUND OF SWAPPED BAKPAT AND
(4)      ;*      THE LOCATIONS TO BE WRITTEN TO SWAPPED BAKPAT WRITTEN TO
(4)      ;*      BAKPAT.
(3)      ;*****
(2) 004104 122737 000012 000404 TST12: CMPB #12,@#TESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
7946 004112 001403 BEQ .+10
7947 004114 004767 002046 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004120 000041 41 ;*****ERROR NUMBER 41*****
(1)
7948
7949 004122 012702 000002 MOV #2,R2 ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1
7950 004126 012703 000400 MOV #400,R3 ;AND 8
7951 004132 112737 000001 000306 1$: MOVB #1,@#PASFLG ;INITIALIZE THE COUNTER FOR THE SUBTEST
7952 004140 010401 2$: MOV R4,R1 ;PLACE THE STARTING ADDRESS OF MEMORY UNDER
7953 ;TEST IN R1
7954 004142 013700 000316 4$: MOV @#BAKPAT,R0
7955 004146 030201 BIT R2,R1 ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET
7956 004150 001004 BNE 8$ ;IF IT IS SET THEN GO TO 8$
7957 004152 030301 BIT R3,R1 ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET
7958 004154 001404 BEQ 12$ ;IF IT IS NOT SET THEN GO TO 12$
7959 004156 005100 6$: COM R0 ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS
7960 ;POINTED BY R2 & POINTED BY R3 = 1 IN WHICH
7961 ;CASE PREPARE TO WRITE THE LOCATION
7962 ;WITH A COMPLEMENT OF LOCATIONS NOT MEETING
7963 ;THIS CONDITION
7964 004160 000402 BR 12$
  
```

D 4

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 63
CNKMAA.P11 27-DEC-82 13:15 T12

WORST CASE TESTING FOR CORE MEMORY

SEQ 0042

7966	004162	030301	8\$:	BIT	R3,R1	;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND
7967			:			
7968			:			
7969			:			
7970	004164	001774		BEO	6\$;CHECK ADDRESS BIT POINTED BY R3
7971						;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 6\$
7972			:			
7973			:			
7974			:			
7975	004166	132737	000002	000306	12\$:	BITB #2,2#PASFLG ;IS IT 2ND OR 3RD PASS OF THE SUBTEST?
7976	004174	001001			BNE 14\$;IF SO THEN READ THE MEMORY
7977			:			
7978			:			

```

7980
7981 004176 010011          MOV    R0,(R1)      ;OTHERWISE WRITE THE MEMORY BFORE READING IT
7982 004200 020011          14$:  CMP    R0,(R1)      ;READ THE MEMORY FOR CORRECT CONTENTS
7983 004202 001403          BEQ    16$
7984 004204 004767 001420  JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 004210 000042          42      ;*****ERROR NUMBER 42*****
(1)
7985 004212 012746 000002  16$:  MOV    #2,-(SP)
7986 004216 005100          18$:  COM    R0
7987 004220 005111          COM    (R1)
7988 004222 020011          CMP    R0,(R1)      ;READ THE MEMORY AGAIN
7989 004224 001404          BEQ    19$
7990 004226 004767 001376  JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 004232 000043          43      ;*****ERROR NUMBER 43*****
(1)
7991 004234 010011          MOV    R0,(R1)      ;RESTORE THE LOCATION (R1)
7992 004236 005316          19$:  DEC    (SP)
7993 004240 001366          BNE    18$
7994 004242 005726          TST    (SP)+
7995 004244 122737 000003 000306  CMPB  #3,@#PASFLG      ;EXECUTE THE CODE FROM 18$ TWICE
7996 004252 001412          BEQ    20$          ;RESTORE THE STACK POINTER
7997 004254 062701 000002          ADD    #2,R1      ;IS IT THE 3RD PASS OF THE SUBTEST ?
7998                                ;IF SO THEN GO TO 20$
7999 004260 020105          CMP    R1,R5          ;IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN
8000 004262 103727          BLO    4$            ;MIN. TO MAX. DIRECTION
8001 004264 105237 000306  INCB  @#PASFLG      ;HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?
8002 004270 122737 000002 000306  CMPB  #2,@#PASFLG      ;IF NOT THEN REPEAT FROM 4$
8003 004276 001720          BEQ    2$
8004 004300 162701 000002  20$:  SUB    #2,R1
8005                                ;IF IT IS THE 2ND PASS OF THE SUBTEST
8006 004304 020104          CMP    R1,R4          ;THEN REPEAT FROM 2$
8007 004306 103315          BHIS  4$            ;OTHERWISE EXECUTE THE TEST IN MAX. TO MIN.
8008 004310 012702 020000          MOV    #20000,R2      ;DIRECTION
8009                                ;HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?
8010 004314 105237 000307          INCB  @#PASFLG+1
8011 004320 123727 000307 000002  CMPB  @#PASFLG+1,#2      ;IF NOT THEN REPEAT FROM 4$
8012 004326 103701          BLO    1$            ;PREPARE TO CHECK THE MEMORY WITH THE XOR OF
8013 004330 101004          BHI    22$          ;ADDRESS BITS 8 AND 13
8014 004332 012702 000010          MOV    #10,R2      ;THE SUB TEST HAS CHECKED THE XOR ONE KIND
8015 004336 006303          ASL    R3            ;HAS TWO XOR COMBINATIONS BEEN CHECKED ?
8016 004340 000674          BR     1$
8017 004342 005137 000316  22$:  COM    @#BAKPAT      ;IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED
8018 004346 105737 000316          TSTB  @#BAKPAT
8019 004352 001654          BEQ    TST12
8020 004354 000625          END12: BR     END10      ;BAKPAT THEN RE-EXECUTE THE TEST
  
```



```

8051 *****
(3) TEST 13 WRITE RECOVERY TEST
(4) THIS TEST DIFFERS FROM 0-12 IN THAT IT CONSISTS OF A SMALL TEST PROGRAM
(4) ACTUALLY RUNNING IN THE 4K BANK UNDER TEST.
(4) THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
(4) TO AID IN THE DEBUG, BEFORE A BANK IS ENTERED 'TST13 BNK XX'
(4) IS TYPED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
(4) BANK FAILED.
(4) THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH 'MOV R2,-(PC)'
(4) AND THE OTHER 1/2 CONTAINING '177667'. '177667' IS THE COMPLEMENT
(4) OF 'JMP (R0)' INSTRUCTION.
(4) R2 CONTAINS 'COM -(R1)' INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS
(4) THE HIGHEST TEST ADDRESS IN THAT BANK. THE HIGHEST TEST ADDRESS IS
(4) USUALLY ON 4K BOUNDARIES. WHEN TESTING BANK 0 RELOCATED, HOWEVER
(4) R1 CONTAINS THE FIRST FREE TEST ADDRESS BELOW THE DIAGNOSTIC.
(4) IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
(4) THE TEST EXECUTION IS AS FOLLOWS:
(4) 1. THE 'MOV R2,-(PC)' INSTRUCTION EXECUTES STORING
(4) THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC)).
(4) 2. SINCE R2 CONTAINS A 'COM -(R1)' INSTRUCTION IT COMPLEMENTS
(4) THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
(4) '177667' SO AFTER THE COM -(R1) IT EQUALS 110
(4) CLEVERLY THIS IS THE 'JMP (R0)' INSTRUCTION.
(4) 3. THIS SEQUENCE CONTINUES UNTIL THE 'MOV R2,-(PC)' INSTRUCTIONS
(4) REACH THE MIDDLE OF THE TEST BANK. THEN THE 'JMP (R0)' INSTRUCTION IS
(4) AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
(4) TO TEST 13.
(4) 4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.
(3) *****
(2) 004356 122737 000013 000404 TST13: CMPB #13,#$TESTN ;CHECK FOR PROPER TEST SEQUENCE
(2)
8052 004364 001403 BEQ .+10
8053 004366 004767 001574 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004372 000044 44 ;*****ERROR NUMBER 44*****
(1)
8054 004374 012702 010247 1$: MOV #10247,R2 ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
8055 ;IN R2.
8056 004400 012700 177667 MOV #177667,R0 ;PLACE THE COMPLEMENT OF THE INSTRUCTION
8057 ;JMP (R0) IN R0
8058 ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
8059 ;SINCE THE TEST STORES 'MOV R2,-(PC)' IN 1/2 AND 177667 IN THE OTHER 1/2
8060
8061 004404 010546 2$: MOV R5,-(SP) ;SAVE R5
8062 004406 010446 MOV R4,-(SP) ;STORE LOWEST ADDRESS ON STACK
8063 004410 000241 29$: CLC
8064 004412 006005 ROR R5 ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
8065 004414 006004 ROR R4 ;DO SAME FOR LOWEST ADDRESS
8066 004416 160405 SUB R4,R5 ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
8067 004420 006005 ROR R5 ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
8068 004422 103002 BCC 30$ ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS.
8069 004424 062716 000002 ADD #2,(SP) ;INCREASE LOWEST TEST ADDRESS BY 2
8070 004430 012604 30$: MOV (SP)+,R4 ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
8071 004432 012605 MOV (SP)+,R5 ;RESTORE HIGHEST TEST ADDRESS
8072 004434 010403 MOV R4,R3 ;PLACE THE LOWEST LOCATION UNDER TEST
8073 ;IN R3

```

8074	004436	000406			BR	28\$:LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0
8075	004440	042703	017776	3\$:	BIC	#17776,R3	:CAUSE R3 TO POINT TO THE LOWEST LOCATION
8076							:IN THE 4K BANK UNDER TEST
8077	004444	001507			BEQ	14\$:IF ADDRESS WENT TO 0, IT MUST BE A 30K SYSTEM
8078	004446	105737	000405		TSTB	@#REL	:ARE WE RELOCATED?
8079	004452	100504			BMI	14\$:BRANCH IF YES-TEST BANK0 ONLY-
8080	004454	020305		28\$:	CMP	R3,R5	:IF R3 IS HIGHER THAN THE HIGHEST LOCATION
8081	004456	103102			BHIS	14\$:UNDER TEST THEN EXIT
8082				:	IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0		
8083	004460	020527	020000		CMP	R5,#20000	:IS HIGHEST TEST ADDRESS BELOW 4K?
8084	004464	103002			BHIS	31\$:BRANCH IF NO
8085	004466	010501			MOV	R5,R1	:SET R1 TO HIGHEST TEST ADDRESS IN BANK0
8086	004470	000407			BR	32\$	
8087							
8088	004472	010301		31\$:	MOV	R3,R1	:SET R1 TO LOWEST CURRENT TEST ADDRESS
8089	004474	052701	017777		BIS	#17777,R1	:SET LOW ORDER ADDRESS BITS
8090	004500	005201			INC	R1	:CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
8091							:OF THE 4K BANK BEING POINTED BY R3
8092	004502	001402			BEQ	32\$:BRANCH IF R1 WENT TO 0 (WHICH MIGHT
8093							:HAVE HAPPENED IF TESTING A 30K LSI SYSTEM)
8094	004504	020105			CMP	R1,R5	:COMPARE R1 TO HIGHEST ADDRESS UNDER TEST
8095	004506	101401			BLOS	33\$:BRANCH IF WITHIN RANGE
8096	004510	010501		32\$:	MOV	R5,R1	:SET R1 TO THE MAXIMUM AVAILABLE MEMORY
8097							
8098	004512	132737	000001 000306	33\$:	BITB	#1,@#PASFLG	:IS THE LOWEST BIT OF LOCATION PASFLG
8099	004520	001101			BNE	16\$:SET? IN WHICH CASE BACK GROUND HAS
8100							:ALREADY BEEN WRITTEN AND WRITE RECOVERY
8101							:TEST IS BEING PERFORMED
8102							
8103	004522	020304		4\$:	CMP	R3,R4	:OTHERWISE WRITE THE BACKGROUND
8104	004524	103430			BLO	8\$:DEFINED AT STEP 3.
8105	004526	105737	000307		TSTB	@#PASFLG+1	:IS THE TEST JUST DOING READ, I.E.
8106	004532	001002			BNE	6\$:IS THE PASFLG+1 LOCATION NON ZERO? IF SO
8107							:THEN GO TO 6\$
8108	004534	012713	010247		MOV	#10247,(R3)	:WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,-(PC)
8109	004540	020213		6\$:	CMP	R2,(R3)	:READ (R3) TO CONTAIN CORRECT DATA
8110	004542	001421			BEQ	8\$	
8111	004544	010046			MOV	R0,-(SP)	:SAVE R0
8112	004546	010146			MOV	R1,-(SP)	:AND R1 ON THE STACK
8113	004550	010301			MOV	R3,R1	
8114	004552	010200			MOV	R2,R0	:SET R0= GOOD DATA FOR ERROR PRINTOUT
8115	004554	004767	001050		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	004560	000045			45		:*****ERROR NUMBER 45*****
(1)							
8116	004562	012601			MOV	(SP)+,R1	:RESTORE R1
8117	004564	012600			MOV	(SP)+,R0	:AND R0
8118	004566	105737	000306		TSTB	@#PASFLG	:IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
8119							:THE PROPER DATA THEN WE DON'T WANT TO GO AND
8120							:EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
8121							:TEST
8122	004572	001005			BNE	8\$:BRANCH IF PASFLG NOT =0
8123							
8124	004574	010200			MOV	R2,R0 ;SAVE FOR ERROR REPORT	
8125	004576	004767	001026		JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	004602	000046			46		:*****ERROR NUMBER 46*****
(1)							

8126	004604	000663			BR	END12	:ABORT TST 13.
8127							
8128	004606	062703	000002		8\$:	ADD #2,R3	:INCREMENT R3 BY 2
8129	004612	162701	000002			SUB #2,R1	:DECREMENT R1 BY 2
8130	004616	020105				CMP R1,R5	:WRITE THE BACKGROUND DEFINED AT STEP 4.
8131	004620	103014				BHIS 12\$	
8132	004622	020103				CMP R1,R3	:HAS STORING THE 177667 REACHED WHERE 'MOV R2,-(PC) IS'
8133	004624	103405				BLO 10\$:BRANCH IF YES DON'T DESTROY THE MOV R2,-(PC) IS.
8134	004626	105737	000307			TSTB @#PASFLG+1	:IS THE READ ONLY CHECK PASS?
8135	004632	001002				BNE 10\$:BRANCH IF YES
8136	004634	012711	177667			MOV #177667,(R1)	:WRITE THE LOCATION WITH THE COMPLEMENT OF THE
8137							:OP CODE JMP (R0)
8138	004640	020011			10\$:	CMP R0,(R1)	:READ R1 TO CONTAIN CORRECT DATA
8139	004642	001403				BEQ 12\$	
8143	004644	004767	000760			JSR PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	004650	000047				47	:*****ERROR NUMBER 47*****
(1)							
8144	004652	020301			12\$:	CMP R3,R1	:IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
8145	004654	103722				BLO 4\$:THEN REPEAT FROM 4\$
8146							
8147							:RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
8148							
8149	004656	062703	020000		13\$:	ADD #20000,R3	:OTHERWISE GO TO THE NEXT 4K BANK
8150	004662	000666				BR 3\$	
8151							
8152	004664	122737	000001	000306	14\$:	CMPB #1,@#PASFLG	:THE PROGRAM CONTROL COMES HERE AS FOLLOWS
8153							:1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
8154							:WRITE/READ CYCLE FOR THE BACK GROUND
8155							:AND WANTS TO BEGIN THE WRITE RECOVERY TEST
8156	004672	001437				BEQ 24\$:2-PASFLG=1, PROGRAM HAS JUST COMPLETED
8157							:THE WRITE RECOVERY TEST AND WANTS TO
8158							:READ MEMORY FOR CORRECT DATA
8159	004674	103627				BLO END12	:3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
8160							:MEMORY AND WANTS TO GO THE NEXT TEST.
8161							
8162	004676	105137	000307			COMB @#PASFLG+1	:ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
8163							:ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
8164							:ENTRY DISABLE READ ONLY
8165	004702	001240				BNE 2\$	
8166	004704	012702	005141			MOV #5141,R2	:PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
8167							:IN R2
8168	004710	012700	177740			MOV #13\$-,-6,R0	:PLACE THE RETURN ADDRESS IN R0 AS 13\$
8169	004714	060700				ADD PC,R0	:THUS WHEN THE READ RECOVERY TEST REACHES
8170							:THE MIDDLE OF THE 4K MEMORY THEN THE
8171							:INSTRUCTION EXECUTED WILL BE JMP (R0)
8172							:BRANCHING THE PROGRAM TO 13\$
8173	004716	105237	000306		15\$:	INCB @#PASFLG	:INCREMENT PASFLG BY 1.
8174	004722	000630				BR 2\$	
8175							
8176	004724	032777	000020	173516	16\$:	BIT #20,@SWR	:HAS THE PRINTOUTS BEEN SUPRESSED ?
8177	004732	001016				BNE 18\$:IF SO THEN GO TO 18\$
8178	004734	105737	000042			TSTB @#42	:IS THE PROGRAM RUNNING UNDER ACT?
8179	004740	001013				BNE 18\$:BRANCH IF YES
8180	004742	004767	001644			JSR PC,PNTMES	:TYPE THE BANK UNDER TEST
8181	004746	051524	030524	020063		.ASCIZ /TST13 BNK/	
	004754	047102	000113				

```
8182  
8183 004760 004767 002454 .EVEN  
8184 004764 004767 001650 JSR PC,GETBNK ;GET BANK NO. UNDER TEST INTO DECWRD FOR PRINT.  
8185 JSR PC,STPDEC ;TYPE BANK NO. UNDER TEST  
8186 004770 000113 18$: JMP (R3) ;BEGIN EXECUTING MOV R2,-(PC) ,COM -(R1) SEQUENCE IN TES  
8187  
8188  
8189 004772 105137 000307 24$: COMB @#PASFLG+1  
8190 004776 012700 000110 MOV #110,R0 ;PLACE THE OP CODE FOR JMP (R0) IN R0  
8191 005002 000745 BR 15$ ; READ THE MEMORY FOR CORRECT DATA AFTER  
8192 ;INCREMMENTING PASFLG TO 2  
8193  
8194 ;TST13 EXITS VIA END12.  
8195
```

```

8200 005004 012737 000377 000316 RELOC: MOV #377,@#BAKPAT
8201 005012 105737 000276 TSTB @#MMAVA ;IS THE MEMORY MANAGEMENT BEING TESTED ?
8202 005016 001065 BNE CONTMM ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
8203 ;MEMORY MANAGEMENT
8204 005020 032777 001000 173422 BIT #1000,@SWR ;RELOCATION WANTED?
8205 005026 001046 BNE CKDONE ;BRANCH IF NO
8206 005030 105737 000405 TSTB @#REL ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
8207 005034 100420 BMI RELOER ; PLACE THE PROGRAM BACK IN LOWER CORE
8208 005036 112737 000200 000405 MOVB #200,@#REL ;OTHERWISE PREPARE TO RELOCATE
8209
8210 ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
8211
8212
8213 005044 004767 001542 JSR PC,PNTMES ;TYPE 'REL'
8214 005050 042522 047514 000103 .ASCIZ /RELOC/
8215 .EVEN
8216 005056 013705 000340 MOV @#MAXMEM,R5 ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
8217 ;AVAILABLE MEMORY
8218 005062 014445 2$: MOV -(R4),-(R5) ;RELOCATE THE PROGRAM
8219 005064 020427 000430 CMP R4,#BEGIN-50 ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN-50
8220 005070 101374 BHI 2$
8221 005072 000165 000050 JMP 50(R5)
8222
8223 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
8224
8225
8226 005076 013705 000346 RELOER: MOV @#SAVR5,R5 ;RESTORE R5
8227 005102 105737 000405 TSTB @#REL ;IS DIAGNOSTIC IN RELOCATED STATE?
8228 005106 100016 BPL CKDONE ;BRANCH IF NO
8229
8230 005110 012704 000430 2$: MOV #BEGIN-50,R4 ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
8231 005114 012524 MOV (R5)+,(R4)+
8232 005116 020537 000340 CMP R5,@#MAXMEM
8233 005122 103774 BLO 2$
8234 005124 105037 000405 CLRB @#REL
8235 005130 010537 000346 MOV R5,@#SAVR5 ;SAVE R5
8236 005134 012706 000500 MOV #BEGIN,SP ;RESET STACK TO LOWER MEMORY
8237 005140 010637 000350 MOV SP,@#SAVR6 ;'BEGIN' USES THIS TO RESET THE STACK.
8238 005144 000137 005150 CKDONE: JMP @#LOWER ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
8239
8240
8241
8242 005150 105737 000315 LOWER: TSTB @#SAVKBB ;HERE DUE TO ^C TYPED?
8243 005154 001073 BNE $TPSTK ;BRANCH IF YES (TYPE ERROR STACK)
8244 005156 004767 001702 TSTM: JSR PC,MEMMNG ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
8245 ;IS AVAILABLE
8246 005162 105737 000276 TSTB @#MMAVA ;IS MEM. MANAG. AVAILABLE ?
8247 005166 001462 BEQ ENDPAS ;BRANCH IF NO
8248 005170 000402 BR $CNTMM ;BEGIN TESTING ABOVE 28K
8249 005172 004767 002036 CONTMM: JSR PC,UPMM ;GO TO UPDATE MEM. MANAG. REGISTERS
8250 005176 012703 000324 $CNTMM: MOV #LOWTWO,R3 ;MAKE R3 POINT TO THE LOCATION LOWTWO
8251 005202 004767 002142 JSR PC,GETSIZ ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
8252 ;OF THE LOWEST ADDRESS UNDER TEST
8253 005206 012704 020000 MOV #20000,R4 ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
8254 ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
8255 005212 020237 172342 CMP R2,@#172342 ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF

```

8256			
8257	005216	103405	
8258	005220	050104	
8259			
8260	005222	162702	000200
8261	005226	004767	001636
8262	005232	004767	002112
8263			
8264			
8265	005236	004767	000020
8266	005242	010005	
8267	005244	004767	002100
8268	005250	004767	000006
8269	005254	010013	
8270	005256	000167	174256
8271			
8272			
8273			
8274			
8275			
8276			
8277			
8278			
8279	005262	010046	
8280	005264	012700	172356
8281			
8282	005270	162716	020000
8283	005274	050116	
8284	005276	020240	
8285	005300	001410	
8286	005302	020027	172340
8287	005306	101370	
8288			
8289	005310	005720	
8290	005312	021002	
8291	005314	003006	
8292	005316	012716	157776
8293	005322	012600	
8294	005324	062700	000002
8295	005330	000207	
8296			
8297	005332	022626	
8298			

```

;MAXADR - SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K.
;REGISTERS:
;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
;R2= PAR BLOCK NO. CURRENTLY UNDER TEST.

```

```

MAXADR:  MOV      R0, -(SP)          ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
          MOV      #172356, R0       ;R0=PAR7 UNIBUS ADDRESS
; **BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2$:      SUB      #20000, (SP)        ;DECREMENT VIRTUAL ADDRESS BY 4K
          BIS      R1, (SP)          ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
          CMP      R2, -(R0)         ;DOES CURRENT PAR= TEST BLOCK NO.?
          BEQ      3$               ;BRANCH IF YES
          CMP      R0, #172340       ;ARE WE AT PAR0?
          BHI      2$               ;NO KEEP TRYING
; **END LOOP TO FIND PAR ADDRESS UNDER TEST
          TST      (R0)+             ;SET TO PAR CURRENT
          CMP      (R0), R2 ; IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
          BGT      4$               ;BRANCH IF YES (FALL INTO ENDPAS)
          MOV      #157776, (SP)     ;EXIT WITH MAXADR= 28K SEGMENT TEST SIZE
3$:      MOV      (SP)+, R0          ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
          ADD      #2, R0            ;MAKE MAXIMUM MEMORY+2
          RTS      PC               ;AND EXIT MAXADR ROUTINE

4$:      CMP      (SP)+, (SP)+       ;FIXUP STACK
          ;AND FALL THRU TO ENDPAS.

```

```

8303      ;* TYPE ROUTINE FOR ERROR STACK
8304      ;* -----
8305      ;*
8306      ;* THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
8307      ;* FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
8308
8309
8310
8311 005334 032777 000020 173106 ENDPAS: BIT    #20,@SWR      ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
8312 005342 001055                BNE     $EOP          ;IF NOT THEN GO TO $EOP
8313 005344 012746 177777      $TPSTK: MOV    #-1,-(SP)    ;THE PROGRAM HAS REACHED THE END AND ERROR
8314                                ;STACK AND END OF PASS WILL BE TYPED OUT
8315 005350 012701 007744                MOV    #ENDPRG,R1  ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
8316                                ;FOR 0 TO 4K MEMORY IN R1
8317 005354 012703 000376      TYPSTK: MOV    #376,R3
8318 005360 005216                INC     (SP)          ;IF WE HAVE GONE THRU THE ENTIRE
8319 005362 020137 000310                CMP    R1,@#ENDSTK ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
8320 005366 103043                BHS     $EOP          ;THEN GO TO TYPE END OF PASS
8321 005370 112702 000022                MOVB   #18.,R2
8322 005374 105302                RETSTK: DECB   R2      ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED.
8323 005376 002766                BLT     TYPSTK        ;BEEN CHECKED FOR ERROR THEN SEE IF THERE
8324                                ;IS ANY MORE 4K MEMORY BANK
8325 005400 105721                TSTB    (R1)+        ;OTHERWISE CHECK THE BYTE STORED AT (R1)
8326 005402 001774                BEQ     RETSTK        ;IF IT IS 0 WE WILL NOT TYPE IT
8327 005404 020227 000020                CMP    R2,#16.  ;IS THE POINTER POINTING TO ERROR STACK BYTE
8328                                ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
8329                                ;THE SPECIFIC MEMORY BANK
8330 005410 103404                BLO     2$           ;IF NOT THEN GO TO TYPE BIT NUMBER
8331 005412 101026                BHI     PARFL        ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
8332                                ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
8333 005414 004767 001000                JSR     PC,TPADER  ;OTHERWISE TYPE "ADDRESS ERROR"
8334 005420 000404                BR      FAILNM
8335 005422 010237 000312      2$:      MOV    R2,@#DECWRD ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
8336                                ;IN DECIMAL
8337 005426 004767 001202                JSR     PC,TYPDEC  ;GO TO TYPE THE BIT NUMBER IN DECIMAL
8338 005432 011637 000312      FAILNM: MOV    (SP),@#DECWRD ;PREPARE TO TYPE THE PAGE NUMBER
8339 005436 004767 001176                JSR     PC,$TPDEC ;IN DECIMAL
8340 005442 005043                CLR     -(R3)
8341 005444 114113                MOVB   -(R1),(R3)      ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
8342                                ;FAILURE OCCURED
8343 005446 105021                CLRB   (R1)+        ;CLEAR THE ERROR STACK
8344 005450 005043                CLR     -(R3)
8345 005452 105237 000314                INCB   @#TYPCNT  ;ENABLE THE TYPE OUT OF 1 WORDS
8346 005456 004767 001316                JSR     PC,RPTOCT ;TYPE THE 4K BANK AND THE NUMBER OF TIMES
8347                                ;THIS FAILURE WAS SEEN
8348 005462 012703 000376                MOV    #376,R3
8349 005466 000742                BR      RETSTK
8350 005470 004767 000750      PARFL: JSR     PC,TPPRER  ;TYPE "PAR ERR"
8351 005474 000756                BR      FAILNM

```

```

8356
8357
8358
8359
8360
8361
8362
8363
8364
8365
8366
8367 005476 005002 $EOP: CLR R2 ;SET R2= PARITY MODULE DISABLE CODE
8368 005500 004767 002014 JSR PC,PARITY ;GO DISABLE PARITY MODULES IF SELECTED.
8369 005504 105737 000315 TSTB @#SAVKBB ;CONTROL-C TYPED?
8370 005510 001043 BNE CTLC ;BRANCH IF YES-RESTORE LOADERS AND HALT-
8371 005512 005237 000406 INC @#$PASS ;INCREMENT PASS COUNT
8372 005516 032777 000040 172724 BIT #40,@SWR ;'PASS#XX' PRINTOUT WANTED?
8373 005524 001012 BNE ACT11 ;BRANCH IF NO
8374 005526 004767 001052 TYPEOP: JSR PC,TPCRLF ; TYPE CR, LF, AND 'PASS#'
8375 005532 040520 051523 000043 .ASCIZ /PASS#/
8376 .EVEN
8377 005540 013737 000406 000312 MOV @#$PASS,@#DECWRD ;GET PASS COUNT
8378 005546 004767 001066 JSR PC,$TPDEC ;TYPE IT
8379 005552 013700 000042 ACT11: MOV @#42,R0 ;GET THE MONITOR ADDRESS
8380 005556 001405 BEQ $DOAGN ;IF NONE
8381 005560 004767 000012 JSR PC,RLODER ;RESTORE XXDP MONITOR
8382 005564 000005 RESET ;RETURN TO ACT11 MONITOR.
8383
8384
8385 ;* SERVICE XXDP/ACT11
8386 005566 000137 000164 JMP @#$ENDAD ;JUMP TO ACT SERVICE
8387
8388 $DOAGN: JMP @#RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP
8389
8390 RLODER: JSR PC,CLRMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
8391 005602 013704 000344 MOV @#SAVR4,R4 ;RESTORE R4 WITH SAVR4
8392 005606 014445 4$: MOV -(R4),-(R5) ;RESTORE LOADERS
8393 005610 020437 000310 CMP R4,@#ENDSTK
8394 005614 101374 BHI 4$
8395 005616 000207 RTS PC ;RETURN FROM RLODER CALL
8396
8397 ;CONTROL C HANDLER
8398
8399 005620 004767 177752 CTLC: JSR PC,RLODER ;RESTORE ABS LOADER
8400 005624 000167 000400 JMP APTHLT ;IF NOT APT HALT AT FATHLT
8401
8402

```



```

8407          ;* ERROR HANDLING ROUTINE
8408          ;* -----
8409          ;*
8410          ;* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
8411          ;* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
8412          ;*
8413
8414 005630 017637 000000 000402 ERROR: MOV @ (SP), @#$FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
8415 005636 010346 1$: MOV R3, -(SP) ;SAVE R3
8416 005640 010046 MOV R0, -(SP) ;AND R0 ON THE STACK
8417
8418          ;SETUP BANK NO. IN FATAL FOR APT
8419
8420 005642 010103 MOV R1, R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
8421 005644 004767 001570 JSR PC, GETBNK ;GET BANK NO. UNDER TEST INTO PBNK
8422 005650 013703 000312 MOV @#PBNK, R3 ;GET BANK UNDER TEST
8423 005654 110337 000403 MOVB R3, @#$FATAL+1 ;STORE FAILING BANK NO. FOR APT
8424
8425          ;
8426
8427 005660 010346 MOV R3, -(SP) ;TEMPORARILY STORE R3
8428 005662 012703 000376 MOV #376, R3 ;MAKE R3 AS THE STACK POINTER
8429 005666 013743 000306 MOV @#PASFLG, -(R3) ;OUTPUT THE WORD STORED AT
8430 005672 005043 2$: CLR -(R3)
8431 005674 113713 000402 MOVB @#$FATAL, (R3) ;PUT ERROR NO. ON ERROR STACK
8432 005700 016643 000006 MOV 6(SP), -(R3) ;PLACE THE RETURN PC AT (R3)
8433 005704 011143 MOV (R1), -(R3) ;PLACE BAD DATA,
8434 005706 010043 MOV R0, -(R3) ;AND GOOD DATA ON THE STACK
8435 005710 005043 CLR -(R3)
8436 005712 016313 000004 MOV 4(R3), (R3) ;TAKE THE
8437 005716 040013 BIC R0, (R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
8438 005720 046300 000004 BIC 4(R3), R0 ;TO FIND THE BITS THAT FAILED
8439 005724 050013 BIS R0, (R3) ;AND PLACE IT ON THE STACK
8440 005726 012700 001766 MOV #ENDPRG--24., R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
8441 005732 060700 ADD PC, R0 ;OF THE STARTING OF THE ERROR STACK
8442 005734 062700 000022 6$: ADD #18., R0 ;FOR THE SPECIFIC 4K BANK
8443 005740 005316 DEC (SP)
8444 005742 002374 BGE 6$
8445 005744 005726 TST (SP)+ ;RESTORE THE STACK POINTER
8446
8447 005746 105037 000277 ERRYP: CLRB @#TYPENB ;DISABLE ANY TYPE OUT
8448 005752 105737 000300 1$: TSTB @#$SPRERR ;IF THIS IS PARITY PROBLEM
8449 005756 001007 BNE 3$ ;THEN GO TO 3$
8450 005760 105720 TSTB (R0)+ ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
8451 005762 105737 000301 TSTB @#$ADERR ;IF THIS IS ADDRESSING PROBLEM
8452 005766 001003 BNE 3$ ;THEN GO TO 3$
8453 005770 105720 TSTB (R0)+ ;INCREMENT THE POINTER R0 BY 1
8454 005772 005713 2$: TST (R3) ;IS BIT 15 OF (R3) SET?
8455 005774 100015 BPL 4$ ;IF NOT THEN GO TO 4$
8456 005776 122710 000377 3$: CMPB #377, (R0) ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
8457 006002 001401 BEQ 5$ ;IF SO DON'T BUMP ERROR COUNT
8458 006004 105210 INCB (R0) ;INCREMENT THE ERROR COUNTER BY 1
8459 006006 122710 000001 5$: CMPB #1, (R0) ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
8460 006012 001404 BEQ 7$ ;BRANCH IF NO
8461 006014 032777 000400 172426 BIT #400, @SWR ;STOP ERROR PRINTOUT AFTER 1 WANTED?
8462 006022 001002 BNE 4$ ;BRANCH IF YES (DON'T TYPE ERROR)

```

8463	006024	105237	000277	7\$:	INCB	@#TYPENB	:ENABLE THE TYPE OUT ROUTINE
8464	006030	105737	000300	4\$:	TSTB	@#\$PRERR	:PARITY ERROR?
8465	006034	001403			BEQ	9\$:BRANCH IF NO
8466	006036	004767	000402		JSR	PC,TPPRER	:ELSE TYPE 'PAR ERR'
8467	006042	000411			BR	8\$:AND DON'T TEST INDIVIDUAL BIT FAILURES.
8468	006044	105737	000301	9\$:	TSTB	@#\$ADERR	:ADDRESS ERROR?
8469	006050	001403			BEQ	6\$:BRANCH IF NO
8470	006052	004767	000342		JSR	PC,TPADERR	:PRINT 'ADR ERR'
8471	006056	000403			BR	8\$	
8472	006060	105720		6\$:	TSTB	(R0)+	:POINT TO NEXT ENTRY IN ERROR STACK
8473	006062	006313			ASL	(R3)	:IS THERE STILL AN ERROR BIT SET IN ERROR.
8474	006064	001342			BNE	2\$:BR IF YES - KEEP FILLING ERROR STACK
8475	006066	112737	000006 000314	8\$:	MOVB	#6,@#TYPCNT	:TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
8476							:THE STACK POINTED BY R3
8477	006074	004767	001142		JSR	PC,PUTADR	:GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
8478							:AT LOCATIONS (R3) AND (R3-2)
8479	006100	004767	000616		JSR	PC,TYPERR	:TYPE ERROR STACK (7 WORDS)
8480							
8481	006104	005037	000300	10\$:	CLR	@#\$PRERR	:CLEAR ADDRESS/PARITY ERROR FLAGS
8482	006110	012600			MOV	(SP)+,R0	:RESTORE R0
8483	006112	012603			MOV	(SP)+,R3	:AND R3
8484	006114	105737	000420	FNDERR:	TSTB	@#\$ENV	:ARE WE RUNNING UNDER APT?
8485	006120	001404			BEQ	2\$:IF NOT THEN TEST FOR HALT
8486	006122	012737	000001 000400		MOV	#1,@#\$MSGTY	:OTHERWISE INFORM THE APT
8487	006130	000442			BR	FATHLT	:GOTO FATHLT AND WAIT FOR APT.
8488							
8489	006132	010246		2\$:	MOV	R2,-(SP)	:SAVE R2 TEMP
8490	006134	005777	172310		TST	@\$WR	:DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
8491							:ON ERROR
8492	006140	100405			BMI	4\$:IF SO THEN HALT ON ERROR
8493							
8494							:CHECK FOR CONTROL-C KEY
8495	006142	004767	001526		JSR	PC,CHECKC	:IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
8496							:AND HALT AT FATHLT.
8497	006146	105737	000042	7\$:	TSTB	@#42	:ARE WE RUNNING UNDER ACT?
8498	006152	001401			BEQ	6\$:BRANCH IF NO
8499							
8500	006154	000000		4\$:	HALT		:PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
8501							:TO A LOCATION WHICH SHOULD HAVE CONTAINED
8502							:THE WORD STORED IN R0
8503	006156	012602		6\$:	MOV	(SP)+,R2	:RESTORE R2
8504	006160	062716	000002		ADD	#2,(SP)	:RESTORE THE RETURN ADDRESS
8505	006164	000207			RTS	PC	:RETURN FROM THE SUBROUTINE
8506							
8507							
8508							
8509	006166			FATERR:			
8510	006166	004767	000412	SEQERR:	JSR	PC,TPCRLF	:TYPE 'ERR #'
8511	006172	051105	020122 000043		.ASCIZ	/ERR #/	
8512					.EVEN		
8513							
8514	006200	017637	000000 000402		MOV	@(SP),@#\$FATAL	:LOAD THE LOCATION \$FATAL WITH THE ERROR NUMBER
8515	006206	105237	000314		INCB	@#TYPCNT	:TELL \$TPNUM TO TYPE 1 WORD
8516	006212	012703	000376		MOV	#376,R3	:\$TPNUM USES R3 AS STACK
8517	006216	013743	000402		MOV	@#\$FATAL,-(R3)	:PUT ERROR NO. ON STACK
8518	006222	005743			TST	-(R3)	:\$TPNUM REQUIRES THIS

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 64-11
 CNKMAA.F11 27-DEC-82 13:15 ERROR HANDLING ROUTINE

SEQ 0054

```

8519 006224 004767 000560      JSR      PC,FATYP      ;TYPE ERROR NO.
8520 006230 105737 000420      APTHLT: TSTB    @#SENV ;RUNNING UNDER APT?
8521 006234 001327              BNE      FNDERR      ;BRANCH IF YES
8522 006236 000000              FATHLT: HALT          ;FATAL ERROR OR ^C HALT.
8523 006240 000137 000250      JMP      @#RESTRT      ;RESTART TST BUT DON'T CLEAR PASS COUNT
8524                                     ;IN CASE ^C RESTART.
8525
8526
8527      ;PARERR
8528      ; PARITY TRAP HANDLER
8529      ; COME HERE FROM A TRAP TO 114.
8530      ; THIS ROUTINE SEARCHES THE AVAILABLE PARITY MODULES AND IF ONE
8531      ; HAS A PARITY ERROR BIT SET THE GET THE PARITY ERROR ADDRESS
8532      ; AND CALL THE 'ERROR' ROUTINE TO PRINT ERROR MESSAGE.
8533      ; IF NO PARITY ERROR BITS CAN BE FOUND A FATAL ERROR IS DONE.
8534
8535      ; REGISTER US AGE.
8536      ; R0= HOLDS PARITY MODULE ADDRESSES.
8537      ; P1= GETS ERROR ADDRESS FOR 'ERROR' CALL.
8538
8539 006244 012637 000356      PARERR: MOV      (SP)+,@#PARSP ;SET PARSP TO RETURN ADDRESS
8540 006250 011637 000360      MOV      (SP),@#PARPS ;SAVE PSW FOR RETURN
8541 006254 013706 000350      MOV      @#SAVR6,SP ;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
8542                                     ; TO COMPLETE THE ERROR SERVICE ROUTINE.
8543 006260 010067 000130      MOV      R0,SAVR0 ;SAVE R0 DURING PARITY SERVICE
8544 006264 010167 000126      MOV      R1,SAVR1 ;SAVE R1 DURING PARITY SERVICE
8545 006270 013701 000352      MOV      @#PARMAP,R1 ;GET PARITY AVAILABLE MAP
8546 006274 012700 172100      MOV      #172100,R0 ;R0= FIRST PARITY ADDRESS.
8547
8548 006300 005701              TST      R1 ;ANY PARITY MODULES AVAILABLE?
8549 006302 001441              BEQ      4$ ;BR IF NO -FATAL ERROR-
8550 006304 006001              1$: ROR      R1 ;SHIFT PARITY MAP BIT INTO C BIT.
8551 006306 103005              BCC      2$ ;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
8552 006310 005710              TST      (R0) ;PARITY MODULE ERROR BIT SET?
8553 006312 100406              BMI      3$ ;BRANCH IF YES -CALL 'ERROR' ROUTINE
8554 006314 020027 172136      CMP      R0,#172136 ;DONE ALL PARITY MODULES?
8555 006320 002032              BGE      4$ ;BR IF YES- GO TO FATAL ERROR CALL-
8556 006322 062700 000002      2$: ADD      #2,R0 ;POINT TO NEXT PARITY ADDRESS
8557 006326 000766              BR       1$ ;AND KEEP TRYING
8558 006330 042710 100000      3$: BIC      #100000,(R0) ;CLEAR PARITY ERROR BIT.
8559 006334 011001              MOV      (R0),R1 ;GET PARITY MODULE CSR
8560 006336 006101              ROL      R1 ;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
8561 006340 006101              ROL      R1
8562 006342 006101              ROL      R1
8563 006344 006101              ROL      R1
8564 006346 042701 000777      BIC      #777,R1 ;SAVE ERROR ADDRESS ONLY
8565 006352 105237 000300      INCB    @#$PRERR ;TELL 'ERROR' PARITY ERROR CALL.
8566 006356 004767 177246      JSR      PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
      (1) 006362 000050      50          ;*****ERROR NUMBER 50*****
      (1)
8567 006364 016700 000024      MOV      SAVR0,R0 ;RESTORE R0
8568 006370 016701 000022      MOV      SAVR1,R1 ;RESTORE R1
8569 006374 013743 000360      MOV      @#PARPS,-(SP) ;SET RETURN PSW ON STACK
8570 006400 013746 000356      MOV      @#PARSP,-(SP) ;AND SET RETURN ADDRESS ON STACK
8571 006404 000002              RTI      ;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
8572

```

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 64-12
 CNKMAA.P11 27-DEC-82 13:15 ERROR HANDLING ROUTINE

SEQ 0055

```

8573      ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
8574 006406 4$:
(1) 006406 004767 177554      JSR      PC,FATERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 006412 000051      51      ;*****ERROR NUMBER 51*****
(1)
8575
8576      ;R0+R1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
8577      ;STACK SPACE BETWEEN 500-450.
8578 006414 000000      SAVR0: 0      ;SAVE R0 DURING PARITY TRAP SERVICE
8579 006414 000000      SAVR1: 0      ;SAVE R1 DURING PARITY TRAP SERVICE
8580
8581
8582 006420 105737 000277      TPADER: TSTB      @#TYPENB      ;TYPE ERROR?
8583 006424 001406      BEQ      1$      ;BRANCH IF NO
8584 006426 004767 000160      JSR      PC,PNTMES      ; TYPE CR, LF AND 'ADR ER'
8585 006432 042101 020122 051105      .ASCIZ /ADR ERR/
      006440 000122
8586
8587 006442 000207      1$:      .EVEN
      RTS      PC
8588
8589 006444 105737 000277      TPPER: TSTB      @#TYPENB      ;ERROR PRINTOUTS ALLOWED?
8590 006450 001406      BEQ      1$      ;BRANCH IF NO
8591 006452 004767 000134      JSR      PC,PNTMES      ;GO TO TYPE CR, LF AND 'PAR ERR'
8592 006456 040520 020122 051105      .ASCIZ /PAR ERR/
      006464 000122
8593
8594 006466 000207      1$:      .EVEN
      RTS      PC

```

```

8599
8600
8601          ;* TYPE OUT ROUTINE
8602          ;* -----
8603          ;*
8604          ;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
8605          ;*
8606
8607 006470 010146 NOTYP: MOV R1,-(SP)
8608 006472 016601 000002 MOV 2(SP),R1
8609 006476 105721 4$: TSTB (R1)+ ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
8610 006500 001376 BNE 4$ ;PREPARE TO RETURN
8611 006502 000412 BR RETTYP
8612 006504 010146 $TYPE: MOV R1,-(SP) ;SAVE R1
8613 006506 010046 MOV R0,-(SP) ;AND R0 ON THE STACK
8614 006510 016601 000004 MOV 4(SP),R1 ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
8615 006514 112100 2$: MOVB (R1)+,R0 ;PLACE THE BYTE TO BE TYPED IN R0
8616 006516 001403 BEQ 4$ ;IF IT IS END OF MESSAGE THEN GO TO 4$
8617 006520 004767 000022 JSR PC,$TPCHR ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
8618 006524 000773 BR 2$
8619 006526 012600 4$: MOV (SP)+,R0 ;RESTORE R0
8620 006530 005201 RETTYP: INC R1 ;CAUSE R1 TO
8621 006532 042701 000001 BIC #1,R1 ;POINT TO EVEN ADDRESS
8622 006536 010166 000002 MOV R1,2(SP) ;MODIFY THE RETURN ADDRESS
8623 006542 012601 MOV (SP)+,R1 ;RESTORE R1
8624 006544 000416 BR EXTYP ;AND RETURN VIA RTS PC
8625
8626 006546 132737 000040 000421 $TPCHR: BITB #40,@#SENVN ;HAVE TYPE OUTS BEEN DISABLED?
8627 006554 001005 4$ BNE 4$ ;IF SO THEN RETURN FROM THE SUBROUTINE
8628 006556 105737 177564 2$: TSTB @#$TPS ;WAIT HERE
8629 006562 100375 BPL 2$ ;UNTIL THE PRINTER IS READY
8630 006564 110037 177566 MOVB R0,@#$TPB ;LOAD DATA TO BE TYPED INTO DATA REG.
8631 006570 000404 4$: BR EXTYP ;RETURN
8632
8633 006572 004767 177706 PCRLF: JSR PC,$TYPE
8634 006576 005015 000 .ASCIZ <15><12> ;CR/LF
8635 006602 006602 .EVEN
8636 006602 000207 EXTYP: RTS PC ;RETURN
8637
8638 006604 004767 177762 TPCRLF: JSR PC,PCRLF ;TYPE CR/LF
8639 006610 000735 BR $TYPE ;NOW GO TO TYPE THE REST OF THE MESSAGE
8640
8641
8642 006612 032777 000020 171630 PNTMES: BIT #20,@#SWR ;PRINTOUTS ALLOWED?
8643 006620 001323 BNE NOTYP ;BRANCH IF NO
8644 006622 123737 000042 000046 CMPB @#42,@#46 ;RUNNING UNDER ACT 11?
8645 006630 001717 BEQ NOTYP ;BRANCH IF YES -NOT PRINTOUT-
8646 006632 000764 BR TPCRLF ;SEND CR/LF AND TYPE MESSAGE.

```

```

8651
8652          ;* ROUTINE TO TYPE OUT A DECIMAL NUMBER
8653          ;* -----
8654          ;*
8655          ;* THIS ROUTINE IS USED TO CONVERT THE CONTENTS OF LOCATION
8656          ;* DECWRD TO DECIMAL NUMBERS AND TYPE THEN FOLLOWING 3 SPACES
8657          ;*
8658
8659 006634 004767 177732      TYPDEC: JSR      PC,PCRLF      ;TYPE CR/LF
8660
8661 006640 005046      STPDEC: CLR      -(SP)
8662 006642 013746 000312      MOV      @#DECWRD,-(SP)      ;GET THE WORD THAT HAS TO BE CONVERTED TO A
8663                                     ;DECIMAL NUMBER
8664 006646 162716 000012      2$:      SUB      #10.,(SP)
8665 006652 002403      BLT      4$      ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
8666                                     ;GO TO 4$
8667 006654 005266 000002      INC      2(SP)      ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
8668 006660 000772      BR      2$      ;AND RETURN TO 2$
8669 006662 062716 000012      4$:      ADD      #10.,(SP)
8670 006666 052716 000060      BIS      #60,(SP)      ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
8671 006672 112667 000020      MOVB     (SP)+,6$-2      ;PLACE THE 1'S DIGIT TO BE TYPED
8672 006676 052716 000060      BIS      #60,(SP)      ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
8673 006702 112667 000007      MOVB     (SP)+,6$-3      ;PLACE THE 10'S DIGIT TO BE TYPED
8674 006706 004767 177572      JSR      PC,$TYPE      ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
8675                                     ;3 SPACES
8676 006712 020040 030040 000060      .ASCIZ / 00/
8677      .EVEN
8678 006720 000207      6$:      RTS      PC      ;RETURN FROM THE SUBROUTINE

```

```

8683
8684
8685
8686
8687
8688
8689
8690
8691
8692
8693
8694
8695
8696
8697
8698 006722 032777 020000 171520 TYPERR: BIT #20000, @SWR ;ERROR PRINTOUT WANTED?
8699 006730 001054 BNE OCTXT ;BRANCH IF NO
8700 006732 004767 177634 JSR PC, PCRLF ;TYPE CR/LF
8701 006736 004767 000012 JSR PC, TYPOCT ;TYPE OCTAL NO.
8702 006742 000447 BR OCTXT ;RETURN VIA RTS PC
8703 006744 012123 OCTTYP: MOV (R1)+, (R3)+ ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
8704 ;BY R3
8705 006746 012113 MOV (R1)+, (R3) ;AND NOW PLACE THE LOW ORDER BITS
8706 006750 105237 000314 INCB @#TYPCNT ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
8707 006754 052743 000004 TYPOCT: BIS #4, -(R3)
8708 006760 106113 2$: ROLB (R3)
8709 006762 103376 BCC 2$
8710 006764 005000 CLR R0
8711 006766 106113 ROLB (R3) ;GET BITS 17 & 16 INTO R0
8712 006770 006100 ROL R0
8713 006772 106113 ROLB (R3)
8714 006774 006100 ROL R0
8715 006776 000405 BR $TPNUM
8716 007000 004767 177500 RPTOCT: JSR PC, $TYPE ; TYPE 3 SPACES
8717 007004 020040 000040 .ASCIZ / /
8718 .EVEN
8719 007010 005000 FATYP: CLR R0
8720 007012 012723 000006 $TPNUM: MOV #6, (R3)+ ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
8721 007016 000241 4$: CLC
8722 007020 006113 ROL (R3)
8723 007022 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
8724 007024 052700 000060 BIS #60, R0 ;OR THE CONTENTS OF R0 WITH AN ASCII 0
8725 007030 004767 177512 JSR PC, $TPCHR ; TYPE THE OCTAL NUMBER STORED IN R0
8726 007034 005000 CLR R0
8727 007036 006113 ROL (R3)
8728 007040 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
8729 007042 006113 ROL (R3)
8730 007044 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
8731 007046 105363 177776 DECB -2(R3) ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
8732 007052 001361 BNE 4$ ;THEN REPEAT FROM 4$
8733 007054 105337 000314 DECB @#TYPCNT ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
8734 007060 001347 BNE RPTOCT ;TYPED THEN REPEAT FROM RPTOCT
8735 007062 000207 OCTXT: RTS PC
  
```

```

8740
8741      ;* ROUTINE TO SET UP MEMORY MANAGEMENT REGISTERS
8742      ;* -----
8743      ;*
8744      ;* PROGRAM CONTROL COMES HERE TO DETERMINE IF THE MEMORY MANAGEMENT
8745      ;* IS AVAILABLE OR NOT, AND IF IT IS AVAILABLE THEN WHETHER
8746      ;* THE MEMORY ABOVE 28K IS REQUIRED TO BE TESTED OR NOT.
8747      ;*
8748
8749 007064 012702 001400      MEMMNG: MOV      #1400,R2
8750 007070 105037 000276      MMREG: CLRB      @#MMAVA      ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
8751                                     ;THAT MEM. MANAG. IS AVAILABLE FOR TESTING
8752 007074 032777 010000 171346      BIT      #10000,@SWR      ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
8753 007102 001441          BEQ      RETMM      ;IF NOT THEN RETURN FROM THE SUBROUTINE
8754 007104 012700 000004      MOV      #4,R0      ;PREPARE TO SETUP TIME OUT VECTOR
8755 007110 012720 007210      MOV      #NOMM,(R0)+    ;RETURN ADDRESS TO NOMM
8756 007114 012710 000300      MOV      #300,(R0)      ;AND WITH A PSW OF 300;VER:1
8757 007120 005037 177572      CLR      @#SRO      ;TRY TO REACH MEM. MANAG. SRO
8758 007124 105237 000276      INCB      @#MMAVA      ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
8759                                     ;BYTE
8760 007130 012701 172340      MOV      #172340,R1      ;R1 IS POINTING TO PAR0
8761 007134 005021          CLR      (R1)+      ;PAR0 WILL POINT TO BANK 0
8762 007136 062702 000200      2$:      ADD      #200,R2
8763 007142 010221          MOV      R2,(R1)+      ;SETUP PAR1-PAR6
8764 007144 020127 172356      CMP      R1,#172356
8765 007150 103772          BLO      2$
8766 007152 012711 007600      MOV      #7600,(R1)      ;PAR7 IS POINTING TO THE I/O PAGE
8767 007156 012701 172300      MOV      #172300,R1
8768 007162 012721 077406      4$:      MOV      #77406,(R1)+    ;SETUP PDR0-PDR7
8769 007166 020127 172316      CMP      R1,#172316
8770 007172 101773          BLOS      4$
8771 007174 005237 177572      INC      @#SRO      ;ENABLE MEM. MANAG.
8772 007200 005010      $RETMM: CLR      (R0)      ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
8773 007202 012740 000104      MOV      #BUSER,-(R0)
8774 007206 000207      RETMM:  RTS      PC
8775
8776 007210 022626      NOMM:  CMP      (SP)+,(SP)+    ;RESTORE STACK POINTER
8777 007212 004767 177366      JSR      PC,TPCRLF      ;TYPE 'NO MEMORY MANAGEMENT MESSAGE
8778 007216 047516 045440 000124      .ASCIZ      /NO KT/
8779      .EVEN
8780 007224 004767 176736      JSR      PC,FATERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
8781 (1) 007230 000052      52      ;*****ERROR NUMBER 52*****
8782 (1)
8781 007232 000762          BR      $RETMM      ; RESTORE TIME OUT TRAP VECTOR
8782
8783 007234 013702 172354      UPM:  MOV      @#172354,R2      ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
8784 007240 000713          BR
  
```



```

8789
8790
8791
8792
8793
8794
8795
8796
8797
8798
8799
8800 007242 005063 177776      PUTADR: CLR      -2(R3)
8801 007246 010113              MOV      R1,(R3)      ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
8802 007250 105737 000276      TSTB      @#MMAVA      ;IS THE MEM. MANAG. AVAILABLE ?
8803 007254 001425              BEQ        6$          ;IF NOT THEN RETURN FROM THE SUBROUTINE
8804 007256 010146              MOV      R1,-(SP)      ;SAVE R1
8805 007260 042701 017777      BIC      #17777,R1      ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
8806 007264 040113              BIC      R1,(R3)      ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
8807 007266 052701 004000      BIS      #4000,R1      ;PREPARE TO SHIFT R1 BY 12 PLACES
8808 007272 006001      2$:      ROR      R1
8809 007274 103376              BCC      2$          ;GET THE NUMBER OF PAR IN R1
8810 007276 062701 172340      ADD      #172340,R1      ;GET THE ADDRESS OF PAR IN R1
8811 007302 011101              MOV      (R1),R1      ;LOAD R1 WITH THE CONTENTS OF PAR
8812 007304 052701 010000      BIS      #10000,R1
8813 007310 006101      4$:      ROL      R1
8814 007312 103376              BCC      4$          ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
8815 007314 006101              ROL      R1
8816 007316 006143              ROL      -(R3)      ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
8817 007320 006101              ROL      R1
8818 007322 006123              ROL      (R3)+      ;PLACE BIT 16 OF THE ADDRESS
8819 007324 050113              BIS      R1,(R3)      ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
8820 007326 012601              MOV      (SP)+,R1      ;RESTORE R1
8821 007330 000207      6$:      RTS      PC          ;RETURN FROM THE SUBROUTINE
8825
8826
8827
8828
8829
8830
8831
8832
8833
8834
8835
8836
8837
8838 007332 016143 000001      GETADR: MOV      1(R1),-(R3)      ;PLACE THE LOW ORDER BITS OF THE ADDRESS
8839 007336 005043              CLR      -(R3)      ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
8840                                ;HAVE TO BE PLACED
8841 007340 116113 177777      MOVB     -1(R1),(R3)      ;PLACE BITS 16 & 17
8842 007344 000207      2$:      RTS      PC          ;RETURN FROM THE SUBROUTINE

```

```

8847
8848
8849
8850
8851
8852
8853
8854
8855
8856
8857 007346 005046          $GTSIZ: CLR      -(SP)          ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
8858                                     ;0-4 OF R2
8859
8860 007350 012301          GETSIZ: MOV      (R3)+,R1
8861 007352 011302          MOV      (R3),R2          ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
8862 007354 042702 017777          BIC      #17777,R2          ;CLEAR ADDRESS BITS 0-12
8863 007360 052702 000040          2$:      BIS      #40,R2
8864 007364 006001          4$:      ROR      R1
8865 007366 006002          ROR      R2          ;ROTATE R1 AND R2 7 TIMES
8866 007370 103375          BCC      4$
8867 007372 005716          TST      (SP)          ;IF RETURN PC IS ZERO THEN IT MUST BE THE
8868 007374 001004          BNE      6$          ;FLAG THAT WAS SET AT $GTSIZ
8869 007376 005726          TST      (SP)+          ;POP THE FLAG OFF STACK
8870 007400 052702 000100          BIS      #100,R2          ;KEEP ROTATING
8871 007404 000767          BR       4$
8872 007406 012301          6$:      MOV      (R3)+,R1          ;PLACE THE LOW ORDER ADDRESS BITS IN R1
8873 007410 012700 160000          MOV      #160000,R0
8874 007414 040001          BIC      R0,R1          ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
8875 007416 000207          RTS      PC          ;RETURN FROM THE SUBORNE
8876
8877
8881
8882
8883
8884
8885
8886
8887
8888 007420 105737 000276          CLRMM: TSTB     @#MMAVA          ;WAS THE MEMORY MANAGEMENT ENABLED ?
8889 007424 001404          BEQ      1$          ;IF NOT THEN GO TO 1$
8890 007426 005037 177572          CLR      @#SRO          ;DISABLE THE MEMORY MANAGEMENT
8891 007432 105037 000276          CLRB     @#MMAVA          ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
8892 007436 000207          1$:      RTS      PC          ;RETURN FROM THE SUBROUTINE
8893
8894
8895
8896
8897
8898
8899
8900
8901
8902 007440 010046          ;* GET BANK NO. UNDER TEST
8903 007442 010346          ;      CALLED BY ERRTP AND TST13 TO GET BANK NO. UNDER TEST INTO PBNK.
8904 007444 042703 017777          ;REGISTERS
8905 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

```

8906	007454	000241			CLC		
8907	007456	006003			1\$: ROR	R3	;SHIFT A BANK BIT
8908	007460	103376			BCC	1\$;UNTIL IN BITS <2:0> OF R3
8909	007462	105737	000276		TSTB	@#MMAVA	;MEMORY MANAGEMENT UNDER TEST?
8910	007466	001407			BEQ	2\$;NO EXIT
8911							
8912							;GET PAR ADDRESS AND PHYSICAL BANK NO.
8913	007470	006303			ASL	R3	;MAKE R3 PAR ADDRESS OFFSET.
8914	007472	062703	172340		ADD	#172340,R3	;MAKE FULL PAR ADDRESS.
8915	007476	011300			MOV	(R3),R0	;GET PAR CONTENTS
8916	007500	006300			ASL	R0	
8917	007502	000300			SWAB	R0	;SHIFT BANK BITS TO BITS <7:0>
8918	007504	110003			MOVB	R0,R3	;SET R3 TO PHYSICAL BANK NO.
8919	007506	010337	000312		2\$: MOV	R3,@#PBNK	;STORE PHYSICAL BANK NO.
8920	007512	012603			MOV	(SP)+,R3	;RESTORE R3
8921	007514	012600			MOV	(SP)+,R0	;RESTORE R0
8922	007516	000207			RTS	PC	;RETURN TO CALLER
8923							
8924							
8925							
8926							
8927							
8928							
8929							
8930							
8931							
8932							
8933							
8934							
8935							
8936							
8937							
8938							
8939							
8940							
8941							
8942							
8943							
8944							
8945	007520	032777	004000	170722	PARITY: BIT	#4000,@SWR	;PARITY TEST WANTED?
8946	007526	001456			BEQ	6\$;BRANCH IF NO
8947							
8948	007530	012700	000004		MOV	#4,R0	;POINT R0 TO BUS TIMEOUT ADDRESS.
8949	007534	012710	000116		MOV	#5\$--6,(R0)	;SET RETURN FROM TIMEOUT TRAP TO 5\$
8950	007540	060710			ADD	PC,(R0)	;IN THE CURRENT BANK.
8951	007542	005037	000352		1\$: CLR	@#PARMAP	;CLEAR PARITY MAP HOLDER.
8952	007546	012701	172140		MOV	#172140,R1	;SET R1 TO LAST PARITY MODULE ADDRESS+2
8953	007552	012703	100000		MOV	#100000,R3	;SET R3 TO PARMAP AVAILABLE CODE BEGIN.
8954	007556	010241			2\$: MOV	R2,-(R1)	;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE.
8955	007560	050337	000352		BIS	R3,@#PARMAP	;NO TRAP TO 5\$, SO SET PARITY AVAILABLE.
8956	007564	000241			CLC		
8957	007566	006003			3\$: ROR	R3	;SETUP NEXT PARMAP BIT
8958	007570	103372			BCC	2\$;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
8959	007572	012710	000104		MOV	#BUSER,(R0)	;RESET BUS TIMEOUT TRAP VECTOR
8960	007576	005702			TST	R2	;IS THIS A DISABLE CALL?
8961	007600	001431			BEQ	6\$;BRANCH IF YES (EXIT)

; PARITY ENABLE/DISABLE ROUTINE
 ;
 ; THIS ROUTINE ENABLES OR DISABLES PARITY MODULES AND PRINTS ASSOCIATED MEESSAGES.
 ; IF PARITY AVAILABLE THEN BIT13 OF 'REL' IS SET AND 'PAR'ITY IS PRINTED.
 ; ALSO THE BACKGROUND TEST PATTERN (LOC. BAKPAT) IS SET=376
 ;
 ; REGISTER USAGE.
 ; R0= POINTS TO BUS TIMEOUT TRAP VECTOR (LOC. 4)
 ; R1= HOLDS PARITY MODULE UNIBUS ADDRESS.
 ; R2= ON ENTRY HOLDS ENABLE/DISABLE CODE .
 ; IF R2=0 THEN DISABLE
 ; IF R2=1 THEN ENABLE
 ; R3= SCRATCH TO SETUP LOC. PARMAP WITH A MAP OF PARITY MODULES PRESENT.
 ; CALL IS
 ; MOV #1,R2 ;ENABLE CODE
 ; JSR PC,PARITY
 ;

8962	007602	005737	000352		TST	@#PARMAP	;WERE ANY PARITY MODULES FOUND?
8963	007606	001011			BNE	4\$;BRANCH IF YES
8964	007610	004767	176770		JSR	PC,TPCRLF	;PRINT 'NO PAR'
8965	007614	047516	050040	051101	.ASCIZ	/NO PAR/	
	007622	000					
8966		007624			.EVEN		
8967	007624	004767	176336		JSR	PC,FATERR	;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1)	007630	000053			53		;*****ERROR NUMBER 53*****
(1)							
8968							
8969	007632	152737	000040	000405	4\$: BISB	#40,@#REL	;SET PARITY UNDER TEST FLAG
8970	007640	012737	000376	000316	MOV	#376,@#BAKPAT	;SET BACKGROUND PATTERN TO
8971							;WORST CASE PARITY CODE.
8972	007646	004767	176732		JSR	PC,TPCRLF	;PRINT 'TST PARITY'
8973	007652	040520	000122		.ASCIZ	/PAR/	
8974					.EVEN		
8975	007656	000405			BR	EXITC	;AND EXIT VIA RTS PC
8976							
8977							
8978							

;GET HERE IF PARITY ADDRESS TIMED OUT TO LOC. 4

```

8980
8981 007660 022626      5$:    CMP      (SP)+,(SP)+      ;RESET STACK FROM TRAP
8982 007662 000741      BR      3$                  ;KEEP TRYING PARITY ADDRESSES.
8983
8984 007664 142737 000040 000405 6$:    BICB      #40,@#REL      ;CLEAR PARITY TESTING FLAG
8985 007672      EXITC:
8986 007672 000207      7$:    RTS      PC              ;RETURN TO CALLER
8987
8988
8989
8990
8991      ;CHECKC
8992      ;      THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
8993      ;TEST OR IN THE ERROR TYPE ROUTINE.
8994      ;IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
8995      ;RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
8996      ;FINALLY IT HALTS AT FATHLT.
8997
8998 007674 105037 000315 CHECKC: CLRB      @#SAVKBB      ;INIT CONTROL-C FLAG.
8999 007700 105737 177560      TSTB      @#TKS          ;ANY CHAR. TYPED?
9000 007704 100372      BPL      EXITC          ;BR IF NO-EXIT VIA RTS PC-
9001 007706 113702 177562      MOV      @#SKBB,R2      ;GET THE CHAR TYPED.
9002 007712 042702 000200      BIC      #200,R2 ;CLEAR THE PARITY BIT.
9003 007716 122702 000003      CMP      #3,R2          ;IS IT CONTROL-C?
9004 007722 001363      BNE      EXITC          ;BRANCH IF NO -EXIT VIA RTS PC-
9005 007724 110237 000315      MOV      R2,@#SAVKBB      ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
9006 007730 004767 176650      JSR      PC,TPCRLF      ;PRINT '^C'
9007 007734 041536      .ASCIIZ /^C/
9008 007740      .EVEN
9009 007740 000167 175132      JMP      RELOER          ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
9010
9017      .=7744
9018 007744 000000      ENDPRG: 0
9019
9020
9021
9022
9023      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

[illegible]

[illegible]

[illegible]

CNKMA MACY11 30(1046) 27-DEC-82 13:16 PAGE 67-1
CNKMAA.P11 27-DEC-82 13:15 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0071

.SERRO 2700#
.SERRT 2896#
.SMULT 4523#
.SPOWE 4229#
.SRAND 4307#
.SRDDE 3891#
.SRDOC 3797#
.SREAU 3395#
.SR2AZ 4958#
.SSAVE 3969#
.SSB2D 4771#
.SSB2O 4874#
.SSCOP 2454#
.SSIZE 4361#
.SSUPR 4913#
.STRAP 4073#
.STYPB 3287#
.STYPD 3209#
.STYPE 2985#
.STYPO 3112#
.S4OCA 972#

. ABS. 007746 000

ERRORS DETECTED: 0

CNKMAA,CNKMAA/CRF/NL:TOC=CNMAC2.SML,CNKMAA.P11
RUN-TIME: 10 12 1 SECONDS
RUN-TIME RATIO: 87/24=3.5
CORE USED: 35K (70 PAGES)