

000583

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZKMA-B-D
PRODUCT NAME: MOS/CORE MEMORY EXERCISER FOR 0 TO 124K
 WITH OR WITHOUT PARITY BITS
PRODUCT DATE: AUGUST, 1976
MAINTAINER: DIAGNOSTIC GROUP

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

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

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

COPYRIGHT (C) 1975, 1976 DIGITAL EQUIPMENT CORPORATION

CONTENTS

| | |
|------|---|
| 1.0 | ABSTRACT |
| 1.1 | GETTING STARTED |
| 2.0 | REQUIREMENTS |
| 2.1 | EQUIPMENT |
| 2.2 | STORAGE |
| 3.0 | LOADING PROCEDURE |
| 4.0 | STARTING PROCEDURE |
| 4.1 | SWITCH SETTINGS |
| 4.2 | CONTROL-C OPTION |
| 4.3 | STARTING ADDRESS =200 RESTART ADDRESS =250 |
| 4.4 | PROGRAM AND/OR OPERATOR ACTION |
| 4.5 | LONG GALLOP OPTION |
| 5.0 | PROGRAM HALTS (NORMAL + ERROR) |
| 6.0 | ERRORS |
| 6.1 | ERROR MESSAGE FORMAT. |
| 6.2 | ERROR DICTIONARY |
| 6.3 | ERROR HISTORY |
| 6.4 | ERROR RECOVERY |
| 7.0 | RESTRICTIONS |
| 8.0 | MISCELLANEOUS |
| 8.1 | ADDRESS/BANK RANGES IN OCTAL AND DECIMAL |
| 8.2 | EXECUTION TIME |
| 8.3 | PASS COUNT AND TEST NO. LOCATIONS |
| 8.4 | STACK POINTER |
| 8.6 | POWER FAIL |
| 9.0 | PROGRAM DESCRIPTION |
| 9.1 | NARRATIVE FLOW CHART |
| 9.2 | TEST TITLES TEST 0: TEST FOR PROPER BANK SELECTION TEST 1: CHECK DATI/DATO LINES TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION TEST 3: DUAL ADDRESS TEST A TEST 4: DUAL ADDRESS TEST B TEST 5: MARCHING 1'S AND 0'S TEST 6: CELLS' VOLATILITY TEST TEST 7: SHIFTING DIAGONAL TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST TEST 12: WORST CASE TESTING FOR CORE MEMORY TEST 13: WRITE RECOVERY TEST |
| 10.0 | RXDP & ACT11 & APT OPERATION |

SR.
450 / 177570 / 176

[1.0] ABSTRACT

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

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

[1.1] GETTING STARTED

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

TO START:

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

!CAUTION! BEFORE "DIGGING" INTO THE LISTING READ SECTION 9.

SWITCH SETTING SUMMARY (SEE SECTION 4.1 FOR DETAILS)

- | | |
|---------------|--|
| BIT15(100000) | HALT ON ERROR |
| BIT14(040000) | LOOP IN SUBTEST DEFINED BY BITS <3,0> |
| BIT13(020000) | INHIBIT ERROR PRINTOUTS |
| BIT12(010000) | ENABLE TESTING ABOVE 20K (MEMORY MANAGEMENT) |
| BIT11(004000) | ENABLE PARITY TESTING |
| BIT10(002000) | HALT AFTER EACH SUBTEST |
| BIT09(001000) | INHIBIT PROGRAM RELOCATION |
| BIT08(000400) | TYPE FIRST FAILING BIT ERROR PER 4K. |
| BIT07(000200) | ENABLE LONG GALLOPING TEST |
| BIT06(000100) | INHIBIT MEMORY SIZING |
| BIT05(000040) | INHIBIT "END PASS #XX" PRINTOUTS |
| BIT04(000020) | INHIBIT PRINTOUTS |
| BIT03-BIT00 | BEGINNING TEST NUMBER. |

[2.0] REQUIREMENTS

[2.1] EQUIPMENT

STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE
AND FROM 4K TO 124K OF MEMORY.

[2.2] STORAGE

PROGRAM STORAGE = 0000 - 7744. PROGRAM EXPANDS FOR ERROR
HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR.
(SEE SECTION 9, FOR DETAILS)

[3.0] LOADING PROCEDURE

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

[4.0] STARTING PROCEDURE

[4.1] SWITCH SETTINGS

SOFTWARE SWITCH REGISTER = LOCATION 176

BIT15(100000) HALT ON ERROR

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

BIT13(020000) INHIBIT ERROR PRINTOUTS

BIT12(010000) ENABLE TESTING ABOVE 28K. (MEMORY MANAGEMENT)

BIT11(004000) ENABLE PARITY MODULES.

! "PARITY" WILL BE TYPED

BIT10(002000) HALT AFTER EACH SUBTEST

! PRESS CONTINUE TO DO NEXT SUBTEST

BIT09(001000) INHIBIT PROGRAM RELOCATION

! IF SET LOCATIONS 430-7776 WILL NOT BE

! TESTED.

BIT08(000400) TYPE FIRST FAILING BIT IN EACH 4K BANK ONLY.

! THE TOTAL ERROR COUNT (UP TO 377) WILL

! BE SAVED IN THE ERROR HISTORY.

BIT07(000200) ENABLE LONG GALLOPING TEST.

! "GLP" WILL BE TYPED.

BIT06(000100) ! CAUTION! INCREASES TEST TIME BY FACTOR OF 25.

INHIBIT MEMORY SIZING.

! THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:

(VALUES TO TEST 0-8K ARE SHOWN)

(LOWTWO=LOCATION 322)

LOWTWO: 0

;STORE BITS 17:16 OF LOW TEST ADDRESS

LOWADD: 0

;STORE REST OF LOW TEST ADDRESS

HIGHTWO: 0 ;STORE BITS 17:16 OF HIGH TEST ADDRESS
HIGHADD: 37776 ;STORE REST OF HIGH TEST ADDRESS
NOTE: HIGHADD MUST BE SET TO A 4K BOUNDARY. (E.G. 37776)

BIT05(000040) INHIBIT "END PASS #XX" PRINTOUTS
BIT04(000020) A. INHIBIT ERROR HISTORY PRINTOUTS. THE
ERROR HISTORY CAN STILL BE OBTAINED
BY TYPING CONTROL-C.
B. INHIBIT PRINTOUTS "PARITY","GLP","TST13 BNK XX".
BIT03-BIT00 NUMBER OF TEST (0-13) TO RUN FIRST.
{NORMALLY USED WITH BIT14 (LOOP ON TEST)}

[4.2] CONTROL-C OPTION

CONTROL C [^C] AFTER COMPLETION OF THE CURRENT TEST,
THE ERROR HISTORY (SEE SEC. 6.3) WILL BE
TYPED. THE PROGRAM WILL HALT IN LOWER MEMORY.
PRESSING CONTINUE WILL RESTART THE DIAGNOSTIC.

[4.3]

STARTING ADDRESS= 200
RESTART ADDRESS = 250 OR 200
RESTART AT 200 CLEARS PASS COUNT (\$PASS) AND PRINTS "DZKMA-B" TITLE.

[4.4] PROGRAM AND/OR OPERATOR ACTION

- 1) LOAD PROGRAM INTO MEMORY USING ABSOLUTE LOADER.
- 2) SET OPTIONS (SEE SEC. 4.1)
- 3) START THE PROGRAM AT 200
- 4) THE FOLLOWING IS AN EXAMPLE WITH EXPLANATIONS
OF THE PRINTOUTS EXPECTED.

"XXXXX-YYYY" ;ADDRESSES OF TEST BOUNDARIES.
"PARITY" ;IF PARITY OPTION SELECTED
"GLP" ;IF LONG GALLOPING OPTION SELECTED.
;PRINTED AS TST11 IS ENTERED.
"TST13 BNK 00" ;ENTERING BANK 00 IN TEST 13.
"TST13 BNK 01" ;AND BANK 1...
ETC... ;UNTIL ALL BANKS (UP TO 6) HAVE BEEN TESTED.
"RELOC" ;THE DIAGNOSTIC RELOCATES TO HIGHEST
;BANK UNDER TEST. AND RUNS TST0-TST13 AGAIN.
"TST13 BNK 00" ;TESTING BANK 00 IN TEST 13 (RELOCATED STATE.)
;NOTE-ONLY BANK 00 IS TESTED IN THE RELOCATED STATE,
"END PASS #XX" ;WHERE "XX" IS THE PASS NO.

ADDITIONAL PRINTOUTS
"NO PAR" ;PRINTED IF PARITY SELECTED BUT NOT AVAILABLE.
"NO MNG" ;PRINTED IF GREATER THAN 28K AND NO MEMORY
;MANAGEMENT AVAILABLE.

4.5 LONG GALLOP OPTION

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

[5.0] PROGRAM HALTS (NORMAL+ ERROR)

THIS IS A LIST OF EXPECTED HALTS. IF THE TEST HALTS
IN A LOCATION NOT IN THIS LIST AND ITS LESS THAN 776, IT
MAY BE DUE TO A DEVICE INTERRUPTING.

NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS
MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND
BY SUBTRACTING 500 FROM 1664(SWHALT) AND ADDING THIS DIFFERENCE TO THE
CONTENTS OF SAVR6 [LOC. 346].

| PC | REASON | RECOVERY |
|------|---|--|
| -- | ----- | ----- |
| 112 | TRAP TO LOC. 4 | EXAMINE R6, IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED. |
| 146 | POWER FAIL | POWER UP WILL RECOVER IF IN CORE MEMORY. |
| 1666 | HALT AT END OF TEST SWITCH SET. | PRESS CONTINUE TO GO TO NEXT SUBTEST. |
| 6132 | HALT ON ERROR SWITCH SET. | PRESS CONTINUE. |
| 6216 | CONTROL=C TYPED OR FATAL ERROR OCCURRED | PRESS CONTINUE TO RE- START TEST. |

[6.0] ERRORS

[6.1] ERROR MESSAGE FORMAT

THE ERROR PRINTOUT CONSISTS OF 6 OCTAL WORDS IN THE FOLLOWING
FORMAT:

"LOCATION GOOD BAD PC ERROR PASFLG"

"ADR ERR" WILL BE PRINTED PRIOR IF AN ADDRESSING ERROR IS SUSPECTED.
"PAR ERR" WILL BE PRINTED PRIOR IF A PARITY ERROR TRAP OCCURRED
!CAUTION! IF PARITY ERROR THE GOOD DATA PRINTOUT IS THE
PARITY MODULE UNIBUS ADDRESS THAT FAILED.

WHERE:

LOCATION= FAILING MEMORY LOCATION
GOOD = GOOD DATA [DATA THAT WAS EXPECTED]
BAD = BAD DATA [DATA THAT WAS FOUND]
PC = PROGRAM COUNTER AT ERROR CALL.
ERROR = FAILING ERROR NO. (SEE SEC 6.2 - ERROR DICTIONARY)
PASFLG = CONTENTS OF LOCATION PASFLG, THIS MAY NOT BE RELEVANT.
(SEE SEC. 6.2-ERROR DICTIONARY)

!THE TEST WILL CONTINUE AFTER THE ERROR PRINTOUT.
!"NO MNG" WILL BE TYPED IF TESTING ABOVE 28K SELECTED AND NO MEMORY
!MANAGEMENT IS FOUND.

!"NO PAR" WILL BE TYPED IF PARITY OPTION SELECTED
!AND NO PARITY MODULES WERE FOUND.

(FATAL ERRORS)

"ERROR #XXXXXX" WILL BE TYPED WHERE "XXXXXX" IS
THE ERROR NUMBER. THE DIAGNOSTIC WILL USUALLY HALT ON THIS TYPE
OF ERROR. SEE SEC. 6.2 -ERROR DICTIONARY - FOR DESCRIPTIONS
OF THE ERROR.

(APT MODE ERRORS)

ALL ERRORS ARE TREATED AS FATAL UNDER APT. WHEN AN
ERROR OCCURS UNDER APT A "1" IS STORED IN LOCATION
\$MSGTY AND THE PROGRAM HALTS AT FATHLT.

\$FATAL CONTAINS THE ERROR NO. IN THE LOW BYTE AND
THE FAILING BANK NO. UNDER TEST IN THE HIGH BYTE.

[6.2] ERROR DICTIONARY

THIS IS A LIST OF ERROR NUMBERS PRINTED AND POSSIBLE
CAUSES FOR THE ERROR.

THE ROUTINE NAME WHERE THE ERROR CALL ORIGINATED IS GIVEN IN
BRACKETS.

NOTE- "BAKPAT" REFERS TO THE BACKGROUND PATTERN WRITTEN INTO MEMORY

FOR VARIOUS TESTS. IF PARITY SELECTED IT HAS A VALUE = 376 ,ELSE=377
"SWAPPED BAKPAT" = 77000 IF PARITY SELECTED, ELSE=77400

.ENDR

```
;ERROR # 0 ;[BUSER] BUS ERROR TRAP TO LOC. 4 OCCURRED  
; THIS ERROR IS NOT PRINTED AND IS FOR "APT" USE.  
  
;ERROR # 1 ;[TSTTRP]FATAL DATA ERROR  
;LOCATIONS 0000-430 FAILED 1'S + 0'S TEST.  
;R0 = GOOD DATA  
;R1 = ADDRESS OF FAILING LOCATION.  
  
;ERROR # 2 ;[APTSIZ] APT FATAL ERROR  
;APT MEMORY TABLES NOT SETUP CORRECTLY.  
;CHECK LOCATIONS $MAMS1 [430] TO $MADR4[446]  
; , FOR CORRECT MEMORY SIZE DATA.  
  
;ERROR # 3 ;[TSTSIZ] OPERATOR FATAL ERROR  
;SELECTED MEMORY SIZE GREATER THAN 28K, BUT  
;SR BIT12 (10000) NOT SET.  
;SET BIT12 AND RESTART AT 200.  
  
;ERROR # 4 ;[TSTSIZ] OPERATOR FATAL ERROR  
;LOWEST SELECTED TEST LIMIT IS HIGHER THAN  
;HIGHEST TEST LIMIT. SET LOCATIONS "LOWTWO"[322]  
;TO "HIGHADD" [330] CORRECTLY AND RESTART  
;AT 200.  
  
;ERROR # 5 ;[TST0] TEST SEQUENCE ERROR  
;TST0 HAS BEEN ENTERED OUT OF SEQUENCE  
;TESTN SHOULD = 00  
;THE DIAGNOSTIC HAS BEEN CORRUPTED.  
;IF POSSIBLE SELECT ANOTHER 4K BANK  
;BANK 0 AND RERUN THE TEST ON THE FAILING MEMORY.  
  
;ERROR # 6 ;[TST0] DUAL ADDRESSING ERROR  
;FOR THIS ERROR THE GOOD DATA PRINTED IS AN  
;ADDRESS. THIS IS THE ADDRESS SELECTED WHEN  
;THE SAME DATA WAS WRITTEN INTO THE FAILING  
;LOCATION. CHECK BANK SELECT CIRCUITRY  
  
;ERROR # 7 ;[TST0] ADDRESS AND DATA ERROR  
;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA  
;WRITTEN INTO THE FAILING LOCATION WAS IN  
;ERROR ALSO.  
  
;ERROR # 10 ;[TST0] DATA ERROR  
;IF BAD DATA = 0000 COULD BE AN ADDRESSING  
;ERROR , ELSE COMPARE GOOD AND BAD DATA FOR FAILING BITS.  
  
;ERROR # 11 ;[TST0] ADDRESSING ERROR  
;THE FAILING ADDRESS RESPONDED BUT IS NON-  
;EXISTENT. MAY BE A DUAL ADDRESSING PROBLEM.
```



```
;ERROR # 12      ;[TST1] TEST SEQUENCE ERROR
                  ;$TEST [404] SHOULD = 01
                  ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 13      ;[TST1] DATA ERROR
                  ;COMPARE GOOD AND BAD PRINTED DATA, FAILING
                  ;DATA BITS MAY SHORTED OR SWAPPED.

;ERROR # 14      ;[TST2] TEST SEQUENCE ERROR
                  ;$TESTN [404] SHOULD = 02
                  ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 15      ;[TST2] ADDRESS OR DATA ERROR
                  ;IF "ADR ERR" NOT PRINTED THEN THE BYTE SELECT
                  ;CIRCUITRY PROBABLY FAILED.

;ERROR # 16      ;[TST3] TEST SEQUENCE ERROR
                  ;$TESTN [404] SHOULD = 03
                  ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 17      ;[TST3] DUAL ADDRESSING ERROR
                  ;DUAL ADDRESSING PROBLEM FOR BITS THAT DIFFER
                  ;IN GOOD AND BAD DATA PRINTOUT.

;ERROR # 20      ;[TST3] DUAL ADDRESSING ERROR
                  ;FOR THIS ERROR THE DATA PRINTED IS AN ADDRESS.
                  ;THIS IS THE ADDRESS THAT WAS SELECTED WHEN THE
                  ;SAME DATA WAS WRITTEN INTO THE FAILING LOCATION.

;ERROR # 21      ;[TST3] DUAL ADDRESSING ERROR
                  ;SAME AS ERROR #20 EXCEPT DIFFERENT DATA
                  ;(SWAPPED BAKPAT) WAS WRITTEN.

;ERROR # 22      ;[TST4] TEST SEQUENCE ERROR
                  ;$TESTN [404] SHOULD = 04.
                  ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 23      ;[TST4] DUAL ADDRESSING ERROR
                  ;IF PASFLG = 0 THEN THE FAILING LOCATION
                  ;AND FAILING DATA ARE DUAL ADDRESSES.

;ERROR # 24      ;[TST5] TEST SEQUENCE ERROR
                  ;$TESTN [404] SHOULD = 05
                  ;      THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 25      ;[TST5] DATA ERROR
                  ;DATA WRITE OR READ ERROR.
;ERROR # 26      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
                  ;IF PASFLG=0  FAILED MARCHING 1'S + 0'S IN
                  ;      MAX TO MIN DIRECTION.
                  ;IF PASFLG=1  FAILED MARCHING 1'S + 0'S IN
                  ;      MIN TO MAX DIRECTION
                  ;IF PASFLG=3  FAILED MARCHING 0'S + 1'S IN
                  ;      MAX TO MIN DIRECTION.

;ERROR # 27      ;[TST5] MARCHING 1'S AND 0'S DATA ERROR
```

```
      ; IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA IS  
      ; CHECKED IMMEDIATELY AFTER BEING WRITTEN.  
  
;ERROR # 30      ; [TST6] TEST SEQUENCE ERROR  
                ; $TESTN SHOULD = 06  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 31      ; [TST6] VOLATILITY/REFRESH TEST ERROR  
                ; IF PASFLG=0 BAKPAT WRITE OR READ ERROR.  
                ; IF PASFLG=1 THE FAILING LOCATION CHANGED WHILE  
                ; ANOTHER LOCATIONS WAS WRITTEN FOR  
                ; 2 MS. THE OTHER LOCATION IS SAVED  
                ; IN SAVLOC [352]  
                ; IF PASFLG=2 SWAPPED BAKPAT (77400 OR 77000)  
                ; WRITE OR READ ERROR.  
                ; IF PASFLG=3 SAME AS IF PASFLG=2 EXCEPT  
                ; THE DATA IS SWAPPED BAKPAT.  
  
;ERROR # 32      ; [TST7] TEST SEQUENCE ERROR  
                ; $TESTN SHOULD = 07  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 33      ; [TST7] SHIFTING DIAGONAL DATA ERROR  
                ; IF PASFLG=0 BAKPAT WRITE OR READ ERROR.  
                ; IF PASFLG=1 BAKPAT READ CHECK ERROR  
                ; IF PASFLG= GREATER THAN 1 BUT EVEN VALUE THEN;  
                ; THE FAILING LOCATION COULD NOT BE WRITTEN INTO.  
                ; IF PASFLG= GREATER THAN 1 BUT ODD VALUE THEN;  
                ; THE FAILING LOCATION WAS WRITTEN CORRECTLY  
                ; BUT LOST THE DATA.  
  
;ERROR # 34      ; [TST10] TEST SEQUENCE ERROR  
                ; $TESTN SHOULD = 10  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 35      ; [TST10] BAKPAT DATA ERROR  
                ; BAKPAT WRITE OR READ ERROR INTO THE FAILING LOCATION,  
  
;ERROR # 36      ; [TST10] READ RECOVERY DATA ERROR  
                ; THIS ERROR CAN BE REPORTED BY TST10 AND TST11,  
                ; (THEY SHARE CODE). SEE $TESTN [404] FOR WHICH TEST FAILED.  
                ; FOR BOTH TESTS COMPARE THE GOOD AND BAD DATA AT THE FAILING  
                ; LOCATION TO SEE WHICH BITS FAILED.  
  
;ERROR # 37      ; [TST10] READ RECOVERY DATA ERROR  
                ; IDENTICAL TO THE PREVIOUS ERROR EXCEPT SWAPPED BAKPAT IS  
                ; USED AS WRITE AND READ DATA.  
  
;ERROR # 40      ; [TST11] TEST SEQUENCE ERROR  
                ; $TESTN SHOULD = 11  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 41      ; [TST12] TEST SEQUENCE ERROR  
                ; $TESTN SHOULD = 12  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
```

```
;ERROR # 42 ;[TST12] WORST CASE CORE TEST DATA ERROR
;IF PASFLG=1 COMPARE GOOD AND BAD DATA FOR FAILING BITS,
;IF PASFLG=2 THE FAILING LOCATION WAS WRITTEN AND READ
; WITH GOOD DATA,BUT FAILED READ CHECK
; READING IN THE MIN. TO MAX DIRECTION,
;IF PASFLG=3 SAME CONDITIONS AS PASFLG=2 EXCEPT FAILED
; DOING THE READ CHECK FROM MAX TO MIN DIRECTION.

;ERROR # 43 ;[TST12] WORST CASE CORE TEST DATA ERROR
; IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA WRITTEN
;AND READ IS COMPLEMENTED.

;ERROR # 44 ;[TST13] TEST SEQUENCE ERROR
;TESTN SHOOULD = 13
; THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 45 ;[TST13] WRITE RECOVERY TEST DATA ERROR
;IF PASFLG=0 COMPARE GOOD AND BAD DATA FOR FAILING BITS,
;IF PASFLG=77400 DATA ERROR FOUND WHILE DOING A SECOND READ CHECK.
;IF PASFLG=77402 DATA ERROR FOUND IN FAILING LOCATION AFTER
; SMALL TEST PROGRAM RUN IN FAILING BANK.

;ERROR # 46 ;[TST13] WRITE RECOVERY TEST DATA ERROR
; DATA ERROR FOUNDJUST BEFORE THE SMALL TEST
;WAS TO BE RUN IN THE FAILING BANK, TO AVOID "BLOWING" UP
;WHEN THE SMALL TEST IS RUN TST13 IS ABORTED.

;ERROR # 47 ;[TST13] WRITE RECOVERY TEST DATA ERROR
; IDENTICAL TO ERROR #XXX EXCEPT THE DATA WRITTEN
;AND READ IS DIFFERENT,(177667).
;177667 IS THE COMPLEMENT OF "JMP (R0)" (110) WHICH IS
;THE ESCAPE FROM THE SMALL TEST PROGRAM RUN IN THE BANK
;UNDER TEST.

;ERROR # 50 ;[PARERR] PARITY TRAP ERROR
; PARITY TRAP TO 114 OCCURRED.
;FOR THIS ERROR PRINTOUT THE "GOOD DATA" IS ACTUALLY
;THE FAILING PARITY MODULE UNIBUS ADDRESS.
; SAVLOC [352] CONTAINS THE PC WHERE THE TRAP OCCURRED.

;ERROR # 51 ;[PARITY] PARITY TRAP FATAL ERROR
; A PARITY TRAP TO 114 OCCURRED, BUT NO PARITY MODULES COULD BE FOUND
;WITH AN ERROR BIT (BIT15) SET.

;ERROR # 52 ;[NOMM] OPERATOR FATAL ERROR
; TESTING ABOVE 28K WAS SELECTED, BUT NO MEMORY MANAGEMENT
;OPTION WAS FOUND.
; RESET SWITCH OPTIONS AND RESTART AT 200.

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

,REPT 0
```

[6.3] ERROR HISTORY

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

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

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

ERROR HISTORY FORMAT:

| ERROR | BANK | COUNT |
|-------|-------|-------|
| ----- | ----- | ----- |

WHERE:

ERROR = BIT THAT FAILED (NUMBER OF THE FAILING BIT IN DECIMAL I.E. 0-15 WILL BE TYPED OUT OR THE WORDS "ADR ERR" OR "PAR ERR" WILL BE TYPED OUT IF ADDRESS ERROR OR PARITY ERROR WAS SEEN IN THE SPECIFIC BANK OF MEMORY)

BANK = 4K MEMORY BANK IN WHICH THIS FAILURE WAS SEEN
A 0 FOR 0 TO 4K, A 1 FOR 4 TO 8K AND SO ON

COUNT = NUMBER OF TIMES THIS MEMORY BANK FAILED.
(377 IS MAXIMUM FAILURE COUNT RECORDED.)

[6.4] ERROR RECOVERY

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

[7.0] RESTRICTIONS

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

[8.0] MISCELLANEOUS

[8.1] ADDRESS/BANK RANGES IN OCTAL AND DECIMAL

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

| BANK NO. | DECIMAL RANGE | OCTAL RANGE | [PAGE ADDRESS REGISTER] | | UNIBUS ADDRESS |
|----------|---------------|---------------|-------------------------|------|----------------|
| | | | USED/CONTENT | | |
| 0 | 0 - 4K | 000000-017776 | 0 | 0000 | 772340 |
| 1 | 4K - 8K | 020000-037776 | NOT USED | | |
| 2 | 8K-12K | 040000-057776 | NOT USED | | |
| 3 | 12K-16K | 060000-077776 | NOT USED | | |
| 4 | 16K-20K | 100000-117776 | NOT USED | | |
| 5 | 20K-24K | 120000-137776 | NOT USED | | |
| 6 | 24K-28K | 140000-157776 | NOT USED | | |
| 7 | 28K-32K | 160000-177776 | 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.

[8.2] EXECUTION TIME

HERE ARE SOME TYPICAL EXECUTION TIMES.

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

[8.2] PASS COUNT AND TEST NO. LOCATIONS

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

\$TESTN [404] = CURRENT TEST NO. AND RELOCATION, PARITY FLAGS.
WHERE;
LOW BYTE = TEST NO.
IF BIT15 = 1 TEST IS RELOCATED
IF BIT13 = 1 PARITY UNDER TEST.

[8.4] STACK POINTER

THE STACK STARTS AT 500 WHEN THE PROGRAM IS NOT RELOCATED.
SAVR6[346] CONTAINS THE STACK STARTING VALUE WHEN THE DIAGNOSTIC
IS RELOCATED.

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

[8.5] POWER FAIL

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

[9.0] PROGRAM DESCRIPTION

[9.1] NARRATIVE FLOW CHART

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

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

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

1. [START] PRINT "DZKMA-A" 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. ENABLE MEMORY MANAGEMENT AND SIZE MEMORY ABOVE 20K.
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 10 BYTES ARE SAVED IN THE FOLLOWING FORMAT:

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

IF GREATER THAN 4K UNDER TEST THE ABSOLUTE LOADER (300 ADDRESSES) IS APPENDED. IF GREATER THAN 4K AND UNDER XXDP CHAIN MODE 5674 (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. IF SR=2000 THEN HALT
IF SR=4000 THEN LOOP ON TEST DEFINED BY <3:0>

ELSE CONTINUE TO NEXT TEST.

12. [TST1-TST12] EXECUTE TST1-TST12 EACH TIME GOING TO STEP 9.
13. [TST13] TEST 13 IS DIFFERENT FROM TESTS 0-12, BECAUSE IT IS A SMALL PROGRAM ACTUALLY RUNNING IN THE MEMORY UNDER TEST. BEFORE THIS SMALL PROGRAM IS STARTED "TST13 BNK XX" IS TYPED. THIS IS DONE IN CASE THE PROGRAM FAILS. THE USER CAN THEN AT LEAST TELL WHICH BANK OF MEMORY FAILED.
14. [RELOC] THE PROGRAM RELOCATES TO HIGH MEMORY TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG). WHERE "ENDPRG" IS THE CONTENTS OF ENDSTK[306]. I.E THE LAST PROGRAM ADDRESS. NOTE "RELOC" IS PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
15. TESTS 0-13 ARE RUN AS DESCRIBED ABOVE EXCEPT ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED.
16. [RELOER] RELOCATE THE PROGRAM BACK TO LOWER MEMORY.
17. [LOWER] IF CONTROL-C TYPED GO PRINT ERROR HISTORY.
18. [TSTMM] IF MEMORY MANAGEMENT SELECTED AND AVAILABLE, RUN TESTS 0-13 ON THE FIRST 24K SLICE ABOVE 28K.
19. [CONTMM] CALL "UPMM" TO UPDATE MEMORY MANAGEMENT PAR REGISTERS TO POINT TO THE NEXT 24K SLICE OF UPPER MEMORY.
20. [MAXADR] REPEAT STEPS 18 + 19 UNTIL ALL MEMORY ABOVE 28K IS TESTED.
21. [ENDPAS] PRINT ERROR HISTORY OF FAILING BITS
22. [\$EOP] DISABLE PARITY MODULES. PRINT "END PASS #XX"

[9.2] TEST TITLES

SEE THE TEST HEADINGS IN THE LISTING FOR DETAILS ON EACH TEST.

TEST 0: TEST FOR PROPER BANK SELECTION
TEST 1: CHECK DATI/DATO LINES
TEST 2: TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
TEST 3: DUAL ADDRESS TEST A
TEST 4: DUAL ADDRESS TEST B
TEST 5: MARCHING 1'S AND 0'S
TEST 6: CELLS' VOLATILITY TEST

TEST 7: SHIFTING DIAGONAL
TEST 10: READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
TEST 11: READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
TEST 12: WORST CASE TESTING FOR CORE MEMORY
TEST 13: WRITE RECOVERY TEST

[10,0] RXDP & ACT11 & APT OPERATION

RXDP CHAIN MODE

OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:

1. NO "DZKMA-B" TITLE IS PRINTED.
2. NO TEST 13 PRINTOUTS SUCH AS "TST13 BNK 00".
3. THE PROGRAM ALWAYS HALTS ON ERROR.
4. AT THE END OF TEST (%ENDAD) CONTROL IS RETURNED TO THE RXDP CHAIN MONITOR VIA LOCATION 42.

ACT11

OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:

1. NO PRINTOUTS EXCEPT ERROR PRINTOUTS.
2. THE PROGRAM ALWAYS HALTS ON ERROR.
3. AT THE END OF TEST (%ENDAD) CONTROL IS RETURNED TO THE ACT11 MONITOR VIA LOCATION 42.

APT

OPERATION IS SIMILAR TO STAND ALONE EXCEPT:

1. THE SOFTWARE SWITCH REGISTER BECOMES LOCATION 422 (%SWREG).
2. AUTO SIZING CAN BE INHIBITED BY SETTING BIT 7 OF BYTE LOCATION 421 (%ENVM).
3. ALL PRINTOUTS CAN BE INHIBITED BY SETTING BIT 5 OF BYTE LOCATION 421 (%ENVM).
4. ALL ERRORS CAUSE LOCATION 400 (%MSGTY) TO BE SET = 0001 AND THE PROGRAM HALTS AT LOCATION 6214 (FATHLT). LOCATION 402 (%FATAL) CONTAINS THE ERROR NO. IN THE LOW BYTE AND THE FAILING MEMORY BANK NO. IN THE HIGH BYTE.

APT MANAGER INFORMATION

THE FOLLOWING IS AN EXAMPLE SCRIPT TO TEST A 4K MEMORY.
IT IS RECOMMENDED THAT DIFFERENT SCRIPTS BE USED FOR
DIFFERENT MEMORY SIZES TO SAVE AUTO SIZING TIME.

THE EXAMPLE ASSUMES YOU ARE LOGGED INTO THE APT MONITOR.

READY

RUN APPLU
APT 11 PAPER TAPE PROGRAM LOAD UTILITY

THE FOLLOWING COMMANDS ARE VALID

ED EDIT A PROGRAM
LI LIST A PROGRAM

COMMAND: ED
PROGRAM NAME TO EDIT: EXAMPL
DO YOU WANT TO LOAD A NEW REV OF THE PROGRAM(Y/N)? N
FIRST PASS RUN TIME IN SECONDS <110>:
LONGEST TEST TIME IN SECONDS <10>:
ADDITIONAL RUN TIME IN SECONDS <0>:
WHICH ETABLE DO YOU WISH TO EDIT? A
SOFTWARE ENVIRONMENT<000>: 1
ENVIRONMENTAL MODE<000>: 240
SWITCH 1 <000000>:
SWITCH 2 <000000>:
CPU OPTIONS<0000>:
MEMORY TYPE 1 <000>:
MAXIMUM ADDRESS<00000000>:
MEMORY TYPE 2 <000>:
MAXIMUM ADDRESS<00000000>:
MEMORY TYPE 3 <000>:
MAXIMUM ADDRESS<00000000>:
MEMORY TYPE 4 <000>: 1
MAXIMUM ADDRESS<00000000>: 17776
WHICH ETABLE DO YOU WISH TO EDIT?
COMMAND: OFF

.ENDR

```

,ABS
,LIST MD,MC,CND
1013 .LIST ME,BIN,SEQ,LOC
1018 .TITLE DZKMA
(1) ;*COPYRIGHT (C) JANUARY 1976
(1) ;*DIGITAL EQUIPMENT CORP.
(1) ;*MAYNARD, MASS, 01754
(1) ;*
(1) ;*PROGRAM BY PERVEZ ZAKI
(1) ;*
(1) ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1) ;*PACKAGE (MAINDEC-11-DZQAC-C1),MAR 24, 1976.
(1) ;*
(1) 160000 $$SWR=160000 ;,HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
1019
1020
1021
1022
1054 ;,TRAP CATCHER OF ,+2 AND HALT FOR 0-776 LOCATIONS
1055
1063
1064
1071
1072 000240 SCOPE =NOP
1073
1074 000042 ,=42
1075 000042 000000 ,WORD 0 ;FOR ACT/XXDP
1076
1077 .SBTTL ACT11 HOOKS
(1) ;*****
(2) ;HOOKS REQUIRED BY ACT11
(1) ;$SVPC=, ;SAVE PC
(1) 000044 ,=46
(1) 000046 000156 $ENDAD ;,1)SET LOC,46 TO ADDRESS OF $ENDAD IN ,EOP
(1) 000052 ,=52
(1) 000052 040000 ,WORD 40000 ;,2)SET LOC,52 TO 40000
(1) 000044 ,=$SVPC ;, RESTORE PC
1078
1079 000070 ,=70
1080 000070 012737 000136 000024 PWRDN: MOV $PWRUP,0*24
1081 000076 000000 HALT
1082

```

```

1084
1085
1086 000104 ,=104
1087 ; GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
1088 000104 005237 000400 BUSER: INC @#$MSGTY ;TELL APT FATAL ERROR#000
1089 000110 000000 HALT ;*ERROR* TRAP TO LOC. 4 OCCURRED.
1090 000112 000000 HALT ;IN CASE CONTINUE PRESSED.
1091 ;,114 AND ,116 ARE RESERVED FOR PARITY TRAP VECTORS, SETUP IN
1092 ;ROUTINE "BEGIN".
1093 000120 ,=120
1094
1095
1099
1100 ;* WRITE MEMORY BACKGROUND
1101 ;* -----
1102 ;*
1103 ;* THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
1104 ;* THE VALUE STORED AT LOCATION BAKPAT, THE ROUTINE ASSUMES
1105 ;* THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE
1106 ;* HIGHEST LOCATION TO BE WRITTEN, THE PROGRAM LEAVES THE
1107 ;* SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
1108 ;*
1109
1110 WRTMEM: MOV R4,R1 ;SET R1 TO LOWEST LOCATION UNDER TEST
1111 000122 013700 000316 MOV @#BAKPAT,R0 ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
1112 000126 010021 2S: MOV R0,(R1)+ ;STARTING FROM THE LOWEST LOCATION WRITE THE
1113 000130 020105 CMP R1,R5 ;MEMORY TO BACK GROUND PATTERN
1114 000132 103775 BLO 2S
1115 000134 000207 RTS PC ;RETURN FROM THE SUBROUTINE
1116
1117
1118 PWRUP: MOV @#SAVR6,SP ;RESTORE STACK POINTER
1119 000142 012700 006066 MOV $PNTMES-BEGIN,R0
1120 000146 060600 ADD SP,R0 ;GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
1121 ;RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE
1122 000150 004710 JSR PC,(R0) ;GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A "P"
1123 000152 000120 ,ASCIZ /P/
1124 ,EVEN
1125 BR START
1126 000154 000411
1127
1128 ;* SERVICE XXDP/ACT11
1129 000156 004710 $ENDAD: JSR PC,(R0) ;RETURN TO ACT11/XXDP MONITOR
1130 000160 000240 NOP ;IF QUICK VERIFY=RESET ELSE NOP
1131 000162 000240 NOP ;IF QUICK VERIFY=CLR #-1 ELSE INC #0
1132 000164 000240 NOP ;IF QUICK VERIFY=BR ,-4 ELSE NOP
1133 000166 000430 BR RESTRT ;REPEAT TEST UNDER ACT11/XXDP
1134
1135 ,=176
1136 000176 000000 SWREG: ,WORD 0
1137
1138
1139 ;*****
1140 .SBTTL START AND RESTART ROUTINES
1141 ;* RESTART AT 200 TO CLEAR APT TABLES
1142 ;*****

```

```

1143 000200 013706 000350   START:  MOV   @#SAVR6,SP   ;SETUP STACK POINTER,
1144 000204 012703 000412     MOV   #UNIT,R3     ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1145 000210 005043           18:   CLR   -(R3)         ;CLEAR A MAILBOX LOCATION
1146 000212 022703 000400     CMP   #MAIL,R3    ;DONE?
1147 000216 001374           BNE   18          ;BRANCH IF NO
1148 000220 105737 000042     TSTB  @#42        ;ACT11 MODE?
1149 000224 001011           BNE   RESTRT      ;BRANCH IF YES
1150 000226 105737 000405     TSTB  @#TESTN+1  ;ARE WE RELOCATED?
1151 000232 100406           BMI   RESTRT      ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
1152 000234 004767 006320     JSR   PC,TPCRLF  ;PRINT TITLE
1153 000240 055104 046513 026501 .ASCIZ /DZKMA-B/
1154 000246 000102           .EVEN
1154
1155
1156 000250 012704 007744   RESTRT: MOV   $ENDPRG,R4 ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1157 000254 012703 000346     MOV   $SAVR5,R3   ;CAUSE R3 TO POINT TO THE LOCATION SAVR5
1158 000260 012305           MOV   (R3)+,R5    ;RESTORE R5
1159 000262 012306           MOV   (R3)+,SP    ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1160 000264 010600           MOV   SP,R0       ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1161 000266 012746 000340     MOV   #340,-(SP) ;SET HIGH PRIORITY FOR RTI
1162 000272 010046           MOV   R0,-(SP)
1163 000274 000002           RTI              ;GO TO "START"-MAY BE RELOCATED,
1164                                     ;IF RELOCATED SEE LOCATION SAVR6 FOR START.
1165
1166
1167
1168
1169
1170
1171
1172
1173                                     .SBTTL  APT PARAMETER BLOCK
(1)
(2)                                     ;;*****
(1)                                     ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(2)                                     ;;*****
(1)                                     .SX=.           ;;SAVE CURRENT LOCATION
(1)                                     .=24           ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024 000200           200           ;;FOR APT START UP
(1)                                     .=44           ;;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044 000276           $APTHDR      ;;POINT TO APT HEADER BLOCK
(1)                                     .=.SX         ;;RESET LOCATION COUNTER
(2)                                     ;;*****
(1)                                     ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(1)                                     ;INTERFACE SPEC.
(1)
(1) 000276           $APTHD:
(1) 000276 000000   $SHBITS: .WORD 0           ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 000300 000400   $MBADR:  .WORD $MAIL      ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 000302 000012   $STMT:   .WORD 10        ;;RUN TIM OF LONGEST TEST
(1) 000304 000156   $PASTM:  .WORD 110       ;;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=$ETABLE(WORDS)
1174
1175
1176 000405           REL=$TESTN+1       ;IT WILL BE 0 IF THE PROGRAM IS IN THE LOWER

```

```

1177                                     ;CORE, BIT 6 OF THE BYTE WILL BE SET IF THE
1178                                     ;PROGRAM IS IN A RELOCATED STATE AND BIT 5
1179                                     ;WILL BE SET IF PARITY BITS ARE BEING TESTED
1180                                     .=$APTHD
1181 000276           MMAVA:  ;THIS BYTE IS USED TO DETERMINE IF MEMORY
1182                                     ;MANAGEMENT IS AVAILABLE OR NOT
1183
1184                                     .=$MMAVA+1
1185 000277           TYPENB: ;THIS BYTE IS USED TO DETERMINE IF THE
1186                                     ;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
1187
1188                                     .=$TYPENB+1
1189 000300           $PRERR: ;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
1190                                     ;A PARITY ERROR
1191
1192                                     .=$PRERR+1
1193 000301           $ADERR: ;THIS BYTE IS USED TO DETERMINE IF THE
1194                                     ;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
1195
1196                                     .=$ADERR+1
1197 000302           STRTDI:
1198                                     .=$STRTDI+2
1199 000304           LOWBNK:
1200                                     .=$LOWBNK+2
1201 000306           PASFLG: ;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
1202                                     ;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
1203                                     ;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
1204
1205                                     .=$PASFLG+2
1206 000310           ENDSTK:
1207 000312           .=$ENDSTK+2
1208 000312           PBNK:   ;HOLDS BANK UNDER TEST FOR "TST BNK XX" PRINTOUT,
1209 000312           DECWRD:
1210                                     .=$DECWRD+2
1211 000314           TYPCNT: .BYTE 0           ;THIS BYTE DETERMINES THE NUMBER OF WORDS
1212                                     ;TO BE TYPED
1213 000315           SAVKBB: .BYTE 0           ;THIS LOCATION IS USED TO SAVE THE CHARACTER
1214                                     ;HIT BY THE OPERATOR
1215                                     ;ALSO IS USED AS TEMP IN ROUTINE $GTSIZ,
1216
1217                                     .EVEN
1218
1219 177560           TKS=   177560
1220 177562           $KBB=  177562
1221 177564           $TPS=  177564
1222 177566           $TPB=  177566
1223 177572           $R0=   177572
1224 000316 000377   BAKPAT:  .WORD 377       ;BACKGROUND PATTERN WRITTEN TO MEMORY.
1225
1226 000320 000000   SWAPAT:  .WORD
1227 000322 000430   RELBOT:  BEGIN=50      ;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED,
1228
1229                                     ;;*****
1230                                     ;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR
1231 000324 000000   LOWTWO: 0             ;HOLDS BITS 17:16 OF LOW TEST ADDRESS
1232 000326 000000   LOWADD: 0             ;HOLDS BITS 15:0 OF LOW TEST ADDRESS

```

```
1233
1234 000330 000000 HIGHTWO: 0 ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
1235 000332 037776 HIGHADD: 37776 ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1236 ;*****
1237
1238 000334 000000 $HIMAX: 0 ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1239 000336 017776 $MAXM: 17776 ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1240
1241 000340 000000 MAXMEM: ,WORD ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1242
1243 000342 000000 SAVMAX: ,WORD
1244 000344 000000 SAVR4: ,WORD
1245 000346 000000 SAVR5: ,WORD
1246
1247 ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1248 000350 000500 SAVR6: ,WORD BEGIN ;CONTAINS START ADDRESS WHEN RELOCATED ALSO,
1249 000352 000000 FARMAP: 0 ;MAP OF PARITY MODULES UNDER TEST,
1250 000354 000000 SAVLOC: 0 ;TEST 6 STORES ERROR INFO HERE
1251 000356 000000 PARSP: 0 ;SAVE SP DURING PARITY ERROR TRAP,
1252 000360 000000 PARPS: 0 ;SAVE PSW DURING PARITY ERROR TRAP,
;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE,
;IN THIS CRUDE FASHION,
1253
1254
1255
1256
1263
1264
1265 ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
```

```
1267 000400 ,=400
1268 ,SBTTL APT MAILBOX-ETABLE
(1) ;*****
(1) $EVEN
(1) 000400 $MAIL: ;APT MAILBOX
(1) 000400 000000 $MSGTY: ,WORD AMSGTY ;MESSAGE TYPE CODE
(1) 000402 000000 $FATAL: ,WORD AFATAL ;FATAL ERROR NUMBER
(1) 000404 000000 $TESTN: ,WORD ATESTN ;TEST NUMBER
(1) 000406 000000 $PASS: ,WORD APASS ;PASS COUNT
(1) 000410 000000 $DEVCT: ,WORD ADEVCT ;DEVICE COUNT
(1) 000412 000000 $UNIT: ,WORD AUNIT ;I/O UNIT NUMBER
(1) 000414 000000 $MSGAD: ,WORD AMSGAD ;MESSAGE ADDRESS
(1) 000416 000000 $MSGLG: ,WORD AMSGLG ;MESSAGE LENGTH
(1) 000420 $ETABLE: ;APT ENVIRONMENT TABLE
(1) 000420 000 $ENV: ,BYTE AENV ;ENVIRONMENT BYTE
(1) 000421 000 $ENVM: ,BYTE AENVM ;ENVIRONMENT MODE BITS
(1) 000422 000000 $SWREG: ,WORD ASWREG ;APT SWITCH REGISTER
(1) 000424 000000 $USWR: ,WORD AUSWR ;USER SWITCHES
(1) 000426 000000 $CPUOP: ,WORD ACPUOP ;CPU TYPE,OPTIONS
(1) ;*
(1) ;* 11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(1) ;* 11/70=06,PDQ=07,Q=10
(1) ;* BIT 10=REAL TIME CLOCK
(1) ;* BIT 9=FLOATING POINT PROCESSOR
(1) ;* BIT 8=MEMORY MANAGEMENT
(1) 000430 000 $MAMS1: ,BYTE AMAMS1 ;HIGH ADDRESS,M,S, BYTE
(1) 000431 000 $MTYP1: ,BYTE AMTYP1 ;MEM,TYPE,BLK#1
(1) ;*
(1) ;* MEM,TYPE BYTE -- (HIGH BYTE)
(1) ;* 900 NSEC CORE=001
(1) ;* 300 NSEC BIPOLAR=002
(1) ;* 500 NSEC MOS=003
(1) 000432 000000 $MADR1: ,WORD AMADR1 ;HIGH ADDRESS,BLK#1
(1) ;*
(1) ;* MEM, LAST ADDR,=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
(1) 000434 000 $MAMS2: ,BYTE AMAMS2 ;HIGH ADDRESS,M,S, BYTE
(1) 000435 000 $MTYP2: ,BYTE AMTYP2 ;MEM,TYPE,BLK#2
(1) 000436 000000 $MADR2: ,WORD AMADR2 ;MEM, LAST ADDRESS, BLK#2
(1) 000440 000 $MAMS3: ,BYTE AMAMS3 ;HIGH ADDRESS,M,S, BYTE
(1) 000441 000 $MTYP3: ,BYTE AMTYP3 ;MEM,TYPE,BLK#3
(1) 000442 000000 $MADR3: ,WORD AMADR3 ;MEM, LAST ADDRESS, BLK#3
(1) 000444 000 $MAMS4: ,BYTE AMAMS4 ;HIGH ADDRESS,M,S, BYTE
(1) 000445 000 $MTYP4: ,BYTE AMTYP4 ;MEM,TYPE,BLK#4
(1) 000446 000000 $MADR4: ,WORD AMADR4 ;MEM, LAST ADDRESS, BLK#4
(1) 000450 $ETEND:
(1) ,MEXIT
1269
1270 ;*****
1271 ,SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES,
```

```

1273 ;*****
1274 000450 177570 SWR: 177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
1275
1276 ;=500
1277 000500 010706 BEGIN: MOV PC,SP ;SET UP STACK POINTER TO EQUAL BEGIN ADDRESS
1278
1279 000502 005746 TST -(SP)
1280 000504 010637 000350 MOV SP,##SAVR6 ;SAVE SP FOR FUTURE USE
1281 000510 012737 000070 000024 MOV #PWRDN,0#24 ;PREPARE FOR ANY FUTURE POWER DOWN
1282 000516 005037 000300 CLR ##PRERR
1283 000522 005037 000314 CLR ##TYPCNT
1284 000526 012700 000114 MOV #114,R0 ;PREPARE TO SETUP PARITY TRAP VECTOR
1285 000532 012710 005456 MOV #PARERR=-,6,(R0)
1286 000536 060720 ADD PC,(R0)+ ;TO PARERR
1287 000540 012710 000340 MOV #340,(R0) ;AND PSW OF 340
1288 000544 105737 000405 TSTB #REL ;IS THIS CODE RELOCATED?
1289 000550 100002 BPL ONEPAS ;BRANCH IF NO
1290 000552 000167 000542 JMP TSTREL ;THIS CODE IS RELOCATED SO GET TEST SIZE,
1291
1292 000556 005737 000406 ONEPAS: TST ##PASS ;IS THIS THE FIRST PASS?
1293 000562 001402 BEQ TSTRP ;BRANCH IF YES (TEST TRAP CATCHER ADDRESSES)
1294 000564 000167 000374 JMP SETSTK ; GET THE TEST SIZE
1295 000570 012704 000744 TSTRP: MOV #ENDPRG ,R4 ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1296 000574 012700 000377 MOV #377,R0
1297 000600 010037 000316 MOV R0,##BAKPAT
1298 000604 005001 CLR R1
1299 000606 012124 28: MOV (R1)+,(R4)+ ;SAVE FROM 0000 TO BEGIN=30 AT END OF PROGRAM FOR NOW
1300 000610 020127 000400 CMP R1,##MAIL
1301 000614 103774 BLO 28
1302 000616 005741 38: TST -(R1) ;PREPARE TO TEST THE TRAP VECTORS
1303 000620 010011 48: MOV R0,(R1) ;CHECK THE TRAP VECTORS FOR THE CAPABILITY
1304 ;OF HOLDING 0'S & 1'S
1305 000622 020011 CMP R0,(R1) ;IS THE DATA OK?
1306 000624 001403 BEQ 68 ;BRANCH IF YES
1307
1308 000626 004767 005304 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1309 (1) 000632 000001 1 ;*****ERROR NUMBER 1*****
1310
1311 000634 000300 68: SWAB R0
1312 000636 001370 BNE 48
1313 000640 005701 TST R1 ;IF WE HAVE NOT REACHED THE LOWEST MEMORY LOCATION
1314 000642 001365 BNE 38 ;THEN REPEAT FROM 38
1315 000644 012701 000400 MOV ##MAIL,R1
1316 000650 014441 88: MOV -(R4),-(R1) ;RESTORE TRAP CATCHER ETC.
1317 000652 005701 TST R1
1318 000654 001375 BNE 88
1319 000656 012700 000006 SETSWR: MOV #6,R0
1320 000662 012710 000340 MOV #340,(R0) ;SET UP TIME OUT TRAP PSW
1321 000666 012740 000700 MOV #48,-(R0) ;AND THE RETURN ADDRESS
1322 000672 005777 177552 28: TST #SWR ;DOES THE SWITCH REGISTER POINTED BY SWR EXIST ?
1323 000676 000404 BR 58 ;BRANCH IF YES
1324 000700 022626 48: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
1325 000702 012737 000176 000450 MOV #SWREG,0#SWR ;AND PLACE THE ADDRESS OF THE SWITCH REGISTER
1326 ;DESIGNED FOR THE COMPUTERS NOT HAVING HARDWARE
1327 ;SWITCH REGISTER AND RUNNING STAND ALONE
1328 ;RUNNING UNDER APT?
1329
1330 000710 105737 000420 58: TSTB #8ENV

```

```

1327 000714 001403 BEQ APTSIZ ;BRANCH IF NO
1328 000716 012737 000422 000450 MOV ##SSWREG,##SWR ;SET SWR EQUAL TO APT SWITCH REGISTER,
1329
1330
1331
1332
1333 ;APTSIZ- THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE AND WHEN
1334 ;A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO GIVEN HIGH ADDRESS,
1335 ; IF APT DEFINES SIZE THE LOW TEST ADDRESS MUST=000000.(DUE TO ETABLE FORMAT)
1336 ;FLOW;
1337 ; IF BLOCK 4 (OR 3,2,1) TYPE NON ZERO THEN GET APT HIGH ADDRESS AND EXIT,
1338 ; ELSE SEND ERROR #3
1339 ;NOTE; THE MEMORY TYPE IS IGNORED SINCE ALL TESTS ARE RUN REGARDLESS OF MEMORY TYPE,
1340
1341 000724 012703 000340 APTSIZ: MOV #MAXMEM,R3 ;POINT R3 TO MAXMEM,
1342 000730 013737 000330 000334 MOV ##HIGHTWO,##SHIMAX ;IN CASE NO SELF SIZING DONE,
1343 000736 013737 000332 000336 MOV ##HIGHADD,##SMAXM ;IN CASE NO SELF SIZING DONE,
1344 000744 105737 000421 TSTB #8ENVM ;DOES APT ALLOW SELF SIZING?
1345 000750 100021 BPL TRYSR ;BRANCH IF YES
1346
1347 000752 012701 000451 MOV ##MTYP4+4,R1 ;POINT R1 TO BLOCK TYPE 4(+4)
1348 000756 162701 000004 18: SUB #4,R1 ;POINT R1 TO NEXT BLOCK TYPE,
1349 000762 105711 TSTB (R1) ;IS THE BLOCK TYPE NON ZERO?
1350 000764 001006 BNE 28 ;BRANCH IF YES (MEMORY EXISTS)
1351 000766 020127 000431 CMP R1,##MTYP1 ;ALL APT BLOCK TYPES BEEN CHECKED?
1352 000772 101371 BHI 18 ;BRANCH IF NO
1353
1354 000774 004767 005136 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1355 (1) 001000 000002 2 ;*****ERROR NUMBER 2*****
1356
1357 001002 004767 006302 28: JSR PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO sMAXM + sHIMAX
1358 001006 004767 006276 JSR PC,GETADR ;GO SET MAXIMUM APT ADDRESS INTO HIGHADD+HIGHTWO
1359 001012 000446 BRTPSZ: BR TYP6IZ ; TYPE THE SIZE OF MEMORY UNDER TEST
1360
1361 001014 032777 000100 177426 TRYSR: BIT #100,0#SWR ;USER DEFINED MEMORY TEST BOUNDARIES??
1362 001022 001042 BNE TYP6IZ ;BRANCH IF YES (DON'T SIZE MEMORY)
1363
1364
1365
1366 001024 010401 SLFSIZ: MOV R4,R1 ;SETUP R1 AND R4 TO THE LOWEST ADDRESS OF MEMORY
1367 001026 012710 001042 MOV #48,(R0) ;SET UP RETURN ADDRESS FROM TIME OUT TRAP TO 48
1368 001032 011111 28: MOV (R1),(R1) ;WRITE A MEMORY LOCATION INTO ITSELF AND TRAP IF NONEXIS
1369
1370 001034 062701 000002 ADD #2,R1 ;ADD 2 TO THE ADDRESS POINTER
1371 001040 000774 BR 28 ;KEEP ON SIZING UP THE MEMORY UNTIL
1372 ;Nxm TRAP (TIME OUT TRAP) IS ENCOUNTERED
1373
1374 001042 022626 48: CMP (SP)+,(SP)+ ;RESTORE THE STACK POINTER
1375 001044 004767 005770 JSR PC,MEMMNG ; SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE
1376 ;AND IF IT HAS TO BE TESTED
1377 001050 105737 000276 TSTB ##MMAVA ;SEE IF MEMORY MANAGEMENT HAS TO BE TESTED
1378 001054 001414 BEQ 128 ;IF NO MEM. MANG. THEN GO TO 128
1379 001056 012710 001070 68: MOV #88,(R0) ;SET UP THE RETURN ADDRESS FROM TRAP TO 88
1380 001062 012701 020000 MOV #20000,R1 ;BEGIN CHECKING MEMORY ABOVE 28K

```

```

1381 001066 000761          BR      20
1382 001070 022626          CMP     (SP)+,(SP)+
1383 001072 022701 160000    80:    CMP     #160000,R1
1384
1385
1386 001076 001003          BNE     120
1387 001100 004767 006106    JSR     PC,UPMM
1388 001104 000764          BR      60
1389 001106 024341          120:   CMP     -(R3),-(R1)
1390
1391 001110 004767 006104    JSR     PC,PUTADR
1392
1393 001114 024343          CMP     -(R3),-(R3)
1394 001116 004767 006076    JSR     PC,PUTADR
1395
1396 001122 005743          TST     -(R3)
1397 001124 005043          CLR     -(R3)
1398 001126 005043          CLR     -(R3)
1399 001130 012720 000104    TYPsiz: MOV    #BUSER,(R0)+
1400 001134 010403          MOV     R4,R3
1401
1402 001136 012701 000324    MOV     #LOWTWO,R1
1403 001142 004767 005400    JSR     PC,PCRLF
1404 001146 004767 005546    JSR     PC,OCTTYP
1405 001152 004767 005302    TYPMEM: JSR    #,STYPE
1406 001156 000055          ,ASCIZ  /-
1407
1408 001160 004767 005534    JSR     PC,OCTTYP
1409 001164 012703 000330    SETSTK: MOV   #HIGHTWO,R3
1410 001170 004767 006130    JSR     PC,#GTsiz
1411
1412 001174 010401          MOV     R4,R1
1413
1414 001176 062704 000022    40:    ADD     #10,,R4
1415
1416 001202 005302          DEC     R2
1417 001204 002374          BGE     40
1418 001206 010437 000310    MOV     P4,#ENDSTK
1419 001212 005021          CLR     (R1)+
1420 001214 020104          CMP     R1,R4
1421 001216 101775          BLOS   60
1422 001220 012737 157776 000340    MOV     #157776,#MAXMEM
1423 001226 005723          TST     (R3)+
1424 001230 001004          SAVLDR (R3)+
1425 001232 021300          CMP     (R3),R0
1426 001234 103002          SAVLDR (R3)+
1427 001236 011363 000002    MOV     (R3),2(R3)
1428
1429
1430
1431 001242 004767 006136    SAVLDR: JSR   PC,CLRMM
1432 001246 005723          TST     (R3)+
1433 001250 011305          MOV     (R3),R5
1434
1435
1436
;IF ONLY 4K BEING TESTED DON'T SAVE LOADERS

```

```

1437 001252 020527 017776    CMP     R5,#17776
1438 001256 103416          BLO     40
1439
1440 001260 162705 000276    30:    SUB     #276,R5
1441 001264 005737 000042    TST     #42
1442 001270 001406          BEQ     20
1443 001272 023737 000042 000046    CMP     #42,#46
1444 001300 001402          BEQ     20
1445 001302 162705 005674    SUB     #<1502,*2>,R5
1446 001306 012524          20:    MOV     (R5)+,(R4)+
1447 001310 020513          CMP     R5,(R3)
1448 001312 101775          BLOS   20
1449 001314 012323          40:    MOV     (R3)+,(R3)+
1450 001316 010423          MOV     R4,(R3)+
1451
1452 001320 010537 000346    TSTREL: MOV   R5,#SAVR5
1453 001324 004767 006054    TSTSiz: JSR   PC,CLRMM
1454 001330 005745          TST     -(R5)
1455 001332 012703 000324    10:    MOV     #LOWTWO,R3
1456 001336 005723          TST     (R3)+
1457
1458 001340 001003          BNE     20
1459 001342 021327 157776    CMP     (R3),#157776
1460
1461 001346 103411          BLO     40
1462 001350 032777 010000 177072 20:    BIT     #10000,#SWR
1463 001356 001003          BNE     30
1464 001360 004767 004552    JSR     PC,FATERR
1465 001364 000003          3
1466
1467 001366 000167 003526    30:    JMP     TSTMM
1468 001372 020423          CMP     R4,(R3)+
1469 001374 103002          BHIS   60
1470 001376 016304 177776    MOV     -2(R3),R4
1471 001402 005723          TST     (R3)+
1472 001404 001003          BNE     80
1473 001406 021305          CMP     (R3),R5
1474
1475 001410 101001          BHI     00
1476 001412 011305          MOV     (R3),R5
1477 001414 105737 000405    80:    TSTB   #REL
1478 001420 100014          BPL    100
1479 001422 013704 000322    MOV     #RELBOT,R4
1480 001426 020527 017776    CMP     R5,#17776
1481 001432 103402          BLO     90
1482 001434 012705 017776    MOV     #17776,R5
1483
1484 001440 020405          90:    CMP     R4,R5
1485 001442 103403          BLO     100
1486 001444 004767 004466    JSR     PC,FATERR
1487 001450 000004          4
1488
1489 001452 012703 000342    100:   MOV     #SAVMAX,R3
1490 001456 011343          MOV     (R3),-(R3)
1491 001460 062713 000002    MEMTST: ADD    #2,(R3)
1492

```

```

1489 001464 005725          TST      (R5)+          ;AND SET R5=MAX MEMORY+2
1490
1491                          ;CLEAR MEMORY UNDER TEST
1492
1493 001466 012702 000001    CLRMEM: MOV     #1,R2          ;SET R2 TO ENABLE PARITY MODULE CODE,
1494 001472 004767 006006    JSR     PC,PARITY        ;ENABLE PARITY IF WANTED AND AVAILABLE,
1495 001476 010500          MOV     R5,R0
1496 001500 005000          2$:    CLR     =(R0)          ;BEGIN CLEARING THE MEMORY FROM THE TOP
1497 001502 020004          CMP     R0,R4            ;UNTIL THE BOTTOM IS REACHED
1498 001504 011375          BHI     2$
1499 001506 012702 000316    MOV     #BAKPAT,R2
1500 001512 012212          MOV     (R2)+,(R2)       ;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
1501 001514 000312          SWAB   (R2)
1502 001516 017702 176726    MOV     @SWR,R2          ;LOAD R2 WITH THE OPTIONS STORED AT $SWREG
1503 001522 042702 177760    BIC     #177760,R2       ;ONLY LEAVE THE LOWER 4 BITS OF $SWREG IN R2 TO GO TO
1504                                     ;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
1505
1506
1507
1508                          ;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
1509
1510 001526 005037 000306    CONT:  CLR     @PASFLG    ;INIT SUBTEST PASS FLAG,
1511 001532 110237 000404    MOVB   R2,@$TESTN       ;SET UP $TESTN WITH THE TEST NUMBER GOING
1512                                     ;TO BE EXECUTED
1513 001536 010401          LOOP:  MOV     R4,R1       ;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
1514 001540 010246          MOV     R2,=(SP)        ;SAVE R2 ON THE STACK
1515 001542 012703 000376    MOV     #376,R3         ;POINT R3 TO SCRATCH STACK
1516 001546 004767 005446    JSR     PC,PUTADR       ;GO TO GENERATE 18 BIT ADDRESS OUT OF THE ADDRESS
1517                                     ;STORED IN R1 AND STORE IT IN LOCATIONS (R3)
1518                                     ;AND (R3-2)
1519 001552 005743          TST     =(R3)           ;CAUSE R3 TO POINT TO THE HIGH ORDER BITS OF THE
1520                                     ;18 BIT ADDRESS
1521 001554 004767 005544    JSR     PC,$GTSIZ      ; PLACE BITS 13-17 OF THE ADDRESS IN BITS
1522                                     ;0-4 OF R2
1523 001560 010400          MOV     R4,R0           ;PLACE THE ADDRESS OF THE LOWEST LOCATION UNDER
1524                                     ;TEST IN R0
1525 001562 010401          MOV     R4,R1           ;IN R1
1526 001564 010403          MOV     R4,R3           ;AND IN R3
1527 001566 012602          MOV     (SP)+,R2       ;RESTORE R2
1528 001570 006302          ASL     R2
1529 001572 006702          ADD     PC,R2
1530 001574 006207 000004    ADD     TBL-,(R2),PC    ;GO TO THE TEST #
1531                                     ;STORED IN BITS 0-3 OF SWITCH REGISTER
1532
1533
1534 001600 000102          TBL:   TST0-TBL        ;RELATIVE ADDRESS OF TEST # 0
1535 001602 000334          TST1-TBL        ;RELATIVE ADDRESS OF TEST # 1
1536 001604 000434          TST2-TBL        ;RELATIVE ADDRESS OF TEST # 2
1537 001606 000544          TST3-TBL        ;RELATIVE ADDRESS OF TEST # 3
1538 001610 001012          TST4-TBL        ;RELATIVE ADDRESS OF TEST # 4
1539 001612 001122          TST5-TBL        ;RELATIVE ADDRESS OF TEST # 5
1540 001614 001270          TST6-TBL        ;RELATIVE ADDRESS OF TEST # 6
1541 001616 001424          TST7-TBL        ;RELATIVE ADDRESS OF TEST # 7
1542 001620 001646          TST10-TBL       ;RELATIVE ADDRESS OF TEST # 10
1543 001622 002174          TST11-TBL       ;RELATIVE ADDRESS OF TEST # 11
1544 001624 002246          TST12-TBL       ;RELATIVE ADDRESS OF TEST # 12

```

```

1545 001626 002520          TST13-TBL        ;RELATIVE ADDRESS OF TEST # 13
1546 001630 003146          RELOC-TBL       ;RELATIVE ADDRESS OF ROUTINE 'RELOC'
1547
1548
1549
1550                          ;R5 IS POINTING TO THE TOP OF THE MEMORY TO BE TESTED+2
1551                          ;R4 & R0 ARE POINTING TO THE LOWEST ADDRESS OF MEMORY TO BE TESTED

```



```

1556                                     ;*      SCOPE ROUTINE
1557                                     ;*      -----
1558                                     ;*
1559                                     ;*
1560                                     ;*      PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
1561                                     ;*      IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE,
1562                                     ;*      IF SR= 2000 (BIT10) THEN HALT
1563                                     ;*      IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SR BITS<3:0>
1564                                     ;*      ELSE CONTINUE TO NEXT TEST.
1565                                     ;*
1566
1567
1568 001632 105737 000420 TSTSCP: TSTB 0##ENV ;ARE WE RUNNING UNDER APT?
1569 001636 001002 BNE CNTSCP ;IF SO THEN GO TO CNTSCP
1570 001640 004767 JSR PC,CHEKCK ;TEST FOR CONTROL-C AND IF TYPED GO
1571 ;PRINT ERROR HISTORY AND HALT AT FATHLT,
1572 001644 113702 CNTSCP: MOVB 0##TESTN,R2 ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
1573 ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
1574 ;OF R2 WILL BE 0
1575 001650 005237 000410 INC 0##DEVCT ;TELL APT WE ARE STILL RUNNING OKAY
1576 001654 032777 002000 176566 BIT #2000,RSWR ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
1577 001662 001401 BEQ TSTGO ;IF NOT THEN GO TO 26
1578 001664 000000 SWHALT: HALT ;HALT AT END OF TEST SWITCH SET,
1579
1580 001666 032777 040000 176554 TSTGO: BIT #40000,RSWR ;IS THE PROGRAM GOING TO LOOP ON TEST
1581 001674 001320 BNE LOOP ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
1582 001676 105202 INCB R2
1583 001700 000712 BR CONT ;GO TO CONT AND CONTINUE EXECUTING THE NEXT TEST
    
```

```

1596                                     ;*****
1597 (3) ;*TEST 0 TEST FOR PROPER BANK SELECTION
1598 (4) ;*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
1599 (4) ;* OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
1600 (4) ;* LOCATION UNDER TEST
1601 (4) ;*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
1602 (4) ;* 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
1603 (4) ;* LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
1604 (4) ;* [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
1605 (4) ;*(3) THIS TEST ALSO CHECKS TO SEE THAT NONE OF THE NON EXIST-
1606 (4) ;* ING BANK RESPOND WHEN THEY ARE ADDRESSED
1607 (3) ;*****
1608 (2) 001702 105737 000404 TST0: TSTB 0##TESTN ;CHECK FOR PROPER TEST SEQUENCE
1609 001706 001403 BEQ .+10
1610 001710 004767 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1611 (1) 001714 000005 5 ;*****ERROR NUMBER 5*****
1612 (1)
1613 1599 001716 012703 177777 MOV #177777,R3
1614 1600 001722 010401 16: MOV R4,R1 ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
1615 1601 001724 010310 MOV R3,(R0) ;SET ALL THE BITS AT (R0)
1616 1602 001726 020001 28: CMP R0,R1 ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
1617 1603 001730 001417 BEQ 48 ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
1618 1604 001732 005711 TST (R1) ;OTHERWISE CHECK (R1) FOR ALL 0'S
1619 1605 001734 001430 BEQ 58
1620 1606 001736 020311 CMP R3,(R1) ;IF R1 IS NOT EQUAL TO R0 AND (R1)
1621 ;DOES NOT CONTAIN ALL 0'S THEN
1622 ;CHECK TO SEE IF (R0) = (R1)
1623 1608
1624 1609 001740 001004 BNE 38
1625 1610 001742 012767 000006 000042 MOV #6,12# ;*ERROR* SETUP ERROR NO. IN 12#
1626 ;*****ERROR NUMBER #6*****
1627 (1)
1628 1611 001750 000403 BR 10#
1629 1612 001752 012767 000007 000032 38: MOV #7,12# ;*ERROR* SETUP ERROR NO. IN 12#
1630 ;*****ERROR NUMBER #7*****
1631 (1)
1632 1613 001760 010046 108: MOV R0,-(SP) ;SAVE R0 ON STACK
1633 1614 001762 105237 000301 INCB 0##ADERR ;AN ADDRESSING ERROR IS SUSPECTED
1634 1615 001766 000407 BR 11#
1635 1616 001770 020311 48: CMP R3,(R1) ;CHECK (R1) FOR ALL 1'S
1636 1617 001772 001411 BEQ 58
1637 1618 001774 012767 000010 000010 MOV #10,12# ;*ERROR* SETUP ERROR NO. IN 12#
1638 ;*****ERROR NUMBER #10*****
1639 (1)
1640 1619 002002 010046 MOV R0,-(SP) ;SAVE R0 ON STACK
1641 1620 002004 010309 MOV R3,R0
1642 1621 002006 004767 003566 JSR PC,ERROR ;GO TO THE ERROR SUBROUTINE
1643 1622 002012 000000 120: MOV (SP)+,R0 ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1644 1623 002014 012600 ;RESTORE R0
1645 1624
1646 1625 002016 013706 000350 58: MOV 0##SAVR6,SP ;RESTORE THE STACK POINTER
1647 1626 002022 062701 020000 ADD #20000,R1 ;CAUSE R1 TO POINT TO THE SAME CHIP
1648 ;LOCATION IN THE NEXT 4K BANK OF MEMORY
1649 ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
1650 1629 002026 020105 CMP R1,R5 ;COMPARE R1 WITH THE HIGHEST MEMORY
1651 ;LOCATION WHICH IS STORED IN R5
1652 1630
1653 1631 002030 103736 BLO 2# ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2#
1654 1632
1655 1633 002032 105737 000421 TSTB 0##ENVH ;HAS APT INHIBITED SIZING?
    
```

```

1634 002036 100430          BMI    00      ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1635 002040 032777 000100 176402 BIT    #100,0SWR ;HAS USER INHIBITED SIZING?
1636 002046 001024          BNE    00      ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1637
1638 002050 020137 000340          CMP    R1,0#MAXMEM ;IS R1 LOWER THAN THE MAXIMUM AVAILABLE
1639                                ;MEMORY ?
1640 002054 103760          BLO    50      ;IF SO THEN GO TO 50
1641 002056 012702 000006          MOV    #6,R2     ;MAKE R2 POINT TO TRAP VECTOR+2 FOR NXM
1642 002062 012712 000340          MOV    #340,(R2) ;SET PSW TO 340
1643 002066 012742 177722          MOV    #50,-6,-(R2) ;SET UP RETURN ADDRESS FROM TRAP TO 50
1644 002072 060712          ADD    PC,(R2)
1645 002074 020127 157776          CMP    R1,#157776 ;SEE IF R1 HAS CROSSED 20K BOUNDRY OF VIRTUAL ADDRESS
1646 002100 101004          BHI    60      ;IN WHICH CASE GO TO 60
1647 002102 011111          MOV    (R1),(R1) ;TRY TO WRITE TO NON-EXISTENT MEMORY (SHOULD TRAP)
1648 002104 004767 004026          JSR    PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002110 000011          11      ;*****ERROR NUMBER 11*****
1649
1650 002112 012722 000006          60:    MOV    #6,(R2)+ ;RESTORE TRAP VECTOR
1651 002116 005012          CLR    (R2)
1652 002120 005010          80:    CLR    (R0)
1653
1654 002122 062700 020000          ADD    #20000,R0 ;CAUSE R0 TO POINT TO THE SAME CHIP
1655                                ;LOCATION IN THE NEXT 4K MEMORY BANK
1656                                ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R0
1657 002126 020005          CMP    R0,R5     ;COMPARE R0 WITH THE HIGHEST MEMORY
1658                                ;LOCATION WHICH IS STORED IN R5,
1659 002130 103674          BLO    10      ;IF R0 LESS THEN REPEAT THE TEST
1660 002132 000637          END0:  BR    TSTSCP
1661
1662
    
```

```

1668                                ;*****
(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) 002134 122737 000001 000404 TST1:  CMPB  #1,0#STESTN ;CHECK FOR PROPER TEST SEQUENCE
1669
1670
1671 002142 001403          REQ    .+10
1672 002144 004767 003766          JSR    PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002150 000012          12      ;*****ERROR NUMBER 12*****
(1)
1673 002152 012700 000001          10:    MOV    #1,R0
1674 002156 010002          MOV    R0,R2     ;SET R2=1
1675 002160 010011          20:    MOV    R0,(R1)  ;MOV 1 AT LOCATION (R1)
1676 002162 020011          30:    CMP    R0,(R1)  ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
1677 002164 001403          BEQ    40
1678 002166 004767 003406          JSR    PC,ERROR  ;*ERROR* REPORT ERROR MESSAGE
(1) 002172 000013          13      ;*****ERROR NUMBER 13*****
(1)
1679
1680 002174 005702          40:    TST    R2       ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
1681 002176 001406          BEQ    50      ;IF SO THEN GO TO 50
1682 002200 006300          ASL    R0       ;SHIFT THE 1 BROUGHT IN AT 10 IN
1683                                ;DATA DIRECTION
1684 002202 103366          BCC    20      ;IF THE 1 HAS NOT BEEN SHIFTED THRU
1685                                ;THE 16 DATA BITS THEN REPEAT FROM 20
1686 002204 005002          CLR    R2       ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
1687 002206 012700 177776          MOV    #177776,R0
1688 002212 000762          BR    20
1689
1690 002214 000261          50:    SEC
1691 002216 006100          ROL    R0       ;SHIFT A 0 16 TIMES IN DATA DIRECTION
1692 002220 103757          BCS    20      ;IF THE 0 HAS NOT BEEN SHIFTED THRU
1693                                ;THE 16 DATA BITS THEN REPEAT FROM 20
1694 002222 062701 020000          ADD    #20000,R1 ;OTHERWISE GO TO THE NEXT BANK OF
1695                                ;4K MEMORY AND REPEAT THE TEST
1696 002226 020105          CMP    R1,R5
1697 002230 103750          BLO    10
1698 002232 000737          END1:  BR    END0
1699
    
```

```

1708                ;*****
(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) 002234 122737 000002 000404 TST2: CMPB #2,##STESTN ;CHECK FOR PROPER TEST SEQUENCE
1709
1710 002242 001403 BEQ #+10
1711 002244 004767 003666 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002250 000014 ;*****ERROR NUMBER 14*****
(1)
1712 002252 013700 000316 15: MOV #BAKPAT,R0
1713 002256 110021 MOVB R0,(R1)+
1714 002260 113721 000317 MOVB #BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
1715 002264 020105 CMP R1,R5
1716 002266 103771 BLO 15
1717
1718 002270 020041 25: CMP R0,-(R1) ;TEST THE MEMORY TO SEE IF IT CONTAINS
1719 ;THE WORD STORED IN BAKPAT
1720 002272 001416 BEQ 0#
1721 002274 062701 000002 ADD #2,R1
1722 002300 123741 000317 CMPB #BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
1723 002304 001402 BEQ 4#
1724 002306 120041 CMPB R0,-(R1) ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
1725 002310 001002 BNE 6#
1726 002312 105237 000301 48: INCB #BADERR ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
1727 002316 042701 000001 68: BIC #1,R1 ;MAKE THE ADDRESS IN R1 EVEN
1728 002322 004767 003252 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 002326 000015 ;*****ERROR NUMBER 15*****
(1)
1729 002330 020104 86: CMP R1,R4 ;KEEP ON TESTING THE MEMORY UNTIL
1730 002332 101356 BHI 2# ;R1 EQUALS THE LOWEST ADDRESS
1731 002334 000337 000316 SWAB #BAKPAT ;CHANGE THE DATA PATTERN
1732 002340 001744 BEQ 1# ;IF THE DATA PATTERN DOES NOT HAVE LOW
1733 ;BYTE =0 THEN FALL THRU
1734 002342 000733 END2: BR END1
1735
1736 ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
1737
    
```

```

1749                ;*****
(3)                ;*TEST 3 DUAL ADDRESS TEST A
(4)                ;*(1) THIS TEST CHECKS FOR DUAL ADDRESSING PROBLEMS BY WRITING A
(4)                ;* BACKGROUND 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) 002344 122737 000003 000404 TST3: CMPB #3,##STESTN ;CHECK FOR PROPER TEST SEQUENCE
1750 002352 001403 BEQ #+10
1751 002354 004767 003556 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002360 000016 ;*****ERROR NUMBER 16*****
(1)
1752 002362 005003 CLR R3
1753 002364 004737 000120 26: JSR PC,##WRMEM ; WRITE MEMORY WITH THE BACKGROUND STORED
1754 ;AT LOCATION BAKPAT
1755 002370 005002 48: CLR R2
1756 002372 003020 68: BIS R3,R2 ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
1757 002374 020204 CMP R2,R4 ;IF R2 IS LESS THAN R4
1758 002376 103465 BLO 16# ;THEN DO NOTHING
1759 002400 020205 CMP R2,R5 ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
1760 002402 103077 BHIS 20# ;TESTED THEN EXIT THE TEST
1761 002404 000312 SWAB (R2) ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
;THE LOCATION POINTED BY R2
1762
1763 002406 005001 CLR R1
1764 002410 003031 78: BIS R3,R1
1765 002412 020104 CMP R1,R4 ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
1766 002414 103445 BLO 12# ;THEN GO TO 12#
1767 002416 020105 CMP R1,R5
1768 002420 103053 BHIS 15#
1769 002422 020102 CMP R1,R2 ;CHECK THE MEMORY FOR CORRECT DATA
1770 002424 001431 BEQ 10#
1771 002426 020011 CMP R0,(R1) ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
;THE SAME WORD AS BAKPAT
1772 ;IN WHICH CASE GO BACK TO 12#
1773 002430 001437 BEQ 12#
1774 002432 012767 000017 000032 MOV #17,22# ;*ERROR* SETUP ERROR NO, IN 22#
(1) ;*****ERROR NUMBER #17*****
1775 002440 010046 88: MOV R0,-(SP) ;PLACE R0 ON THE STACK
1776 002442 000316 SWAB (SP)
1777 002444 022611 CMP (SP)+,(R1) ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
;AS A SWAPPED R0
1778 ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
1779 002446 001003 BNE 9# ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
1780 ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
1781 ;ENTIRE WORD
1782 1783 002450 012767 000020 000014 MOV #20,22# ;*ERROR* SETUP ERROR NO, IN 22#
(1) ;*****ERROR NUMBER #20*****
1784 002456 105237 000301 98: INCB #BADERR ;ADDRESSING PROBLEM IS DETECTED
1785 002462 010046 MOV R0,-(SP) ;SAVE R0
1786 002464 010200 MOV R2,R0 ;SET R0=GOOD ADDRESS FOR ERROR REPORT
1787 002466 004767 003106 JSR PC,ERROR ;GO TO THE ERROR SUBROUTINE
1788 002472 000000 22: ,WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1789 002474 012600 MOV (SP)+,R0 ;RESTORE R0
    
```

```

1790 002476 010011      MOV    R0,(R1)      ;RESTORE (R1)
1791 002500 020037      CMP    R0,#BAKPAT  ;IF THE CONTROL CAME HERE FROM 15#-2 THEN
1792 002504 001411      BEQ    12#
1793 002506 000407      BR     11#         ;RETURN TO 11#
1794 002510 000300      R0     SWAB        ;MAKE R0 SAME AS SWAPPED BAKPAT
1795 002512 020011      CMP    R0,(R1)     ;IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
1796                               ;EQUAL TO SWAPPED R0
1797 002514 001404      BEQ    11#         ;IN WHICH CASE GO BACK TO 11#
1798 002516 012767      MOV    #21,22#     ;*ERROR* SETUP ERROR NO, IN 22#
1799                               ;*****ERROR NUMBER #21*****
1800 002524 000745      BR     8#         ;AND GO TO 8#
1801 002526 000300      SWAB   R0         ;RESTORE R0 TO BAKPAT
1802 002530 040301      BIC   R3,R1       ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
1803 002532 005701      TST   R1         ;IF R1 IS 0 THEN PLACE A 1 IN R1
1804 002534 001001      BNE   13#        ;OTHERWISE GO TO 13#
1805 002536 005201      INC   R1
1806 002540 006101      ROL   R1
1807 002542 020127      CMP    R1,#2000#   ;IF R1 IS LESS THAN A 4K BOUNDRY
1808 002544 103720      BLO   7#         ;THEN REPEAT FROM 7#
1809 002546 103720      BLO   7#
1810 002550 000312      SWAB   (R2)       ;RESTORE (R2) TO BAKPAT
1811 002552 040302      BIC   R3,R2       ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
1812                               ;STORED IN R2
1813 002554 005702      TST   R2         ;IF R2 = 0 THEN MOVE A 1 TO R2
1814 002556 001001      BNE   18#        ;OTHERWISE GO TO 18#
1815 002560 005202      INC   R2
1816 002562 006102      ROL   R2
1817 002564 020000      CMP    R2,#2000#   ;SHIFT A ONE IN THE ADDRESS WORD
1818 002566 103720      BLO   7#         ;IS THE ADDRESS IN R2 MORE THAN THE BOUNDRY
1819 002568 103720      BLO   7#         ;OF 4K
1820 002570 103700      BLO   6#         ;IF NOT THEN GO TO 6#
1821 002572 060203      ADD   R2,R3       ;OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
1822 002574 020337      CMP    R3,#MAXMEM ;IF R3 IS POINTING TO A BANK THAT IS LOWER
1823                               ;THAN MAXMEM
1824 002600 103673      BLO   4#         ;THEN REPEAT FROM 4#
1825 002602 000337      SWAB   #BAKPAT    ;REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
1826 002604 001656      BEQ    TST3       ;THE LOWER BYTE OF BAKPAT IS 0
1827 002610 000654      END3# BR    END2
    
```

```

1832                               ;*****
1833 (3)                               ;*TEST 4 DUAL ADDRESS TEST B
1834 (4)                               ;*(1) THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
1835 (4)                               ;* AND READING THE ADDRESS IN THE LOCATION AND THEN
1836 (4)                               ;* WRITING AND READING ADDRESS COMPLEMENT
1837 (3)                               ;*****
1838 (2) 002612 122737 000004 000404 TST4# CMPB #4,0#TESTN ;CHECK FOR PROPER TEST SEQUENCE
1839 002620 001403      BEQ    .+10
1840 002622 004767      JSR    PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1841 (1) 002626 000022      22     ;*****ERROR NUMBER 22*****
1842 (1)
1843 002630 005303      CLR   R3
1844 002632 010100      MOV   R1,R0      15# ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
1845 002634 005703      TST   R3         ;IN THE LOCATION
1846 002636 001401      BEQ   26#        ;OTHERWISE STORE COMPLEMENT
1847 002640 005100      COM   R0         ;OF THE ADDRESS
1848 002642 010021      MOV   R0,(R1)+   28# ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
1849 002644 020105      CMP   R1,R5
1850 002646 103771      BLO   1#
1851 002650 020041      CMP   R0,-(R1)   38# ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
1852 002652 001405      BEQ   4#
1853 002654 105237      INCB #ADERR      ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
1854                               ;BIT PROBLEM
1855 002660 004767      JSR   PC,ERROR   ;*ERROR* REPORT ERROR MESSAGE
1856 (1) 002664 000023      23     ;*****ERROR NUMBER 23*****
1857 (1)
1858 002666 010100      MOV   R1,R0      48# ;CHECK THAT THE ADDRESS IS STORED AT
1859 002670 162700      SUB   #2,R0      ;LOCATION IF R3 IS NOT 0
1860 002674 005703      TST   R3         ;OTHERWISE CHECK FOR
1861 002676 001401      BEQ   58#        ;ADDRESS COMPLEMENT
1862 002700 005100      COM   R0
1863 002702 020104      CMP   R1,R4      58#
1864 002704 101361      BHI   3#
1865 002706 112737      MOVB #1,#PASFLG  ;SET PASFLG FOR ERROR REPORT,
1866 002710 005103      COM   R3         ;COMPLEMENT THE CONTENTS OF R3
1867 002714 001345      BNE   1#         ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
1868                               ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU
1869 002720 000733      END4# BR    END3
1870 1861
    
```

```

1876 ;*****
(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)
(4) ;*****
(2) 002722 122737 000005 000404 TST5: CMPB #5,##TESTN ;CHECK FOR PROPER TEST SEQUENCE
1877
1878 002730 001403 BEQ ,+10
1879 002732 004767 003200 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002736 000024 24 ;*****ERROR NUMBER 24*****
(1)
1880 002740 004737 000120 18: JSR PC,##WRITMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1881 ;WORD STORED IN BAKPAT
1882 002744 020041 28: CMP R0,=(R1) ;READ THE CONTENTS OF LOCATION POINTED BY R1
1883 002746 001403 BEQ 38 ;TO SEE IF IT HAS THE SAME VALUE AS R0
1884 002750 004767 002624 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 002754 000025 25 ;*****ERROR NUMBER 25*****
(1)
1885 002756 000300 38: SWAB R0
1886 002760 010011 MOV R0,(R1) ;SWAP THE BYTES AT (R1)
1887 002762 021100 CMP (R1),R0 ;READ (R1) FOR CORRECT VALUE
1888 002764 001403 BEQ 48
1889 002766 004767 002606 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 002772 000026 26 ;*****ERROR NUMBER 26*****
(1)
1890
1891 002774 000300 48: SWAB R0 ;SWAP THE BYTES OF THE REGISTER
1892 ;CONTAINING BACKGROUND PATTERN
1893 002776 001023 BNE 98 ;IF THE LOWER BYTE OF THE REGISTER
1894 ;IS NOT 0 THEN THE PROGRAM IS READING
1895 ;THE MEMORY TO CONTAIN A BACK GROUND OF
1896 ;BAKPAT AND WRITING THE SWAPPED WORD
1897
1898 ;IN WHICH CASE GO TO 98
1899
1900
1901 003000 005703 58: TST R3 ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1902 ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
1903 ;IF R3 EQUAL 0 THEN THE PROGRAM IS
1904 ;READING/WRTING MIN, TO MAX, OTHERWISE
1905 ;IT IS GOING IN MAX, TO MIN, DIRECTION
1906 003002 001023 BNE 108 ;IF R3 IS NOT CLEAR THEN GO TO 108
1907 003004 062701 000002 68: ADD #2,R1 ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1908 003010 020105 CMP R1,R5 ;COMPARE R1 WITH THE MAX, MEMORY LOCATION TO
1909 ;BE TESTED
1910 003012 103006 BHIS 88 ;IF R1>R5 THEN GO TO 88 OTHERWISE
1911 003014 020011 78: CMP R0,(R1) ;READ (R1) FOR THE CORRECT DATA
    
```

```

1912 003016 001757 BEQ 38 ; WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1913 ;AND REPEAT UNTIL R1 > R5
1914 003020 004767 002554 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003024 000027 27 ;*****ERROR NUMBER 27*****
(1)
1915 003026 000753 BR 38
1916 003030 105237 000306 88: INCB ##PASFLG
1917 003034 000300 SWAB R0
1918 003036 001742 BEQ 28 ;IF THE LOWER BYTE OF R0 IS ALL 0'S
1919 ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1920 ;AND READING BAKPAT GOING FROM MAX, TO MIN, [PASFLG=4]
1921 003040 005103 COM R3
1922 003042 010401 MOV R4,R1 ;OTHERWISE CLEAR R0
1923 003044 000763 BR 78 ;PUT THE LOWEST TESTING ADDRESS IN R1
1924 ;AND BEGIN READING 0'S, WRITING 1'S AND
1925 ;READING 1'S IN MIN, TO MAX, DIRECTION [PASFLG=3]
1926 003046 005703 98: TST R3 ;IF R3 IS NON 0, I.E. PASFLG=3
1927 003050 001353 BNE 58 ;THEN READ BAKPAT, WRITE
1928 ;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1929 ;IN MIN, TO MAX, DIRECTION
1930 003052 020104 108: CMP R1,R4 ;OTHERWISE TEST IS PROCEEDING IN MAX, TO
1931 ;MIN, DIRECTION.
1932 003054 101333 BHI 28 ;KEEP ON LOOPING UNTIL R1=R4
1933 003056 105237 000306 88: INCB ##PASFLG
1934 003062 000300 SWAB R0
1935 003064 001753 BEQ 78 ;IF R0 SWAPPED HAS LOWER BYTE=0
1936 ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1937 ;AND READ BAKPAT GOING FROM MIN, TO MAX.
1938 003066 000714 END5: BR END4
1939
    
```

```
1956 ;*****  
(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) 003070 122737 000006 000404 TST6: CMPB #6,0#$TESTN ;CHECK FOR PROPER TEST SEQUENCE  
1957  
1958  
1959 003076 001403 BEQ .+10  
1960 003100 004767 003032 JSR PC,SEGGERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT  
(1) 003104 000030 30 ;*****ERROR NUMBER 30*****  
(1)  
1961 003106 004737 000120 RPT6: JSR PC,0$WRTEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE  
1962 ;WORD STORED AT LOCATION BAKPAT  
1963 003112 005037 000306 CLR 0$PASFLG  
1964 003116 010403 10: MOV R4,R3 ;SET R3  
1965 003120 010401 20: MOV R4,R1 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST  
1966 003122 020011 30: CMP R0,(R1) ;CHECK (R1) FOR CORRECT DATA  
1967 003124 001403 BEQ 4$  
1968 003126 004767 002446 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 003132 000031 31 ;*****ERROR NUMBER 31*****  
(1)  
1969 003134 062701 000002 4$: ADD #2,R1 ;INCREMENT R1 BY 2  
1970 003140 020105 CMP R1,R5 ;SEE IF R1 HAS REACHED THE MAX, OF MEMORY  
1971 003142 103767 BLO 3$  
1972 003144 132737 000001 000306 BITB #1,0$PASFLG ;CHECK TO SEE IF PASFLG=0 OR 2  
1973 003152 001002 BNE 5$  
1974 003154 105237 000306 INCB 0$PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1  
1975  
1976 003160 020305 50: CMP R3,R5 ;SEE IF R3 HAS REACHED THE MAX, OF THE MEMORY  
1977 003162 103012 BHIS 7$  
1978 003164 012702 037776 MOV #37776,R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)  
1979 003170 000313 60: SWAB (R3)  
1980 003172 005302 DEC R2  
1981 003174 001375 BNE 6$  
1982 003176 010337 000354 MOV R3,0$SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT,  
1983 003202 062703 020000 ADD #20000,R3 ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE  
1984 ;R3 TO POINT TO A LOCATION IN THE NEXT  
1985 ;4K BANK OF MEMORY  
1986 003206 000744 BR 2$  
1987 003210 105237 000306 7$: INCB 0$PASFLG ;MAKE PASFLG=2  
1988 003214 000337 000316 SWAB 0$BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND  
1989 003220 001732 BEQ RPT6 ;THEN GO BACK TO THE LOCATION RPT6  
1990 003222 000721 END6: BR ENDS  
1991
```

1992

```

2004                ;*****
(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) 003224 122737 000007 000404 TST7:  CMPB    #7,0#TESTN    ;CHECK FOR PROPER TEST SEQUENCE
2005
2006 003232 001403                BEQ     ,+10
2007 003234 004767 002676        JSR     PC,SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003240 000032                ;*****ERROR NUMBER 32*****
(1)
2008 003242 005037 000306        28:    CLR     0#PASFLG
2009 003246 010337 000304        MOV     R3,0#LOWBNK ;LOWBNK CONTAINS ADDRESS OF THE LOWEST LOCATION
2010                ;IN THE 4K BANK THAT CAN BE TESTED
2011 003252 010302                MOV     R3,R2
2012 003254 052702 017776        BIS     #1776,R2    ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK
2013 003260 005722                TST    (R2)+        ;ADD 2 TO R2
2014 003262 020502                CMP     R5,R2
2015 003264 103001                BHIS   4#           ;IF R2 IS GREATER THAN R5 THEN GO TO 4#
2016 003266 010502                MOV     R5,R2       ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION
2017                ;THAT CAN BE TESTED
2018 003270 010337 000302        48:    MOV     R3,0#STRTDI ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE
2019                ;DIAGONAL
2020 003274 013701 000304        MOV     0#LOWBNK,R1 ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K
2021                ;BANK
2022 003300 013700 000316        68:    MOV     0#BAKPAT,R0 ;STORE THE CONTENTS OF BAKPAT IN R0
2023 003304 020103                CMP     R1,R3        ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?
2024 003306 001010                BNE    10#          ;IF NOT THEN GO TO 10#
2025 003310 062703 000002        ADD     #2,R3        ;THE FOLLOWING CODE IS USED TO PLACE THE
2026 003314 032703 000176        BIT    #176,R3      ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL
2027 003320 001402                BEQ     8#           ;IN R3
2028 003322 062703 000200        ADD     #200,R3     ;
2029 003326 000300                88:    SWAB   R0        ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN
2030 003330 132737 000001 000306 108:   BITB   #1,0#PASFLG ;CONTENTS OF LOCATION PASFLG WILL BE EVEN IF THE
2031                ;MEMORY IS BEING WRITTEN AND IT WILL BE ODD
2032                ;IF IT IS ONLY BEING READ
2033 003336 001001                BNE    12#          ;IF IT IS BEING READ ONLY THEN GO TO 12#
2034 003340 010011                MOV     R0,(R1)     ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS
2035                ;OF R0
2036 003342 020011                128:   CMP     R0,(R1)   ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN
2037                ;PROPER DATA
2038 003344 001403                BEQ     14#          ;IF IT IS OK THEN GO TO 14#
2039 003346 004767 002226        JSR     PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
(1) 003352 000033                ;*****ERROR NUMBER 33*****
(1)
2040 003354 062701 000002        148:   ADD     #2,R1        ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION
2041 003360 020102                CMP     R1,R2        ;IS IT THE END OF THE BANK ?
2042 003362 103746                BLO    6#           ;IF NOT THEN GO TO 6#
2043 003364 005237 000410        168:   INC     0#DEVCT     ;TELL APT WE ARE STILL RUNNING OKAY
2044 003370 105237 000306        INCB   0#PASFLG
    
```

```

2045 003374 013703 000302        MOV     0#STRTDI,R3 ;LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL
2046 003400 132737 000001 000306  BITB   #1,0#PASFLG ;HAS THE READ OF THE MEMORY BEEN DONE ?
2047 003406 001330                BNE    4#           ;IF NOT THEN GO TO 4#
2048 003410 005723                TST    (R3)+        ;ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL
2049 003412 020302                CMP     R3,R2        ;AND UNLESS THE END OF THE BANK IS REACHED
2050 003414 103003                BHIS   18#          ;
2051 003416 105737 000306        TSTB   0#PASFLG    ;OR THE DIAGONAL HAS BEEN ROTATED 64 TIMES
2052 003422 100322                BPL    4#           ;REPEAT FROM 4#
2053 003424 013703 000304        188:   MOV     0#LOWBNK,R3 ;MAKE R3 POINT TO THE LOWEST LOCATION IN THE
2054                ;IN THE BANK UNDER TEST
2055 003430 000337 000316        SWAB   0#BAKPAT
2056 003434 001715                BEQ     4#           ;AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE
2057                ;SWAPPED BACK GROUND PATTERN THEN GO TO 4#
2058 003436 010203                MOV     R2,R3        ;MAKE THE PRESENT HIGH BOUNDRY AS THE NEXT
2059                ;LOW BOUNDRY
2060 003440 020205                CMP     R2,R5        ;UNLESS THE PRESENT HIGH BOUNDRY IS ALSO THE
2061                ;HIGH BOUNDRY FOR THE MEMORY UNDER TEST
2062 003442 103677                BLO    2#
2063 003444 000666                END7:  BR     END6
    
```

```
2091 J*****  
(3) J*TEST 10 READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL  
(4)  
(4) J*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN  
(4) J* STORED AT LOCATION BAKPAT  
(4) J*(2) TEST BEGINS AT LOWEST LOCATION BEING TESTED  
(4) J* (LETS NAME IT 'A')  
(4) J*(3) LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B',  
(4) J*(4) SWAPS BYTES FOR LOCATION 'A'.  
(4) J*(5) READS 'A', READS 'B'  
(4) J*(6) 'B' = 'B'+200 (MAKES 'B'=64TH CELL I.E, 200TH OCTAL  
(4) J* LOCATION FROM THE PRESENT LOCATION OF 'B')  
(4) J*(7) REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE  
(4) J* END OF THE 4K BANK OF THE MEMORY IN WHICH 'A' IS RESIDING  
(4) J*(8) A = A+2  
(4) J*(9) REPEATS STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK  
(4) J*(10) GOES TO THE NEXT 4K BANK OF MEMORY AND REPEATS STEPS  
(4) J* 3-9 UNTIL THE END OF THE MEMORY  
(4) J*(11) AFTER EXECUTING THE TEST BYTES ARE SWAPPED AT  
(4) J* LOCATION BAKPAT AND STEPS 1-10 ARE REPEATED  
(4) J*(12) IN THIS TEST R0 IS POINTING TO LOCATION 'A', R1 TO  
(4) J* LOCATION 'B', R2 TO THE END OF THE 4K BANK IN WHICH THE  
(4) J* TEST IS TAKING PLACE AND R3 TO THE LOWEST LOCATION IN THE  
(4) J* COLUMN/ROW CONTAINING 'A' AND 'B'  
(4) J*(13) MOST OF THE CODE USED BY THIS TEST IS ALSO USED BY TEST 11  
(4)  
(3) J*****  
(2) 003446 122737 000010 000404 TST10: CMPB #10,0#8TESTN ;CHECK FOR PROPER TEST SEQUENCE  
2092  
2093 003454 001403 BEQ +10  
2094 003456 004767 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT  
(1) 003462 000034 ;*****ERROR NUMBER 34*****  
(1)  
2095 003464 010402 MOV R4,R2 ;SET R2 TO THE LOWEST MEMORY UNDER TEST  
2096 003466 052702 RPT10: BIS #17776,R2 ;MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K  
(1) ;BANK FOR WHICH GALLOPING WILL BE PERFORMED  
2097 ;INCREMENT R2 BY 2  
2098 003472 062702 GALLOP: ADD #2,R2 ;IF THE HIGH BOUNDRY OF THE TEST IS HIGHER THAN  
2099 003476 020205 CMP R2,R5 ;THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2  
2100 003500 101401 BLOS 28  
2101 003502 010502 MOV R5,R2  
2102 003504 005046 28: CLR -(SP)  
2103 003506 010200 MOV R2,R0  
2104 003510 013740 48: MOV #BAKPAT,-(R0) ;WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF  
(1) ;BAKPAT  
2105  
2106 003514 020003 CMP R0,R3  
2107 003516 101374 BHI 48  
2108 003520 010301 68: MOV R3,R1 ;R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT  
(1) ;CAN BE TESTED IN THIS BLOCK  
2109 003522 023710 000316 CMP #BAKPAT,(R0) ;BEFORE STARTING THE GALLOPING TEST FOR LOCATION  
(1) ;(R0) CHECK IT  
2110 ;CONTINUE IF OK  
2111 003526 001410 BEQ 88 ;OTHERWISE PREPARE TO REPORT THE ERROR  
2112 003530 010001 MOV R0,R1  
2113 003532 013700 000316 MOV #BAKPAT,R0  
2114 003536 004767 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 003542 000035 ;*****ERROR NUMBER 35*****  
(1)
```

```
2116 003544 010011 MOV R0,(R1) ;RESTORE THE CONTENTS OF (R1)  
2117 003546 010100 MOV R1,R0 ;RESTORE R0  
2118  
2119 003550 000310 88: SWAB (R0)  
2120 003552 031011 108: BIT (R0),(R1) ;CHECK TO SEE THAT NONE OF THE BITS SET  
(1) ;IN (R0) ARE SET IN (R1) AND VICE VERSA  
2121 ;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1  
2122 003554 020001 CMP R0,R1  
2123  
2124 003556 001412 BEQ 128  
2125 003560 021137 000316 CMP (R1),#BAKPAT ;CHECK THAT (R1) HAS BAKPAT IN IT  
2126 003564 001407 BEQ 128  
2127 003566 010046 MOV R0,-(SP) ;SAVE R0 ON STACK  
2128 003570 013700 000316 MOV #BAKPAT,R0 ;PLACE THE PATTERN WORD IN R0  
2129 003574 004767 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 003600 000036 ;*****ERROR NUMBER 36*****  
(1)  
2130 003602 012600 MOV (SP)+,R0 ;RESTORE R0  
2131 003604 021037 128: CMP (R0),#SWAPAT ;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT  
2132 003610 001412 BEQ 148  
2133 003612 010146 MOV R1,-(SP) ;SAVE R1 ON THE STACK  
2134 003614 010001 MOV R0,R1 ;MAKE R1 POINT TO THE FAILING LOCATION  
2135 003616 013700 000320 MOV #SWAPAT,R0 ;LOAD R0 WITH THE EXPECTED RESULT IN (R1)  
2136 003622 004767 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 003626 000037 ;*****ERROR NUMBER 37*****  
(1)  
2137 003630 010011 MOV R0,(R1) ;RECOVER (R1) FROM THE ERROR  
2138 003632 010100 MOV R1,R0 ;RESTORE R0  
2139 003634 012601 MOV (SP)+,R1 ;AND RESTORE R1  
2140 003636 122737 000011 000404 148: CMPB #11,0#8TESTN ;IS THE PROGRAM EXECUTING TEST # 11 ?  
2141 003644 001402 BEQ 168 ;IF SO THEN GO TO 168  
2142 003646 062701 ADD #176,R1  
2143 003652 062701 168: ADD #2,R1 ;MAKE R1 POINT TO THE NEXT ADJACENT CELL  
2144 003656 020102 CMP R1,R2 ;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDRY  
2145 003660 103734 BLO 108 ;THEN REPEAT FROM 108  
2146 003662 000320 SWAB (R0)+ ;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST  
(1) ;WAS BEING PERFORMED  
2147 ;IS IT TEST 11 ?  
2148 003664 122737 000011 000404 CMPB #11,0#8TESTN ;IS IT TEST 11 ?  
2149 003672 001407 BEQ 176 ;IF SO THEN GO TO 176  
2150 003674 005723 TST (R3)+ ;OTHERWISE INCREMENT R3 BY 2  
2151 003676 062716 000002 ADD #2,(SP) ;FOR EVERY ROW/COLUMN TESTED ADD 2  
2152 003702 105716 TSTB (SP)  
2153 003704 100002 BPL 178 ;UNTIL (SP) IS 200  
2154 003706 161603 SUB (SP),R3 ;SUBTRACT 200 FROM R3  
2155 003710 005016 CLR (SP)  
2156 003712 032700 000177 BIT #177,R0 ;AT A 64TH CALL BOUNDRY?  
2157 003716 001002 BNE 188 ;BRANCH IF NO  
2158 003720 005237 000410 INC #8DEVCT ;TELL APT WE ARE STILL RUNNING  
2159 003724 020002 CMP R0,R2 ;IF R0 HAS NOT REACHED THE END OF THE BOUNDRY  
2160 003726 103674 BLO 68 ;THEN REPEAT FROM 68  
2161 003730 162603 SUB (SP)+,R3 ;RESTORE SP AND R3  
2162 003732 000337 000320 SWAB #SWAPAT  
2163 003736 000337 000316 SWAB #BAKPAT  
2164 003742 001660 BEQ 28 ;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 28  
2165 003744 010203 MOV R2,R3 ;OTHERWISE MAKE THE PRESENT HIGH BOUNDRY AS THE  
(1) ;NEXT LOW BOUNDRY  
2166  
2167 003746 020205 CMP R2,R5
```



```

2168 003750 001410      BEQ     END10      ;IF PREVIOUS HIGH BOUNDRY WAS THE END OF THE
2169                                ;TEST BOUNDRY THEN EXIT THE TEST
2170 003752 032702 017776      BIT     #17776,R2   ;WAS IT A 4K BOUNDRY ?
2171 003756 001025      BNE     RPT11      ;IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG
2172                                ;GALLOPING TEST DISABLED
2173 003760 122737 000011 000404  CMPB   #11,0##TESTN ;IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11
2174 003766 001421      BEQ     RPT11      ;
2175 003770 000636      BR      RPT10      ;OTHERWISE REPEAT TEST 10
2176 003772 000624      END10: BR     END7
2177
2178
2179
    
```

```

2208                                ;*****
(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
(4)
(3)                                ;*****
(2) 003774 122737 000011 000404 TST11: CMPB   #11,0##TESTN ;CHECK FOR PROPER TEST SEQUENCE
2209
2210 004002 001403      BEQ     ,+10
2211 004004 004767 002126      JSR     PC,SEQERR   ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004010 000040      40
(1)
2212 004012 010402      MOV     R4,R2       ;MAKE R2 TO POINT TO THE LOWEST LOCATION
2213                                ;UNDER TEST
2214 004014 105777 174430      TSTB   @SWR        ;LONG GALLOP ENABLED?
2215 004020 100004      BPL     RPT11      ;BRANCH IF NO
2216 004022 004767 002540      JSR     PC,PNTMES   ;TYPE "GLP"
2217 004026 046107 000120      ,ASCIZ /GLP/
2218 004032 105777 174412      RPT11: TSTB   @SWR   ;LONG GALLOPING ENABLED?
2219 004036 100613      BMI     RPT10      ;BRANCH IF YES
2220                                ;TO RPT10
2221 004040 052702 000176      BIS     #176,R2     ;OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
2222                                ;TO GET THE HIGH BOUNDRY
2223
2224 004044 000612      BR      GALLOP     ; PERFORM GALLOPING TEST
    
```

```
2247 ;*****  
(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) ;* 0 = 1 ARE WRITTEN TO A VALUE OF SWAPPED BAKPAT  
(4) ;*(2) STARTING FROM THE LOWEST LOCATION THE MEMORY IS CHECKED  
(4) ;* TO OBTAIN 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 0 & 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) 004046 122737 000012 000404 TST121 CMPB #12,0#TESTN ;CHECK FOR PROPER TEST SEQUENCE  
2248 004054 001403 BEQ ,+10  
2249 004056 004767 002054 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT  
(1) 004062 000041 41 ;*****ERROR NUMBER 41*****  
2250  
2251 004064 012702 000001 MOV #1,R2 ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1  
2252 004070 012703 000400 MOV #400,R3 ;AND 0  
2253 004074 112737 000001 000306 18: MOVB #1,#PASFLG ;INITIALIZE THE COUNTER FOR THE SUBTEST  
2254 004102 010401 28: MOV R4,R1 ;PLACE THE STARTING ADDRESS OF MEMORY UNDER  
2255 ;TEST IN R1  
2256 004104 013700 000316 48: MOV 0#BAKPAT,R0  
2257 004110 030201 BIT R2,R1 ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET  
2258 004112 001004 BNE 88 ;IF IT IS SET THEN GO TO 88  
2259 004114 030301 BIT R3,R1 ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET  
2260 004116 001404 BEQ 128 ;IF IT IS NOT SET THEN GO TO 128  
2261 004120 005100 66: COM R0 ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS  
2262 ;POINTED BY R2 & POINTED BY R3 = 1 IN WHICH  
2263 ;CASE PREPARE TO WRITE THE LOCATION  
2264 ;WITH A COMPLEMENT OF LOCATIONS NOT MEETING  
2265 ;THIS CONDITION  
2266 004122 000402 BR 128  
2267 004124 030301 88: BIT R3,R1 ;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND  
2268 ;CHECK ADDRESS BIT POINTED BY R3  
2269 004126 001774 BEQ 68 ;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 68  
2270 004130 132737 000002 000306 128: BITB #2,#PASFLG ;IS IT 2ND OR 3RD PASS OF THE SUBTEST ?  
2271 004136 001001 BNE 148 ;IF SO THEN READ THE MEMORY
```

```
2289 004140 010011 MOV R0,(R1) ;OTHERWISE WRITE THE MEMORY BEFORE READING IT  
2290 004142 020011 148: CMP R0,(R1) ;READ THE MEMORY FOR CORRECT CONTENTS  
2291 004144 001403 BEQ 168  
2292 004146 004767 001426 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 004152 000042 42 ;*****ERROR NUMBER 42*****  
2293 004154 012746 000002 168: MOV #2,=(SP)  
2294 004160 005100 188: COM R0  
2295 004162 005111 COM (R1)  
2296 004164 020011 CMP R0,(R1) ;READ THE MEMORY AGAIN  
2297 004166 001404 BEQ 198  
2298 004170 004767 001404 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE  
(1) 004174 000043 43 ;*****ERROR NUMBER 43*****  
2299 004176 010011 MOV R0,(R1) ;RESTORE THE LOCATION (R1)  
2300 004200 005316 198: DEC (SP)  
2301 004202 001366 BNE 188 ;EXECUTE THE CODE FROM 188 TWICE  
2302 004204 005726 TST (SP)+ ;RESTORE THE STACK POINTER  
2303 004206 122737 000003 000306 CMPB #3,#PASFLG ;IS IT THE 3RD PASS OF THE SUBTEST ?  
2304 004214 001412 BEQ 208 ;IF SO THEN GO TO 208  
2305 004216 062701 000002 ADD #2,R1 ;IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN  
2306 ;MIN. TO MAX. DIRECTION  
2307 004222 020105 CMP R1,R5 ;HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?  
2308 004224 103727 BLO 48 ;IF NOT THEN REPEAT FROM 48  
2309 004226 105237 000306 INCB 0#PASFLG  
2310 004232 122737 000002 000306 CMPB #2,#PASFLG ;IF IT IS THE 2ND PASS OF THE SUBTEST  
2311 004240 001720 BEQ 28 ;THEN REPEAT FROM 28  
2312 004242 162701 000002 208: SUB #2,R1 ;OTHERWISE EXECUTE THE TEST IN MAX. TO MIN,  
2313 ;DIRECTION  
2314 004246 020104 CMP R1,R4 ;HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?  
2315 004250 103315 BHS 48 ;IF NOT THEN REPEAT FROM 48  
2316 004252 012702 020000 MOV #20000,R2 ;PREPARE TO CHECK THE MEMORY WITH THE XOR OF  
2317 ;ADDRESS BITS 0 AND 13  
2318 004256 105237 000307 INCB 0#PASFLG+1 ;THE SUB TEST HAS CHECKED THE XOR ONE KIND  
2319 004262 123727 000306 000002 CMPB 0#PASFLG,#2 ;HAS TWO XOR COMBINATIONS BEEN CHECKED ?  
2320 004270 103701 BLO 18 ;IF NOT THEN GO TO 18  
2321 004272 101004 BHI 228 ;IF ALL THREE HAVE BEEN CHECKED THEN GO TO 228  
2322 004274 012702 000010 MOV #10,R2 ;IF IT IS THE 2ND XOR COMBINATION THEN CHECK  
2323 004300 006303 ASL R3 ;FOR ADDRESS BITS 3 & 8  
2324 004302 000674 BR 18  
2325 004304 005137 000316 228: COM 0#BAKPAT ;IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED  
2326 004310 105737 000316 TSTB 0#BAKPAT  
2327 004314 001654 BEQ TST12 ;BAKPAT THEN RE-EXECUTE THE TEST  
2328 004316 000625 END12: BR END10
```

```

2359                ;*****
(3)                ;*TEST 13   WRITE RECOVERY TEST
(4)                ;*
(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 BANK 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)                ;*****
2360 004320 122737 000013 000404 TST131 CMPB #13,0##TESTN ;CHECK FOR PROPER TEST SEQUENCE
2361 004326 001403 BEQ #+10
2361 004330 004767 001602 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004334 000044 44 ;*****ERROR NUMBER 44*****
2362 004336 012702 010247 18: MOV #10247,R2 ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
2363 ;IN R2.
2364 004342 012700 177667 MOV #177667,R0 ;PLACE THE COMPLEMENT OF THE INSTRUCTION
2365 ;JMP (R0) IN R0
2366 ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
2367 ;SINCE THE TEST STORES "MOV R2,-(PC) IN 1/2 AND 177667 IN THE OTHER 1/2
2368
2369 004346 010546 28: MOV R5,-(SP) ;SAVE R5
2370 004350 010446 MOV R4,-(SP) ;STORE LOWEST ADDRESS ON STACK
2371 004352 000241 298: CLC
2372 004354 006005 ROR R5 ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
2373 004356 006004 ROR R4 ;DO SAME FOR LOWEST ADDRESS
2374 004360 160405 SUB R4,R5 ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
2375 004362 006005 ROR R5 ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
2376 004364 103002 BCC 308 ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS,
2377 004366 062716 000002 ADD #2,(SP) ;INCREASE LOWEST TEST ADDRESS BY 2
2378 004372 012604 308: MOV (SP)+,R4 ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
2379 004374 012605 MOV (SP)+,R5 ;RESTORE HIGHEST TEST ADDRESS
2380 004376 010403 MOV R4,R3 ;PLACE THE LOWEST LOCATION UNDER TEST
2381 ;IN R3
2382 004400 000405 BR 288 ;LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0

```

```

2383 004402 042703 017776 38: BIC #17776,R3 ;CAUSE R3 TO POINT TO THE LOWEST LOCATION
2384 ;IN THE 4K BANK UNDER TEST
2385 004406 105737 000405 TSTB 0#REL ;ARE WE RELOCATED?
2386 004412 100504 BMI 148 ;BRANCH IF YES-TEST BANK0 ONLY-
2387 004414 020305 288: CMP R3,R5 ;IF R3 IS HIGHER THAN THE HIGHEST LOCATION
2388 004416 103102 BHIS 148 ;UNDER TEST THEN EXIT
2389 ;IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0
2390 004420 020527 020000 CMP R5,#20000 ;IS HIGHEST TEST ADDRESS BELOW 4K?
2391 004424 103002 BHIS 318 ;BRANCH IF NO
2392 004426 010501 MOV R5,R1 ;SET R1 TO HIGHEST TEST ADDRESS IN BANK0
2393 004430 000405 BR 328
2394
2395 004432 010301 318: MOV R3,R1 ;SET R1 TO LOWEST CURRENT TEST ADDRESS
2396 004434 042701 017776 BIC #17776,R1 ;CLEAR LOW ORDER ADDRESS BITS
2397 004440 062701 020000 ADD #20000,R1 ;CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
2398 ;OF THE 4K BANK BEING POINTED BY R3
2399 004444 020137 000340 328: CMP R1,0#MAXMEM ;IF R1 IS HIGHER THAN MAX, OF THE
2400 004450 101065 BHI 148 ;MEMORY+2 ALTHOUGH R3 IS LESS THAN R5
2401 ;THEN THE HIGHEST LOCATION UNDER
2402 ;TEST IS NOT IN A 4K BANK EXIT
2403
2404 004452 132737 000001 000306 BITB #1,0#PASFLG ;IS THE LOWEST BIT OF LOCATION PASFLG
2405 004460 001101 BNE 168 ;SET? IN WHICH CASE BACK GROUND HAS
2406 ;ALREADY BEEN WRITTEN AND WRITE RECOVERY
2407 ;TEST IS BEING PERFORMED
2408
2409 004462 020304 48: CMP R3,R4 ;OTHERWISE WRITE THE BACKGROUND
2410 004464 103430 BLO 88 ;DEFINED AT STEP 3.
2411 004466 105737 000307 TSTB 0#PASFLG+1 ;IS THE TEST JUST DOING READ, I.E.
2412 004472 001002 BNE 68 ;IS THE PASFLG+1 LOCATION NON ZERO? IF SO
2413 ;THEN GO TO 68
2414 004474 012713 010247 MOV #10247,(R3) ;WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,-(PC)
2415 004500 020213 68: CMP R2,(R3) ;READ (R3) TO CONTAIN CORRECT DATA
2416 004502 001421 BEQ 88
2417 004504 010046 MOV R0,-(SP) ;SAVE R0
2418 004506 010146 MOV R1,-(SP) ;AND R1 ON THE STACK
2419 004510 010301 MOV R3,R1
2420 004512 010200 MOV R2,R0 ;SET R0= GOOD DATA FOR ERROR PRINTOUT
2421 004514 004767 001060 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 004520 000045 45 ;*****ERROR NUMBER 45*****
(1)
2422 004522 012601 MOV (SP)+,R1 ;RESTORE R1
2423 004524 012600 MOV (SP)+,R0 ;AND R0
2424 004526 105737 000306 TSTB 0#PASFLG ;IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
2425 ;THE PROPER DATA THEN WE DON'T WANT TO GO AND
2426 ;EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
2427 ;TEST
2428 004532 001005 BNE 88 ;BRANCH IF PASFLG NOT =0
2429
2430 004534 010200 MOV R2,R0 ;SAVE FOR ERROR REPORT
2431 004536 004767 001036 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 004542 000046 46 ;*****ERROR NUMBER 46*****
(1)
2432 004544 000664 BR END12 ;ABORT TST 13.
2433
2434 004546 062703 000002 88: ADD #2,R3 ;INCREMENT R3 BY 2

```

```

2435 004552 162701 000002 SUB #2,R1 ;DECREMENT R1 BY 2
2436 004556 020105 CMP R1,R5 ;WRITE THE BACKGROUND DEFINED AT STEP 4,
2437 004560 103014 BHIS 12#
2438 004562 020103 CMP R1,R3 ;HAS STORING THE 177667 REACHED WHERE "MOV R2,=(PC) IS?
2439 004564 103405 BLO 10# ;BRANCH IF YES DON'T DESTROY THE MOV R2,=(PC) IS,
2440 004566 105737 000307 TSTB 0#PASFLG+1 ;IS THE THE READ ONLY CHECK PASS?
2441 004572 001002 BNE 10# ;BRANCH IF YES
2442 004574 012711 177667 MOV #177667,(R1) ;WRITE THE LOCATION WITH THE COMPLEMENT OF THE
2443 ;OP CODE JMP (R0)
2444 004600 020011 10# CMP R0,(R1) ;READ R1 TO CONTAIN CORRECT DATA
2445 004602 001403 BEQ 12#
2449 004604 004767 000770 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 004610 000047 47 ;*****ERROR NUMBER 47*****
(1)
2450 004612 020301 12# CMP R3,R1 ;IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
2451 004614 103722 BLO 4# ;THEN REPEAT FROM 4#
2452 ;RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
2453
2454 ;OTHERWISE GO TO THE NEXT 4K BANK
2455 004616 062703 020000 13# ADD #2000,R3
2456 004622 000667 BR 3#
2457
2458 004624 122737 000001 000306 14# CMPB #1,0#PASFLG ;THE PROGRAM CONTROL COMES HERE AS FOLLOWS
2459 ;1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
2460 ; WRITE/READ CYCLE FOR THE BACK GROUND
2461 ; AND WANTS TO BEGIN THE WRITE RECOVERY TEST
2462 004632 001440 BEQ 24# ;2-PASFLG=1, PROGRAM HAS JUST COMPLETED
2463 ; THE WRITE RECOVERY TEST AND WANTS TO
2464 ; READ MEMORY FOR CORRECT DATA
2465 004634 103630 BLO END12 ;3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
2466 ; MEMORY AND WANTS TO GO THE NEXT TEST,
2467
2468 004636 105137 000307 COMB 0#PASFLG+1 ;ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
2469 ;ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
2470 ;ENTRY DISABLE READ ONLY
2471 004642 001241 BNE 2#
2472 004644 012702 005141 MOV #5141,R2 ;PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
2473 ;IN R2
2474 004650 012700 177740 MOV #13#,-6,R0 ;PLACE THE RETURN ADDRESS IN R0 AS 13#
2475 004654 060700 ADD PC,R0 ;THUS WHEN THE READ RECOVERY TEST REACHES
2476 ;THE MIDDLE OF THE 4K MEMORY THEN THE
2477 ;INSTRUCTION EXECUTED WILL BE JMP (R0)
2478 ;BRANCHING THE PROGRAM TO 13#
2479 004656 105237 000306 15# INCB 0#PASFLG ;INCREMENT PASFLG BY 1,
2480 004662 000631 BR 2#
2481
2482 004664 032777 000020 173556 16# BIT #20,0SWR ;HAS THE PRINTOUTS BEEN SUPRESSED ?
2483 004672 001017 BNE 18# ;IF SO THEN GO TO 18#
2484 004674 105737 000042 TSTB 0#42 ;IS THE PROGRAM RUNNING UNDER ACT?
2485 004700 001014 BNE 18# ;BRANCH IF YES
2486 004702 004767 001660 JSR PC,PNTMES ; TYPE THE BANK UNDER TEST
2487 004706 051524 030524 020063 .ASCIZ /TST13 BANK/
2488 004714 040502 045516 000
2489 ;EVEN
2490 004722 004722 JSR PC,GETBNK ;GET BANK NO. UNDER TEST INTO DECWRD FOR PRINT,
2491 004722 004767 002476 JSR PC,$TPDEC ;TYPE BANK NO. UNDER TEST
2492 004726 004767 001662

```

```

2491
2492 004732 000113 18# JMP (R3) ;BEGIN EXECUTING MOV R2,=(PC) ,COM -(R1) SEQUENCE IN TES
2493
2494
2495 004734 105137 000307 24# COMB 0#PASFLG+1
2496 004740 012700 000110 MOV #110,R0 ;PLACE THE OP CODE FOR JMP (R0) IN R0
2497 004744 000744 BR 15# ; READ THE MEMORY FOR CORRECT DATA AFTER
2498 ;INCREMENTING PASFLG TO 2
2499
2500 ;TST13 EXITS VIA END12.
2501

```

```

2506 004746 012737 000377 000316 RELOC: MOV #377,0#BAKPAT
2507 004754 105737 000276 TSTB 0#MMAVA ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2508 004760 001065 BNE CONTMM ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2509 ;MEMORY MANAGEMENT
2510 004762 032777 001000 173460 BIT #1000,0#SWR ;RELOCATION WANTED?
2511 004770 001046 BNE CKDONE ;BRANCH IF NO
2512 004772 105737 000405 TSTB 0#REL ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2513 004776 100420 BMI RELOER ; PLACE THE PROGRAM BACK IN LOWER CORE
2514 005000 112737 000200 000405 MOV# #200,0#REL ;OTHERWISE PREPARE TO RELOCATE
2515
2516 ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
2517
2518
2519 005006 004767 001554 JSR PC,PNTMES ;TYPE "RELOC"
2520 005012 042522 047514 000103 .ASCIZ /RELOC/
2521 .EVEN
2522 005020 013705 000340 MOV 0#MAXMEM,R5 ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
2523 ;AVAILABLE MEMORY
2524 005024 014445 28: MOV =(R4),=(R5) ;RELOCATE THE PROGRAM
2525 005026 020427 000430 CMP R4,#BEGIN-50 ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN-50
2526 005032 101374 BHI 28
2527 005034 000165 000050 JMP 50(R5)
2528
2529 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2530
2531
2532 005040 013705 000346 RELOER: MOV 0#SAVR5,R5 ;RESTORE R5
2533 005044 105737 000405 TSTB 0#REL ;IS DIAGNOSTIC IN RELOCATED STATE?
2534 005050 100016 BPL CKDONE ;BRANCH IF NO
2535
2536 005052 012704 000430 MOV #BEGIN-50,R4 ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2537 005056 012524 28: MOV (R5)+,(R4)+
2538 005060 020537 000340 CMP R5,0#MAXMEM
2539 005064 103774 BLO 28
2540 005066 105037 000405 CLR# 0#REL
2541 005072 010537 000346 MOV R5,0#SAVR5 ;SAVE R5
2542 005076 012706 000346 MOV #BEGIN,SP ;RESET STACK TO LOWER MEMORY
2543 005102 010637 000350 MOV SP,0#SAVR6 ;"BEGIN" USES THIS TO RESET THE STACK,
2544 005106 000137 005112 CKDONE: JMP 0#LOWER ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2545
2546
2547
2548 005112 105737 000315 LOWER: TSTB 0#SAVKBB ;HERE DUE TO "C TYPED?
2549 005116 001073 BNE $TPSTK ;BRANCH IF YES (TYPE ERROR STACK)
2550 005120 004767 001714 TSTMM: JSR PC,MEMMNG ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
2551 ;IS AVAILABLE
2552 005124 105737 000276 TSTB 0#MMAVA ;IS MEM, MANAG, AVAILABLE ?
2553 005130 001462 ENDPAS ;BRANCH IF NO
2554 005132 000402 BEQ SCNTMM ;BEGIN TESTING ABOVE 28K
2555 005134 004767 002052 CONTMM: JSR PC,UPHM ;GO TO UPDATE MEM, MANAG, REGISTERS
2556 005140 012703 000324 SCNTMM: MOV #LOWTWO,R3 ;MAKE R3 POINT TO THE LOCATION LOWTWO
2557 005144 004767 002160 JSR PC,GETSIZ ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2558 ;OF THE LOWEST ADDRESS UNDER TEST
2559 005150 012704 020000 MOV #20000,R4 ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2560 ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
2561 005154 020237 172342 CMP R2,0#172342 ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
    
```

```

2562 ;PAR1 ?
2563 005160 103405 BLO 28 ;IF SO THEN GO 28
2564 005162 050104 BIS R1,R4 ;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2565 ;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2566 005164 162702 000200 SUB #200,R2
2567 005170 004767 001650 JSR PC,MMREG ; SET MEM, MANAG, REGISTERS
2568 005174 004767 002130 28: JSR PC,GETSIZ ;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2569 ;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2570 ;OF LOCATION HIGHADD IN R1
2571 005200 004767 000020 JSR PC,MAXADR ; GET THE ADDRESS OF MAX, MEM, UNDER TEST
2572 005204 010005 MOV R0,R5
2573 005206 004767 002116 JSR PC,GETSIZ ; PREPARE TO SET UP LOCATION MAXMEM
2574 005212 004767 000006 JSR PC,MAXADR ; GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2575 005216 010013 MOV R0,(R3) ;AND STORE INTO "MAXMEM"
2576 005220 000167 174242 JMP CLRMEM ;GO TEST A 24K SLICE ABOVE 28K,
2577
2578
2579 ;MAXADR = SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K,
2580 ;REGISTERS:
2581 ;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR, UNDER TEST
2582 ;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
2583 ;R2= PAR BLOCK NO, CURRENTLY UNDER TEST,
2584
2585 005224 010046 MAXADR: MOV R0,=(SP) ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
2586 005226 012700 172356 MOV #172356,R0 ;R0=PAR7 UNIBUS ADDRESS
2587 ;**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2588 005232 162716 020000 28: SUB #20000,(SP) ;DECREMENT VIRTUAL ADDRESS BY 4K
2589 005236 050116 BIS R1,(SP) ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
2590 005240 020240 CMP R2,=(R0) ;DOES CURRENT PAR= TEST BLOCK NO.?
2591 005242 001410 BEQ 38 ;BRANCH IF YES
2592 005244 020027 172340 CMP R0,#172340 ;ARE WE AT PAR0?
2593 005250 101370 BHI 28 ;NO KEEP TRYING
2594 ;**END LOOP TO FIND PAR ADDRESS UNDER TEST
2595 005252 005720 TST (R0)+ ;SET TO PAR CURRENT
2596 005254 021002 CMP (R0),R2 ;IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
2597 005256 003006 BGT 48 ;BRANCH IF YES (FALL INTO ENDPAS)
2598 005260 012716 157776 MOV #157776,(SP) ;EXIT WITH MAXADR= 28K SEGMENT TEST SIZE
2599 005264 012600 38: MOV (SP)+,R0 ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2600 005266 062700 000002 ADD #2,R0 ;MAKE MAXIMUM MEMORY+2
2601 005272 000207 RTS PC ;AND EXIT MAXADR ROUTINE
2602
2603 005274 022626 48: CMP (SP)+,(SP)+ ;FIXUP STACK
2604 ;AND FALL THRU TO ENDPAS,
    
```

```

2609                                     ;* TYPE ROUTINE FOR ERROR STACK
2610                                     ;* -----
2611                                     ;*
2612                                     ;* THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
2613                                     ;* FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
2614
2615
2616
2617 005276 032777 000020 173144 ENDPAS: BIT #20,0$WR ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
2618 005304 001055 ;EOP ;IF NOT THEN GO TO $EOP
2619 005306 012746 177777 $TPSTK: MOV #=-1,-(SP) ;THE PROGRAM HAS REACHED THE END AND ERROR
2620 ;STACK AND END OF PASS WILL BE TYPED OUT
2621 005312 012701 007744 MOV #ENDPRG,R1 ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
2622 ;FOR 0 TO 4K MEMORY IN R1
2623 005316 012703 000376 TYPSTK: MOV #376,R3
2624 005322 005216 INC (SP) ;IF WE HAVE GONE THRU THE ENTIRE
2625 005324 020137 000310 CMP R1,#ENDSTK ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
2626 005330 103043 $EOP ;THEN GO TO TYPE END OF PASS
2627 005332 112702 000022 MOV# #18,,R2
2628 005336 105302 RETSTK: DECB R2 ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED,
2629 005340 002766 BLT TYPSTK ;BEEN CHECKED FOR ERROR THEN SEE IF THERE
2630 ;IS ANY MORE 4K MEMORY BANK
2631 005342 105721 TSTB (R1)+ ;OTHERWISE CHECK THE BYTE STORED AT (R1)
2632 005344 001774 BEQ RETSTK ;IF IT IS 0 WE WILL NOT TYPE IT
2633 005346 020227 000020 CMP R2,#16. ;IS THE POINTER POINTING TO ERROR STACK BYTE
2634 ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
2635 ;THE SPECIFIC MEMORY BANK
2636 005352 103404 BLO 2# ;IF NOT THEN GO TO TYPE BIT NUMBER
2637 005354 101026 BHI PARFL ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
2638 ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
2639 005356 004767 001012 JSR PC,TPADER ;OTHERWISE TYPE "ADDRESS ERROR"
2640 005362 000404 BR FAILNM
2641 005364 010237 000312 2#: MOV R2,#$DECNWRD ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
2642 ;IN DECIMAL
2643 005370 004767 001214 JSR PC,TYPDEC ;GO TO TYPE THE BIT NUMBER IN DECIMAL
2644 005374 011637 000312 FAILNM: MOV (SP),#$DECNWRD ;PREPARE TO TYPE THE PAGE NUMBER
2645 005400 004767 001210 JSR PC,$TPDEC ;IN DECIMAL
2646 005404 005043 CLR -(R3)
2647 005406 114113 MOV# -(R1),(R3) ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
2648 ;FAILURE OCCURED
2649 005410 105021 CLR# (R1)+ ;CLEAR THE ERROR STACK
2650 005412 005043 CLR -(R3)
2651 005414 105237 000314 INCB #TYPCNT ;ENABLE THE TYPE OUT OF 1 WORDS
2652 005420 004767 001330 JSR PC,RPTOCT ;TYPE THE 4K BANK AND THE NUMBER OF TIMES
2653 ;THIS FAILURE WAS SEEN
2654 005424 012703 000376 MOV #376,R3 ;RESET SCRATCH STACK FOR EACH BIT PRINTED,
2655 005430 000742 BR RETSTK
2656 005432 004767 000762 PARFL: JSR PC,TPPRER ;TYPE "PAR ERR"
2657 005436 000756 BR FAILNM
    
```

```

2662
2663
2664                                     ;* END OF PASS
2665                                     ;* -----
2666                                     ;*
2667                                     ;* TYPE "END PASS" AND DISABLE PARITY.
2668                                     ;* ALSO SERVICE ACT11.
2669                                     ;* AND EVERY CONSECUTIVE PASSES UNLESS BIT 4 OF $SWREG IS HIGH
2670                                     ;*
2671
2672
2673 005440 005002 $EOP: CLR R2 ;SET R2= PARITY MODULE DISABLE CODE
2674 005442 004767 002036 JSR PC,PARITY ;GO DISABLE PARITY MODULES IF SELECTED.
2675 005446 105737 000315 TSTB #0$AVKBB ;CONTROL-C TYPED?
2676 005452 001046 BNE CTLC ;BRANCH IF YES-RESTORE LOADERS AND HALT-
2677 005454 005237 000406 INC #0$PASS ;INCREMENT PASS COUNT
2678 005460 032777 000040 172762 BIT #40,0$WR ;"END PASS #XX" PRINTOUT WANTED?
2679 005466 001015 BNE ACT11 ;BRANCH IF NO
2680 005470 004767 001064 TYPEOP: JSR PC,TPCRLF ;TYPE CR, LF, AND "END PASS #"
2681 005474 047105 020104 040520 ,ASCIZ /END PASS #/
2682 005502 051523 021440 000 ,EVEN
2683 005510 013737 000406 000312 MOV #0$PASS,#$DECNWRD ;GET PASS COUNT
2684 005516 004767 001072 JSR PC,$TPDEC ;TYPE IT
2685 005522 013700 000042 ACT11: MOV #042,R0 ;GET THE MONITOR ADDRESS
2686 005526 001405 BEQ $DOAGN ;IF NONE
2687 005530 004767 000012 JSR PC,RLODER ;RESTORE XXDP MONITOR
2688 005534 000005 RESET ;RETURN TO ACT11 MONITOR.
2689
2690
2691
2692 005536 000137 000156 ;* SERVICE XXDP/ACT11
2693 JMP #0$ENDAD ;JUMP TO ACT SERVICE
2694
2695
2696 005542 000137 000250 $DOAGN: JMP #0$RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP
2697
2698
2699 005546 004767 001632 RLODER: JSR PC,CLRMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
2700 005552 013704 000344 MOV #0$SAVR4,R4 ;RESTORE R4 WITH SAVR4
2701 005556 014445 4#; MOV -(R4),-(R5) ;RESTORE LOADERS
2702 005560 020437 000310 CMP R4,#ENDSTK
2703 005564 101374 BHI 4#
2704 005566 000207 RTS PC ;RETURN FROM RLODER CALL
2705
2706
2707
2708 ;CONTROL C HANDLER
2709 CTLC: JSR PC,RLODER ;RESTORE ABS LOADER
2710 JMP APTHLT ;IF NOT APT HALT AT FATHLT
    
```

```

2713                ;* ERROR HANDLING ROUTINE
2714                ;* -----
2715                ;*
2716                ;* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
2717                ;* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
2718                ;*
2719
2720 005600 017637 000000 000402 ERROR: MOV    R(SP),R##FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2721 005606 010346          18: MOV    R3,-(SP) ;SAVE R3
2722 005610 010046          MOV    R0,-(SP) ;AND R0 ON THE STACK
2723
2724                ;SETUP BANK NO. IN FATAL FOR APT
2725
2726 005612 010103          MOV    R1,R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
2727 005614 004767 001604 JSR    PC,GETBNK ;GET BANK NO. UNDER TEST INTO PBANK
2728 005620 013703 000312 MOV    R##PBANK,R3 ;GET BANK UNDER TEST
2729 005624 110337 000403 MOV    R3,R##FATAL+1 ;STORE FAILING BANK NO. FOR APT
2730
2731                ;
2732
2733 005630 010346          MOV    R3,-(SP) ;TEMPORARILY STORE R3
2734 005632 012703 000376 MOV    #376,R3 ;MAKE R3 AS THE STACK POINTER
2735 005636 013743 000306 MOV    R##PASPLG,-(R3) ;OUTPUT THE WORD STORED AT
2736 005642 005043          CLR    -(R3)
2737 005644 113713 000402 28: MOV    R##FATAL,(R3) ;PUT ERROR NO. ON ERROR STACK
2738 005650 016643 000006 MOV    6(SP),-(R3) ;PLACE THE RETURN PC AT (R3)
2739 005654 011143          MOV    (R1),-(R3) ;PLACE BAD DATA,
2740 005656 010043          MOV    R0,-(R3) ;AND GOOD DATA ON THE STACK
2741 005660 005043          CLR    -(R3)
2742 005662 016313 000004 MOV    4(R3),(R3) ;TAKE THE
2743 005666 040013          BIC    R0,(R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
2744 005670 046300 000004 BIC    4(R3),R0 ;TO FIND THE BITS THAT FAILED
2745 005674 050013          BIS    R0,(R3) ;AND PLACE IT ON THE STACK
2746 005676 012700 002016 MOV    #ENDPRG-,-24,R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
2747 005702 060700          ADD    PC,R0 ;OF THE STARTING OF THE ERROR STACK
2748 005704 062700 000022 68: ADD    #18,R0 ;FOR THE SPECIFIC 4K BANK
2749 005710 005316          DEC    (SP)
2750 005712 002374          BGE    68
2751 005714 005726          TST    (SP)+ ;RESTORE THE STACK POINTER
2752
2753 005716 105037 000277 ERRTP: CLR    B 0#TYPENB ;DISABLE ANY TYPE OUT
2754 005722 105737 000300 18: TST    B##PRERR ;IF THIS IS PARITY PROBLEM
2755 005726 001007          BNE    38 ;THEN GO TO 38
2756 005730 105720          TST    (R0)+ ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
2757 005732 105737 000301 TST    B##ADERR ;IF THIS IS ADDRESSING PROBLEM
2758 005736 001003          BNE    38 ;THEN GO TO 38
2759 005740 105720          TST    (R0)+ ;INCREMENT THE POINTER R0 BY 1
2760 005742 005713 28: TST    (R3) ;IS BIT 15 OF (R3) SET?
2761 005744 100015          BPL    48 ;IF NOT THEN GO TO 48
2762 005746 122710 000377 38: CMP    B 377,(R0) ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
2763 005752 001401          BEQ    58 ;IF SO DON'T BUMP ERROR COUNT
2764 005754 105210          INCB   (R0) ;INCREMENT THE ERROR COUNTER BY 1
2765 005756 122710 000001 58: CMP    B 1,(R0) ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
2766 005762 001404          BEQ    78 ;BRANCH IF NO
2767 005764 032777 000400 172456 BIT    #400,BSWR ;STOP ERROR PRINTOUT AFTER 1 WANTED?
2768 005772 001002          BNE    48 ;BRANCH IF YES (DON'T TYPE ERROR)
    
```

```

2769 005774 105237 000277 78: INCB   0#TYPENB ;ENABLE THE TYPE OUT ROUTINE
2770 006000 105737 000300 48: TST    B##PRERR ;PARITY ERROR?
2771 006004 001411          BEQ    68 ;BRANCH IF NO
2772 006006 004767 000406 JSR    PC,TPPRER ;ELSE TYPE "PAR ERR"
2773 006012 000411          BR     88 ;AND DON'T TEST INDIVIDUAL BIT FAILURES,
2774 006014 105737 000301 TST    B##ADERR ;ADDRESS ERROR?
2775 006020 001403          BEQ    68 ;BRANCH IF NO
2776 006022 004767 000346 JSR    PC,TPADERR ;PRINT "ADR ERR"
2777 006026 000403          BR     88
2778 006030 105720 68: TST    (R0)+ ;POINT TO NEXT ENTRY IN ERROR STACK
2779 006032 006313          ASL    (R3) ;IS THERE STILL AN ERROR BIT SET IN ERROR,
2780 006034 001342          BNE    28 ;BR IF YES - KEEP FILLING ERROR STACK
2781 006036 112737 000006 000314 88: MOV    #6,0#TYPCNT ;TELL TYPCTO TO TYPE 6 WORDS OF ERROR STACK,
2782                ;THE STACK POINTED BY R3
2783 006044 004767 001150 JSR    PC,PUTADR ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
2784                ;AT LOCATIONS (R3) AND (R3-2)
2785 006050 004767 000622 JSR    PC,TYPERR ;TYPE ERROR STACK (7 WORDS)
2786
2787 006054 005037 000300 108: CLR    B##PRERR ;CLEAR ADDRESS/PARITY ERROR FLAGS
2788 006060 012600          MOV    (SP)+,R0 ;RESTORE R0
2789 006062 012603          MOV    (SP)+,R3 ;AND R3
2790 006064 105737 000420 FNDERR: TST    B##ENV ;ARE WE RUNNING UNDER APT?
2791 006070 001404          BEQ    28 ;IF NOT THEN TEST FOR HALT
2792 006072 012737 000001 000400 MOV    #1,B##MSGTY ;OTHERWISE INFORM THE APT
2793 006100 000443          BR     FATHLT ;GOTO FATHLT AND WAIT FOR APT,
2794
2795 006102 010246 28: MOV    R2,-(SP) ;SAVE R2 TEMP
2796 006104 005777 172340 TST    BSWR ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
2797                ;ON ERROR
2798 006110 100405          BMI    48 ;IF SO THEN HALT ON ERROR
2799                ;CHECK FOR CONTROL=C KEY
2800
2801 006112 004767 001546 JSR    PC,CHECKC ;IF CONTROL=C TYPED THEN PRINT ERROR HISTORY
2802                ;AND HALT AT FATHLT.
2803 006116 105737 000042 78: TST    B 42 ;ARE WE RUNNING UNDER ACT?
2804 006122 001401          BEQ    68 ;BRANCH IF NO
2805
2806 006124 000000 46: HALT ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2807                ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2808                ;THE WORD STORED IN R0
2809 006126 012602 68: MOV    (SP)+,R2 ;RESTORE R2
2810 006130 062716 000002 ADD    #2,(SP) ;RESTORE THE RETURN ADDRESS
2811 006134 000207          RTS    PC ;RETURN FROM THE SUBROUTINE
2812
2813
2814
2815 006136          FATERR:
2816 006136 004767 000416 SEQERR: JSR    PC,TPCRLF ;TYPE "ERROR #"
2817 006142 051105 047522 020122 ,ASCIZ /ERROR #/
2818                ,EVEN
2819
2820 006152 017637 000000 000402 MOV    0(SP),R##FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2821 006160 105237 000314 INCB   0#TYPCNT ;TELL $TPNUM TO TYPE 1 WORD
2822 006164 012703 000376 MOV    #376,R3 ;$TPNUM USES R3 AS STACK
2823 006170 013743 000402 MOV    R##FATAL,-(R3) ;PUT ERROR NO. ON STACK
    
```

```

2824 006174 005743          TST      -(R3)          ;#TPNUM REQUIRES THIS
2825 006176 004767 000562    JSR      PC,FATYP     ;TYPE ERROR NO.
2826 006202 105737 000420    APTHLT: TSTB     00#ENV ;RUNNING UNDER APT?
2827 006206 001326          BNE      FNDERR       ;BRANCH IF YES
2828 006210 000000          FATHLT: HALT     ;FATAL ERROR OR "C HALT,
2829 006212 000137 000250    JMP      00#RESTRT    ;RESTART TST BUT DON'T CLEAR PASS COUNT
2830                                     ;IN CASE "C RESTART.
2831
2832
2833 ;PARERR
2834 ; PARITY TRAP HANDLER
2835 ;COME HERE FROM A TRAP TO 114.
2836 ; THIS ROUTINE SEARCHES THE AVAILABLE PARITY MODULES AND IF ONE
2837 ;HAS A PARITY ERROR BIT SET THE GET THE PARITY ERROR ADDRESS
2838 ;AND CALL THE "ERROR" ROUTINE TO PRINT ERROR MESSAGE.
2839 ;IF NO PARITY ERROR BITS CAN BE FOUND A FATAL ERROR IS DONE.
2840 ;
2841 ;REGISTER US AGE.
2842 ;R0= HOLDS PARITY MODULE ADDRESSES
2843 ;R1= GETS ERROR ADDRESS FOR "ERROR" CALL.
2844
2845 006216 012637 000356    PARERR: MOV      (SP)+,0#PARSP ;SET PARSP TO RETURN ADDRESS
2846 006222 011637 000360    MOV      (SP),0#PARPS     ;SAVE PSW FOR RETURN
2847 006226 013706 000350    MOV      00#SAVR6,SP      ;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
2848                                     ;TO COMPLETE THE ERROR SERVICE ROUTINE.
2849 006232 010067 000132    MOV      R0,SAVR0         ;SAVE R0 DURING PARITY SERVICE
2850 006236 010167 000130    MOV      R1,SAVR1         ;SAVE R1 DURING PARITY SERVICE
2851 006242 013701 000352    MOV      0#PARMAP,R1     ;GET PARITY AVAILABLE MAP
2852 006246 012700 172100    MOV      #172100,R0       ;R0= FIRST PARITY ADDRESS.
2853
2854 006252 005701          TST      R1              ;ANY PARITY MODULES AVAILABLE?
2855 006254 001442          BEQ      4$              ;BR IF NO -FATAL ERROR-
2856 006256 000241          CLC
2857 006260 006001          ROR      R1              ;SHIFT PARITY MAP BIT INTO C BIT.
2858 006262 103005          BCC      2$              ;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
2859 006264 005710          TST      (R0)            ;PARITY MODULE ERROR BIT SET?
2860 006266 100406          BMI      3$              ;BRANCH IF YES -CALL "ERROR" ROUTINE
2861 006270 020027 172136    CMP      R0,#172136      ;DONE ALL PARITY MODULES?
2862 006274 002032          BGE      4$              ;BR IF YES- GO TO FATAL ERROR CALL-
2863 006276 062700 000002    2$: ADD      #2,R0        ;POINT TO NEXT PARITY ADDRESS
2864 006302 000766          BR       1$              ;AND KEEP TRYING
2865 006304 042710 100000    3$: BIC      #100000,(R0)  ;CLEAR PARITY ERROR BIT.
2866 006310 011001          MOV      (R0),R1         ;GET PARITY MODULE CSR
2867 006312 006101          ROL      R1              ;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
2868 006314 006101          ROL      R1
2869 006316 006101          ROL      R1
2870 006320 006101          ROL      R1
2871 006322 042701 000777    BIC      #777,R1 ;SAVE ERROR ADDRESS ONLY
2872 006326 105237 000300    INCB    00#PRERR         ;TELL "ERROR" PARITY ERROR CALL.
2873 006332 004767 177242    JSR      PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2874 (1) 006336 000050          S0
2875                                     ;*****ERROR NUMBER 50*****
2874 006340 016700 000024    MOV      SAVR0,R0        ;RESTORE R0
2875 006344 016701 000022    MOV      SAVR1,R1        ;RESTORE R1
2876 006350 013746 000360    MOV      0#PARPS,-(SP)   ;SET RETURN PSW ON STACK
2877 006354 013746 000356    MOV      0#PARSP,-(SP)   ;AND SET RETURN ADDRESS ON STACK
    
```

```

2878 006360 000002          RTI                      ;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
2879
2880 ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
2881 006362          4$:
(1) 006362 004767 177550    JSR      PC,FATERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 006366 000051          S1
(1)                                     ;*****ERROR NUMBER 51*****
2882
2883 ;R0+R1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
2884 ;STACK SPACE BETWEEN 500-450.
2885 006370 000000          SAVR0: 0                ;SAVE R0 DURING PARITY TRAP SERVICE
2886 006372 000000          SAVR1: 0                ;SAVE R1 DURING PARITY TRAP SERVICE
2887
2888
2889 006374 105737 000277    TPADER: TSTB     0#TYPENB ;TYPE ERROR?
2890 006400 001406          BEQ      1$              ;BRANCH IF NO
2891 006402 004767 000160    JSR      PC,PNTMES       ; TYPE CR, LF AND "ADR ER"
2892 006406 042101 020122 051105 ;ASCIZ /ADR ERR/
2893                                     ;
2894 006416 000207          1$: .EVEN
2895                                     RTS      PC
2896
2896 006420 105737 000277    TPPRER: TSTB     0#TYPENB ;ERROR PRINTOUTS ALLOWED?
2897 006424 001406          BEQ      1$              ;BRANCH IF NO
2898 006426 004767 000134    JSR      PC,PNTMES       ;GO TO TYPE CR, LF AND "PAR ERR"
2899 006432 040520 020122 051105 ;ASCIZ /PAR ERR/
2900                                     ;
2901 006442 000207          1$: .EVEN
2901                                     RTS      PC
    
```



```

2906
2907
2908
2909
2910
2911
2912
2913
2914 006444 010146          NOTYP: MOV    R1,-(SP)
2915 006446 016601 000002    MOV    2(SP),R1
2916 006452 105721          4$: TSTB  (R1)+      ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
2917 006454 001376          BNE    4$           ;PREPARE TO RETURN
2918 006456 000412          BR     RETTYP
2919 006460 010146          $TYPE: MOV    R1,-(SP) ;SAVE R1
2920 006462 010046          MOV    R0,-(SP)      ;AND R0 ON THE STACK
2921 006464 016601 000004    MOV    4(SP),R1      ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
2922 006470 112100          2$: MOVB  (R1)+,R0    ;PLACE THE BYTE TO BE TYPED IN R0
2923 006472 001403          BEQ    4$           ;IF IT IS END OF MESSAGE THEN GO TO 4$
2924 006474 004767 000022    JSR    PC,$TPCHR     ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
2925 006500 000773          BR     2$
2926 006502 012600          4$: MOV    (SP)+,R0   ;RESTORE R0
2927 006504 005201          RETTYP: INC   R1      ;CAUSE R1 TO
2928 006506 042701 000001    BIC    #1,R1         ;POINT TO EVEN ADDRESS
2929 006512 010166 000002    MOV    R1,2(SP)     ;MODIFY THE RETURN ADDRESS
2930 006516 012601          MOV    (SP)+,R1     ;RESTORE R1
2931 006520 000416          BR     EXTYP        ;AND RETURN VIA RTS PC
2932
2933 006522 132737 000040 000421 $TPCHR: BITB  #40,0##ENVM ;HAVE TYPE OUTS BEEN DISABLED?
2934 006530 001005          BNE    4$           ;IF SO THEN RETURN FROM THE SUBROUTINE
2935 006532 105737 177564          2$: TSTB  0##TPS     ;WAIT HERE
2936 006536 100375          BPL    2$           ;UNTIL THE PRINTER IS READY
2937 006540 110037 177566          MOVB  R0,0##STPB    ;LOAD DATA TO BE TYPED INTO DATA REG.
2938 006544 000404          BR     EXTYP        ;RETURN
2939
2940 006546 004767 177706          PCRLF: JSR    PC,$TYPE ;TYPE CR/LF
2941 006552 005015 000     ,ASCIZ  <15><12> ;CR/LF
2942 006556          ,EVEN
2943 006556 000207          EXTYP: RTS     PC    ;RETURN
2944
2945 006560 004767 177762          TPCRLF: JSR   PC,PCRLF ;TYPE CR/LF
2946 006564 000735          BR     $TYPE       ;NOW GO TO TYPE THE REST OF THE MESSAGE
2947
2948
2949 006566 032777 000020 171654 PNTMES: BIT  #20,0SWR   ;PRINTOUTS ALLOWED?
2950 006574 001323          BNE    NOTYP        ;BRANCH IF NO
2951 006576 123737 000042 000046  CMPB  #42,0#46     ;RUNNING UNDER ACT 11?
2952 006604 001717          BEQ    NOTYP        ;BRANCH IF YES -NOT PRINTOUT-
2953 006606 000764          BR     TPCRLF      ;SEND CR/LF AND TYPE MESSAGE,
    
```

```

2958
2959
2960
2961
2962
2963
2964
2965
2966 006610 004767 177732          TYPDEC: JSR    PC,PCRLF ;TYPE CR/LF
2967
2968 006614 005046          $TPDEC: CLR    -(SP)
2969 006616 013746 000312    MOV    0#DECRD,-(SP) ;GET THE WORD THAT HAS TO BE CONVERTED TO A
2970                                ;DECIMAL NUMBER
2971 006622 162716 000012          2$: SUB    #10,.(SP)
2972 006626 002403          BLT    4$           ;IF THE NUMBER IN (SP) WAS LESS THAN 10, THEN
2973                                ;GO TO 4$
2974 006630 005266 000002          INC    2(SP)        ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
2975 006634 000772          BR     2$           ;AND RETURN TO 2$
2976 006636 062716 000012          4$: ADD    #10,.(SP)
2977 006642 052716 000060          BIS    #60,(SP)     ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2978 006646 112667 000020          MOVB  (SP)+,6$-2    ;PLACE THE 1'S DIGIT TO BE TYPED
2979 006652 052716 000060          BIS    #60,(SP)     ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
2980 006656 112667 000007          MOVB  (SP)+,6$-3    ;PLACE THE 10'S DIGIT TO BE TYPED
2981 006662 004767 177572          JSR    PC,$TYPE     ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
2982                                ;3 SPACES
2983 006666 020040 030040 000060    ,ASCIZ  / 00/
2984 006666          ,EVEN
2985 006674 000207          6$: RTS     PC      ;RETURN FROM THE SUBROUTINE
    
```

```

2990
2991
2992
2993
2994
2995
2996
2997
2998
2999
3000
3001
3002
3003
3004
3005 006676 032777 020000 171544 TYPERR: BIT #20000,0SWR ;ERROR PRINTOUT WANTED?
3006 006704 001054 BNE OCTXT ;BRANCH IF NO
3007 006706 004767 177634 JSR PC,PCRLF ;TYPE CR/LF
3008 006712 004767 000012 JSR PC,TYPOCT ;TYPE OCTAL NO.
3009 006716 000447 BR OCTXT ;RETURN VIA RTS PC
3010 006720 012123 OCTTYP: MOV (R1)+,(R3)+ ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
3011 ;BY R3
3012 006722 012113 MOV (R1)+,(R3) ;AND NOW PLACE THE LOW ORDER BITS
3013 006724 105237 000314 INCB #TYPCNT ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3014 006730 052743 000004 TYPOCT: BIS #4,(R3)
3015 006734 106113 28: ROLB (R3)
3016 006736 103376 BCC 28
3017 006740 005000 CLR R0
3018 006742 106113 ROLB (R3) ;GET BITS 17 & 16 INTO R0
3019 006744 006100 ROL R0
3020 006746 106113 ROLB (R3)
3021 006750 006100 ROL R0
3022 006752 000405 BR $TPNUM
3023 006754 004767 177500 RPTOCT: JSR PC,$TYPE ; TYPE 3 SPACES
3024 006760 020040 000040 ,ASCIZ / /
3025 ,EVEN
3026 006764 005000 FATYP: CLR R0
3027 006766 012723 000006 $TPNUM: MOV #6,(R3)+ ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3028 006772 000241 48: CLC
3029 006774 006113 ROL (R3)
3030 006776 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
3031 007000 052700 000060 BIS #60,R0 ;OR THE CONTENTS OF R0 WITH AN ASCII 0
3032 007004 004767 177512 JSR PC,$TPCHR ; TYPE THE OCTAL NUMBER STORED IN R0
3033 007010 005000 CLR R0
3034 007012 006113 ROL (R3)
3035 007014 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
3036 007016 006113 ROL (R3)
3037 007020 006100 ROL R0 ;PLACE THE CARRY FROM (R3) IN R0
3038 007022 105363 177776 DECB =2(R3) ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3039 007026 001361 BNE 48 ;THEN REPEAT FROM 48
3040 007030 105337 000314 DECB #TYPCNT ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
3041 007034 001347 RPTOCT BNE RPTOCT ;TYPED THEN REPEAT FROM RPTOCT
3042 007036 000207 OCTXT: RTS PC
    
```

```

3047
3048
3049
3050
3051
3052
3053
3054
3055
3056 007040 012702 001400 MEMMNG: MOV #1400,R2
3057 007044 105037 000276 MMREG: CLR B #MMMAVA ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
3058 ;THAT MEM, MANAG, IS AVAILABLE FOR TESTING
3059 007050 032777 010000 171372 BIT #10000,0SWR ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG, ?
3060 007056 001441 BEQ RETMM ;IF NOT THEN RETURN FROM THE SUBROUTINE
3061 007060 012700 000004 MOV #4,R0 ;PREPARE TO SETUP TIME OUT VECTOR
3062 007064 012720 007164 MOV #NOMM,(R0)+ ;RETURN ADDRESS TO NOMM
3063 007070 012710 000340 MOV #340,(R0) ;AND WITH A PSW OF 340
3064 007074 005037 177572 CLR #SR0 ;TRY TO REACH MEM, MANAG, SR0
3065 007100 105237 000276 INCB #MMMAVA ;IF IT IS AVAILABLE THEN SET MEM, MANAG, AVAILABLE
3066 ;BYTE
3067 007104 012701 172340 MOV #172340,R1 ;R1 IS POINTING TO PAR0
3068 007110 005021 CLR (R1)+ ;PAR0 WILL POINT TO BANK 0
3069 007112 062702 000200 28: ADD #200,R2
3070 007116 010221 MOV R2,(R1)+ ;SETUP PAR1-PAR6
3071 007120 020127 172356 CMP R1,#172356
3072 007124 103772 BLO 28
3073 007126 012711 007600 MOV #7600,(R1) ;PAR7 IS POINTING TO THE I/O PAGE
3074 007132 012701 172300 MOV #172300,R1
3075 007136 012721 007406 48: MOV #77406,(R1)+ ;SETUP PDR0-PDR7
3076 007142 020127 172316 CMP R1,#172316
3077 007146 101773 BLOS 48
3078 007150 005237 177572 INC #SR0 ;ENABLE MEM, MANAG.
3079 007154 005010 $RETMM: CLR (R0) ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
3080 007156 012740 000104 MOV #BUSER,-(R0)
3081 007162 000207 RETMM: RTS PC
3082
3083 007164 022626 NOMM: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
3084 007166 004767 177366 JSR PC,TPCRLF ;TYPE "NO MEMORY MANAGEMENT MESSAGE
3085 007172 047516 046440 043516 ,ASCIZ /NO MNG/
3086 007202 ,EVEN
3087 007202 004767 176730 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
3088 (1) 007206 000052 52 ;*****ERROR NUMBER 52*****
3089 BR $RETMM ; RESTORE TIME OUT TRAP VECTOR
3090 007212 013702 172354 UPMN: MOV #172354,R2
3091 007216 000712 BR MMREG
    
```

```

3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106
3107 007220 005063 177776 PUTADR: CLR =2(R3)
3108 007224 010113 MOV R1,(R3) ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
3109 007226 105737 000276 TSTB @#MMAVA ;IS THE MEM, MANAG, AVAILABLE ?
3110 007232 001425 BEQ 68 ;IF NOT THEN RETURN FROM THE SUBROUTINE
3111 007234 010146 MOV R1,=(SP) ;SAVE R1
3112 007236 042701 017777 BIC #17777,R1 ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
3113 007242 040113 BIC R1,(R3) ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
3114 007244 052701 004000 BIS #4000,R1 ;PREPARE TO SHIFT R1 BY 12 PLACES
3115 007250 006001 28: ROR R1
3116 007252 103376 ROR R1 ;GET THE NUMBER OF PAR IN R1
3117 007254 062701 172340 ADD #172340,R1 ;GET THE ADDRESS OF PAR IN R1
3118 007260 011101 MOV (R1),R1 ;LOAD R1 WITH THE CONTENTS OF PAR
3119 007262 052701 010000 BIS #10000,R1
3120 007266 006101 48: ROL R1
3121 007270 103376 ROR R1 ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
3122 007272 006101 ROL R1
3123 007274 006143 ROL =-(R3) ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
3124 007276 006101 ROL R1
3125 007300 006123 ROL (R3)+ ;PLACE BIT 16 OF THE ADDRESS
3126 007302 050113 BIS R1,(R3) ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
3127 007304 012601 MOV (SP)+,R1 ;RESTORE R1
3128 007306 000207 68: RTS PC ;RETURN FROM THE SUBROUTINE
3132
3133 ;* GET ADDRESS FROM THE APT MAILBOX
3134 ;* -----
3135 ;*
3136 ;* THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
3137 ;* PLACE IT IN THE LOCATION USED BY THE PROGRAM TO DEFINE THE
3138 ;* MEMORY BOUNDRIES.
3139 ;* PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
3140 ;* ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
3141 ;* THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
3142 ;* TO BE PLACED
3143 ;*
3144 ;*
3145 007310 016143 000001 GETADR: MOV 1(R1),-(R3) ;PLACE THE LOW ORDER BITS OF THE ADDRESS
3146 007314 005043 CLR -(R3) ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
3147 ;* HAVE TO BE PLACED
3148 007316 116113 177777 MOVB -1(R1),(R3) ;PLACE BITS 16 & 17
3149 007322 000207 28: RTS PC ;RETURN FROM THE SUBROUTINE
    
```

```

3154
3155
3156
3157
3158
3159
3160
3161
3162
3163
3164 007324 105237 000315 8GTSIZ: INCB @#SAVKBB ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3165 ;0-4 OF R2
3166
3167 007330 012301 GETSIZ: MOV (R3)+,R1
3168 007332 011302 MOV (R3),R2 ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3169 007334 042702 017777 BIC #17777,R2 ;CLEAR ADDRESS BITS 0-12
3170 007340 052702 000040 28: BIS #40,R2
3171 007344 006001 48: ROR R1
3172 007346 006002 ROR R2 ;ROTATE R1 AND R2 7 TIMES
3173 007350 103375 ROR R1
3174 007352 105737 000315 TSTB @#SAVKBB
3175 007356 001405 BEQ 68
3176 007360 105037 000315 CLR @#SAVKBB
3177 007364 052702 000100 BIS #100,R2
3178 007370 000765 BR 48
3179 007372 012301 68: MOV (R3)+,R1 ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3180 007374 012700 MOV #160000,R0
3181 007400 040001 BIC R0,R1 ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3182 007402 000207 RTS PC ;RETURN FROM THE SUBROUTINE
3183
3184
3185
3186
3187
3188
3189 ;* SUBROUTINE TO DISABLE MEMORY MANAGEMENT
3190 ;* -----
3191 ;*
3192 ;* THIS SUBROUTINE IS CALLED TO DISABLE THE MEMORY MANAGEMENT
3193 ;* UNIT
3194 ;*
3195 007404 105737 000276 CLRMM: TSTB @#MMAVA ;WAS THE MEMORY MANAGEMENT ENABLED ?
3196 007410 001404 BEQ 18 ;IF NOT THEN GO TO 18
3197 007412 005037 177572 CLR @#SR0 ;DISABLE THE MEMORY MANAGEMENT
3198 007416 105737 000276 CLRB @#MMAVA ;AND DO NOT ATTEMPT TO TEST MEM, MANAG,
3199 007422 000207 18: RTS PC ;RETURN FROM THE SUBROUTINE
3200
3201 ;* GET BANK NO, UNDER TEST
3202 ;* CALLED BY ERRYP AND TST13 TO GET BANK NO, UNDER TEST INTO PBK,
3203 ;* REGISTERS
3204 ;* R0=POINTER TO PAR UNDER TEST
3205 ;* R3=VIRTUAL ADDRESS ON ENTRY
3206 ;* R0+R3 ARE RESTORED ON EXIT.
3207
3208
3209 007424 010046 GETBNK: MOV R0,=(SP) ;SAVE R0
3210 007426 010346 MOV R3,=(SP) ;SAVE R3
3211 007430 042703 017777 BIC #17777,R3 ;SAVE ONLY VIRTUAL BANK BITS
3212 007434 052703 010000 BIS #10000,R3 ;SETUP R3 SHIPT BIT
    
```

```

3213 007440 000241          CLC
3214 007442 006003          10: ROR   R3           ;SHIFT A BANK BIT
3215 007444 103376          BCC   10           ;UNTIL IN BITS <210> OF R3
3216 007446 105737 000276  TSTB  0#HMAVA     ;MEMORY MANAGEMENT UNDER TEST?
3217 007452 001407          BEQ   20           ;NO EXIT
3218
3219
3220 007454 006303          ;GET PAR ADDRESS AND PHYSICAL BANK NO.
3221 007456 062703 172340  ADD   #172340,R3   ;MAKE R3 PAR ADDRESS OFFSET,
3222 007462 011300          MOV   (R3),R0     ;MAKE FULL PAR ADDRESS,
3223 007464 006300          ASL   R0           ;GET PAR CONTENTS
3224 007466 000300          SWAB  R0           ;SHIFT BANK BITS TO BITS <710>
3225 007470 110003          MOVB  R0,R3       ;SET R3 TO PHYSICAL BANK NO.
3226 007472 010337 000312 20: MOV   R3,0#PBNK   ;STORE PHYSICAL BANK NO.
3227 007476 012603          MOV   (SP)+,R3    ;RESTORE R3
3228 007500 012600          MOV   (SP)+,R0    ;RESTORE R0
3229 007502 000207          RTS   PC           ;RETURN TO CALLER
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252 007504 032777 004000 170736 PARITY: BIT  #4000,0SWR   ;PARITY TEST WANTED?
3253 007512 001460          BEQ   60           ;BRANCH IF NO
3254
3255 007514 012700 000004          MOV   #4,R0       ;POINT R0 TO BUS TIMEOUT ADDRESS.
3256 007520 012710 000122          MOV   #50,-6,(R0) ;SET RETURN FROM TIMEOUT TRAP TO 50
3257 007524 060710          ADD   PC,(R0)     ;IN THE CURRENT BANK,
3258 007526 005037 000352          CLR   0#PARMAP   ;CLEAR PARITY MAP HOLDER,
3259 007532 012701 172140          MOV   #172140,R1  ;SET R1 TO LAST PARITY MODULE ADDRESS+2
3260 007536 012703 100000          MOV   #100000,R3  ;SET R3 TO PARMAP AVAILABLE CODE BEGIN,
3261 007542 010241          MOV   R2,-(R1)    ;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE,
3262 007544 050337 000352          BIS   R3,0#PARMAP ;NO TRAP TO 50, SO SET PARITY AVAILABLE.
3263 007550 000241          CLC
3264 007552 006003          30: ROR   R3           ;SETUP NEXT PARMAP BIT
3265 007554 103372          RCC   20           ;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
3266 007556 012710 000104          MOV   #BUSER,(R0) ;RESET BUS TIMEOUT TRAP VECTOR
3267 007562 005702          TST  R2           ;IS THIS A DISABLE CALL?
3268 007564 001433          BEQ   60           ;BRANCH IF YES (EXIT)
  
```

```

3269 007566 005737 000352          TST  0#PARMAP     ;WERE ANY PARITY MODULES FOUND?
3270 007572 001011          BNE  40           ;BRANCH IF YES
3271 007574 004767 176760          JSR  PC,TPCRLF    ;PRINT "NO PAR"
3272 007600 047516 050040 051101  .ASCIZ /NO PAR/
3273 007606 000
3274 007610 007610 176322          .EVEN
3275 007614 000053          JSR  PC,FATERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
3276 007616 152737 000040 000405 40: .EVEN          ;*****ERROR NUMBER 53*****
3277 007624 012737 000376 000316 40: BISB  #40,0#REL   ;SET PARITY UNDER TEST FLAG
3278 007632 004767 176722          MOV   #376,0#BAKPAT ;SET BACKGROUND PATTERN TO
3279 007636 040520 044522 054524  JSR  PC,TPCRLF    ;WORST CASE PARITY CODE.
3280 007644 000
3281 007646 007646          .ASCIZ /PARITY/   ;PRINT "TST PARITY"
3282 007646 000405          .EVEN
3283          BR    EXIT      ;AND EXIT VIA RTS PC
3284
3285          ;GET HERE IF PARITY ADDRESS TIMED OUT TO LOC. 4
  
```

```

3296 007650 022626          58:  CMP      (SP)+,(SP)+      ;RESET STACK FROM TRAP
3297 007652 000737          BR      38          ;KEEP TRYING PARITY ADDRESSES.
3298
3299 007654 142737 000040 000405 68:  BICB    #40,0*REL      ;CLEAR PARITY TESTING FLAG
3300 007662          EXITC:
3301 007662 000207          78:  RTS     PC          ;RETURN TO CALLER
3302
3303
3304
3305
3306          ;CHECKC
3307          ; THIS ROUTINE CHECKS IF CONTROL=C WAS TYPED AT THE END OF EACH
3308          ;TEST OR IN THE ERROR TYPE ROUTINE.
3309          ;IF CONTROL=C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
3310          ;RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
3311          ;FINALLY IT HALTS AT FATHLT.
3312
3313 007664 105037 000315      CHECKC: CLRB   0*SAVKBB      ;INIT CONTROL-C FLAG.
3314 007670 105737 177560      TSTB   0*TKS          ;ANY CHAR. TYPED?
3315 007674 100372          BPL    EXITC         ;BR IF NO-EXIT VIA RTS PC-
3316 007676 113702          MOVB   0*#KBB,R2     ;GET THE CHAR TYPED.
3317 007702 042702 000200      BIC    #200,R2 ;CLEAR THE PARITY BIT.
3318 007706 122702 000003      CMPB   #3,R2        ;IS IT CONTROL-C?
3319 007712 001363          BNE    EXITC         ;BRANCH IF NO -EXIT VIA RTS PC-
3320 007714 110237 000315      MOVB   R2,0*SAVKBB   ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
3321 007720 004767 176634      JSP    PC,TPCRLF     ;PRINT "C"
3322 007724 041536 000          .ASCIZ /"C/
3323          .EVEN
3324 007730 000167 175104      JMP    RELOER        ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
3325
3326          .=7744
3327 007744 000000      ENDPRG: 0
3328
3329          ;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
3330          ;STACK, FOR EACH 4K BANK 18, BYTES ARE SAVED.
3331          ;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
3332          ;AFTER THE ERROR STACK.
3333          ;FOR 4K MEMORY SIZE THEN PROGRAM=7744+22=7776
3334
3335
3336
3337
3338          .END
  
```

| | | | | |
|-----------------|---------------|----------------|----------------|-----------------|
| ABASE = 000000 | BRTPSZ 001012 | NOTYP 006444 | START 000200 | SETABL 000420 |
| ACDW1 = 000000 | BUSER 000104 | OCTYTP 006720 | STRTDI 000302 | SETEND 000450 |
| ACDW2 = 000000 | CHECKC 007664 | OCTXT 007036 | SWAPAT 000320 | SFATAL 000402 |
| ACPUOP = 000000 | CKDONE 005106 | ONFAPS 000556 | SMHALT 001664 | SGTSIZ 007324 |
| ACT11 = 005522 | CLRMEM 001466 | PARERP 006216 | SMR 000450 | SHD = 000002 |
| ADDW0 = 000000 | CLRMM 007404 | PARFL 005432 | SMREG 000176 | SHBTS 000276 |
| ADDW1 = 000000 | CNTSCP 001644 | PARITY 007504 | SM11 = 004000 | SHIMAX 000334 |
| ADDW10 = 000000 | CONT 001526 | PARMAP 000352 | TBL 001600 | SKBB = 177562 |
| ADDW11 = 000000 | CONTMM 005134 | PARPS 000360 | TKS = 177560 | SMADR1 000432 |
| ADDW12 = 000000 | CTLC 005570 | PARSP 000356 | TPADER 006374 | SMADR2 000436 |
| ADDW13 = 000000 | DECWRD 000312 | PASFLG 000306 | TPCRLF 006560 | SMADR3 000442 |
| ADDW14 = 000000 | ENDPAS 005276 | PRNK 000312 | TPPRER 006420 | SMADR4 000446 |
| ADDW15 = 000000 | ENDPRG 007744 | PC = 000007 | TPYSR 001014 | SMAIL 000400 |
| ADDW2 = 000000 | ENDSTK 000310 | PCRLF 006546 | TSTGO 001666 | SMAMS1 000430 |
| ADDW3 = 000000 | END0 002132 | PNTMES 006566 | TSTMM 005120 | SMAMS2 000434 |
| ADDW4 = 000000 | END1 002232 | PUTADR 007220 | TSTREL 001320 | SMAMS3 000440 |
| ADDW5 = 000000 | END10 003772 | PWRDN 000070 | TSTRP 000570 | SMAMS4 000444 |
| ADDW6 = 000000 | END12 004316 | PWRUP 000136 | TSTSCP 001632 | SMAX 000336 |
| ADDW7 = 000000 | END2 002342 | REL = 000405 | TSTSIZ 001324 | SMADR 000300 |
| ADDW8 = 000000 | END3 002610 | RELBOT 000322 | TST0 001702 | SMGAD 000414 |
| ADDW9 = 000000 | END4 002720 | RELOC 004746 | TST1 002134 | SMGLG 000416 |
| ADEVCT = 000000 | END5 003066 | RELOER 005040 | TST10 003446 | SMSGT 000400 |
| ADEVN = 000000 | END6 003222 | RESTR 000250 | TST11 003774 | SMIYP1 000431 |
| AENV = 000000 | END7 003444 | PETMM 007162 | TST12 004046 | SMIYP2 000435 |
| AENVN = 000000 | EKROR 005600 | RETSTK 005336 | TST13 004320 | SMIYP3 000441 |
| AFATAL = 000000 | ERRTYP 005716 | RETTY 006504 | TST2 002234 | SMIYP4 000445 |
| AMADR1 = 000000 | EXITC 007662 | RLODER 005546 | TST3 002344 | SNWTST = 000001 |
| AMADR2 = 000000 | EXTYP 006556 | RPTOCT 006754 | TST4 002612 | SPASS 000406 |
| AMADR3 = 000000 | FAILNM 005374 | RPT10 003466 | TST5 002722 | SPASTH 000304 |
| AMADR4 = 000000 | FATERR 006136 | RPT11 004032 | TST6 003070 | SPRERR 000300 |
| AMAMS1 = 000000 | FATHLT 006210 | RPT6 003106 | TST7 003224 | SPRETM 007154 |
| AMAMS2 = 000000 | FATYP 006764 | R0 = 00000000 | TYPCNT 000314 | SSVPC = 000044 |
| AMAMS3 = 000000 | FNDEPR 006064 | R1 = 00000001 | TYPDEC 006610 | SSWR = 000000 |
| AMAMS4 = 000000 | GALLOP 003472 | R2 = 00000002 | TYPENB 000277 | SSWREG 000422 |
| AMSGAD = 000000 | GETADR 007310 | R3 = 00000003 | TYPEOP 005470 | STESTN 000404 |
| AMSLG = 000000 | GETBNK 007424 | R4 = 00000004 | TYPERR 006676 | STN = 000014 |
| AMSGTY = 000000 | GETSIZ 007330 | R5 = 00000005 | TYPHEM 001152 | STPB = 177566 |
| AMTYP1 = 000000 | HIGHAD 000332 | SAVKBB 000315 | TYPPOCT 006730 | STPCHR 000522 |
| AMTYP2 = 000000 | HIGHTW 000330 | SAVLDR 001242 | TYPSIZ 001130 | STPDEC 000614 |
| AMTYP3 = 000000 | LOOP 001536 | SAVLOC 000354 | TYPSTK 005316 | STPNUM 006766 |
| AMTYP4 = 000000 | LOWADD 000326 | SAVMAX 000342 | UPMM 007212 | STPS = 177564 |
| APASS = 000000 | LOWBNR 000304 | SAVR0 006370 | WRHEM = 000120 | STPSTK 005306 |
| APRTOR = 000000 | LOWER 005112 | SAVP1 006372 | SA = 000001 | STSTM 000302 |
| APTHLT 006202 | LOWTWO 000324 | SAVR4 000344 | \$ADERR 000301 | STYPE 000460 |
| APTSIZ 000724 | M = 000200 | SAVR5 000346 | \$APTHD 000276 | STUNIT 000412 |
| ASWREG = 000000 | MAXADR 005224 | SAVR6 000350 | \$CNTMM 005140 | SUNITH 000306 |
| ATESTN = 000000 | MAXHEM 000340 | SCOPE = 000240 | \$CPUOP 000426 | SUSNR 000424 |
| AUNIT = 000000 | MEMMNG 007040 | SEGERR 006136 | \$DEVCT 000410 | SZ = 000362 |
| AUSR = 000000 | MEMTST 001460 | SETSTK 001164 | \$DOAGN 005542 | SZZ = 007734 |
| AVECT1 = 000000 | MMAVA 000276 | SETSWR 000656 | \$ENDAD 000156 | SM = 000200 |
| AVECT2 = 000000 | MMREG 007044 | SLSFTZ 001024 | \$ENV 000420 | \$X = 000276 |
| BAKPAT 000316 | N = 000054 | SP = 00000006 | \$ENVN 000421 | . = 007746 |

DZKMA MACY11 27(732) 05-AUG-76 13:43 PAGE 34-1
DZKMAB,P11 SYMBOL TABLE

SEQ 0074

DEFAULT GLOBALS GENERATED: 0

*DZKMAB,DZKMAB=DZKMAB,P11
RUN-TIME: 25 26 ,8 SECONDS
RUN-TIME RATIO: 104/54=1,9
CORE USED: 12K (23 PAGES)