PDP11

MEMORY EXERCISER

MD-11-DZKMA-C

EP-DZKMA-C-DL-B JUN 1977

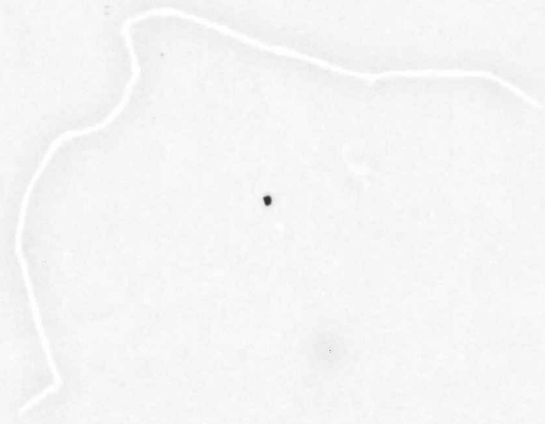
FICHE 1 OF 1

digital

MADE IN USA

B01

EOF1DZLPKGSEQ	00010000	770608	PDP10 411	HDR1DZKMACSEQ	00010000	770608



```
        .REPT   0
```

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZKMA-C-D
PRODUCT NAME:	MOS/CORE MEMORY EXERCISER FOR 0 TO 124K WITH OR WITHOUT PARITY BITS
DATE CREATED:	FEBRUARY-1977
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	PERVEZ A. ZAKI
REVISED:	FRED STRAIGHT DAN CASALETTO (AUGUST 1976)

CONTENTS

[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 28K (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.
!CAUTION! INCREASES TEST TIME BY FACTOR OF 25.
BIT06(000100) 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-C" 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-YYYYY"           ;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
                        ;STESTN 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] USED/CONTENT	UNIBUS ADDRESS
0	0 - 4K	000000-017776	0 0000	772340
1	4K - 8K	020000-037776	NOT USED	
2	8K-12K	040000-057776	NOT USED	
3	12K-16K	060000-077776	NOT USED	
4	16K-20K	100000-117776	NOT USED	
5	20K-24K	120000-137776	NOT USED	
6	24K-28K	140000-157776	NOT USED	
7	28K-32K	160000-177776	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	7 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	7 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. [ISTRP] 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 28K.

5. [TYPSIZ] TYPE MEMORY TEST LIMITS.

6. [SETSTK] SPACE IS SAVED AT THE END OF THE TEST FOR AN ERROR HISTORY. FOR EACH 4K BANK 18 BYTES ARE SAVED IN THE FOLLOWING FORMAT:

!ADR ERR!	PAR ERR!
!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=40000 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. [SEOP] 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-C" 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
```

```
DZKMA   MACY11 27(1006)  18-FEB-77  12:35  PAGE 18
DZKMAC.P11    18-FEB-77 12:30

                    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
                .NLIST  MD,MC,CND
  985                                   .LIST   ME,BIN,SEQ,LOC
  986                                   .TITLE  DZKMA
  987                                   ;*COPYRIGHT (C) JANUARY 1976
  988                                   ;*DIGITAL EQUIPMENT CORP.
  989                                   ;*MAYNARD, MASS. 01754
  990                                   ;*
  991                                   ;*PROGRAM BY PERVEZ ZAKI
  992                                   ;*
  993                                   ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
  994                                   ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
  995                                   ;*
  996           160000                  $SWR=160000     ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYPOUT
  997
  998
  999
 1000                                           ;;TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS
 1001
 1002
 1003
 1004
 1005
 1006           000240                  SCOPE   =NOP
 1007
 1008           000042                          .=42
 1009  000042   000000                          .WORD   0       ;FOR ACT/XXDP
 1010
 1011                                   .SBTTL  ACT11 HOOKS
 1012
 1013                                   ;;**************************************************************
 1014                                   ;HOOKS REQUIRED BY ACT11
 1015           000044                          $SVPC=.                         ;SAVE PC
 1016           000046                          .=46
 1017  000046   000156                          $ENDAD                          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
 1018           000052                          .=52
 1019  000052   040000                          .WORD   40000                   ;;2)SET LOC.52 TO 40000
 1020           000044                          .=$SVPC                         ;; RESTORE PC
 1021
 1022           000070                          .=70
 1023  000070   012737  000136  000024  PWRDN:  MOV     #PWRUP,@#24
 1024  000076   000000                          HALT
 1025
```

```
1026
1027
1028            000104                  .=104
1029                                    ; GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
1030  000104  005237  000400    BUSER:  INC     @#$MSGTY        ;TELL APT FATAL ERROR#000
1031  000110  000000                    HALT                    ;*ERROR* TRAP TO LOC. 4 OCCURRED.
1032  000112  000000                    HALT                    ;IN CASE CONTINUE PRESSED.
1033                                    ;114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS.  SETUP IN
1034                                    ;ROUTINE "BEGIN".
1035            000120                  .=120
1036
1037
1038                                    ;* WRITE MEMORY BACKGROUND
1039                                    ;* ----------------------
1040                                    ;*
1041                                    ;*      THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
1042                                    ;*      THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES
1043                                    ;*      THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE
1044                                    ;*      HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE
1045                                    ;*      SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
1046                                    ;*
1047
1048
1049  000120  010401    WRTMEM: MOV     R4,R1           ;SET R1 TO LOWEST LOCATION UNDER TEST
1050  000122  013700  000316            MOV     @#BAKPAT,R0     ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
1051  000126  010021    2$:     MOV     R0,(R1)+        ;STARTING FROM THE LOWEST LOCATION WRITE THE
1052  000130  020105            CMP     R1,R5           ;MEMORY TO BACK GROUND PATTERN
1053  000132  103775            BLO     2$
1054  000134  000207            RTS     PC              ;RETURN FROM THE SUBROUTINE
1055
1056
1057  000136  013706  000350    PWRUP:  MOV     @#SAVR6,SP      ;RESTORE STACK POINTER
1058  000142  012700  006066            MOV     #PNTMES-BEGIN,R0
1059  000146  060600                    ADD     SP,R0           ;GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
1060                                                            ;RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE
1061  000150  004710                    JSR     PC,(R0)         ;GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A "P"
1062  000152  000120                    .ASCIZ  /P/
1063                                    .EVEN
1064
1065  000154  000411                    BR      START
1066
1067                                    ;* SERVICE XXDP/ACT11
1068  000156  004710    $ENDAD: JSR     PC,(R0)         ;RETURN TO ACT11/XXDP MONITOR
1069  000160  000240                    NOP                     ;IF QUICK VERIFY=RESET ELSE NOP
1070  000162  000240                    NOP                     ;IF QUICK VERIFY=CLR @#-1 ELSE INC @#0
1071  000164  000240                    NOP                     ;IF QUICK VERIFY=BR .-4 ELSE NOP
1072  000166  000430                    BR      RESTRT          ;REPEAT TEST UNDER ACT11/XXDP
1073
1074            000176                  .=176
1075  000176  000000    SWREG:  .WORD   0
1076
1077
1078                                    ;;*****************************************************************
1079                                    .SBTTL  START AND RESTART ROUTINES
1080                                    ;*      RESTART AT 200 TO CLEAR APT TABLES
1081                                    ;;*****************************************************************
```

```
  1082  000200  013706  000350          START:  MOV     @#SAVR6,SP      ;SETUP STACK POINTER.
  1083  000204  012703  000412                  MOV     #$UNIT,R3       ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
  1084  000210  005043                  1$:     CLR     -(R3)           ;CLEAR A MAILBOX LOCATION
  1085  000212  022703  000400                  CMP     #$MAIL,R3       ;DONE?
  1086  000216  001374                          BNE     1$              ;BRANCH IF NO
  1087  000220  105737  000042                  TSTB    @#42            ;ACT11 MODE?
  1088  000224  001011                          BNE     RESTRT          ;BRANCH IF YES
  1089  000226  105737  000405                  TSTB    @#$TESTN+1              ;ARE WE RELOCATED?
  1090  000232  100406                          BMI     RESTRT          ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
  1091  000234  004767  006320                  JSR     PC,TPCRLF       ;PRINT TITLE
  1092  000240  055104  046513  026501          .ASCIZ  /DZKMA-C/
  1093  000246  000103
  1094                                          .EVEN
  1095
  1096  000250  012704  007744          RESTRT: MOV     #ENDPRG,R4      ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
  1097  000254  012703  000346                  MOV     #SAVR5,R3       ;CAUSE R3 TO POINT TO THE LOCATION SAVR5
  1098  000260  012305                          MOV     (R3)+,R5        ;RESTORE R5
  1099  000262  012306                          MOV     (R3)+,SP        ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
  1100  000264  010600                          MOV     SP,R0           ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
  1101  000266  012746  000340                  MOV     #340,-(SP)      ;SET HIGH PRIORITY FOR RTI
  1102  000272  010046                          MOV     R0,-(SP)
  1103  000274  000002                          RTI                     ;GO TO "START"-MAY BE RELOCATED.
  1104                                                                  ;IF RELOCATED SEE LOCATION SAVR6 FOR START.
  1105
  1106
  1107
  1108
  1109
  1110
  1111
  1112
  1113                                  .SBTTL  APT PARAMETER BLOCK
  1114
  1115                                  ;;*****************************************************************
  1116                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
  1117                                  ;;*****************************************************************
  1118  000276                                  .$X=.   ;;SAVE CURRENT LOCATION
  1119  000024                                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
  1120  000024  000200                          200     ;;FOR APT START UP
  1121  000044                                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
  1122  000044  000276                          $APTHDR ;;POINT TO APT HEADER BLOCK
  1123  000276                                  .=.$X   ;;RESET LOCATION COUNTER
  1124                                  ;;*****************************************************************
  1125                                  ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
  1126                                  ;INTERFACE SPEC.
  1127
  1128  000276                          $APTHD:
  1129  000276  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
  1130  000300  000400                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
  1131  000302  000012                  $TSTM:  .WORD   10.     ;;RUN TIM OF LONGEST TEST
  1132  000304  000156                  $PASTM: .WORD   110.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
  1133  000306  000000                  $UNITM: .WORD           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
  1134  000310  000024                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
  1135
  1136
  1137          000405                  REL=$TESTN+1                    ;IT WILL BE 0 IF THE PROGRAM IS IN THE LOWER
```

```
                                ;CORE. BIT 7 OF THE BYTE WILL BE SET IF THE
                                ;PROGRAM IS IN A RELOCATED STATE AND BIT 5
                                ;WILL BE SET IF PARITY BITS ARE BEING TESTED
                    000276      .=$APTHD
000276              MMAVA:      ;THIS BYTE IS USED TO DETERMINE IF MEMORY
                                ;MANAGEMENT IS AVAILABLE OR NOT
                    000277      .=MMAVA+1
000277              TYPENB:     ;THIS BYTE IS USED TO DETERMINE IF THE
                                ;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
                    000300      .=TYPENB+1
000300              $PRERR:     ;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
                                ;A PARITY ERROR
                    000301      .=$PRERR+1
000301              $ADERR:     ;THIS BYTE IS USED TO DETERMINE IF THE
                                ;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
                    000302      .=$ADERR+1
000302              STRTDI:
                    000304      .=STRTDI+2
000304              LOWBNK:
                    000306      .=LOWBNK+2
000306              PASFLG:     ;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
                                ;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
                                ;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
                    000310      .=PASFLG+2
000310              ENDSTK:
                    000312      .=ENDSTK+2
000312              PBNK:               ;HOLDS BANK UNDER TEST FOR "TST BNK XX" PRINTOUT.
000312              DECWRD:
                    000314      .=DECWRD+2
000314    000       TYPCNT: .BYTE   0   ;THIS BYTE DETERMINES THE NUMBER OF WORDS
                                        ;TO BE TYPED
000315    000       SAVKBB: .BYTE   0   ;THIS LOCATION IS USED TO SAVE THE CHARACTER
                                        ;HIT BY THE OPERATOR
                                        ;ALSO IS USED AS TEMP IN ROUTINE $GTSIZ.
                    .EVEN

                    177560  TKS=     177560
                    177562  $KBB=    177562
                    177564  $TPS=    177564
                    177566  $TPB=    177566
                    177572  SR0=     177572
000316  000377      BAKPAT: .WORD   377     ;BACKGROUND PATTERN WRITTEN TO MEMORY.

000320  000000      SWAPAT: .WORD
000322  000430      RELBOT: BEGIN-50        ;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.

                    ;;*****************************************************************
                    ;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR
000324  000000      LOWTWO: 0               ;HOLDS BITS 17:16 OF LOW TEST ADDRESS
000326  000000      LOWADD: 0               ;HOLDS BITS 15:0 OF LOW TEST ADDRESS
```

```
1194
1195  000330  000000                HIGHTWO:        0                       ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
1196  000332  037776                HIGHADD:        37776                   ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1197                                ;;**************************************************************
1198
1199  000334  000000                $HIMAX: 0                               ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1200  000336  017776                $MAXM:  17776                           ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1201
1202  000340  000000                MAXMEM: .WORD   ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1203
1204  000342  000000                SAVMAX: .WORD
1205  000344  000000                SAVR4:  .WORD
1206  000346  000000                SAVR5:  .WORD
1207
1208                                ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1209  000350  000500                SAVR6:  .WORD   BEGIN           ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1210  000352  000000                PARMAP: 0                       ;MAP OF PARITY MODULES UNDER TEST.
1211  000354  000000                SAVLOC: 0                       ;TEST 6 STORES ERROR INFO HERE
1212  000356  000000                PARSP:  0       ;SAVE SP DURING PARITY ERROR TRAP.
1213  000360  000000                PARPS:  0                       ;SAVE PSW DURING PARITY ERROR TRAP.
1214                                                                ;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
1215                                                                ;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
1216                                                                ;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE.
1217                                                                ;IN THIS CRUDE FASHION.
1218
1219
1220                                ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
```

```
                    000400                          .=400
                                                .SBTTL  APT MAILBOX-ETABLE
                                                ;;**********************************************************************
                                                .EVEN
000400                                          $MAIL:                          ;;APT MAILBOX
000400  000000                                  $MSGTY: .WORD   AMSGTY          ;;MESSAGE TYPE CODE
000402  000000                                  $FATAL: .WORD   AFATAL          ;;FATAL ERROR NUMBER
000404  000000                                  $TESTN: .WORD   ATESTN          ;;TEST NUMBER
000406  000000                                  $PASS:  .WORD   APASS           ;;PASS COUNT
000410  000000                                  $DEVCT: .WORD   ADEVCT          ;;DEVICE COUNT
000412  000000                                  $UNIT:  .WORD   AUNIT           ;;I/O UNIT NUMBER
000414  000000                                  $MSGAD: .WORD   AMSGAD          ;;MESSAGE ADDRESS
000416  000000                                  $MSGLG: .WORD   AMSGLG          ;;MESSAGE LENGTH
000420                                          $ETABLE:                        ;;APT ENVIRONMENT TABLE
000420     000                                  $ENV:   .BYTE   AENV            ;;ENVIRONMENT BYTE
000421     000                                  $ENVM:  .BYTE   AENVM           ;;ENVIRONMENT MODE BITS
000422  000000                                  $SWREG: .WORD   ASWREG          ;;APT SWITCH REGISTER
000424  000000                                  $USWR:  .WORD   AUSWR           ;;USER SWITCHES
000426  000000                                  $CPUOP: .WORD   ACPUOP          ;;CPU TYPE,OPTIONS
                                                ;*                              BITS 15-11=CPU TYPE
                                                ;*                                   11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                                                ;*                                   11/70=06,PDQ=07,Q=10
                                                ;*                              BIT 10=REAL TIME CLOCK
                                                ;*                              BIT  9=FLOATING POINT PROCESSOR
                                                ;*                              BIT  8=MEMORY MANAGEMENT
000430     000                                  $MAMS1: .BYTE   AMAMS1          ;;HIGH ADDRESS,M.S. BYTE
000431     000                                  $MTYP1: .BYTE   AMTYP1          ;;MEM. TYPE,BLK#1
                                                ;*                              MEM.TYPE BYTE   --  (HIGH BYTE)
                                                ;*                                       900 NSEC CORE=001
                                                ;*                                       300 NSEC BIPOLAR=002
                                                ;*                                       500 NSEC MOS=003
000432  000000                                  $MADR1: .WORD   AMADR1          ;;HIGH ADDRESS,BLK#1
                                                ;*                                MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
000434     000                                  $MAMS2: .BYTE   AMAMS2          ;;HIGH ADDRESS,M.S. BYTE
000435     000                                  $MTYP2: .BYTE   AMTYP2          ;;MEM.TYPE,BLK#2
000436  000000                                  $MADR2: .WORD   AMADR2          ;;MEM.LAST ADDRESS,BLK#2
000440     000                                  $MAMS3: .BYTE   AMAMS3          ;;HIGH ADDRESS,M.S.BYTE
000441     000                                  $MTYP3: .BYTE   AMTYP3          ;;MEM.TYPE,BLK#3
000442  000000                                  $MADR3: .WORD   AMADR3          ;;MEM.LAST ADDRESS,BLK#3
000444     000                                  $MAMS4: .BYTE   AMAMS4          ;;HIGH ADDRESS,M.S.BYTE
000445     000                                  $MTYP4: .BYTE   AMTYP4          ;;MEM.TYPE,BLK#4
000446  000000                                  $MADR4: .WORD   AMADR4          ;;MEM.LAST ADDRESS,BLK#4
000450                                          $ETEND:
                                                .MEXIT

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

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

1269                    ;;*****************************************************************
1270  000450  177570    SWR:    177570  ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
1271
1272          000500            .=500
1273  000500  010706    BEGIN:  MOV     PC,SP           ;SET UP STACK POINTER TO EQUAL BEGIN ADDRESS
1274
1275  000502  005746            TST     -(SP)
1276  000504  010637  000350    MOV     SP,@#SAVR6      ;SAVE SP FOR FUTURE USE
1277  000510  012737  000070  000024  MOV     #PWRDN,@#24     ;PREPARE FOR ANY FUTURE POWER DOWN
1278  000516  005037  000300    CLR     @#$PRERR
1279  000522  005037  000314    CLR     @#TYPCNT
1280  000526  012700  000114    MOV     #114,R0         ;PREPARE TO SETUP PARITY TRAP VECTOR
1281  000532  012710  005456    MOV     #PARERR-.-6,(R0)
1282  000536  060720            ADD     PC,(R0)+        ;TO PARERR
1283  000540  012710  000340    MOV     #340,(R0)       ;AND PSW OF 340
1284  000544  105737  000405    TSTB    @#REL           ;IS THIS CODE RELOCATED?
1285  000550  100002            BPL     ONEPAS          ;BRANCH IF NO
1286  000552  000167  000542    JMP     TSTREL          ;THIS CODE IS RELOCATED SO GET TEST SIZE.
1287
1288  000556  005737  000406    ONEPAS: TST     @#$PASS         ;IS THIS THE FIRST PASS?
1289  000562  001402            BEQ     TSTRP           ;BRANCH IF YES (TEST TRAP CATCHER ADDRESSES)
1290  000564  000167  000374    JMP     SETSTK          ; GET THE TEST SIZE
1291  000570  012704  007744    TSTRP:  MOV     #ENDPRG ,R4     ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1292  000574  012700  000377            MOV     #377,R0
1293  000600  010037  000316            MOV     R0,@#BAKPAT
1294  000604  005001            CLR     R1
1295  000606  012124    2$:     MOV     (R1)+,(R4)+     ;SAVE FROM 0000 TO BEGIN-30 AT END OF PROGRAM FOR NOW
1296  000610  020127  000400            CMP     R1,#$MAIL
1297  000614  103774            BLO     2$
1298  000616  005741    3$:     TST     -(R1)           ;PREPARE TO TEST THE TRAP VECTORS
1299  000620  010011    4$:     MOV     R0,(R1)         ;CHECK THE TRAP VECTORS FOR THE CAPABILITY
1300                                                    ;OF HOLDING 0'S & 1'S
1301  000622  020011            CMP     R0,(R1)         ;IS THE DATA OK?
1302  000624  001403            BEQ     6$              ;BRANCH IF YES
1303
1304  000626  004767  005304            JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1305  000632  000001            1                       ;******ERROR NUMBER 1******
1306
1307  000634  000300    6$:     SWAB    R0
1308  000636  001370            BNE     4$
1309  000640  005701            TST     R1              ;IF WE HAVE NOT REACHED THE LOWEST MEMORY LOCATION
1310  000642  001365            BNE     3$              ;THEN REPEAT FROM 3$
1311  000644  012701  000400            MOV     #$MAIL,R1
1312  000650  014441    8$:     MOV     -(R4),-(R1)     ;RESTORE TRAP CATCHER ETC.
1313  000652  005701            TST     R1
1314  000654  001375            BNE     8$
1315  000656  012700  000006    SETSWR: MOV     #6,R0
1316  000662  012710  000340            MOV     #340,(R0)       ;SET UP TIME OUT TRAP PSW
1317  000666  012740  000700            MOV     #4$,-(R0)       ;AND THE RETURN ADDRESS
1318  000672  005777  177552    2$:     TST     @SWR            ;DOES THE SWITCH REGISTER POINTED BY SWR EXIST ?
1319  000676  000404            BR      5$              ;BRANCH IF YES
1320  000700  022626    4$:     CMP     (SP)+,(SP)+     ;RESTORE THE STACK POINTER
1321  000702  012737  000176  000450  MOV     #SWREG,@#SWR    ;AND PLACE THE ADDRESS OF THE SWITCH REGISTER
1322                                                    ;DESIGNED FOR THE COMPUTERS NOT HAVING HARDWARE
1323                                                    ;SWITCH REGISTER AND RUNNING STAND ALONE
1324  000710  105737  000420    5$:     TSTB    @#$ENV          ;RUNNING UNDER APT?
```

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

1325  000714  001403                        BEQ     APTSIZ          ;BRANCH IF NO
1326  000716  012737  000422  000450        MOV     #$SWREG,@#SWR   ;SET SWR EQUAL TO APT SWITCH REGISTER.
1327
1328
1329
1330
1331                                ;APTSIZ- THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE AND WHEN
1332                                ;A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO GIVEN HIGH ADDRESS.
1333                                ;       IF APT DEFINES SIZE THE LOW TEST ADDRESS MUST=00000.(DUE TO ETABLE FORMAT)
1334                                ;FLOW;
1335                                ;       IF BLOCK 4 (OR 3,2,1) TYPE NON ZERO THEN GET APT HIGH ADDRESS AND EXIT.
1336                                ;       ELSE    SEND ERROR #3
1337                                ;NOTE; THE MEMORY TYPE IS IGNORED SINCE ALL TESTS ARE RUN REGARDLESS OF MEMORY TYPE.
1338
1339  000724  012703  000340        APTSIZ: MOV     #MAXMEM,R3      ;POINT R3 TO MAXMEM.
1340  000730  013737  000330  000334        MOV     @#HIGHTWO,@#SHIMAX      ;IN CASE NO SELF SIZING DONE.
1341  000736  013737  000332  000336        MOV     @#HIGHADD,@#SMAXM       ;IN CASE NO SELF SIZING DONE.
1342  000744  105737  000421                TSTB    @#SENVM         ;DOES APT ALLOW SELF SIZING?
1343  000750  100021                        BPL     TRYSR           ;BRANCH IF YES
1344
1345  000752  012701  000451                MOV     #SMTYP4+4,R1    ;POINT R1 TO BLOCK TYPE 4(+4)
1346  000756  162701  000004        1$:     SUB     #4,R1           ;POINT R1 TO NEXT BLOCK TYPE.
1347  000762  105711                        TSTB    (R1)            ;IS THE BLOCK TYPE NON ZERO?
1348  000764  001006                        BNE     2$              ;BRANCH IF YES (MEMORY EXISTS)
1349  000766  020127  000431                CMP     R1,#SMTYP1      ;ALL APT BLOCK TYPES BEEN CHECKED?
1350  000772  101371                        BHI     1$              ;BRANCH IF NO
1351
1352  000774  004767  005136                JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1353  001000  000002                        2                       ;******ERROR NUMBER 2******
1354
1355  001002  004767  006302        2$:     JSR     PC,GETADR       ;GO SET MAXIMUM APT ADDRESS INTO SMAXM + SHIMAX
1356  001006  004767  006276                JSR     PC,GETADR       ;GO SET MAXIMUM APT ADDRESS INTO HIGHADD+HIGHTWO
1357  001012  000446        BRTPSZ:         BR      TYPSIZ          ; TYPE THE SIZE OF MEMORY UNDER TEST
1358
1359  001014  032777  000100  177426  TRYSR:  BIT     #100,@SWR       ;USER DEFINED MEMORY TEST BOUNDARIES??
1360  001022  001042                        BNE     TYPSIZ          ;BRANCH IF YES (DON'T SIZE MEMORY)
1361
1362
1363
1364
1365
1366  001024  010401        SLFSIZ:         MOV     R4,R1           ;SETUP R1 AND R4 TO THE LOWEST ADDRESS OF MEMORY
1367  001026  012710  001042                MOV     #4$,(R0)        ;SET UP RETURN ADDRESS FROM TIME OUT TRAP TO 4$
1368  001032  011111        2$:             MOV     (R1),(R1)       ;WRITE A MEMORY LOCATION INTO ITSELF AND TRAP IF NONEXIS
1369
1370  001034  062701  000002                ADD     #2,R1           ;ADD 2 TO THE ADDRESS POINTER
1371  001040  000774                        BR      2$              ;KEEP ON SIZING UP THE MEMORY UNTIL
1372                                                                ;NXM TRAP (TIME OUT TRAP) IS ENCOUNTERED
1373
1374  001042  022626        4$:             CMP     (SP)+,(SP)+     ;RESTORE THE STACK POINTER
1375  001044  004767  005770                JSR     PC,MEMMNG       ; SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE
1376                                                                ;AND IF IT HAS TO BE TESTED
1377  001050  105737  000276                TSTB    @#MMAVA         ;SEE IF MEMORY MANAGEMENT HAS TO BE TESTED
1378  001054  001414                        BEQ     12$             ;IF NO MEM. MANG. THEN GO TO 12$
1379  001056  012710  001070        6$:     MOV     #8$,(R0)        ;SET UP THE RETURN ADDRESS FROM TRAP TO 8$
1380  001062  012701  020000                MOV     #20000,R1       ;BEGIN CHECKING MEMORY ABOVE 28K
```

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

1381  001066  000761                  BR      2$
1382  001070  022626          8$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
1383  001072  022701  160000          CMP     #160000,R1      ;IF R1 DID NOT READ ALL THE LOCATIONS POINTED BY
1384                                                          ;PAGE ADDRESS REGISTER 6 THEN IT HAS REACHED THE
1385                                                          ;MAXIMUM AVAILABLE MEMORY
1386  001076  001003                  BNE     12$             ;IN WHICH CASE GO TO 12$
1387  001100  004767  006106          JSR     PC,UPMM         ;OTHERWISE GO TO UPDATE MEM. MANG. REGISTERS
1388  001104  000764                  BR      6$
1389  001106  024341          12$:    CMP     -(R3),-(R1)     ;CAUSE R3 TO POINT TO LOCATION SMAXM AND R1
1390                                                          ;TO THE MAXIMUM AVAILABLE MEMORY
1391  001110  004767  006104          JSR     PC,PUTADR       ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
1392                                                          ;AT LOCATIONS SMAXM AND SHIMAX
1393  001114  024343                  CMP     -(R3),-(R3)     ;MAKE R3 POINT TO HIGHADD
1394  001116  004767  006076          JSR     PC,PUTADR       ;PLACE THE ADDRESS IN R1 AT LOCATIONS HIGHADD
1395                                                          ;AND HIGHTWO
1396  001122  005743                  TST     -(R3)
1397  001124  005043                  CLR     -(R3)           ;CLEAR THE LOCATION LOWADD
1398  001126  005043                  CLR     -(R3)           ;AND LOWTWO
1399  001130  012720  000104  TYPSIZ: MOV     #BUSER,(R0)+    ;SET UP VECTOR FOR ANY FUTURE TRAP
1400  001134  010403                  MOV     R4,R3           ;SET R3 TO POINT TO THE LOWEST AVAILABLE MEMORY
1401                                                          ;LOCATION
1402  001136  012701  000324          MOV     #LOWTWO,R1
1403  001142  004767  005400          JSR     PC,PCRLF        ;TYPE CR/LF
1404  001146  004767  005546          JSR     PC,OCTTYP       ;TYPE LOW TEST ADDRESSE (LOWTWO+LOWADD)
1405  001152  004767  005302  TYPMEM: JSR     PC,$TYPE        ; TYPE "-"
1406  001156  000055                  .ASCIZ  /-/
1407                                   .EVEN
1408  001160  004767  005534          JSR     PC,OCTTYP       ;TYPE HIGHEST TEST ADDRESS (HIGHTWO+HIGHADD)
1409  001164  012703  000330  SETSTK: MOV     #HIGHTWO,R3     ;MAKE R3 POINT TO THE HIGH ORDER BITS OF TOP ADDRESS
1410  001170  004767  006130          JSR     PC,$GTSIZ       ; GET THE BITS 13-17 OF THE TOP ADDRESS
1411                                                          ;PLACED IN BITS 0-4 OF R2
1412  001174  010401                  MOV     R4,R1           ;SET R1 TO LOWEST TEST ADDRESS
1413
1414  001176  062704  000022  4$:     ADD     #18.,R4         ;APPEND THE ERROR STACK FOR THE MEMORY UNDER
1415                                                          ;TEST TO THE END OF THE PROGRAM
1416  001202  005302                  DEC     R2
1417  001204  002374                  BGE     4$
1418  001206  010437  000310          MOV     R4,@#ENDSTK     ;SAVE THE ADDRESS OF THE END OF THE ERROR STACK
1419  001212  005021          6$:     CLR     (R1)+           ;CLEAR THE ERROR STACK
1420  001214  020104                  CMP     R1,R4
1421  001216  101775                  BLOS    6$
1422  001220  012737  157776  000340  MOV     #157776,@#MAXMEM                ;SET MAXMEM TO MAXIMUM VIRTUAL ADDRESS
1423  001226  005723                  TST     (R3)+           ;TESTING MEMORY MANAGEMENT?
1424  001230  001004                  BNE     SAVLDR          ;BRANCH IF YES (GO SAVE LOADERS AT TOP OF VIRTUAL MEMORY
1425  001232  021300                  CMP     (R3),R0         ;IS THE VIRTUAL ADDRESS ABOVE 157776?
1426  001234  103002                  BHIS    SAVLDR          ;BRANCH IF YES (GO SAVE LOADERS)
1427  001236  011363  000002          MOV     (R3),2(R3)      ;OTHERWISE MAKE THE CONTENTS OF LOCATION MAXMEM
1428                                                          ;EQUAL TO THE MAXIMUM AVAILABLE MEMORY
1429                                                          ;AND FALL INTO SAVE LOADERS.
1430
1431  001242  004767  006136  SAVLDR: JSR     PC,CLRMM        ; DISABLE THE MEMORY MANAGEMENT UNIT
1432  001246  005723                  TST     (R3)+           ;MAKE R3 TO POINT TO THE LOCATION MAXMEM
1433  001250  011305                  MOV     (R3),R5         ;R5 CONTAINS THE ADDRESS OF MAXIMUM AVAILABLE MEM.
1434
1435                          ;IF ONLY 4K BEING TESTED DON'T SAVE LOADERS
1436
```

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

1437  001252  020527  017776          CMP     R5,#17776       ;ONLY TESTING 4K MAX?
1438  001256  103416                  BLO     4$              ;BRANCH IF YES (DON'T SAVE LOADERS)
1439
1440  001260  162705  000276    3$:   SUB     #276,R5         ;PREPARE TO SAVE 300 BYTES OF THE LOADERS
1441  001264  005737  000042          TST     @#42            ;IS THE PROGRAM RUNNING UNDER ACT OR XXDP ?
1442  001270  001406                  BEQ     2$              ;IF NOT THEN GO TO 2$
1443  001272  023737  000042  000046  CMP     @#42,@#46       ;ARE WE RUNNING UNDER XXDP CHAIN MODE?
1444  001300  001402                  BEQ     2$              ;BRANCH IF NO
1445  001302  162705  005674          SUB     #<1502.*2>,R5   ;SAVE 1500. WORDS FOR XXDP CHAIN MODE
1446  001306  012524          2$:     MOV     (R5)+,(R4)+     ;SAVE LOADER
1447  001310  020513                  CMP     R5,(R3)
1448  001312  101775                  BLOS    2$
1449  001314  012323          4$:     MOV     (R3)+,(R3)+     ;SAVE THE CONTENTS OF LOCATION MAXMEM IN SAVMAX
1450  001316  010423                  MOV     R4,(R3)+        ;AND THE CONTENTS OF R4 AT SAVR4
1451
1452  001320  010537  000346  TSTREL:MOV     R5,@#SAVR5      ;SAVE HIGHEST VIRTUAL ADDRESS+2
1453  001324  004767  006054  TSTSIZ:JSR     PC,CLRMM        ;GO TO DISABLE MEMORY MANAGEMENT UNIT
1454  001330  005745                  TST     -(R5)           ;SET R5 BACK TO HIGHEST VIRTUAL ADDRESS
1455  001332  012703  000324    1$:   MOV     #LOWTWO,R3      ;PREPARE TO LOAD R4 AND R5 WITH THE MEMORY BOUNDRIES
1456  001336  005723                  TST     (R3)+           ;IF THE BITS 16,17 OF THE LOWEST LOCATION UNDER
1457                                                          ;TEST ARE NON ZERO
1458  001340  001003                  BNE     2$              ;THEN GO TO 2$
1459  001342  021327  157776          CMP     (R3),#157776    ;IF THE LOWEST LOCATION UNDER TEST IS HIGHER THAN
1460                                                          ;157776 THEN GO TO TEST MEMORY MANAGEMENT
1461  001346  103411                  BLO     4$
1462  001350  032777  010000  177072  2$:     BIT     #10000,@SWR     ;IS MEMORY MANAGEMENT SELECTED?
1463  001356  001003                  BNE     3$              ;YES ALL IS WELL
1464  001360  004767  004552          JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1465  001364  000003                  3                       ;******ERROR NUMBER 3******
1466
1467  001366  000167  003526    3$:   JMP     TSTMM           ;GO TO TEST MEMORY MANAGEMENT
1468  001372  020423          4$:     CMP     R4,(R3)+
1469  001374  103002                  BHIS    6$
1470  001376  016304  177776          MOV     -2(R3),R4       ;ADJUST R4 TO POINT TO THE LOWEST LOCATION UNDER TEST
1471  001402  005723          6$:     TST     (R3)+           ;IF BITS 16-17 OF HIGHEST LOCATION TO BE TESTED
1472  001404  001003                  BNE     8$              ;ARE NON ZERO THEN GO TO 8$
1473  001406  021305                  CMP     (R3),R5         ;OTHERWISE SEE IF THE HIGHEST LOCATION TO BE
1474                                                          ;TESTED IS HIGHER THAN 157776
1475  001410  101001                  BHI     8$              ;IF SO THEN GO TO 8$
1476  001412  011305                  MOV     (R3),R5         ;MODIFY R5
1477  001414  105737  000405    8$:   TSTB    @#REL           ;ARE WE RELOCATED.?
1478  001420  100014                  BPL     10$             ;BRANCH IF NO
1479  001422  013704  000322          MOV     @#RELBOT,R4     ;SET BOTTOM TEST ADDRESS WHEN RELOCATED.
1480  001426  020527  017776          CMP     R5,#17776       ;ARE WE RELOCATED IN BANK 0?
1481  001432  103402                  BLO     9$              ;BRANCH IF YES
1482  001434  012705  017776          MOV     #17776,R5       ;ELSE SET HIGH MEMORY UNDER TEST=4K
1483
1484  001440  020405          9$:     CMP     R4,R5           ;IS LOW LIMIT LOWER THAN HIGH LIMIT?
1485  001442  103403                  BLO     10$             ;BRANCH IF YES
1486  001444  004767  004466          JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1487  001450  000004                  4                       ;******ERROR NUMBER 4******
1488
1489  001452  012703  000342    10$:  MOV     #SAVMAX,R3
1490  001456  011343                  MOV     (R3),-(R3)      ;RESTORE THE CONTENTS OF MAXMEM
1491  001460  062713  000002  MEMTST: ADD     #2,(R3)         ;MAKE THE CONTENTS OF MAXMEM = MAXIMUM AVAILABLE
1492                                                          ;MEMORY +2
```

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

1493  001464  005725                  TST     (R5)+           ;AND SET R5=MAX MEMORY+2
1494
1495                                  ;CLEAR MEMORY UNDER TEST
1496
1497  001466  012702  000001  CLRMEM: MOV     #1,R2           ;SET R2 TO ENABLE PARITY MODULE CODE.
1498  001472  004767  006006          JSR     PC,PARITY       ;ENABLE PARITY IF WANTED AND AVAILABLE.
1499  001476  010500                  MOV     R5,R0
1500  001500  005040          2$:     CLR     -(R0)           ;BEGIN CLEARING THE MEMORY FROM THE TOP
1501  001502  020004                  CMP     R0,R4           ;UNTIL THE BOTTOM IS REACHED
1502  001504  101375                  BHI     2$
1503  001506  012702  000316          MOV     #BAKPAT,R2
1504  001512  012212                  MOV     (R2)+,(R2)      ;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
1505  001514  000312                  SWAB    (R2)
1506  001516  017702  176726          MOV     @SWR,R2         ;LOAD R2 WITH THE OPTIONS STORED AT $SWREG
1507  001522  042702  177760          BIC     #177760,R2      ;ONLY LEAVE THE LOWER 4 BITS OF $SWREG IN R2 TO GO TO
1508                                                          ;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
1509
1510
1511
1512                                  ;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
1513
1514  001526  005037  000306  CONT:   CLR     @#PASFLG        ;INIT SUBTEST PASS FLAG.
1515  001532  110237  000404          MOVB    R2,@#STESTN     ;SET UP STESTN WITH THE TEST NUMBER GOING
1516                                                          ;TO BE EXECUTED
1517  001536  010401          LOOP:   MOV     R4,R1           ;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
1518  001540  010246                  MOV     R2,-(SP)        ;SAVE R2 ON THE STACK
1519  001542  012703  000376          MOV     #376,R3         ;POINT R3 TO SCRATCH STACK
1520  001546  004767  005446          JSR     PC,PUTADR       ;GO TO GENERATE 18 BIT ADDRESS OUT OF THE ADDRESS
1521                                                          ;STORED IN R1 AND STORE IT IN LOCATIONS (R3)
1522                                                          ;AND (R3-2)
1523  001552  005743                  TST     -(R3)           ;CAUSE R3 TO POINT TO THE HIGH ORDER BITS OF THE
1524                                                          ;18 BIT ADDRESS
1525  001554  004767  005544          JSR     PC,$GTSIZ       ; PLACE BITS 13-17 OF THE ADDRESS IN BITS
1526                                                          ;0-4 OF R2
1527  001560  010400                  MOV     R4,R0           ;PLACE THE ADDRESS OF THE LOWEST LOCATION UNDER
1528                                                          ;TEST IN R0
1529  001562  010401                  MOV     R4,R1           ;IN R1
1530  001564  010403                  MOV     R4,R3           ;AND IN R3
1531  001566  012602                  MOV     (SP)+,R2        ;RESTORE R2
1532  001570  006302                  ASL     R2
1533  001572  060702                  ADD     PC,R2
1534  001574  066207  000004          ADD     TBL-.(R2),PC            ;GO TO THE TEST #
1535                                                          ;STORED IN BITS 0-3 OF SWITCH REGISTER
1536
1537
1538  001600  000102          TBL:    TST0-TBL                ;RELATIVE ADDRESS OF TEST # 0
1539  001602  000334                  TST1-TBL                ;RELATIVE ADDRESS OF TEST # 1
1540  001604  000434                  TST2-TBL                ;RELATIVE ADDRESS OF TEST # 2
1541  001606  000544                  TST3-TBL                ;RELATIVE ADDRESS OF TEST # 3
1542  001610  001012                  TST4-TBL                ;RELATIVE ADDRESS OF TEST # 4
1543  001612  001122                  TST5-TBL                ;RELATIVE ADDRESS OF TEST # 5
1544  001614  001270                  TST6-TBL                ;RELATIVE ADDRESS OF TEST # 6
1545  001616  001424                  TST7-TBL                ;RELATIVE ADDRESS OF TEST # 7
1546  001620  001646                  TST10-TBL               ;RELATIVE ADDRESS OF TEST # 10
1547  001622  002174                  TST11-TBL               ;RELATIVE ADDRESS OF TEST # 11
1548  001624  002246                  TST12-TBL               ;RELATIVE ADDRESS OF TEST # 12
```

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

1549  001626  002520                    TST13-TBL               ;RELATIVE ADDRESS OF TEST # 13
1550  001630  003146                    RELOC-TBL               ;RELATIVE ADDRESS OF ROUTINE 'RELOC'
1551
1552
1553
1554                                    ;R5 IS POINTING TO THE TOP OF THE MEMORY TO BE TESTED+2
1555                                    ;R4 & R0 ARE POINTING TO THE LOWEST ADDRESS OF MEMORY TO BE TESTED
```

```
                                ;*      SCOPE ROUTINE
                                ;*      -------------
                                ;*
                                ;*
                                ;*
                                ;*      PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
                                ;*      IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE.
                                ;*      IF SR= 2000 (BIT10) THEN HALT
                                ;*      IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SR BITS<3:0>
                                ;*      ELSE CONTINUE TO NEXT TEST.
                                ;*

001632  105737  000420          TSTSCP: TSTB    @#$ENV          ;ARE WE RUNNING UNDER APT?
001636  001002                          BNE     CNTSCP          ;IF SO THEN GO TO CNTSCP
001640  004767  006020                  JSR     PC,CHECKC       ;TEST FOR CONTROL-C AND IF TYPED GO
                                                                ;PRINT ERROR HISTORY AND HALT AT FATHLT.
001644  113702  000404          CNTSCP: MOVB    @#$TESTN,R2     ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
                                                                ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
                                                                ;OF R2 WILL BE 0
001650  005237  000410                  INC     @#$DEVCT        ;TELL APT WE ARE STILL RUNNING OKAY
001654  032777  002000  176566          BIT     #2000,@SWR      ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
001662  001401                          BEQ     TSTGO           ;IF NOT THEN GO TO 2$
001664  000000          SWHALT: HALT                            ;HALT AT END OF TEST SWITCH SET.

001666  032777  040000  176554  TSTGO:  BIT     #40000,@SWR     ;IS THE PROGRAM GOING TO LOOP ON TEST
001674  001320                          BNE     LOOP            ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
001676  105202                          INCB    R2
001700  000712                          BR      CONT            ;GO TO CONT AND CONTINUE EXECUTING THE NEXT TEST
```

```
                                ;;*****************************************************************
                                ;*TEST 0         TEST FOR PROPER BANK SELECTION
                                        ;*(1)   THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
                                        ;*      OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
                                        ;*      LOCATION UNDER TEST
                                        ;*(2)   IT CHECKS FOR PROPER BANK SELECTION BY WRITING
                                        ;*      1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
                                        ;*      LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
                                        ;*      [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
                                        ;*(3)   THIS TEST ALSO CHECKS TO SEE THAT NONE OF THE NON EXIST-
                                        ;*      ING BANK RESPOND WHEN THEY ARE ADDRESSED
                                ;;*****************************************************************
001702  105737  000404          TST0:   TSTB    @#$TESTN        ;CHECK FOR PROPER TEST SEQUENCE
001706  001403                  BEQ     .+10
001710  004767  004222          JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
001714  000005                  5                       ;******ERROR NUMBER 5******

001716  012703  177777          MOV     #177777,R3
001722  010401          1$:     MOV     R4,R1           ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
001724  010310                  MOV     R3,(R0)         ;SET ALL THE BITS AT (R0)
001726  020001          2$:     CMP     R0,R1           ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
001730  001417                  BEQ     4$              ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
001732  005711                  TST     (R1)            ;OTHERWISE CHECK (R1) FOR ALL 0'S
001734  001430                  BEQ     5$
001736  020311                  CMP     R3,(R1)         ;IF R1 IS NOT EQUAL TO R0 AND (R1)
                                                        ;DOES NOT CONTAIN ALL 0'S THEN
                                                        ;CHECK TO SEE IF (R0) = (R1)
001740  001004                  BNE     3$
001742  012767  000006  000042  MOV     #6,12$          ;*ERROR* SETUP ERROR NO. IN 12$
                                                        ;******ERROR NUMBER #6******
001750  000403                  BR      10$
001752                  3$:
001752  012767  000007  000032  MOV     #7,12$          ;*ERROR* SETUP ERROR NO. IN 12$
                                                        ;******ERROR NUMBER #7******
001760  010046          10$:    MOV     R0,-(SP)        ;SAVE R0 ON STACK
001762  105237  000301          INCB    @#$ADERR        ;AN ADDRESSING ERROR IS SUSPECTED
001766  000407                  BR      11$
001770  020311          4$:     CMP     R3,(R1)         ;CHECK (R1) FOR ALL 1'S
001772  001411                  BEQ     5$
001774  012767  000010  000010  MOV     #10,12$         ;*ERROR* SETUP ERROR NO. IN 12$
                                                        ;******ERROR NUMBER #10******
002002  010046                  MOV     R0,-(SP)        ;SAVE R0 ON STACK
002004  010300                  MOV     R3,R0
002006  004767  003566  11$:    JSR     PC,ERROR        ;GO TO THE ERROR SUBROUTINE
002012  000000  12$:    .WORD                           ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
002014  012600                  MOV     (SP)+,R0        ;RESTORE R0

002016  013706  000350  5$:     MOV     @#SAVR6,SP      ;RESTORE THE STACK POINTER
002022  062701  020000          ADD     #20000,R1       ;CAUSE R1 TO POINT TO THE SAME CHIP
                                                        ;LOCATION IN THE NEXT 4K BANK OF MEMORY
                                                        ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
002026  020105                  CMP     R1,R5           ;COMPARE R1 WITH THE HIGHEST MEMORY
                                                        ;LOCATION WHICH IS STORED IN R5
002030  103736                  BLO     2$              ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$

002032  105737  000421          TSTB    @#$ENVM         ;HAS APT INHIBITED SIZING?
```

```
DZKMA   MACY11 27(1006)  18-FEB-77  12:35  PAGE 33
DZKMAC.P11    18-FEB-77 12:30            TO      TEST FOR PROPER BANK SELECTION

 1640  002036  100430                          BMI     8$              ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
 1641  002040  032777  000100  176402          BIT     #100,@SWR       ;HAS USER INHIBITED SIZING?
 1642  002046  001024                          BNE     8$              ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
 1643
 1644  002050  020137  000340                  CMP     R1,@#MAXMEM     ;IS R1 LOWER THAN THE MAXIMUM AVAILABLE
 1645                                                                  ;MEMORY ?
 1646  002054  103760                          BLO     5$              ;IF SO THEN GO TO 5$
 1647  002056  012702  000006                  MOV     #6,R2           ;MAKE R2 POINT TO TRAP VECTOR+2 FOR NXM
 1648  002062  012712  000340                  MOV     #340,(R2)       ;SET PSW TO 340
 1649  002066  012742  177722                  MOV     #5$-.-6,-(R2)   ;SET UP RETURN ADDRESS FROM TRAP TO 5$
 1650  002072  060712                          ADD     PC,(R2)
 1651  002074  020127  157776                  CMP     R1,#157776      ;SEE IF R1 HAS CROSSED 28K BOUNDRY OF VIRTUAL ADDRESS
 1652  002100  101004                          BHI     6$              ;IN WHICH CASE GO TO 6$
 1653  002102  011111                          MOV     (R1),(R1)       ;TRY TO WRITE TO NON-EXISTENT MEMORY (SHOULD TRAP)
 1654  002104  004767  004026                  JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
 1655  002110  000011                          11                      ;******ERROR NUMBER 11******
 1656
 1657
 1658  002112  012722  000006          6$:     MOV     #6,(R2)+        ;RESTORE TRAP VECTOR
 1659  002116  005012                          CLR     (R2)
 1660  002120  005010                  8$:     CLR     (R0)
 1661
 1662  002122  062700  020000                  ADD     #20000,R0       ;CAUSE R0 TO POINT TO THE SAME CHIP
 1663                                                                  ;LOCATION IN THE NEXT 4K MEMORY BANK
 1664                                                                  ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R0
 1665  002126  020005                          CMP     R0,R5           ;COMPARE R0 WITH THE HIGHEST MEMORY
 1666                                                                  ;LOCATION WHICH IS STORED IN R5.
 1667  002130  103674                          BLO     1$              ;IF R0 LESS THEN REPEAT THE TEST
 1668  002132  000637                  ENDO:   BR      TSTSCP
 1669
 1670
```

```
1671                                    ;;*****************************************************************
1672                                    ;*TEST 1         CHECK DI/DO LINES
1673                                            ;*(1)   THIS TEST CHECKS THE DATI/DATO LINES BY SHIFTING
1674                                            ;*      A 1 IN THE WORD DIRECTION
1675                                    ;;*****************************************************************
1676  002134  122737  000001  000404  TST1:   CMPB    #1,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE
1677
1678
1679  002142  001403                          BEQ     .+10
1680  002144  004767  003766                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1681  002150  000012                          12                      ;******ERROR NUMBER 12******
1682
1683  002152  012700  000001          1$:     MOV     #1,R0
1684  002156  010002                          MOV     R0,R2           ;SET R2=1
1685  002160  010011                  2$:     MOV     R0,(R1)         ;MOV 1 AT LOCATION (R1)
1686  002162  020011                  3$:     CMP     R0,(R1)         ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
1687  002164  001403                          BEQ     4$
1688  002166  004767  003406                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1689  002172  000013                          13                      ;******ERROR NUMBER 13******
1690
1691
1692  002174  005702                  4$:     TST     R2              ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
1693  002176  001406                          BEQ     5$              ;IF SO THEN GO TO 5$
1694  002200  006300                          ASL     R0              ;SHIFT THE 1 BROUGHT IN AT 1$ IN
1695                                                                  ;DATA DIRECTION
1696  002202  103366                          BCC     2$              ;IF THE 1 HAS NOT BEEN SHIFTED THRU
1697                                                                  ;THE 16 DATA BITS THEN REPEAT FROM 2$
1698  002204  005002                          CLR     R2              ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
1699  002206  012700  177776                  MOV     #177776,R0
1700  002212  000762                          BR      2$
1701
1702  002214  000261                  5$:     SEC                     ;SET C BIT
1703  002216  006100                          ROL     R0              ;SHIFT A 0 16 TIMES IN DATA DIRECTION
1704  002220  103757                          BCS     2$              ;IF THE 0 HAS NOT BEEN SHIFTED THRU
1705                                                                  ;THE 16 DATA BITS THEN REPEAT FROM 2$
1706  002222  062701  020000                  ADD     #20000,R1       ;OTHERWISE GO TO THE NEXT BANK OF
1707                                                                  ;4K MEMORY AND REPEAT THE TEST
1708  002226  020105                          CMP     R1,R5
1709  002230  103750                          BLO     1$
1710  002232  000737                  END1:   BR      ENDO
1711
```

```
1712                                    ;;*****************************************************************
1713                                    ;*TEST 2         TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
1714                                            ;*(1)   THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
1715                                            ;*      OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
1716                                            ;*      OF BAKPAT AND READING IT
1717                                            ;*(2)   MEMORY IS WRITTEN USING A BYTE AT A TIME
1718                                            ;*(3)   STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
1719                                    ;;*****************************************************************
1720  002234  122737  000002  000404  TST2:   CMPB    #2,@#TESTN      ;CHECK FOR PROPER TEST SEQUENCE
1721
1722  002242  001403                          BEQ     .+10
1723  002244  004767  003666                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1724  002250  000014                          14                      ;******ERROR NUMBER 14******
1725
1726  002252  013700  000316          1$:     MOV     @#BAKPAT,R0
1727  002256  110021                          MOVB    R0,(R1)+
1728  002260  113721  000317                  MOVB    @#BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
1729  002264  020105                          CMP     R1,R5
1730  002266  103771                          BLO     1$
1731
1732  002270  020041          2$:     CMP     R0,-(R1)        ;TEST THE MEMORY TO SEE IF IT CONTAINS
1733                                                          ;THE WORD STORED IN BAKPAT
1734  002272  001416                          BEQ     8$
1735  002274  062701  000002                  ADD     #2,R1
1736  002300  123741  000317                  CMPB    @#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
1737  002304  001402                          BEQ     4$
1738  002306  120041                          CMPB    R0,-(R1)        ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
1739  002310  001002                          BNE     6$
1740  002312  105237  000301          4$:     INCB    @#$ADERR        ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
1741  002316  042701  000001          6$:     BIC     #1,R1           ;MAKE THE ADDRESS IN R1 EVEN
1742  002322  004767  003252                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1743  002326  000015                          15                      ;******ERROR NUMBER 15******
1744
1745  002330  020104          8$:     CMP     R1,R4           ;KEEP ON TESTING THE MEMORY UNTIL
1746  002332  101356                          BHI     2$              ;R1 EQUALS THE LOWEST ADDRESS
1747  002334  000337  000316                  SWAB    @#BAKPAT        ;CHANGE THE DATA PATTERN
1748  002340  001744                          BEQ     1$              ;IF THE DATA PATTERN DOES NOT HAVE LOW
1749                                                                  ; BYTE =0 THEN FALL THRU
1750  002342  000733          END2:   BR      END1
1751
1752                                    ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
1753
```

```
1754                                    ;;*****************************************************************
1755                                    ;*TEST 3         DUAL ADDRESS TEST A
1756                                    ;
1757                                    ;*(1)    THIS TEST CHECKS FOR DUAL ADDRESSING PROBLEMS BY WRITING A
1758                                    ;*       BACK GROUND OF BAKPAT.
1759                                    ;*(2)    STARTING FROM THE LOWEST LOCATION IN THE BANK THE TEST WRITES A
1760                                    ;*       LOCATION WITH SWAPPED BAKPAT
1761                                    ;*(3)    READS THE MEMORY FOR PROPER CONTENTS
1762                                    ;*(4)    SHIFTS A 1 ALONG THE ADDRESS DIRECTION AND REPEATS STEPS 1-3
1763                                    ;*(5)    REPEATS STEP 1-4 FOR EACH 4K BANK
1764                                    ;;*****************************************************************
1765  002344  122737  000003  000404  TST3:   CMPB    #3,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE
1766  002352  001403                          BEQ     .+10
1767  002354  004767  003556                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1768  002360  000016                          16                      ;******ERROR NUMBER 16******
1769
1770  002362  005003                          CLR     R3
1771  002364  004737  000120          2$:     JSR     PC,@#WRTMEM     ; WRITE MEMORY WITH THE BACKGROUND STORED
1772                                                                  ;AT LOCATION BAKPAT
1773  002370  005002                  4$:     CLR     R2
1774  002372  050302                  6$:     BIS     R3,R2           ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
1775  002374  020204                          CMP     R2,R4           ;IF R2 IS LESS THAN R4
1776  002376  103465                          BLO     16$             ;THEN DO NOTHING
1777  002400  020205                          CMP     R2,R5           ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
1778  002402  103077                          BHIS    20$             ;TESTED THEN EXIT THE TEST
1779  002404  000312                          SWAB    (R2)            ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
1780                                                                  ;THE LOCATION POINTED BY R2
1781  002406  005001                          CLR     R1
1782  002410  050301                  7$:     BIS     R3,R1
1783  002412  020104                          CMP     R1,R4           ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
1784  002414  103445                          BLO     12$             ;THEN GO TO 12$
1785  002416  020105                          CMP     R1,R5
1786  002420  103053                          BHIS    15$
1787  002422  020102                          CMP     R1,R2           ;CHECK THE MEMORY FOR CORRECT DATA
1788  002424  001431                          BEQ     10$
1789  002426  020011                          CMP     R0,(R1)         ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
1790                                                                  ;THE SAME WORD AS BAKPAT
1791  002430  001437                          BEQ     12$             ;IN WHICH CASE GO BACK TO 12$
1792  002432  012767  000017  000032          MOV     #17,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1793                                                                  ;******ERROR NUMBER #17******
1794  002440  010046                  8$:     MOV     R0,-(SP)        ;PLACE R0 ON THE STACK
1795  002442  000316                          SWAB    (SP)
1796  002444  022611                          CMP     (SP)+,(R1)      ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
1797                                                                  ;AS A SWAPPED R0
1798  002446  001003                          BNE     9$              ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
1799                                                                  ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
1800                                                                  ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
1801                                                                  ;ENTIRE WORD
1802  002450  012767  000020  000014          MOV     #20,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1803                                                                  ;******ERROR NUMBER #20******
1804  002456  105237  000301          9$:     INCB    @#$ADERR        ;ADDRESSING PROBLEM IS DETECTED
1805  002462  010046                          MOV     R0,-(SP)        ;SAVE R0
1806  002464  010200                          MOV     R2,R0           ;SET R0=GOOD ADDRESS FOR ERROR REPORT
1807  002466  004767  003106                  JSR     PC,ERROR        ;GO TO THE ERROR SUBROUTINE
1808  002472  000000                  22$:    .WORD                   ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1809  002474  012600                          MOV     (SP)+,R0        ;RESTORE R0
```

```
1810  002476  010011                        MOV     RO,(R1)         ;RESTORE (R1)
1811  002500  020037  000316                CMP     RO,@#BAKPAT     ;IF THE CONTROL CAME HERE FROM 15$-2 THEN
1812  002504  001411                        BEQ     12$
1813  002506  000407                        BR      11$             ;RETURN TO 11$
1814  002510  000300                10$:    SWAB    RO              ;MAKE RO SAME AS SWAPPED BAKPAT
1815  002512  020011                        CMP     RO,(R1)         ;IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
1816                                                                ;EQUAL TO SWAPPED RO
1817  002514  001404                        BEQ     11$             ;IN WHICH CASE GO BACK TO 11$
1818  002516  012767  000021  177746        MOV     #21,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1819                                                                ;******ERROR NUMBER #21******
1820  002524  000745                        BR      8$              ;AND GO TO 8$
1821  002526  000300                11$:    SWAB    RO              ;RESTORE RO TO BAKPAT
1822  002530  040301                12$:    BIC     R3,R1           ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
1823  002532  005701                        TST     R1              ;IF R1 IS 0 THEN PLACE A 1 IN R1
1824  002534  001001                        BNE     13$             ;OTHERWISE GO TO 13$
1825  002536  005201                        INC     R1
1826  002540  006101                13$:    ROL     R1
1827  002542  020127  020000                CMP     R1,#20000       ;IF R1 IS LESS THAN A 4K BOUNDRY
1828  002546  103720                        BLO     7$              ;THEN REPEAT FROM 7$
1829  002550  000312                15$:    SWAB    (R2)            ;RESTORE (R2) TO BAKPAT
1830  002552  040302                16$:    BIC     R3,R2           ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
1831                                                                ;STORED IN R2
1832  002554  005702                        TST     R2              ;IF R2 = 0 THEN MOVE A 1 TO R2
1833  002556  001001                        BNE     18$             ;OTHERWISE GO TO 18$
1834  002560  005202                        INC     R2
1835  002562  006102                18$:    ROL     R2              ;SHIFT A ONE IN THE ADDRESS WORD
1836  002564  020227  020000                CMP     R2,#20000       ;IS THE ADDRESS IN R2 MORE THAN THE BOUNDRY
1837                                                                ;OF 4K
1838  002570  103700                        BLO     6$              ;IF NOT THEN GO TO 6$
1839  002572  060203                        ADD     R2,R3           ;OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
1840  002574  020337  000340                CMP     R3,@#MAXMEM     ;IF R3 IS POINTING TO A BANK THAT IS LOWER
1841                                                                ;THAN MAXMEM
1842  002600  103673                        BLO     4$              ;THEN REPEAT FROM 4$
1843  002602  000337  000316        20$:    SWAB    @#BAKPAT
1844  002606  001656                        BEQ     TST3            ;REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
1845                                                                ;THE LOWER BYTE OF BAKPAT IS 0
1846  002610  000654                END3:   BR      END2
```

```
1847                                    ;;*****************************************************************
1848                                    ;*TEST 4         DUAL ADDRESS TEST B
1849                                            ;*(1)   THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
1850                                            ;*      AND READING THE ADDRESS IN THE LOCATION AND THEN
1851                                            ;*      WRITING AND READING ADDRESS COMPLEMENT
1852                                    ;;*****************************************************************
1853  002612  122737  000004  000404    TST4:   CMPB    #4,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE
1854  002620  001403                            BEQ     .+10
1855  002622  004767  003310                    JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1856  002626  000022                            22                      ;******ERROR NUMBER 22******
1857
1858  002630  005003                            CLR     R3
1859  002632  010100                    1$:     MOV     R1,R0
1860  002634  005703                            TST     R3              ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
1861  002636  001401                            BEQ     2$              ;IN THE LOCATION
1862  002640  005100                            COM     R0              ;OTHERWISE STORE COMPLEMENT
1863  002642  010021                    2$:     MOV     R0,(R1)+        ;OF THE ADDRESS
1864  002644  020105                            CMP     R1,R5           ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
1865  002646  103771                            BLO     1$
1866
1867  002650  020041                    3$:     CMP     R0,-(R1)        ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
1868  002652  001405                            BEQ     4$
1869  002654  105237  000301                    INCB    @#SADERR        ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
1870                                            ;BIT PROBLEM
1871  002660  004767  002714                    JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1872  002664  000023                            23                      ;******ERROR NUMBER 23******
1873
1874  002666  010100                    4$:     MOV     R1,R0
1875  002670  162700  000002                    SUB     #2,R0           ;CHECK THAT THE ADDRESS IS STORED AT
1876  002674  005703                            TST     R3              ;LOCATION IF R3 IS NOT 0
1877  002676  001401                            BEQ     5$              ;OTHERWISE CHECK FOR
1878  002700  005100                            COM     R0              ;ADDRESS COMPLEMENT
1879  002702  020104                    5$:     CMP     R1,R4
1880  002704  101361                            BHI     3$
1881  002706  112737  000001  000306            MOVB    #1,@#PASFLG     ;SET PASFLG FOR ERROR REPORT.
1882  002714  005103                            COM     R3              ;COMPLEMENT THE CONTENTS OF R3
1883  002716  001345                            BNE     1$              ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
1884                                            ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU
1885  002720  000733                    END4:   BR      END3
1886
```

```
1887                                ;;*****************************************************************
1888                                ;*TEST 5          MARCHING 1'S AND 0'S
1889                                        ;*(1)    THIS TEST WRITES A BACK GROUND OF THE WORD STORED
1890                                        ;*       AT BAKPAT.
1891                                        ;*(2)    READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
1892                                        ;*       AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
1893                                        ;*       DIRECTION OF MEMORY LOCATIONS.
1894                                        ;*(3)    READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
1895                                        ;*       WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
1896                                        ;*       IN  MIN. TO MAX. DIRECTION
1897                                        ;*(4)    REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
1898                                        ;*(5)    REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION
1899                                ;;*****************************************************************
1900
1901  002722  122737  000005  000404  TST5:  CMPB    #5,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE
1902
1903  002730  001403                        BEQ     .+10
1904  002732  004767  003200                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1905  002736  000024                        24                      ;******ERROR NUMBER 24*******
1906
1907  002740  004737  000120          1$:   JSR     PC,@#WRTMEM     ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1908                                                                ;WORD STORED IN BAKPAT
1909  002744  020041                2$:   CMP     R0,-(R1)        ;READ THE CONTENTS OF LOCATION POINTED BY R1
1910  002746  001403                        BEQ     3$              ;TO SEE IF IT HAS THE SAME VALUE AS R0
1911  002750  004767  002624                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1912  002754  000025                        25                      ;******ERROR NUMBER 25******
1913
1914  002756  000300                3$:   SWAB    R0
1915  002760  010011                        MOV     R0,(R1)         ;SWAP THE BYTES AT (R1)
1916  002762  021100                        CMP     (R1),R0         ;READ (R1) FOR CORRECT VALUE
1917  002764  001403                        BEQ     4$
1918  002766  004767  002606                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1919  002772  000026                        26                      ;******ERROR NUMBER 26******
1920
1921
1922  002774  000300                4$:   SWAB    R0              ;SWAP THE BYTES OF THE REGISTER
1923                                                                ;CONTAINING BACKGROUND PATTERN
1924  002776  001023                        BNE     9$              ;IF THE LOWER BYTE OF THE REGISTER
1925                                                                ;IS NOT 0 THEN THE PROGRAM IS READING
1926                                                                ;THE MEMORY TO CONTAIN A BACK GROUND OF
1927
1928                                                                ;BAKPAT AND WRITING THE SWAPPED WORD
1929
1930                                                                ;IN WHICH CASE GO TO 9$
1931
1932  003000  005703                5$:   TST     R3              ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1933                                                                ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
1934                                                                ;IF R3 EQUAL 0 THEN THE PROGRAM IS
1935                                                                ;READING/WRITING MIN. TO MAX. OTHERWISE
1936                                                                ;IT IS GOING IN MAX. TO MIN. DIRECTION
1937  003002  001023                        BNE     10$             ;IF R3 IS NOT CLEAR THEN GO TO 10$
1938  003004  062701  000002          6$:   ADD     #2,R1           ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1939  003010  020105                        CMP     R1,R5           ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
1940                                                                ;BE TESTED
1941  003012  103006                        BHIS    8$              ;IF R1>R5 THEN GO TO 8$ OTHERWISE
1942  003014  020011                7$:   CMP     R0,(R1)         ;READ (R1) FOR THE CORRECT DATA
```

```
1943  003016  001757                        BEQ     3$              ;WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1944                                                                  ;AND REPEAT UNTIL R1 > R5
1945  003020  004767  002554                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1946  003024  000027                        27                      ;******ERROR NUMBER 27******
1947
1948  003026  000753                        BR      3$
1949  003030  105237  000306        8$:     INCB    @#PASFLG
1950  003034  000300                        SWAB    R0
1951  003036  001742                        BEQ     2$              ;IF THE LOWER BYTE OF R0 IS ALL 0'S
1952                                                                  ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1953                                                                  ;AND READING BAKPAT GOING FROM MAX. TO MIN.[PASFLG=4]
1954  003040  005103                        COM     R3              ;OTHERWISE CLEAR R0
1955  003042  010401                        MOV     R4,R1           ;PUT THE LOWEST TESTING ADDRESS IN R1
1956  003044  000763                        BR      7$              ;AND BEGIN READING 0'S, WRITING 1'S AND
1957                                                                  ;READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]
1958
1959  003046  005703                9$:     TST     R3              ;IF R3 IS NON 0, I.E. PASFLG=3
1960  003050  001353                        BNE     5$              ;THEN READ BAKPAT, WRITE
1961                                                                  ;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1962                                                                  ;IN MIN. TO MAX. DIRECTION
1963  003052  020104                10$:     CMP     R1,R4           ;OTHERWISE TEST IS PROCEEDING IN MAX. TO
1964                                                                  ;MIN. DIRECTION.
1965  003054  101333                        BHI     2$              ;KEEP ON LOOPING UNTIL R1=R4
1966  003056  105237  000306                INCB    @#PASFLG
1967  003062  000300                        SWAB    R0
1968  003064  001753                        BEQ     7$              ;IF R0 SWAPPED HAS LOWER BYTE=0
1969                                                                  ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1970                                                                  ;AND READ BAKPAT GOING FROM MIN. TO MAX.
1971  003066  000714                ENDS:   BR      END4
1972
```

```
1973                                    ;;*****************************************************************
1974                                    ;*TEST 6         CELLS' VOLATILITY TEST
1975
1976                                            ;*(1)   THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
1977                                            ;*(2)   WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
1978                                            ;*      AND THEN INCREMENTS PASFLG
1979                                            ;*(3)   IT THEN READS/SWAPS BYTES/WRITES A LOCATION X FOR
1980                                            ;*      OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
1981                                            ;*(4)   REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
1982                                            ;*(5)   IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
1983                                            ;*      BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
1984                                            ;*      SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
1985                                            ;*(6)   REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
1986                                            ;*      BAKPAT INSTEAD OF BAKPAT.
1987
1988                                    ;;*****************************************************************
1989  003070  122737  000006  000404    TST6:   CMPB    #6,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE
1990
1991
1992  003076  001403                            BEQ     .+10
1993  003100  004767  003032                    JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1994  003104  000030                            30                      ;******ERROR NUMBER 30******
1995
1996  003106  004737  000120            RPT6:   JSR     PC,@#WRTMEM     ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1997                                                                    ;WORD STORED AT LOCATION BAKPAT
1998  003112  005037  000306                    CLR     @#PASFLG
1999  003116  010403                    1$:     MOV     R4,R3           ;SET R3
2000  003120  010401                    2$:     MOV     R4,R1           ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
2001  003122  020011                    3$:     CMP     R0,(R1)         ;CHECK (R1) FOR CORRECT DATA
2002  003124  001403                            BEQ     4$
2003  003126  004767  002446                    JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2004  003132  000031                            31                      ;******ERROR NUMBER 31******
2005
2006  003134  062701  000002            4$:     ADD     #2,R1           ;INCREMENT R1 BY 2
2007  003140  020105                            CMP     R1,R5           ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
2008  003142  103767                            BLO     3$
2009  003144  132737  000001  000306            BITB    #1,@#PASFLG     ;CHECK TO SEE IF PASFLG=0 OR 2
2010  003152  001002                            BNE     5$
2011  003154  105237  000306                    INCB    @#PASFLG        ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
2012
2013  003160  020305                    5$:     CMP     R3,R5           ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
2014  003162  103012                            BHIS    7$
2015  003164  012702  037776                    MOV     #37776,R2       ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
2016  003170  000313                    6$:     SWAB    (R3)
2017  003172  005302                            DEC     R2
2018  003174  001375                            BNE     6$
2019  003176  010337  000354                    MOV     R3,@#SAVLOC     ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
2020  003202  062703  020000                    ADD     #20000,R3       ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
2021                                                                    ;R3 TO POINT TO A LOCATION IN THE NEXT
2022                                                                    ;4K BANK OF MEMORY
2023  003206  000744                            BR      2$
2024  003210  105237  000306            7$:     INCB    @#PASFLG        ;MAKE PASFLG=2
2025  003214  000337  000316                    SWAB    @#BAKPAT        ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
2026  003220  001732                            BEQ     RPT6            ;THEN GO BACK TO THE LOCATION RPT6
2027  003222  000721                    END6:   BR      END5
2028
```

```
2029
```

```
                                ;;*****************************************************************
                                ;*TEST 7         SHIFTING DIAGONAL
                                ;
                                        ;*(1)   THIS TEST WRITES THE MEMORY WITH A BACKGROUND OF BAKPAT
                                        ;*(2)   IT WRITES A DIAGONAL OF SWAPPED BAKPAT THROUGH EACH MEMORY BANK
                                        ;*(3)   READS THE MEMORY FOR CORRECT DATA
                                        ;*(4)   SHIFTS THE DIAGONAL AND REPEATS STEP 3 UNTIL THE
                                        ;*      DIAGONAL HAS BEEN SHIFTED 64 TIMES
                                        ;*(5)   WRITES A BACKGROUND OF SWAPPED BAKPAT, A DIAGONAL OF
                                        ;*      BAKPAT AND REPEATS FROM STEP 3
                                ;;*****************************************************************
003224  122737  000007  000404  TST7:  CMPB    #7,@#STESTN     ;CHECK FOR PROPER TEST SEQUENCE

003232  001403                        BEQ     .+10
003234  004767  002676                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
003240  000032                        32                      ;******ERROR NUMBER 32******

003242  005037  000306          2$:   CLR     @#PASFLG
003246  010337  000304                MOV     R3,@#LOWBNK     ;LOWBNK CONTAINS ADDRESS OF THE LOWEST LOCATION
                                                              ;IN THE 4K BANK THAT CAN BE TESTED
003252  010302                        MOV     R3,R2
003254  052702  017776                BIS     #17776,R2       ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK
003260  005722                        TST     (R2)+           ;ADD 2 TO R2
003262  020502                        CMP     R5,R2
003264  103001                        BHIS    4$              ;IF R2 IS GREATER THAN R5 THEN GO TO 4$
003266  010502                        MOV     R5,R2           ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION
                                                              ;THAT CAN BE TESTED
003270  010337  000302          4$:   MOV     R3,@#STRTDI     ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE
                                                              ;DIAGONAL
003274  013701  000304                MOV     @#LOWBNK,R1     ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K
                                                              ;BANK
003300  013700  000316          6$:   MOV     @#BAKPAT,R0     ;STORE THE CONTENTS OF BAKPAT IN R0
003304  020103                        CMP     R1,R3           ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?
003306  001010                        BNE     10$             ;IF NOT THEN GO TO 10$
003310  062703  000002                ADD     #2,R3           ;THE FOLLOWING CODE IS USED TO PLACE THE
003314  032703  000176                BIT     #176,R3         ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL
003320  001402                        BEQ     8$              ;IN R3
003322  062703  000200                ADD     #200,R3
003326  000300                  8$:   SWAB    R0              ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN
003330  132737  000001  000306  10$:  BITB    #1,@#PASFLG     ;CONTENTS OF LOCATION PASFLG WILL BE EVEN IF THE
                                                              ;MEMORY IS BEING WRITTEN AND IT WILL BE ODD
                                                              ;IF IT IS ONLY BEING READ
003336  001001                        BNE     12$             ;IF IT IS BEING READ ONLY THEN GO TO 12$
003340  010011                        MOV     R0,(R1)         ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS
                                                              ;OF R0
003342  020011                  12$:  CMP     R0,(R1)         ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN
                                                              ;PROPER DATA
003344  001403                        BEQ     14$             ;IF IT IS OK THEN GO TO 14$
003346  004767  002226                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
003352  000033                        33                      ;******ERROR NUMBER 33******

003354  062701  000002          14$:  ADD     #2,R1           ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION
003360  020102                        CMP     R1,R2           ;IS IT THE END OF THE BANK ?
003362  103746                        BLO     6$              ;IF NOT THEN GO TO 6$
003364  005237  000410          16$:  INC     @#$DEVCT                ;TELL APT WE ARE STIL RUNNING OKAY
003370  105237  000306                INCB    @#PASFLG
```

```
 2086  003374  013703  000302                MOV     @#STRTDI,R3     ;LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL
 2087  003400  132737  000001  000306        BITB    #1,@#PASFLG     ;HAS THE READ OF THE MEMORY BEEN DONE ?
 2088  003406  001330                        BNE     4$              ;IF NOT THEN GO TO 4$
 2089  003410  005723                        TST     (R3)+           ;ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL
 2090  003412  020302                        CMP     R3,R2           ;AND UNLESS THE END OF THE BANK IS REACHED
 2091  003414  103003                        BHIS    18$             ;
 2092  003416  105737  000306                TSTB    @#PASFLG        ;OR THE DIAGONAL HAS BEEN ROTATED 64 TIMES
 2093  003422  100322                        BPL     4$              ;REPEAT FROM 4$
 2094  003424  013703  000304        18$:    MOV     @#LOWBNK,R3     ;MAKE R3 POINT TO THE LOWEST LOCATION IN THE
 2095                                                                ;IN THE BANK UNDER TEST
 2096  003430  000337  000316                SWAB    @#BAKPAT
 2097  003434  001715                        BEQ     4$              ;AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE
 2098                                                                ;SWAPPED BACK GROUND PATTERN THEN GO TO 4$
 2099  003436  010203                        MOV     R2,R3           ;MAKE THE PRESENT HIGH BOUNDRY AS THE NEXT
 2100                                                                ;LOW BOUNDRY
 2101  003440  020205                        CMP     R2,R5           ;UNLESS THE PRESENT HIGH BOUNDRY IS ALSO THE
 2102                                                                ;HIGH BOUNDRY FOR THE MEMORY UNDER TEST
 2103  003442  103677                        BLO     2$
 2104  003444  000666                END7:   BR      END6
```

```
                                ;;*********************************************************************
                                ;*TEST 10        READ RECOVERY GALLOPING TEST THROUGH EVERY 64TH CELL
                                ;
                                ;*(1)    THIS TEST WRITES THE MEMORY WITH A BACK GROUND PATTERN
                                ;*       STORED AT LOCATION BAKPAT
                                ;*(2)    TEST BEGINS AT LOWEST LOCATION BEING TESTED
                                ;*       (LETS NAME IT 'A')
                                ;*(3)    LETS NAME THE 1ST LOCATION IN THE ROW/COLUMN UNDER TEST AS 'B'.
                                ;*(4)    SWAPS BYTES FOR LOCATION 'A'.
                                ;*(5)    READS 'A'  READS 'B'
                                ;*(6)    'B' = 'B'+200 (MAKES 'B'=64TH CELL I.E. 200TH OCTAL
                                ;*       LOCATION FROM THE PRESENT LOCATION OF 'B')
                                ;*(7)    REPEATS STEPS 5 AND 6 UNTIL 'B' IS GREATER THAN THE
                                ;*       END OF THE 4K BANK OF THE MEMORY IN WHICH 'A' IS RESIDING
                                ;*(8)    A = A+2
                                ;*(9)    REPEATS  STEPS 3-8 UNTILL 'A' REACHES THE END OF THE BANK
                                ;*(10)   GOES TO THE NEXT 4K BANK OF MEMORY AND REPEATS STEPS
                                ;*       3-9 UNTIL THE END OF THE MEMORY
                                ;*(11)   AFTER EXECUTING THE TEST BYTES ARE SWAPPED AT
                                ;*       LOCATION BAKPAT AND STEPS 1-10 ARE REPEATED
                                ;*(12)   IN THIS TEST R0 IS POINTING TO LOCATION 'A', R1 TO
                                ;*       LOCATION 'B', R2 TO THE END OF THE 4K BANK IN WHICH THE
                                ;*       TEST IS TAKING PLACE AND R3 TO THE LOWEST LOCATION IN THE
                                ;*       COLUMN/ROW CONTAINING 'A' AND 'B'
                                ;*(13)   MOST OF THE CODE USED BY THIS TEST IS ALSO USED BY TEST 11
                                ;;*********************************************************************
003446  122737  000010  000404  TST10:  CMPB    #10,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE
003454  001403                         BEQ     .+10
003456  004767  002454                 JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
003462  000034                         34                      ;******ERROR NUMBER 34******
003464  010402                         MOV     R4,R2           ;SET R2 TO THE LOWEST MEMORY UNDER TEST
003466  052702  017776          RPT10:  BIS     #17776,R2       ;MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K
                                                                ;BANK FOR WHICH GALLOPING WILL BE PERFOMED
003472  062702  000002          GALLOP: ADD     #2,R2           ;INCREMENT R2 BY 2
003476  020205                         CMP     R2,R5           ;IF THE HIGH BOUNDRY OF THE TEST IS HIGHER THAN
003500  101401                         BLOS    2$
003502  010502                         MOV     R5,R2           ;THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2
003504  005046                  2$:    CLR     -(SP)
003506  010200                         MOV     R2,R0
003510  013740  000316          4$:    MOV     @#BAKPAT,-(R0)  ;WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF
                                                                ;BAKPAT
003514  020003                         CMP     R0,R3
003516  101374                         BHI     4$
003520  010301                  6$:    MOV     R3,R1           ;R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT
                                                                ;CAN BE TESTED IN THIS BLOCK
003522  023710  000316                 CMP     @#BAKPAT,(R0)   ;BEFORE STARTING THE GALLOPING TEST FOR LOCATION
                                                                ;(R0) CHECK IT
003526  001410                         BEQ     8$              ;CONTINUE IF OK
003530  010001                         MOV     R0,R1           ;OTHERWISE PREPARE TO REPORT THE ERROR
003532  013700  000316                 MOV     @#BAKPAT,R0
003536  004767  002036                 JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
003542  000035                         35                      ;******ERROR NUMBER 35******
```

```
2161  003544  010011                        MOV     R0,(R1)         ;RESTORE THE CONTENTS OF (R1)
2162  003546  010100                        MOV     R1,R0           ;RESTORE R0
2163
2164  003550  000310                 8$:     SWAB    (R0)
2165  003552  031011                 10$:    BIT     (R0),(R1)       ;CHECK TO SEE THAT NONE OF THE BITS SET
2166                                                                 ;IN (R0) ARE SET IN (R1) AND VICE VERSA
2167  003554  020001                        CMP     R0,R1           ;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1
2168
2169  003556  001412                        BEQ     12$
2170  003560  021137  000316                CMP     (R1),@#BAKPAT   ;CHECK THAT (R1) HAS BAKPAT IN IT
2171  003564  001407                        BEQ     12$
2172  003566  010046                        MOV     R0,-(SP)        ;SAVE R0 ON STACK
2173  003570  013700  000316                MOV     @#BAKPAT,R0     ;PLACE THE PATTERN WORD IN R0
2174  003574  004767  002000                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2175  003600  000036                        36                      ;******ERROR NUMBER 36******
2176
2177  003602  012600                        MOV     (SP)+,R0        ;RESTORE R0
2178  003604  021037  000320         12$:    CMP     (R0),@#SWAPAT   ;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT
2179  003610  001412                        BEQ     14$
2180  003612  010146                        MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
2181  003614  010001                        MOV     R0,R1           ;MAKE R1 POINT TO THE FAILING LOCATION
2182  003616  013700  000320                MOV     @#SWAPAT,R0     ;LOAD R0 WITH THE EXPECTED RESULT IN (R1)
2183  003622  004767  001752                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2184  003626  000037                        37                      ;******ERROR NUMBER 37******
2185
2186  003630  010011                        MOV     R0,(R1)         ;RECOVER (R1) FROM THE ERROR
2187  003632  010100                        MOV     R1,R0           ;RESTORE R0
2188  003634  012601                        MOV     (SP)+,R1        ;AND RESTORE R1
2189  003636  122737  000011  000404 14$:    CMPB    #11,@#$TESTN    ;IS THE PROGRAM EXECUTING TEST # 11 ?
2190  003644  001402                        BEQ     16$             ;IF SO THEN GO TO 16$
2191  003646  062701  000176                ADD     #176,R1
2192  003652  062701  000002         16$:    ADD     #2,R1           ;MAKE R1 POINT TO THE NEXT ADJACENT CELL
2193  003656  020102                        CMP     R1,R2           ;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDRY
2194  003660  103734                        BLO     10$             ;THEN REPEAT FROM 10$
2195  003662  000320                        SWAB    (R0)+           ;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST
2196                                                                 ;WAS BEING PERFORMED
2197  003664  122737  000011  000404        CMPB    #11,@#$TESTN    ;IS IT TEST 11 ?
2198  003672  001407                        BEQ     17$             ;IF SO THEN GO TO 17$
2199  003674  005723                        TST     (R3)+           ;OTHERWISE INCREMENT R3 BY 2
2200  003676  062716  000002                ADD     #2,(SP)         ;FOR EVERY ROW/COLUMN TESTED ADD 2
2201  003702  105716                        TSTB    (SP)
2202  003704  100002                        BPL     17$             ;UNTIL (SP) IS 200
2203  003706  161603                        SUB     (SP),R3         ;SUBTRACT 200 FROM R3
2204  003710  005016                        CLR     (SP)
2205  003712  032700  000177         17$:    BIT     #177,R0         ;AT A 64TH CALL BOUNDRY?
2206  003716  001002                        BNE     18$             ;BRANCH IF NO
2207  003720  005237  000410                INC     @#$DEVCT        ;TELL APT WE ARE STILL RUNNING
2208  003724  020002                 18$:    CMP     R0,R2           ;IF R0 HAS NOT REACHED THE END OF THE BOUNDRY
2209  003726  103674                        BLO     6$              ;THEN REPEAT FROM 6$
2210  003730  162603                        SUB     (SP)+,R3        ;RESTORE SP AND R3
2211  003732  000337  000320                SWAB    @#SWAPAT
2212  003736  000337  000316                SWAB    @#BAKPAT
2213  003742  001660                        BEQ     2$              ;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 2$
2214  003744  010203                        MOV     R2,R3           ;OTHERWISE MAKE THE PRESENT HIGH BOUNDRY AS THE
2215                                                                 ;NEXT LOW BOUNDRY
2216  003746  020205                        CMP     R2,R5
```

```
2217  003750  001410                          BEQ     END10           ;IF PREVIOUS HIGH BOUNDRY WAS THE END OF THE
2218                                                                    ;TEST BOUNDRY THEN EXIT THE TEST
2219  003752  032702  017776                  BIT     #17776,R2       ;WAS IT A 4K BOUNDRY ?
2220  003756  001025                          BNE     RPT11           ;IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG
2221                                                                    ;GALLOPING TEST DISABLED
2222  003760  122737  000011  000404          CMPB    #11,@#STESTN    ;IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11
2223  003766  001421                          BEQ     RPT11
2224  003770  000636                          BR      RPT10           ;OTHERWISE REPEAT TEST 10
2225  003772  000624                  END10:  BR      END7
2226
2227
2228
```

```
                                ;;*****************************************************************
                                ;*TEST 11        READ RECOVERY LONG GALLOPING/FAST GALLOPING TEST
                                ;
                                ;*(1)    THIS TEST WRITES MEMORY WITH BAKPAT
                                ;*(2)    THE TEST BEGINS AT THE LOWEST LOCATION BEING TESTED
                                ;*       (LETS NAME II 'B')
                                ;*(3)    'A'←'B' [MOVE THE ADDRESS OF 'B' TO THE POINTER FOR LOCATION 'A']
                                ;*(4)    SWAPS BYTES FOR LOCATION 'A'
                                ;*(5)    READS 'A', READS 'B'
                                ;*(6)    'B'='B'+2
                                ;*(7)     IF GALLOPING OPTION BIT AT $SWREG IS HIGH THEN STEPS 4 AND 5
                                ;*       ARE REPEATED UNTIL 'B' REACHES THE HIGHEST MEMORY LOCATION
                                ;*       OF THE 4K BANK IN WHICH 'A' IS RESIDING, THEN 'A' IS
                                ;*       DECREMENTED BY 2 AND AFTER MAKING 'B' TO POINT TO THE LOWEST
                                ;*       LOCATION OF THE 4K MEMORY BANK CONTAINING 'A' STEPS 3,4,5 AND
                                ;*       6 ARE REPEATED UNTIL 'A' EQUALS THE END OF THE ENTIRE MEMORY
                                ;*(8)    IF GALLOPING OPTION BIT IS NOT HIGH THEN STEPS 4 AND 5 ARE
                                ;*       REPEATED UNTIL 'B' IS POINTING TO A CELL IN THE NEXT COLUMN
                                ;*       IF SEQUENTIAL CELLS LIE ALONG THE ROW, OR THE NEXT ROW
                                ;*       IF SEQUENTIAL CELLS LIE ALONG THE COLUMN, AT WHICH TIME
                                ;*       STEPS 2,3,4,5 AND 7 ARE REPEATED UNTIL THE END OF THE MEMORY
                                ;*(9)    TEST IS REPEATED FOR THE OPPOSITE BACKGROUND DATA
                                ;*(10)   IN THIS TEST R0 POINTS TO LOCATION 'A', R1 TO LOCATION
                                ;*       'B', R2 TO THE HIGHEST LOCATION AND R3 TO THE LOWEST
                                ;*       LOCATION IN A 64/4K CELL BOUNDRY
                                ;*(11)   MOST OF THE CODE USED BY TEST 10 IS ALSO USED BY THIS TEST

                                ;;*****************************************************************
003774  122737  000011  000404  T$T11:  CMPB    #11,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE

004002  001403                        BEQ     .+10
004004  004767  002126                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
004010  000040                        40                      ;******ERROR NUMBER 40******

004012  010402                        MOV     R4,R2           ;MAKE R2 TO POINT TO THE LOWEST LOCATION
                                                              ;UNDER TEST
004014  105777  174430                TSTB    @SWR            ;LONG GALLOP ENABLED?
004020  100004                        BPL     RPT11           ;BRANCH IF NO
004022  004767  002540                JSR     PC,PNTMES       ;TYPE "GLP"
004026  046107  000120                .ASCIZ  /GLP/
004032  105777  174412        RPT11:  TSTB    @SWR            ;LONG GALLOPING ENABLED?
004036  100613                        BMI     RPT10           ;BRANCH IF YES
                                                              ;TO RPT10
004040  052702  000176                BIS     #176,R2         ;OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
                                                              ;TO GET THE HIGH BOUNDRY

004044  000612                        BR      GALLOP          ; PERFORM GALLOPING TEST
```

```
                                ;;*****************************************************************
                                ;*TEST 12        WORST CASE TESTING FOR CORE MEMORY
                                ;*(1)    STARTING FROM THE LOWEST LOCATION UNDER TEST THE MEMORY
                                ;*       IS WRITTEN WITH A BACKGROUND OF BAKPAT, HOWEVER LOCATIONS
                                ;*       HAVING ADDRESS SUCH THAT EXCLUSIVE OR OF ADDRESS BITS 1 &
                                ;*       8 = 1 ARE WRITTEN TO A VALUE OF SWAPPED BAKPAT
                                ;*(2)    STARTING FROM THE LOWEST LOCATION THE MEMORY IS CHECKED
                                ;*       TO CONTAIN THE CORRECT DATA AS EXPLAINED IN STEPS 3 & 4,
                                ;*       UNTILL THE HIGHEST LOCATION UNDER TEST IS REACHED
                                ;*(3)    READ EACH LOCATION FOR THE CORRECT CONTENT
                                ;*(4)    COMPLEMENT THE LOCATION AND READ IT; COMPLEMENT THE LOCATION
                                ;*       BACK TO ITS ORIGINAL VALUE AND READ IT AGAIN
                                ;*(5)    STARTING FROM THE HIGHEST LOCATION UNDER TEST REPEAT STEPS
                                ;*       3 & 4 UNTIL THE LOWEST LOCATION UNDER TEST IS REACHED
                                ;*(6)    REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
                                ;*       OF ADDRESS BITS 8 & 13 =1 ARE WRITTEN TO SWAPPED BAKPAT
                                ;*(7)    REPEAT STEPS 1-5, HOWEVER THIS TIME LOCATIONS WITH XOR
                                ;*       OF ADDRESS BITS 3 & 9 =1 ARE WRITTEN TO SWAPPED BAKPAT
                                ;*(8)    REPEAT STEPS 1-7 WITH A BACKGROUND OF SWAPPED BAKPAT AND
                                ;*       THE LOCATIONS TO BE WRITTEN TO SWAPPED BAKPAT WRITTEN TO
                                ;*       BAKPAT.
                                ;;*****************************************************************
004046  122737  000012  000404  TST12:  CMPB    #12,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE
004054  001403                         BEQ     .+10
004056  004767  002054                 JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
004062  000041                         41                      ;******ERROR NUMBER 41******

004064  012702  000002                 MOV     #2,R2           ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1
004070  012703  000400                 MOV     #400,R3         ;AND 8
004074  112737  000001  000306  1$:    MOVB    #1,@#PASFLG     ;INITIALIZE THE COUNTER FOR THE SUBTEST
004102  010401                 2$:     MOV     R4,R1           ;PLACE THE STARTING ADDRESS OF MEMORY UNDER
                                                               ;TEST IN R1
004104  013700  000316          4$:     MOV     @#BAKPAT,R0
004110  030201                         BIT     R2,R1           ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET
004112  001004                         BNE     8$              ;IF IT IS SET THEN GO TO 8$
004114  030301                         BIT     R3,R1           ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET
004116  001404                         BEQ     12$             ;IF IT IS NOT SET THEN GO TO 12$
004120  005100                 6$:     COM     R0              ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS
                                                               ;POINTED BY R2 & POINTED BY R3 = 1 IN WHICH
                                                               ;CASE PREPARE TO WRITE THE LOCATION
                                                               ;WITH A COMPLEMENT OF LOCATIONS NOT MEETING
                                                               ;THIS CONDITION
004122  000402                         BR      12$
004124  030301                 8$:     BIT     R3,R1           ;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND
                                                               ;CHECK ADDRESS BIT POINTED BY R3
004126  001774                         BEQ     6$              ;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 6$
004130  132737  000002  000306  12$:    BITB    #2,@#PASFLG     ;IS IT 2ND OR 3RD PASS OF THE SUBTEST ?
004136  001001                         BNE     14$             ;IF SO THEN READ THE MEMORY
```

```
2325  004140  010011                      MOV     R0,(R1)         ;OTHERWISE WRITE THE MEMORY BFORE READING IT
2326  004142  020011                14$:  CMP     R0,(R1)         ;READ THE MEMORY FOR CORRECT CONTENTS
2327  004144  001403                      BEQ     16$
2328  004146  004767  001426              JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2329  004152  000042                      42                      ;******ERROR NUMBER 42******
2330
2331  004154  012746  000002          16$:  MOV     #2,-(SP)
2332  004160  005100                18$:  COM     R0
2333  004162  005111                      COM     (R1)
2334  004164  020011                      CMP     R0,(R1)         ;READ THE MEMORY AGAIN
2335  004166  001404                      BEQ     19$
2336  004170  004767  001404              JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2337  004174  000043                      43                      ;******ERROR NUMBER 43******
2338
2339  004176  010011                      MOV     R0,(R1)         ;RESTORE THE LOCATION (R1)
2340  004200  005316                19$:  DEC     (SP)
2341  004202  001366                      BNE     18$             ;EXECUTE THE CODE FROM 18$ TWICE
2342  004204  005726                      TST     (SP)+           ;RESTORE THE STACK POINTER
2343  004206  122737  000003  000306      CMPB    #3,@#PASFLG     ;IS IT THE 3RD PASS OF THE SUBTEST ?
2344  004214  001412                      BEQ     20$             ;IF SO THEN GO TO 20$
2345  004216  062701  000002              ADD     #2,R1           ;IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN
2346                                                              ;MIN. TO MAX. DIRECTION
2347  004222  020105                      CMP     R1,R5           ;HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?
2348  004224  103727                      BLO     4$              ;IF NOT THEN REPEAT FROM 4$
2349  004226  105237  000306              INCB    @#PASFLG
2350  004232  122737  000002  000306      CMPB    #2,@#PASFLG     ;IF IT IS THE 2ND PASS OF THE SUBTEST
2351  004240  001720                      BEQ     2$              ;THEN REPEAT FROM 2$
2352  004242  162701  000002          20$:  SUB     #2,R1           ;OTHERWISE EXECUTE THE TEST IN MAX. TO MIN.
2353                                                              ;DIRECTION
2354  004246  020104                      CMP     R1,R4           ;HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?
2355  004250  103315                      BHIS    4$              ;IF NOT THEN REPEAT FROM 4$
2356  004252  012702  020000              MOV     #20000,R2       ;PREPARE TO CHECK THE MEMORY WITH THE XOR OF
2357                                                              ;ADDRESS BITS 8 AND 13
2358  004256  105237  000307              INCB    @#PASFLG+1      ;THE SUB TEST HAS CHECKED THE XOR ONE KIND
2359  004262  123727  000307  000002      CMPB    @#PASFLG+1,#2   ;HAS TWO XOR COMBINATIONS BEEN CHECKED ?
2360  004270  103701                      BLO     1$              ;IF NOT THEN GO TO 1$
2361  004272  101004                      BHI     22$             ;IF ALL THREE HAVE BEEN CHECKED THEN GO TO 22$
2362  004274  012702  000010              MOV     #10,R2          ;IF IT IS THE 2ND XOR COMBINATION THEN CHECK
2363  004300  006303                      ASL     R3              ;FOR ADDRESS BITS 3 & 8
2364  004302  000674                      BR      1$
2365  004304  005137  000316          22$:  COM     @#BAKPAT        ;IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED
2366  004310  105737  000316              TSTB    @#BAKPAT
2367  004314  001654                      BEQ     TST12           ;BAKPAT THEN RE-EXECUTE THE TEST
2368  004316  000625                END12:  BR      END10
```

```
2369                    ;;*****************************************************************
2370                    ;*TEST 13        WRITE RECOVERY TEST
2371                    ;*      THIS TEST DIFFERS FROM 0-12 IN THAT IT CONSISTS OF A SMALL TEST PROGRAM
2372                    ;*      ACTUALLY RUNNING IN THE 4K BANK UNDER TEST.
2373                    ;*              THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
2374                    ;*      TO AID IN THE DEBUG, BEFORE A A BANK IS ENTERED "TST13 BANK XX"
2375                    ;*      IS TYPED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
2376                    ;*      BANK FAILED.
2377                    ;*      THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH "MOV R2,-(PC)"
2378                    ;*      AND THE OTHER 1/2 CONTAINING "177667". "177667" IS THE COMPLEMENT
2379                    ;*      OF "JMP (R0)" INSTRUCTION.
2380                    ;*      R2 CONTAINS "COM -(R1)" INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS
2381                    ;*      THE HIGHEST TEST ADDRESS IN THAT BANK. THE HIGHEST TEST ADDRESS IS
2382                    ;*      USUALLY ON 4K BOUNDARIES. WHEN TESTING BANK 0 RELOCATED,HOWEVER
2383                    ;*      R1 CONTAINS THE FIRST FREE TEST ADDRESS BELOW THE DIAGNOSTIC.
2384                    ;*      IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
2385                    ;*              THE TEST EXECUTION IS AS FOLLOWS:
2386                    ;*                      1. THE "MOV R2,-(PC)" INSTRUCTION EXECUTES STORING
2387                    ;*                        THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC).
2388                    ;*                      2. SINCE R2 CONTAINS A "COM -(R1)" INSTRUCTION IT COMPLEMENTS
2389                    ;*                        THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
2390                    ;*                        "177667" SO AFTER THE COM -(R1) IT EQUALS 110
2391                    ;*                        CLEVERLY THIS IS THE "JMP (R0)" INSTRUCTION.
2392                    ;*                      3. THIS SEQUENCE CONTINUES UNTIL THE "MOV R2,-(PC) INSTRUCTIONS
2393                    ;*                        REACH THE MIDDLE OF THE TEST BANK. THEN THE "JMP (R0)" INSTRUCTION IS
2394                    ;*                        AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
2395                    ;*                        TO TEST 13.
2396                    ;*                      4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.
2397                    ;*
2398                    ;;*****************************************************************
2399  004320  122737  000013  000404  TST13:  CMPB    #13,@#STESTN    ;CHECK FOR PROPER TEST SEQUENCE
2400  004326  001403                          BEQ     .+10
2401  004330  004767  001602                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2402  004334  000044                          44                      ;******ERROR NUMBER 44******
2403
2404  004336  012702  010247          1$:     MOV     #10247,R2       ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
2405                                                                  ;IN R2.
2406  004342  012700  177667                  MOV     #177667,R0      ;PLACE THE COMPLEMENT OF THE INSTRUCTION
2407                                                                  ;JMP (R0) IN R0
2408                          ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
2409                          ;SINCE THE TEST STORES "MOV R2,-(PC) IN 1/2 AND 177667 IN THE OTHER 1/2
2410
2411  004346  010546          2$:     MOV     R5,-(SP)        ;SAVE R5
2412  004350  010446                  MOV     R4,-(SP)        ;STORE LOWEST ADDRESS ON STACK
2413  004352  000241          29$:    CLC
2414  004354  006005                  ROR     R5              ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
2415  004356  006004                  ROR     R4              ;DO SAME FOR LOWEST ADDRESS
2416  004360  160405                  SUB     R4,R5           ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
2417  004362  006005                  ROR     R5              ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
2418  004364  103002                  BCC     30$             ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS.
2419  004366  062716  000002          ADD     #2,(SP)         ;INCREASE LOWEST TEST ADDRESS BY 2
2420  004372  012604          30$:    MOV     (SP)+,R4        ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
2421  004374  012605                  MOV     (SP)+,R5        ;RESTORE HIGHEST TEST ADDRESS
2422  004376  010403                  MOV     R4,R3           ;PLACE THE LOWEST LOCATION UNDER TEST
2423                                                          ;IN R3
2424  004400  000405                  BR      28$             ;LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0
```

```
DZKMA   MACY11 27(1006)  18-FEB-77  12:35  PAGE 52
DZKMAC.P11    18-FEB-77 12:30          T13     WRITE RECOVERY TEST

2425  004402  042703  017776          3$:     BIC     #17776,R3       ;CAUSE R3 TO POINT TO THE LOWEST LOCATION
2426                                                                  ;IN THE 4K BANK UNDER TEST
2427  004406  105737  000405                  TSTB    @#REL           ;ARE WE RELOCATED?
2428  004412  100504                          BMI     14$             ;BRANCH IF YES-TEST BANK0 ONLY-
2429  004414  020305                  28$:    CMP     R3,R5           ;IF R3 IS HIGHER THAN THE HIGHEST LOCATION
2430  004416  103102                          BHIS    14$             ;UNDER TEST THEN EXIT
2431                                  ;IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0
2432  004420  020527  020000                  CMP     R5,#20000       ;IS HIGHEST TEST ADDRESS BELOW 4K?
2433  004424  103002                          BHIS    31$             ;BRANCH IF NO
2434  004426  010501                          MOV     R5,R1           ;SET R1 TO HIGHEST TEST ADDRESS IN BANK0
2435  004430  000405                          BR      32$
2436
2437  004432  010301                  31$:    MOV     R3,R1           ;SET R1 TO LOWEST CURRENT TEST ADDRESS
2438  004434  042701  017776                  BIC     #17776,R1       ;CLEAR LOW ORDER ADDRESS BITS
2439  004440  062701  020000                  ADD     #20000,R1       ;CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
2440                                                                  ;OF THE 4K BANK BEING POINTED BY R3
2441  004444  020137  000340          32$:    CMP     R1,@#MAXMEM     ;IF R1 IS HIGHER THAN MAX. OF THE
2442  004450  101065                          BHI     14$             ;MEMORY+2 ALTHOUGH R3 IS LESS THAN R5
2443                                                                  ;THEN THE HIGHEST LOCATION UNDER
2444                                                                  ;TEST IS NOT IN A 4K BANK EXIT
2445
2446  004452  132737  000001  000306          BITB    #1,@#PASFLG     ;IS THE LOWEST BIT OF LOCATION PASFLG
2447  004460  001101                          BNE     16$             ;SET? IN WHICH CASE BACK GROUND HAS
2448                                                                  ;ALREADY BEEN WRITTEN AND WRITE RECOVERY
2449                                                                  ;TEST IS BEING PERFORMED
2450
2451  004462  020304                  4$:     CMP     R3,R4           ;OTHERWISE WRITE THE BACKGROUND
2452  004464  103430                          BLO     8$              ;DEFINED AT STEP 3.
2453  004466  105737  000307                  TSTB    @#PASFLG+1      ;IS THE TEST JUST DOING READ, I.E.
2454  004472  001002                          BNE     6$              ;IS THE PASFLG+1 LOCATION NON ZERO? IF SO
2455                                                                  ;THEN GO TO 6$
2456  004474  012713  010247                  MOV     #10247,(R3)     ;WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,-(PC)
2457  004500  020213                  6$:     CMP     R2,(R3)         ;READ (R3) TO CONTAIN CORRECT DATA
2458  004502  001421                          BEQ     8$
2459  004504  010046                          MOV     R0,-(SP)        ;SAVE R0
2460  004506  010146                          MOV     R1,-(SP)        ;AND R1 ON THE STACK
2461  004510  010301                          MOV     R3,R1
2462  004512  010200                          MOV     R2,R0           ;SET R0= GOOD DATA FOR ERROR PRINTOUT
2463  004514  004767  001060                  JSR     PC,ERROR        ;#ERROR# REPORT ERROR MESSAGE
2464  004520  000045                          45                      ;******ERROR NUMBER 45******
2465
2466  004522  012601                          MOV     (SP)+,R1        ;RESTORE R1
2467  004524  012600                          MOV     (SP)+,R0        ;AND R0
2468  004526  105737  000306                  TSTB    @#PASFLG        ;IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
2469                                                                  ;THE PROPER DATA THEN WE DON'T WANT TO GO AND
2470                                                                  ;EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
2471                                                                  ;TEST
2472  004532  001005                          BNE     8$              ;BRANCH IF PASFLG NOT =0
2473
2474  004534  010200                          MOV     R2,R0   ;SAVE FOR ERROR REPORT
2475  004536  004767  001036                  JSR     PC,ERROR        ;#ERROR# REPORT ERROR MESSAGE
2476  004542  000046                          46                      ;******ERROR NUMBER 46******
2477
2478  004544  000664                          BR      END12           ;ABORT TST 13.
2479
2480  004546  062703  000002          8$:     ADD     #2,R3           ;INCREMENT R3 BY 2
```

```
2481  004552  162701  000002            SUB     #2,R1           ;DECREMENT R1 BY 2
2482  004556  020105                    CMP     R1,R5           ;WRITE THE BACKGROUND DEFINED AT STEP 4.
2483  004560  103014                    BHIS    12$
2484  004562  020103                    CMP     R1,R3           ;HAS STORING THE 177667 REACHED WHERE "MOV R2,-(PC) IS?
2485  004564  103405                    BLO     10$             ;BRANCH IF YES DON'T DESTROY THE MOV R2,-(PC) IS.
2486  004566  105737  000307            TSTB    @#PASFLG+1      ;IS THE THE READ ONLY CHECK PASS?
2487  004572  001002                    BNE     10$             ;BRANCH IF YES
2488  004574  012711  177667            MOV     #177667,(R1)    ;WRITE THE LOCATION WITH THE COMPLEMENT OF THE
2489                                                            ;OP CODE JMP (R0)
2490  004600  020011            10$:    CMP     R0,(R1)         ;READ R1 TO CONTAIN CORRECT DATA
2491  004602  001403                    BEQ     12$
2492  004604  004767  000770            JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2493  004610  000047                    47                      ;******ERROR NUMBER 47******
2494
2495  004612  020301            12$:    CMP     R3,R1           ;IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
2496  004614  103722                    BLO     4$              ;THEN REPEAT FROM 4$
2497
2498                            ;RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
2499
2500  004616  062703  020000    13$:    ADD     #20000,R3       ;OTHERWISE GO TO THE NEXT 4K BANK
2501  004622  000667                    BR      3$
2502
2503  004624  122737  000001  000306  14$:    CMPB    #1,@#PASFLG     ;THE PROGRAM CONTROL COMES HERE AS FOLLOWS
2504                                                                    ;1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
2505                                                                    ;  WRITE/READ CYCLE FOR THE BACK GROUND
2506                                                                    ;  AND WANTS TO BEGIN THE WRITE RECOVERY TEST
2507  004632  001440                    BEQ     24$             ;2-PASFLG=1, PROGRAM HAS JUST COMPLETED
2508                                                            ;  THE WRITE RECOVERY TEST AND WANTS TO
2509                                                            ;  READ MEMORY FOR CORRECT DATA
2510  004634  103630                    BLO     END12           ;3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
2511                                                            ;  MEMORY AND WANTS TO GO THE NEXT TEST.
2512
2513  004636  105137  000307            COMB    @#PASFLG+1      ;ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
2514                                                            ;ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
2515                                                            ;ENTRY DISABLE READ ONLY
2516  004642  001241                    BNE     2$
2517  004644  012702  005141            MOV     #5141,R2        ;PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
2518                                                            ;IN R2
2519  004650  012700  177740            MOV     #13$-.-6,R0     ;PLACE THE RETURN ADDRESS IN R0 AS 13$
2520  004654  060700                    ADD     PC,R0           ;THUS WHEN THE READ RECOVERY TEST REACHES
2521                                                            ;THE MIDDLE OF THE 4K MEMORY THEN THE
2522                                                            ;INSTRUCTION EXECUTED WILL BE JMP (R0)
2523                                                            ;BRANCHINGING THE PROGRAM TO 13$
2524  004656  105237  000306    15$:    INCB    @#PASFLG        ;INCREMENT PASFLG BY 1.
2525  004662  000631                    BR      2$
2526
2527  004664  032777  000020  173556  16$:    BIT     #20,@SWR        ;HAS THE PRINTOUTS BEEN SUPRESSED ?
2528  004672  001017                    BNE     18$             ;IF SO THEN GO TO 18$
2529  004674  105737  000042            TSTB    @#42            ;IS THE PROGRAM RUNNING UNDER ACT?
2530  004700  001014                    BNE     18$             ;BRANCH IF YES
2531  004702  004767  001660            JSR     PC,PNTMES       ;  TYPE THE BANK UNDER TEST
2532  004706  051524  030524  020063    .ASCIZ  /TST13 BANK/
2533  004714  040502  045516      000
2534          004722                    .EVEN
2535  004722  004767  002476            JSR     PC,GETBNK       ;GET BANK NO. UNDER TEST INTO DECWRD FOR PRINT.
2536  004726  004767  001662            JSR     PC,$TPDEC       ;TYPE BANK NO. UNDER TEST
```

```
2537
2538  004732  000113                  18$:    JMP     (R3)            ;BEGIN EXECUTING MOV R2,-(PC) ,COM -(R1) SEQUENCE IN TES
2539
2540
2541  004734  105137  000307          24$:    COMB    @#PASFLG+1
2542  004740  012700  000110                  MOV     #110,R0         ;PLACE THE OP CODE FOR JMP (R0) IN R0
2543  004744  000744                          BR      15$             ; READ THE MEMORY FOR CORRECT DATA AFTER
2544                                                                  ;INCREMENTING PASFLG TO 2
2545
2546                                  ;TST13 EXITS VIA END12.
2547
```

```
                                        PARITY OR RELOCATE?

2548  004746  012737  000377  000316  RELOC:  MOV     #377,@#BAKPAT
2549  004754  105737  000276                  TSTB    @#MMAVA         ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2550  004760  001065                          BNE     CONTMM          ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2551                                                                  ;MEMORY MANAGEMENT
2552  004762  032777  001000  173460          BIT     #1000,@SWR      ;RELOCATION WANTED?
2553  004770  001046                          BNE     CKDONE          ;BRANCH IF NO
2554  004772  105737  000405                  TSTB    @#REL           ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2555  004776  100420                          BMI     RELOER          ; PLACE THE PROGRAM BACK IN LOWER CORE
2556  005000  112737  000200  000405          MOVB    #200,@#REL      ;OTHERWISE PREPARE TO RELOCATE
2557
2558                                  ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
2559
2560
2561  005006  004767  001554                  JSR     PC,PNTMES       ;TYPE "RELOC"
2562  005012  042522  047514  000103          .ASCIZ /RELOC/
2563                                          .EVEN
2564  005020  013705  000340                  MOV     @#MAXMEM,R5     ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
2565                                                                  ;AVAILABLE MEMORY
2566  005024  014445          2$:     MOV     -(R4),-(R5)             ;RELOCATE THE PROGRAM
2567  005026  020427  000430          CMP     R4,#BEGIN-50    ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN-50
2568  005032  101374                  BHI     2$
2569  005034  000165  000050          JMP     50(R5)
2570
2571                                  ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2572
2573
2574  005040  013705  000346  RELOER: MOV     @#SAVR5,R5      ;RESTORE R5
2575  005044  105737  000405          TSTB    @#REL           ;IS DIAGNOSTIC IN RELOCATED STATE?
2576  005050  100016                  BPL     CKDONE          ;BRANCH IF NO
2577
2578  005052  012704  000430          MOV     #BEGIN-50,R4    ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2579  005056  012524          2$:     MOV     (R5)+,(R4)+
2580  005060  020537  000340          CMP     R5,@#MAXMEM
2581  005064  103774                  BLO     2$
2582  005066  105037  000405          CLRB    @#REL
2583  005072  010537  000346          MOV     R5,@#SAVR5      ;SAVE R5
2584  005076  012706  000500          MOV     #BEGIN,SP       ;RESET STACK TO LOWER MEMORY
2585  005102  010637  000350          MOV     SP,@#SAVR6      ;"BEGIN" USES THIS TO RESET THE STACK.
2586  005106  000137  005112  CKDONE: JMP     @#LOWER         ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2587
2588
2589
2590  005112  105737  000315  LOWER:  TSTB    @#SAVKBB        ;HERE DUE TO ↑C TYPED?
2591  005116  001073                  BNE     $TPSTK          ;BRANCH IF YES (TYPE ERROR STACK)
2592  005120  004767  001714  TSTMM:  JSR     PC,MEMMNG       ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
2593                                                          ;IS AVAILABLE
2594  005124  105737  000276          TSTB    @#MMAVA         ;IS MEM. MANAG. AVAILABLE ?
2595  005130  001462                  BEQ     ENDPAS          ;BRANCH IF NO
2596  005132  000402                  BR      $CNTMM          ;BEGIN TESTING ABOVE 28K
2597  005134  004767  002052  CONTMM: JSR     PC,UPMM         ;GO TO UPDATE MEM. MANAG. REGISTERS
2598  005140  012703  000324  $CNTMM: MOV     #LOWTWO,R3      ;MAKE R3 POINT TO THE LOCATION LOWTWO
2599  005144  004767  002160          JSR     PC,GETSIZ       ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2600                                                          ;OF THE LOWEST ADDRESS UNDER TEST
2601  005150  012704  020000          MOV     #20000,R4       ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2602                                                          ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
2603  005154  020237  172342          CMP     R2,@#172342     ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
```

```
2604
2605  005160  103405                      BLO     2$              ;PAR1 ?
2606  005162  050104                      BIS     R1,R4           ;IF SO THEN GO 2$
2607                                                              ;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2608  005164  162702  000200              SUB     #200,R2         ;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2609  005170  004767  001650              JSR     PC,MMREG        ; SET MEM. MANAG. REGISTERS
2610  005174  004767  002130      2$:     JSR     PC,GETSIZ       ;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2611                                                              ;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2612                                                              ;OF LOCATION HIGHADD IN R1
2613  005200  004767  000020              JSR     PC,MAXADR       ; GET THE ADDRESS OF MAX. MEM. UNDER TEST
2614  005204  010005                      MOV     R0,R5
2615  005206  004767  002116              JSR     PC,GETSIZ       ; PREPARE TO SET UP LOCATION MAXMEM
2616  005212  004767  000006              JSR     PC,MAXADR       ; GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2617  005216  010013                      MOV     R0,(R3)         ;AND STORE INTO "MAXMEM"
2618  005220  000167  174242              JMP     CLRMEM          ;GO TEST A 24K SLICE ABOVE 28K.
2619
2620
2621                                  ;MAXADR - SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K.
2622                                  ;REGISTERS:
2623                                  ;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
2624                                  ;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
2625                                  ;R2= PAR BLOCK NO. CURRENTLY UNDER TEST.
2626
2627  005224  010046          MAXADR: MOV     R0,-(SP)        ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
2628  005226  012700  172356          MOV     #172356,R0      ;R0=PAR7 UNIBUS ADDRESS
2629                                  ;**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2630  005232  162716  020000  2$:     SUB     #20000,(SP)     ;DECREMENT VIRTUAL ADDRESS BY 4K
2631  005236  050116                  BIS     R1,(SP)         ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
2632  005240  020240                  CMP     R2,-(R0)        ;DOES CURRENT PAR= TEST BLOCK NO.?
2633  005242  001410                  BEQ     3$              ;BRANCH IF YES
2634  005244  020027  172340          CMP     R0,#172340      ;ARE WE AT PAR0?
2635  005250  101370                  BHI     2$              ;NO KEEP TRYING
2636                                  ;**END LOOP TO FIND PAR ADDRESS UNDER TEST
2637  005252  005720                  TST     (R0)+           ;SET TO PAR CURRENT
2638  005254  021002                  CMP     (R0),R2 ;IS THE PAR BLOCK UNDER TEST [illegible] THAN ALLOWED?
2639  005256  003006                  BGT     4$              ;BRANCH IF YES (FALL [illegible]O ENDPAS)
2640  005260  012716  157776          MOV     #157776,(SP)    ;EXIT WITH MAXADR= 28[illegible]K SEGMENT TEST SIZE
2641  005264  012600          3$:     MOV     (SP)+,R0        ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2642  005266  062700  000002          ADD     #2,R0           ;MAKE MAXIMUM MEMORY+2
2643  005272  000207                  RTS     PC              ;AND EXIT MAXADR ROUTINE
2644
2645  005274  022626          4$:     CMP     (SP)+,(SP)+     ;FIXUP STACK
2646                                                          ;AND FALL THRU TO ENDPAS.
```

```
2647                                    ;* TYPE ROUTINE FOR ERROR STACK
2648                                    ;* ---------------------------
2649                                    ;*
2650                                    ;*      THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
2651                                    ;*      FOR ONLY THE FAILING BITS IS REQUIRED OR NOT
2652
2653
2654
2655  005276  032777  000020  173144  ENDPAS: BIT     #20,@SWR        ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
2656  005304  001055                  BNE     $EOP            ;IF NOT THEN GO TO $EOP
2657  005306  012746  177777          $TPSTK: MOV     #-1,-(SP)       ;THE PROGRAM HAS REACHED THE END AND ERROR
2658                                                          ;STACK AND END OF PASS WILL BE TYPED OUT
2659  005312  012701  007744                  MOV     #ENDPRG,R1      ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
2660                                                          ;FOR 0 TO 4K MEMORY IN R1
2661  005316  012703  000376          TYPSTK: MOV     #376,R3
2662  005322  005216                  INC     (SP)            ;IF WE HAVE GONE THRU THE ENTIRE
2663  005324  020137  000310                  CMP     R1,@#ENDSTK     ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
2664  005330  103043                  BHIS    $EOP            ;THEN GO TO TYPE END OF PASS
2665  005332  112702  000022                  MOVB    #18.,R2
2666  005336  105302                  RETSTK: DECB    R2      ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED.
2667  005340  002766                  BLT     TYPSTK          ;BEEN CHECKED FOR ERROR THEN SEE IF THERE
2668                                                          ;IS ANY MORE 4K MEMORY BANK
2669  005342  105721                  TSTB    (R1)+           ;OTHERWISE CHECK THE BYTE STORED AT (R1)
2670  005344  001774                  BEQ     RETSTK          ;IF IT IS 0 WE WILL NOT TYPE IT
2671  005346  020227  000020                  CMP     R2,#16.         ;IS THE POINTER POINTING TO ERROR STACK BYTE
2672                                                          ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
2673                                                          ;THE SPECIFIC MEMORY BANK
2674  005352  103404                  BLO     2$              ;IF NOT THEN GO TO TYPE BIT NUMBER
2675  005354  101026                  BHI     PARFL           ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
2676                                                          ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
2677  005356  004767  001012                  JSR     PC,TPADER       ;OTHERWISE TYPE "ADDRESS ERROR"
2678  005362  000404                  BR      FAILNM
2679  005364  010237  000312          2$:     MOV     R2,@#DECWRD     ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
2680                                                          ;IN DECIMAL
2681  005370  004767  001214                  JSR     PC,TYPDEC       ;GO TO TYPE THE BIT NUMBER IN DECIMAL
2682  005374  011637  000312          FAILNM: MOV     (SP),@#DECWRD   ;PREPARE TO TYPE THE PAGE NUMBER
2683  005400  004767  001210                  JSR     PC,$TPDEC       ;IN DECIMAL
2684  005404  005043                  CLR     -(R3)
2685  005406  114113                  MOVB    -(R1),(R3)      ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
2686                                                          ;FAILURE OCCURED
2687  005410  105021                  CLRB    (R1)+           ;CLEAR THE ERROR STACK
2688  005412  005043                  CLR     -(R3)
2689  005414  105237  000314                  INCB    @#TYPCNT        ;ENABLE THE TYPE OUT OF 1 WORDS
2690  005420  004767  001330                  JSR     PC,RPTOCT       ; TYPE THE 4K BANK AND THE NUMBER OF TIMES
2691                                                          ;THIS FAILURE WAS SEEN
2692  005424  012703  000376                  MOV     #376,R3         ;RESET SCRATCH STACK FOR EACH BIT PRINTED.
2693  005430  000742                  BR      RETSTK
2694  005432  004767  000762          PARFL:  JSR     PC,TPPRER       ; TYPE "PAR ERR"
2695  005436  000756                  BR      FAILNM
```

```
2696
2697
2698                                    ;* END OF PASS
2699                                    ;* ----------
2700                                    ;*
2701                                    ;*      TYPE "END PASS" AND DISABLE PARITY.
2702                                    ;*      ALSO SERVICE ACT11.
2703                                    ;*      AND EVERY CONSECUTIVE PASSES UNLESS BIT 4 OF $SWREG IS HIGH
2704                                    ;*
2705
2706
2707  005440  005002                SEOP:   CLR     R2              ;SET R2= PARITY MODULE DISABLE CODE
2708  005442  004767  002036               JSR     PC,PARITY       ;GO DISABLE PARITY MODULES IF SELECTED.
2709  005446  105737  000315               TSTB    @#SAVKBB        ;CONTROL-C TYPED?
2710  005452  001046                       BNE     CTLC            ;BRANCH IF YES-RESTORE LOADERS AND HALT-
2711  005454  005237  000406               INC     @#$PASS         ;INCREMENT PASS COUNT
2712  005460  032777  000040  172762       BIT     #40,@SWR        ;"END PASS #XX" PRINTOUT WANTED?
2713  005466  001015                       BNE     ACT11           ;BRANCH IF NO
2714  005470  004767  001064          TYPEOP: JSR     PC,TPCRLF       ; TYPE CR, LF, AND "END PASS #"
2715  005474  047105  020104  040520       .ASCIZ  /END PASS #/
2716  005502  051523  021440     000
2717          005510                       .EVEN
2718  005510  013737  000406  000312       MOV     @#$PASS,@#DECWRD        ;GET PASS COUNT
2719  005516  004767  001072               JSR     PC,$TPDEC               ;TYPE IT
2720  005522  013700  000042          ACT11: MOV     @#42,R0         ;GET THE MONITOR ADDRESS
2721  005526  001405                       BEQ     $DOAGN          ;IF NONE
2722  005530  004767  000012               JSR     PC,RLODER       ;RESTORE XXDP MONITOR
2723  005534  000005                       RESET                   ;RETURN TO ACT11 MONITOR.
2724
2725
2726                                    ;* SERVICE XXDP/ACT11
2727  005536  000137  000156               JMP     @#$ENDAD        ;JUMP TO ACT SERVICE
2728
2729  005542  000137  000250          $DOAGN: JMP   @#RESTRT        ;REPEAT TEST IF NOT UNDER ACT11/XXDP
2730
2731  005546  004767  001632          RLODER: JSR   PC,CLRMM        ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
2732  005552  013704  000344               MOV     @#SAVR4,R4      ;RESTORE R4 WITH SAVR4
2733  005556  014445                  4$:  MOV     -(R4),-(R5)     ;RESTORE LOADERS
2734  005560  020437  000310               CMP     R4,@#ENDSTK
2735  005564  101374                       BHI     4$
2736  005566  000207                       RTS     PC              ;RETURN FROM RLODER CALL
2737
2738                                    ;CONTROL C HANDLER
2739
2740  005570  004767  177752          CTLC:   JSR   PC,RLODER       ;RESTORE ABS LOADER
2741  005574  000167  000402               JMP     APTHLT          ;IF NOT APT HALT AT FATHLT
2742
2743
```

```
                                                ;* ERROR HANDLING ROUTINE
                                                ;* ---------------------
                                                ;*
                                                ;*      PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
                                                ;*      ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
005600  017637  000000  000402  ERROR:  MOV     @(SP),@#$FATAL  ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
005606  010346            1$:   MOV     R3,-(SP)        ;SAVE R3
005610  010046                  MOV     R0,-(SP)        ;AND R0 ON THE STACK

                              ;SETUP BANK NO. IN FATAL FOR APT

005612  010103                  MOV     R1,R3           ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
005614  004767  001604          JSR     PC,GETBNK       ;GET BANK NO. UNDER TEST INTO PBNK
005620  013703  000312          MOV     @#PBNK,R3       ;GET BANK UNDER TEST
005624  110337  000403          MOVB    R3,@#$FATAL+1   ;STORE FAILING BANK NO. FOR APT

                              ;
005630  010346                  MOV     R3,-(SP)        ;TEMPORARILY STORE R3
005632  012703  000376          MOV     #376,R3         ;MAKE R3 AS THE STACK POINTER
005636  013743  000306          MOV     @#PASFLG,-(R3)  ;OUTPUT THE WORD STORED AT
005642  005043            2$:   CLR     -(R3)
005644  113713  000402          MOVB    @#$FATAL,(R3)   ;PUT ERROR NO. ON ERROR STACK
005650  016643  000006          MOV     6(SP),-(R3)     ;PLACE THE RETURN PC AT (R3)
005654  011143                  MOV     (R1),-(R3)      ;PLACE BAD DATA,
005656  010043                  MOV     R0,-(R3)        ;AND GOOD DATA ON THE STACK
005660  005043                  CLR     -(R3)
005662  016313  000004          MOV     4(R3),(R3)      ;TAKE THE
005666  040013                  BIC     R0,(R3)         ;EXCLUSIVE OR OF GOOD AND BAD DATA
005670  046300  000004          BIC     4(R3),R0        ;TO FIND THE BITS THAT FAILED
005674  050013                  BIS     R0,(R3)         ;AND PLACE IT ON THE STACK
005676  012700  002016          MOV     #ENDPRG-.-24.,R0;THIS CODE BRINGS THE RELATIVE ADDRESS
005702  060700                  ADD     PC,R0           ;OF THE STARTING OF THE ERROR STACK
005704  062700  000022    6$:   ADD     #18.,R0         ;FOR THE SPECIFIC 4K BANK
005710  005316                  DEC     (SP)
005712  002374                  BGE     6$
005714  005726                  TST     (SP)+           ;RESTORE THE STACK POINTER

005716  105037  000277  ERRTYP: CLRB    @#TYPENB        ;DISABLE ANY TYPE OUT
005722  105737  000300    1$:   TSTB    @#$PRERR        ;IF THIS IS PARITY PROBLEM
005726  001007                  BNE     3$              ;THEN GO TO 3$
005730  105720                  TSTB    (R0)+           ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
005732  105737  000301          TSTB    @#$ADERR        ;IF THIS IS ADDRESSING PROBLEM
005736  001003                  BNE     3$              ;THEN GO TO 3$
005740  105720                  TSTB    (R0)+           ;INCREMENT THE POINTER R0 BY 1
005742  005713            2$:   TST     (R3)            ;IS BIT 15 OF (R3) SET?
005744  100015                  BPL     4$              ;IF NOT THEN GO TO 4$
005746  122710  000377    3$:   CMPB    #377,(R0)       ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
005752  001401                  BEQ     5$              ;IF SO DON'T BUMP ERROR COUNT
005754  105210                  INCB    (R0)            ;INCREMENT THE ERROR COUNTER BY 1
005756  122710  000001    5$:   CMPB    #1,(R0)         ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
005762  001404                  BEQ     7$              ;BRANCH IF NO
005764  032777  000400  172456  BIT     #400,@SWR       ;STOP ERROR PRINTOUT AFTER 1 WANTED?
005772  001002                  BNE     4$              ;BRANCH IF YES (DON'T TYPE ERROR)
```

```
2800  005774  105237  000277          7$:   INCB   @#TYPENB      ;ENABLE THE TYPE OUT ROUTINE
2801  006000  105737  000300          4$:   TSTB   @#$PRERR      ;PARITY ERROR?
2802  006004  001411                        BEQ    6$            ;BRANCH IF NO
2803  006006  004767  000406                JSR    PC,TPPRER     ;ELSE TYPE "PAR ERR"
2804  006012  000411                        BR     8$            ;AND DON'T TEST INDIVIDUAL BIT FAILURES.
2805  006014  105737  000301                TSTB   @#$ADERR      ;ADDRESS ERROR?
2806  006020  001403                        BEQ    6$            ;BRANCH IF NO
2807  006022  004767  000346                JSR    PC,TPADERR    ;PRINT "ADR ERR"
2808  006026  000403                        BR     8$
2809  006030  105720                6$:     TSTB   (R0)+         ;POINT TO NEXT ENTRY IN ERROR STACK
2810  006032  006313                        ASL    (R3)          ;IS THERE STILL AN ERROR BIT SET IN ERROR.
2811  006034  001342                        BNE    2$            ;BR IF YES - KEEP FILLING ERROR STACK
2812  006036  112737  000006  000314  8$:   MOVB   #6,@#TYPCNT   ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
2813                                                              ;THE STACK POINTED BY R3
2814  006044  004767  001150                JSR    PC,PUTADR     ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
2815                                                              ;AT LOCATIONS (R3) AND (R3-2)
2816  006050  004767  000622                JSR    PC,TYPERR     ;TYPE ERROR STACK (7 WORDS)
2817
2818  006054  005037  000300          10$:  CLR    @#$PRERR      ;CLEAR ADDRESS/PARITY ERROR FLAGS
2819  006060  012600                        MOV    (SP)+,R0      ;RESTORE R0
2820  006062  012603                        MOV    (SP)+,R3      ;AND R3
2821  006064  105737  000420          FNDERR: TSTB  @#$ENV        ;ARE WE RUNNING UNDER APT?
2822  006070  001404                        BEQ    2$            ;IF NOT THEN TEST FOR HALT
2823  006072  012737  000001  000400        MOV    #1,@#$MSGTY   ;OTHERWISE INFORM THE APT
2824  006100  000443                        BR     FATHLT        ;GOTO FATHLT AND WAIT FOR APT.
2825
2826  006102  010246                2$:     MOV    R2,-(SP)      ;SAVE R2 TEMP
2827  006104  005777  172340                TST    @SWR          ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
2828                                                              ;ON ERROR
2829  006110  100405                        BMI    4$            ;IF SO THEN HALT ON ERROR
2830                                ;CHECK FOR CONTROL-C KEY
2831
2832  006112  004767  001546                JSR    PC,CHECKC     ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
2833                                                              ;AND HALT AT FATHLT.
2834  006116  105737  000042          7$:   TSTB   @#42          ;ARE WE RUNNING UNDER ACT?
2835  006122  001401                        BEQ    6$            ;BRANCH IF NO
2836
2837  006124  000000                4$:     HALT                 ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2838                                                              ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2839                                                              ;THE WORD STORED IN R0
2840  006126  012602                6$:     MOV    (SP)+,R2      ;RESTORE R2
2841  006130  062716  000002                ADD    #2,(SP)       ;RESTORE THE RETURN ADDRESS
2842  006134  000207                        RTS    PC            ;RETURN FROM THE SUBROUTINE
2843
2844
2845
2846  006136                        FATERR:
2847  006136  004767  000416        SEQERR: JSR    PC,TPCRLF     ;TYPE "ERROR #"
2848  006142  051105  047522  020122        .ASCIZ  /ERROR #/
2849  006150  000043
2850                                        .EVEN
2851
2852  006152  017637  000000  000402        MOV    @(SP),@#$FATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2853  006160  105237  000314                INCB   @#TYPCNT      ;TELL $TPNUM TO TYPE 1 WORD
2854  006164  012703  000376                MOV    #376,R3       ;$TPNUM USES R3 AS STACK
2855  006170  013743  000402                MOV    @#$FATAL,-(R3) ;PUT ERROR NO. ON STACK
```

```
006174  005743                  TST     -(R3)           ;$TPNUM REQUIRES THIS
006176  004767  000562          JSR     PC,FATYP        ;TYPE ERROR NO.
006202  105737  000420  APTHLT: TSTB    @#$ENV  ;RUNNING UNDER APT?
006206  001326                  BNE     FNDERR          ;BRANCH IF YES
006210  000000          FATHLT: HALT                    ;FATAL ERROR OR ↑C HALT.
006212  000137  000250          JMP     @#RESTRT                ;RESTART TST BUT DON'T CLEAR PASS COUNT
                                                        ;IN CASE ↑C RESTART.


                        ;PARERR
                        ;       PARITY TRAP HANDLER
                        ;COME HERE FROM A TRAP TO 114.
                        ;       THIS ROUTINE SEARCHES THE AVAILABLE PARITY MODULES AND IF ONE
                        ;HAS A PARITY ERROR BIT SET THE GET THE PARITY ERROR ADDRESS
                        ;AND CALL THE "ERROR" ROUTINE TO PRINT ERROR MESSAGE.
                        ;IF NO PARITY ERROR BITS CAN BE FOUND A FATAL ERROR IS DONE.
                        ;
                        ;REGISTER US AGE.
                        ;R0= HOLDS PARITY MODULE ADDRESSES
                        ;R1= GETS ERROR ADDRESS FOR "ERROR" CALL.

006216  012637  000356  PARERR: MOV     (SP)+,@#PARSP   ;SET PARSP TO RETURN ADDRESS
006222  011637  000360          MOV     (SP),@#PARPS    ;SAVE PSW FOR RETURN
006226  013706  000350          MOV     @#SAVR6,SP      ;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
                                                        ;TO COMPLETE THE ERROR SERVICE ROUTINE.
006232  010067  000132          MOV     R0,SAVR0        ;SAVE R0 DURING PARITY SERVICE
006236  010167  000130          MOV     R1,SAVR1        ;SAVE R1 DURING PARITY SERVICE
006242  013701  000352          MOV     @#PARMAP,R1     ;GET PARITY AVAILABLE MAP
006246  012700  172100          MOV     #172100,R0      ;R0= FIRST PARITY ADDRESS.

006252  005701                  TST     R1              ;ANY PARITY MODULES AVAILABLE?
006254  001442                  BEQ     4$              ;BR IF NO -FATAL ERROR-
006256  000241                  CLC
006260  006001          1$:     ROR     R1              ;SHIFT PARITY MAP BIT INTO C BIT.
006262  103005                  BCC     2$              ;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
006264  005710                  TST     (R0)            ;PARITY MODULE ERROR BIT SET?
006266  100406                  BMI     3$              ;BRANCH IF YES -CALL "ERROR" ROUTINE
006270  020027  172136          CMP     R0,#172136      ;DONE ALL PARITY MODULES?
006274  002032                  BGE     4$              ;BR IF YES- GO TO FATAL ERROR CALL-
006276  062700  000002  2$:     ADD     #2,R0   ;POINT TO NEXT PARITY ADDRESS
006302  000766                  BR      1$              ;AND KEEP TRYING
006304  042710  100000  3$:     BIC     #100000,(R0)    ;CLEAR PARITY ERROR BIT.
006310  011001                  MOV     (R0),R1         ;GET PARITY MODULE CSR
006312  006101                  ROL     R1              ;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
006314  006101                  ROL     R1
006316  006101                  ROL     R1
006320  006101                  ROL     R1
006322  042701  000777          BIC     #777,R1 ;SAVE ERROR ADDRESS ONLY
006326  105237  000300          INCB    @#$PRERR        ;TELL "ERROR" PARITY ERROR CALL.
006332  004767  177242          JSR     PC,ERROR        ;#ERROR# REPORT ERROR MESSAGE
006336  000050                  50                      ;******ERROR NUMBER 50******

006340  016700  000024          MOV     SAVR0,R0        ;RESTORE R0
006344  016701  000022          MOV     SAVR1,R1        ;RESTORE R1
006350  013746  000360          MOV     @#PARPS,-(SP)   ;SET RETURN PSW ON STACK
006354  013746  000356          MOV     @#PARSP,-(SP)   ;AND SET RETURN ADDRESS ON STACK
```

```
                                ERROR HANDLING ROUTINE

2912  006360  000002                      RTI                     ;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
2913
2914                                  ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
2915  006362                          4$:
2916  006362  004767  177550              JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
2917  006366  000051                      51                      ;******ERROR NUMBER 51******
2918
2919
2920                                  ;R0+R1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
2921                                  ;STACK SPACE BETWEEN 500-450.
2922  006370  000000                  SAVR0:  0                   ;SAVE R0 DURING PARITY TRAP SERVICE
2923  006372  000000                  SAVR1:  0       ;SAVE R1 DURING PARITY TRAP SERVICE
2924
2925
2926  006374  105737  000277          TPADER: TSTB    @#TYPENB        ;TYPE ERROR?
2927  006400  001406                          BEQ     1$              ;BRANCH IF NO
2928  006402  004767  000160                  JSR     PC,PNTMES       ; TYPE CR, LF AND "ADR ER"
2929  006406  042101  020122  051105          .ASCIZ  /ADR ERR/
2930  006414  000122
2931                                          .EVEN
2932  006416  000207                  1$:     RTS     PC
2933
2934  006420  105737  000277          TPPRER: TSTB    @#TYPENB        ;ERROR PRINTOUTS ALLOWED?
2935  006424  001406                          BEQ     1$              ;BRANCH IF NO
2936  006426  004767  000134                  JSR     PC,PNTMES       ;GO TO TYPE CR, LF AND "PAR ERR"
2937  006432  040520  020122  051105          .ASCIZ  /PAR ERR/
2938  006440  000122
2939                                          .EVEN
2940  006442  000207                  1$:     RTS     PC
```

```
                                        ;* TYPE OUT ROUTINE
                                        ;* ---------------
                                        ;*
                                        ;*      THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
                                        ;*
006444 010146                 NOTYP:    MOV     R1,-(SP)
006446 016601 000002                    MOV     2(SP),R1
006452 105721                 4$:       TSTB    (R1)+           ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
006454 001376                           BNE     4$              ;PREPARE TO RETURN
006456 000413                           BR      RETTYP
006460 010146                 $TYPE:    MOV     R1,-(SP)        ;SAVE R1
006462 010046                           MOV     R0,-(SP)        ;AND R0 ON THE STACK
006464 016601 000004                    MOV     4(SP),R1        ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
006470 112100                 2$:       MOVB    (R1)+,R0        ;PLACE THE BYTE TO BE TYPED IN R0
006472 001403                           BEQ     4$              ;IF IT IS END OF MESSAGE THEN GO TO 4$
006474 004767 000022                    JSR     PC,$TPCHR       ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
006500 000773                           BR      2$
006502 012600                 4$:       MOV     (SP)+,R0        ;RESTORE R0
006504 005201                 RETTYP:   INC     R1              ;CAUSE R1 TO
006506 042701 000001                    BIC     #1,R1           ;POINT TO EVEN ADDRESS
006512 010166 000002                    MOV     R1,2(SP)        ;MODIFY THE RETURN ADDRESS
006516 012601                           MOV     (SP)+,R1        ;RESTORE R1
006520 000416                           BR      EXTYP           ;AND RETURN VIA RTS PC

006522 132737 000040 000421   $TPCHR:   BITB    #40,@#SENVM     ;HAVE TYPE OUTS BEEN DISABLED?
006530 001005                           BNE     4$              ;IF SO THEN RETURN FROM THE SUBROUTINE
006532 105737 177564          2$:       TSTB    @#STPS          ;WAIT HERE
006536 100375                           BPL     2$              ;UNTIL THE PRINTER IS READY
006540 110037 177566                    MOVB    R0,@#STPB       ;LOAD DATA TO BE TYPED INTO DATA REG.
006544 000404                 4$:       BR      EXTYP           ;RETURN

006546 004767 177706          PCRLF:    JSR     PC,$TYPE
006552 005015 000                       .ASCIZ  <15><12>        ;CR/LF
006556                                  .EVEN
006556 000207                 EXTYP:    RTS     PC      ;RETURN

006560 004767 177762          TPCRLF:   JSR     PC,PCRLF        ;TYPE CR/LF
006564 000735                           BR      $TYPE           ;NOW GO TO TYPE THE REST OF THE MESSAGE

006566 032777 000020 171654   PNTMES:   BIT     #20,@SWR        ;PRINTOUTS ALLOWED?
006574 001323                           BNE     NOTYP           ;BRANCH IF NO
006576 123737 000042 000046             CMPB    @#42,@#46       ;RUNNING UNDER ACT 11?
006604 001717                           BEQ     NOTYP           ;BRANCH IF YES -NOT PRINTOUT-
006606 000764                           BR      TPCRLF          ;SEND CR/LF AND TYPE MESSAGE.
```

```
                                        ;* ROUTINE TO TYPE OUT A DECIMAL NUMBER
                                        ;* ----------------------------------
                                        ;*
                                        ;*      THIS ROUTINE IS USED TO CONVERT THE CONTENTS OF LOCATION
                                        ;*      DECWRD TO DECIMAL NUMBERS AND TYPE THEN FOLLOWING 3 SPACES
                                        ;*
006610  004767  177732          TYPDEC: JSR     PC,PCRLF        ;TYPE CR/LF

006614  005046                  $TPDEC: CLR     -(SP)
006616  013746  000312                  MOV     @#DECWRD,-(SP)  ;GET THE WORD THAT HAS TO BE CONVERTED TO A
                                                                ;DECIMAL NUMBER
006622  162716  000012          2$:     SUB     #10.,(SP)
006626  002403                          BLT     4$              ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
                                                                ;GO TO 4$
006630  005266  000002                  INC     2(SP)           ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
006634  000772                          BR      2$              ;AND RETURN TO 2$
006636  062716  000012          4$:     ADD     #10.,(SP)
006642  052716  000060                  BIS     #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
006646  112667  000020                  MOVB    (SP)+,6$-2      ;PLACE THE 1'S DIGIT TO BE TYPED
006652  052716  000060                  BIS     #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
006656  112667  000007                  MOVB    (SP)+,6$-3      ;PLACE THE 10'S DIGIT TO BE TYPED
006662  004767  177572                  JSR     PC,$TYPE        ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
                                                                ;3 SPACES
006666  020040  030040  000060          .ASCIZ  /   00/
                                        .EVEN
006674  000207                  6$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
```

```
3017
3018
3019                                    ;* OCTAL TYPE OUT ROUTINE
3020                                    ;* ----------------------
3021                                    ;*
3022                                    ;*      THIS ROUTINE IS USED TO TYPE OUT THE OCTAL VALUES
3023                                    ;*      CONTROL SHOULD COME TO THIS ROUTINE WITH R3 POINTING TO
3024                                    ;*      THE LOW ORDER BITS (I.E. BITS 0-15) OF THE ADDRESS TO
3025                                    ;*      BE TYPED WHERE AS R3-2 SHOULD CONTAIN THE HIGH ORDER BITS
3026                                    ;*      (I.E. BITS 16 & 17). CONTENTS OF LOCATION R3-1 AND R0 ARE
3027                                    ;*      DESTROYED BY THIS SUBROUTINE
3028                                    ;*      BYTE TYPCNT SHOULD BE SET TO THE NUMBER OF WORDS THAT HAVE
3029                                    ;*      TO BE TYPED.
3030                                    ;*
3031
3032  006676  032777  020000  171544  TYPERR: BIT     #20000,@SWR     ;ERROR PRINTOUT WANTED?
3033  006704  001054                  BNE     OCTXT           ;BRANCH IF NO
3034  006706  004767  177634          JSR     PC,PCRLF        ;TYPE CR/LF
3035  006712  004767  000012          JSR     PC,TYPOCT       ;TYPE OCTAL NO.
3036  006716  000447                  BR      OCTXT           ;RETURN VIA RTS PC
3037  006720  012123          OCTTYP: MOV     (R1)+,(R3)+     ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
3038                                                          ;BY R3
3039  006722  012113                  MOV     (R1)+,(R3)      ;AND NOW PLACE THE LOW ORDER BITS
3040  006724  105237  000314          INCB    @#TYPCNT        ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3041  006730  052743  000004  TYPOCT: BIS     #4,-(R3)
3042  006734  106113          2$:     ROLB    (R3)
3043  006736  103376                  BCC     2$
3044  006740  005000                  CLR     R0
3045  006742  106113                  ROLB    (R3)            ;GET BITS 17 & 16 INTO R0
3046  006744  006100                  ROL     R0
3047  006746  106113                  ROLB    (R3)
3048  006750  006100                  ROL     R0
3049  006752  000405                  BR      $TPNUM
3050  006754  004767  177500  RPTOCT: JSR     PC,$TYPE        ; TYPE 3 SPACES
3051  006760  020040  000040          .ASCIZ  /   /
3052                                  .EVEN
3053  006764  005000          FATYP:  CLR     R0
3054  006766  012723  000006  $TPNUM: MOV     #6,(R3)+        ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3055  006772  000241          4$:     CLC
3056  006774  006113                  ROL     (R3)
3057  006776  006100                  ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3058  007000  052700  000060          BIS     #60,R0          ;OR THE CONTENTS OF R0 WITH AN ASCII 0
3059  007004  004767  177512          JSR     PC,$TPCHR       ; TYPE THE OCTAL NUMBER STORED IN R0
3060  007010  005000                  CLR     R0
3061  007012  006113                  ROL     (R3)
3062  007014  006100                  ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3063  007016  006113                  ROL     (R3)
3064  007020  006100                  ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3065  007022  105363  177776          DECB    -2(R3)          ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3066  007026  001361                  BNE     4$              ;THEN REPEAT FROM 4$
3067  007030  105337  000314          DECB    @#TYPCNT        ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
3068  007034  001347                  BNE     RPTOCT          ;TYPED THEN REPEAT FROM RPTOCT
3069  007036  000207          OCTXT:  RTS     PC
```

```
                                        ;* ROUTINE TO SET UP MEMORY MANAGEMENT REGISTERS
                                        ;* ----------------------------------------------
                                        ;*
                                        ;*      PROGRAM CONTROL COMES HERE TO DETERMINE IF THE MEMORY MANAGEMENT
                                        ;*      IS AVAILABLE OR NOT, AND IF IT IS AVAILABLE THEN WHETHER
                                        ;*      THE MEMORY ABOVE 28K IS REQUIRED TO BE TESTED OR NOT.
                                        ;*
007040  012702  001400          MEMMNG: MOV     #1400,R2
007044  105037  000276          MMREG:  CLRB    @#MMAVA         ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
                                                                ;THAT MEM. MANAG. IS AVAILABLE FOR TESTING
007050  032777  010000  171372          BIT     #10000,@SWR     ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
007056  001441                        BEQ     RETMM           ;IF NOT THEN RETURN FROM THE SUBROUTINE
007060  012700  000004                MOV     #4,R0           ;PREPARE TO SETUP TIME OUT VECTOR
007064  012720  007164                MOV     #NOMM,(R0)+     ;RETURN ADDRESS TO NOMM
007070  012710  000340                MOV     #340,(R0)       ;AND WITH A PSW OF 340
007074  005037  177572                CLR     @#SR0           ;TRY TO REACH MEM. MANAG. SR0
007100  105237  000276                INCB    @#MMAVA         ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
                                                                ;BYTE
007104  012701  172340                MOV     #172340,R1      ;R1 IS POINTING TO PAR0
007110  005021                        CLR     (R1)+           ;PAR0 WILL POINT TO BANK 0
007112  062702  000200          2$:     ADD     #200,R2
007116  010221                        MOV     R2,(R1)+        ;SETUP PAR1-PAR6
007120  020127  172356                CMP     R1,#172356
007124  103772                        BLO     2$
007126  012711  007600                MOV     #7600,(R1)      ;PAR7 IS POINTING TO THE I/O PAGE
007132  012701  172300                MOV     #172300,R1
007136  012721  077406          4$:     MOV     #77406,(R1)+    ;SETUP PDR0-PDR7
007142  020127  172316                CMP     R1,#172316
007146  101773                        BLOS    4$
007150  005237  177572                INC     @#SR0           ;ENABLE MEM. MANAG.
007154  005010                SRETMM: CLR     (R0)            ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
007156  012740  000104                MOV     #BUSER,-(R0)
007162  000207                RETMM:  RTS     PC

007164  022626                NOMM:   CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
007166  004767  177366                JSR     PC,TPCRLF       ;TYPE "NO MEMORY MANAGEMENT MESSAGE
007172  047516  046440  043516        .ASCIZ  /NO MMG/
007200     000
007202                              .EVEN
007202  004767  176730                JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
007206  000052                        52                      ;******ERROR NUMBER 52******

007210  000761                        BR      SRETMM          ; RESTORE TIME OUT TRAP VECTOR

007212  013702  172354          UPMM:   MOV     @#172354,R2     ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
007216  000712                        BR      MMREG
```

```
                                        ;* 18 BIT ADDRESS GENERATOR
                                        ;* ------------------------
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO PLACE THE ADDRESS STORED IN R1
                                        ;*      IN THE LOCATION POINTED BY R3. THE ADDRESS IN R1 IS CONVERTED
                                        ;*      TO AN 18 BIT ADDRESS ONLY IF MEM. MANAG. IS AVILABLE IN WHICH
                                        ;*      CASE THE HIGH ORDER BITS OF THE ADDRESS ARE PLACED IN LOCATION
                                        ;*      POINTED BY R3-2
                                        ;*

007220  005063  177776          PUTADR: CLR     -2(R3)
007224  010113                          MOV     R1,(R3)         ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
007226  105737  000276                  TSTB    @#MMAVA         ;IS THE MEM. MANAG. AVAILABLE ?
007232  001425                          BEQ     6$              ;IF NOT THEN RETURN FROM THE SUBROUTINE
007234  010146                          MOV     R1,-(SP)        ;SAVE R1
007236  042701  017777                  BIC     #17777,R1       ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
007242  040113                          BIC     R1,(R3)         ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
007244  052701  004000                  BIS     #4000,R1        ;PREPARE TO SHIFT R1 BY 12 PLACES
007250  006001                  2$:     ROR     R1
007252  103376                          BCC     2$              ;GET THE NUMBER OF PAR IN R1
007254  062701  172340                  ADD     #172340,R1      ;GET THE ADDRESS OF PAR IN R1
007260  011101                          MOV     (R1),R1         ;LOAD R1 WITH THE CONTENTS OF PAR
007262  052701  010000                  BIS     #10000,R1
007266  006101                  4$:     ROL     R1
007270  103376                          BCC     4$              ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
007272  006101                          ROL     R1
007274  006143                          ROL     -(R3)           ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
007276  006101                          ROL     R1
007300  006123                          ROL     (R3)+           ;PLACE BIT 16 OF THE ADDRESS
007302  050113                          BIS     R1,(R3)         ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
007304  012601                          MOV     (SP)+,R1        ;RESTORE R1
007306  000207                  6$:     RTS     PC              ;RETURN FROM THE SUBROUTINE

                                        ;* GET ADDRESS FROM THE APT MAILBOX
                                        ;* --------------------------------
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
                                        ;*      PLACE IT IN THE LOCATION USED BY THE PROGRAM TO  DEFINE THE
                                        ;*      MEMORY BOUNDRIES.
                                        ;*      PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
                                        ;*      ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
                                        ;*      THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
                                        ;*      TO BE PLACED
                                        ;*

007310  016143  000001          GETADR: MOV     1(R1),-(R3)     ;PLACE THE LOW ORDER BITS OF THE ADDRESS
007314  005043                          CLR     -(R3)           ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
                                                                ;HAVE TO BE PLACED
007316  116113  177777                  MOVB    -1(R1),(R3)     ;PLACE BITS 16 & 17
007322  000207                  2$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
```

```
                                        ;* CONVERT 18 BIT ADDRESS TO THE PAR FORM
                                        ;* --------------------------------------
                                        ;*
                                        ;*      THIS SUBROUTINE IS USED TO CONVERT 18 BIT ADDRESS STORED IN
                                        ;*      LOCATIONS POINTED BY R3 AND R3+2 TO THE FORM IT WILL BE STORED
                                        ;*      IN A PAR. THE RESULT IS LEFT IN R2. R1 IS LOADED WITH BITS
                                        ;*      0-12 OF THE ADDRESS AND R0 WITH 160000
                                        ;*
007324  105237  000315          $GTSIZ: INCB    @#SAVKBB        ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
                                                                ;0-4 OF R2
007330  012301                  GETSIZ: MOV     (R3)+,R1
007332  011302                          MOV     (R3),R2         ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
007334  042702  017777                  BIC     #17777,R2       ;CLEAR ADDRESS BITS 0-12
007340  052702  000040          2$:     BIS     #40,R2
007344  006001                  4$:     ROR     R1
007346  006002                          ROR     R2              ;ROTATE R1 AND R2 7 TIMES
007350  103375                          BCC     4$
007352  105737  000315                  TSTB    @#SAVKBB
007356  001405                          BEQ     6$
007360  105037  000315                  CLRB    @#SAVKBB
007364  052702  000100                  BIS     #100,R2
007370  000765                          BR      4$
007372  012301                  6$:     MOV     (R3)+,R1        ;PLACE THE LOW ORDER ADDRESS BITS IN R1
007374  012700  160000                  MOV     #160000,R0
007400  040001                          BIC     R0,R1           ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
007402  000207                          RTS     PC              ;RETURN FROM THE SUBRORNE


                                        ;* SUBROUTINE TO DISABLE MEMORY MANAGEMENT
                                        ;* --------------------------------------
                                        ;*
                                        ;*      THIS SUBROUTINE IS CALLED TO DISABLE THE MEMORY MANAGEMENT
                                        ;*      UNIT

007404  105737  000276          CLRMM:  TSTB    @#MMAVA         ;WAS THE MEMORY MANAGEMENT ENABLED ?
007410  001404                          BEQ     1$              ;IF NOT THEN GO TO 1$
007412  005037  177572                  CLR     @#SR0           ;DISABLE THE MEMORY MANAGEMENT
007416  105037  000276                  CLRB    @#MMAVA         ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
007422  000207                  1$:     RTS     PC              ;RETURN FROM THE SUBROUTINE


                                        ;* GET BANK NO. UNDER TEST
                                        ;       CALLED BY ERRTYP AND TST13 TO GET BANK NO. UNDER TEST INTO PBNK.
                                        ;REGISTERS
                                        ;R0=POINTER TO PAR UNDER TEST
                                        ;R3=VIRTUAL ADDRESS ON ENTRY
                                        ;R0+R3 ARE RESTORED ON EXIT.

007424  010046                  GETBNK: MOV     R0,-(SP)        ;SAVE R0
007426  010346                          MOV     R3,-(SP)        ;SAVE R3
007430  042703  017777                  BIC     #17777,R3       ;SAVE ONLY VIRTUAL BANK BITS
007434  052703  010000                  BIS     #10000,R3               ;SETUP R3 SHIFT BIT
```

```
                        SUBROUTINE TO DISABLE MEMORY MANAGEMENT

 3225  007440  000241                    CLC
 3226  007442  006003            1$:     ROR     R3              ;SHIFT A BANK BIT
 3227  007444  103376                    BCC     1$              ;UNTIL IN BITS <2:0> OF R3
 3228  007446  105737  000276            TSTB    @#MMAVA         ;MEMORY MANAGEMENT UNDER TEST?
 3229  007452  001407                    BEQ     2$              ;NO EXIT
 3230
 3231                                    ;GET PAR ADDRESS AND PHYSICAL BANK NO.
 3232  007454  006303                    ASL     R3              ;MAKE R3 PAR ADDRESS OFFSET.
 3233  007456  062703  172340            ADD     #172340,R3      ;MAKE FULL PAR ADDRESS.
 3234  007462  011300                    MOV     (R3),R0         ;GET PAR CONTENTS
 3235  007464  006300                    ASL     R0
 3236  007466  000300                    SWAB    R0              ;SHIFT BANK BITS TO BITS <7:0>
 3237  007470  110003                    MOVB    R0,R3           ;SET R3 TO PHYSICAL BANK NO.
 3238  007472  010337  000312    2$:     MOV     R3,@#PBNK       ;STORE PHYSICAL BANK NO.
 3239  007476  012603                    MOV     (SP)+,R3        ;RESTORE R3
 3240  007500  012600                    MOV     (SP)+,R0        ;RESTORE R0
 3241  007502  000207                    RTS     PC              ;RETURN TO CALLER
 3242
 3243
 3244
 3245                                    ; PARITY ENABLE/DISABLE ROUTINE
 3246                                    ;
 3247                                    ;       THIS ROUTINE ENABLES OR DISABLES PARITY MODULES AND PRINTS ASSOCIATED MEASSAGES.
 3248                                    ;       IF PARITY AVAILABLE THEN BIT13 OF "REL" IS SET AND "PAR"ITY IS PRINTED.
 3249                                    ;       ALSO THE BACKGROUND TEST PATTERN (LOC. BAKPAT) IS SET=376
 3250                                    ;
 3251                                    ;REGISTER USAGE.
 3252                                    ;R0= POINTS TO BUS TIMEOUT TRAP VECTOR (LOC. 4)
 3253                                    ;R1= HOLDS PARITY MODULE UNIBUS ADDRESS.
 3254                                    ;R2= ON ENTRY HOLDS ENABLE/DISABLE CODE .
 3255                                    ;       IF R2=0 THEN DISABLE
 3256                                    ;       IF R2=1 THEN ENABLE
 3257                                    ;R3= SCRATCH TO SETUP LOC. PARMAP WITH A MAP OF PARITY MODULES PRESENT.
 3258
 3259                                    ;CALL IS
 3260                                    ;       MOV     #1,R2   ;ENABLE CODE
 3261                                    ;       JSR     PC,PARITY
 3262
 3263
 3264  007504  032777  004000  170736  PARITY: BIT     #4000,@SWR      ;PARITY TEST WANTED?
 3265  007512  001460                    BEQ     6$                      ;BRANCH IF NO
 3266
 3267  007514  012700  000004                    MOV     #4,R0           ;POINT R0 TO BUS TIMEOUT ADDRESS.
 3268  007520  012710  000122                    MOV     #5$-.-6,(R0)    ;SET RETURN FROM TIMEOUT TRAP TO 5$
 3269  007524  060710                    ADD     PC,(R0)         ;IN THE CURRENT BANK.
 3270  007526  005037  000352    1$:     CLR     @#PARMAP        ;CLEAR PARITY MAP HOLDER.
 3271  007532  012701  172140            MOV     #172140,R1      ;SET R1 TO LAST PARITY MODULE ADDRESS+2
 3272  007536  012703  100000            MOV     #100000,R3      ;SET R3 TO PARMAP AVAILABLE CODE BEGIN.
 3273  007542  010241            2$:     MOV     R2,-(R1)        ;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE.
 3274  007544  050337  000352            BIS     R3,@#PARMAP     ;NO TRAP TO 5$, SO SET PARITY AVAILABLE.
 3275  007550  000241                    CLC
 3276  007552  006003            3$:     ROR     R3              ;SETUP NEXT PARMAP BIT
 3277  007554  103372                    BCC     2$              ;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
 3278  007556  012710  000104            MOV     #BUSER,(R0)     ;RESET BUS TIMEOUT TRAP VECTOR
 3279  007562  005702                    TST     R2              ;IS THIS A DISABLE CALL?
 3280  007564  001433                    BEQ     6$              ;BRANCH IF YES (EXIT)
```

```
3281  007566  005737  000352          TST     @#PARMAP        ;WERE ANY PARITY MODULES FOUND?
3282  007572  001011                  BNE     4$              ;BRANCH IF YES
3283  007574  004767  176760          JSR     PC,TPCRLF       ;PRINT "NO PAR"
3284  007600  047516  050040  051101  .ASCIZ  /NO PAR/
3285  007606     000
3286             007610                .EVEN
3287  007610  004767  176322          JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
3288  007614  000053                  53                      ;******ERROR NUMBER 53******
3289
3290
3291  007616  152737  000040  000405  4$:     BISB    #40,@#REL       ;SET PARITY UNDER TEST FLAG
3292  007624  012737  000376  000316          MOV     #376,@#BAKPAT   ;SET BACKGROUND PATTERN TO
3293                                                                  ;WORST CASE PARITY CODE.
3294  007632  004767  176722                  JSR     PC,TPCRLF       ;PRINT "TST PARITY"
3295  007636  040520  044522  054524          .ASCIZ  /PARITY/
3296  007644     000
3297             007646                .EVEN
3298  007646  000405                  BR      EXITC           ;AND EXIT VIA RTS PC
3299
3300                          ;GET HERE IF PARITY ADDRESS TIMED OUT TO LOC. 4
3301
```

```
DZKMA   MACY11 27(1006)  18-FEB-77  12:35  PAGE 71
DZKMAC.P11    18-FEB-77 12:30          SUBROUTINE TO DISABLE MEMORY MANAGEMENT

  3302  007650  022626                 5$:     CMP     (SP)+,(SP)+     ;RESET STACK FROM TRAP
  3303  007652  000737                         BR      3$              ;KEEP TRYING PARITY ADDRESES.
  3304
  3305  007654  142737  000040  000405 6$:     BICB    #40,@#REL       ;CLEAR PARITY TESTING FLAG
  3306  007662                         EXITC:
  3307  007662  000207                 7$:     RTS     PC              ;RETURN TO CALLER
  3308
  3309
  3310
  3311
  3312                                 ;CHECKC
  3313                                 ;       THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
  3314                                 ;TEST OR IN THE ERROR TYPE ROUTINE.
  3315                                 ;IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
  3316                                 ;RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
  3317                                 ;FINALLY IT HALTS AT FATHLT.
  3318
  3319  007664  105037  000315        CHECKC: CLRB    @#SAVKBB        ;INIT CONTROL-C FLAG.
  3320  007670  105737  177560                TSTB    @#TKS           ;ANY CHAR. TYPED?
  3321  007674  100372                        BPL     EXITC           ;BR IF NO-EXIT VIA RTS PC-
  3322  007676  113702  177562                MOVB    @#$KBB,R2       ;GET THE CHAR TYPED.
  3323  007702  042702  000200                BIC     #200,R2 ;CLEAR THE PARITY BIT.
  3324  007706  122702  000003                CMPB    #3,R2           ;IS IT CONTROL-C?
  3325  007712  001363                        BNE     EXITC           ;BRANCH IF NO -EXIT VIA RTS PC-
  3326  007714  110237  000315                MOVB    R2,@#SAVKBB     ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
  3327  007720  004767  176634                JSR     PC,TPCRLF       ;PRINT "↑C"
  3328  007724  041536     000                .ASCIZ  /↑C/
  3329          007730                      .EVEN
  3330  007730  000167  175104                JMP     RELOER          ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
  3331
  3332          007744                 .=7744
  3333  007744  000000                ENDPRG: 0               ;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
  3334                                                         ;STACK.FOR EACH 4K BANK 18. BYTES ARE SAVED.
  3335                                                         ;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
  3336                                                         ;AFTER THE ERROR STACK.
  3337                                                         ;FOR 4K MEMORY SIZE THEN PROGRAM=7744+22=7776
  3338          000001                         .END
```

```
ABASE = 000000          1225
ACDW1 = 000000          1225
ACDW2 = 000000          1225
ACPUOP= 000000          1225    1240
ACT11   005522          2713    2720#
ADDW0 = 000000          1225
ADDW1 = 000000          1225
ADDW10= 000000          1225
ADDW11= 000000          1225
ADDW12= 000000          1225
ADDW13= 000000          1225
ADDW14= 000000          1225
ADDW15= 000000          1225
ADDW2 = 000000          1225
ADDW3 = 000000          1225
ADDW4 = 000000          1225
ADDW5 = 000000          1225
ADDW6 = 000000          1225
ADDW7 = 000000          1225
ADDW8 = 000000          1225
ADDW9 = 000000          1225
ADEVCT= 000000          1225    1231
ADEVM = 000000          1225
AENV  = 000000          1225    1236
AENVM = 000000          1225    1237
AFATAL= 000000          1225    1228
AMADR1= 000000          1225    1253
AMADR2= 000000          1225    1257
AMADR3= 000000          1225    1260
AMADR4= 000000          1225    1263
AMAMS1= 000000          1225    1247
AMAMS2= 000000          1225    1255
AMAMS3= 000000          1225    1258
AMAMS4= 000000          1225    1261
AMSGAD= 000000          1225    1233
AMSGLG= 000000          1225    1234
AMSGTY= 000000          1225    1227
AMTYP1= 000000          1225    1248
AMTYP2= 000000          1225    1256
AMTYP3= 000000          1225    1259
AMTYP4= 000000          1225    1262
APASS = 000000          1225    1230
APRIOR= 000000          1225
APTHLT  006202          2741    2858#
APTSIZ  000724          1325    1339#
ASWREG= 000000          1225    1238
ATESTM= 000000          1225    1229
AUNIT = 000000          1225    1232
AUSWR = 000000          1225    1239
AVECT1= 000000          1225
AVECT2= 000000          1225
BAKPAT  000316          1050    1185#   1293#   1503    1726    1728    1736    1747#   1811    1843#   2025#   2061    2096#
                        2147    2153    2157    2170    2173    2212#   2309    2365#   2366    2548#   3292#
BEGIN   000500          1058    1188    1209    1273#   2567    2578    2584
BRTPSZ  001012          1357#
BUSER   000104          1030#   1399    3103    3278
```

CROSS REFERENCE TABLE -- USER SYMBOLS

```
CHECKC  007664          1570    2832    3319#
CKDONE  005106          2553    2576    2586#
CLRMEM  001466          1497#   2618
CLRMM   007404          1431    1453    2731    3207#
CNTSCP  001644          1569    1572#
CONT    001526          1514#   1583
CONTMM  005134          2550    2597#
CTLC    005570          2710    2740#
DECWRD  000312          1170#   1171    2679*   2682*   2718*   3000
ENDPAS  005276          2595    2655#
ENDPRG  007744          1096    1291    2659    2777    3333#
ENDSTK  000310          1167#   1168    1418*   2663    2734
END0    002132          1668#   1710
END1    002232          1710#   1750
END10   003772          2217    2225#   2368
END12   004316          2368#   2478    2510
END2    002342          1750#   1846
END3    002610          1846#   1885
END4    002720          1885#   1971
END5    003066          1971#   2027
END6    003222          2027#   2104
END7    003444          2104#   2225
ERROR   005600          1627    1688    1742    1807    1871    1911    1918    1945    2003    2078    2158    2174    2183
                        2328    2336    2463    2475    2492    2751#   2905
ERRTYP  005716          2784#
EXITC   007662          3298    3306#   3321    3325
EXTYP   006556          2966    2973    2978#
FAILMM  005374          2678    2682#   2695
FATERR  006136          1304    1352    1464    1486    1654    2846#   2916    3111    3287
FATHLT  006210          2824    2860#
FATYP   006764          2857    3053#
FNDERR  006064          2821#   2859
GALLOP  003472          2141#   2275
GETADR  007310          1355    1356    3164#
GETBNK  007424          2535    2758    3221#
GETSIZ  007330          2599    2610    2615    3182#
HIGHAD  000332          1196#   1341
HIGHTW  000330          1195#   1340    1409
LOOP    001536          1517#   1581
LOWADD  000326          1193#
LOWBNK  000304          1160#   1161    2048*   2059    2094
LOWER   005112          2586    2590#
LOWTWO  000324          1192#   1402    1455    2598
M       =  000200       1005#   2492
MAXADR  005224          2613    2616    2627#
MAXMEM  000340          1202#   1339    1422*   1644    1840    2441    2564    2580
MEMMNG  007040          1375    2592    3079#
MEMTST  001460          1491#
MMAVA   000276          1142#   1145    1377    2549    2594    3080*   3088*   3131    3207    3210*   3228
MMREG   007044          2609    3080#   3117
N       =  000054       1005#   1304    1307#   1352    1355#   1464    1467#   1486    1489#   1598    1601#   1612    1614#
                        1615    1618#   1623    1625#   1654    1657#   1680    1683#   1688    1691#   1723    1726#   1742
                        1745#   1767    1770#   1792    1794#   1802    1804#   1818    1820#   1855    1858#   1871    1874#
                        1904    1907#   1911    1914#   1918    1921#   1945    1948#   1993    1996#   2003    2006#   2044
                        2047#   2078    2081#   2135    2138#   2158    2161#   2174    2177#   2183    2186#   2260    2263#
                        2300    2303#   2328    2331#   2336    2339#   2401    2404#   2463    2466#   2475    2478#   2492
```

```
                2495#  2905   2908#  2915   2919#  3111   3114#  3287   3290#
NOMM    007164  3085   3106#
NOTYP   006444  2949#  2985   2987
OCTTYP  006720  1404   1408   3037#
OCTXT   007036  3033   3036   3069#
ONEPAS  000556  1285   1288#
PARERR  006216  1281   2877#
PARFL   005432  2675   2694#
PARITY  007504  1498   2708   3264#
PARMAP  000352  1210#  2883   3270*  3274*  3281
PARPS   000360  1213#  2878*  2910
PARSP   000356  1212#  2877*  2911
PASFLG  000306  1162#  1166   1514*  1881*  1949*  1966*  1998*  2009   2011*  2024*  2047*  2069   2085*
                2087   2092   2306*  2323   2343   2349*  2350   2358*  2359   2446   2453   2468   2486
                2503   2513*  2524*  2541*  2766
PBNK    000312  1169#  2759   3238*
PCRLF   006546  1403   2975#  2980   2997   3034
PNTHES  006566  1058   2267   2531   2561   2928   2936   2984#
PUTADR  007220  1391   1394   1520   2814   3129#
PWRDN   000070  1023#  1277
PWRUP   000136  1023   1057#
REL     =000405 1137#  1284   1477   2427   2554   2556*  2575   2582*  3291*  3305*
RELBOT  000322  1188#  1479
RELOC   004746  1550   2548#
RELOER  005040  2555   2574#  3330
RESTRT  000250  1072   1088   1090   1096#  2729   2861
RETMM   007162  3083   3104#
RETSTK  005336  2666#  2670   2693
RETTYP  006504  2953   2962#
RLODER  005546  2722   2731#  2740
RPTOCT  006754  2690   3050#  3068
RPT10   003466  2139#  2224   2270
RPT11   004032  2220   2223   2266   2269#
RPT6    003106  1996#  2026
SAVKBB  000315  1174#  2590   2709   3179*  3189   3191*  3319*  3326*
SAVLOR  001242  1424   1426   1431#
SAVLOC  000354  1211#  2019*
SAVMAX  000342  1204#  1489
SAVR0   006370  2881*  2908   2922#
SAVR1   006372  2882*  2909   2923#
SAVR4   000344  1205#  2732
SAVR5   000346  1097   1206#  1452*  2574   2583*
SAVR6   000350  1057   1082   1209#  1276*  1631   2585*  2879
SCOPE   =000240 1006#
SEQERR  006136  1598   1680   1723   1767   1855   1904   1993   2044   2135   2260   2300   2401   2847#
SETSTK  001164  1290   1409#
SETSWR  000656  1315#
SLFSIZ  001024  1366#
SR0     =177572 1184#  3087*  3101*  3209*
START   000200  1065   1082#
STRTDI  000302  1158#  1159   2057*  2086
SWAPAT  000320  1187#  2178   2182   2211*
SWHALT  001664  1578#
SWR     000450  1270#  1318   1321*  1326*  1359   1462   1506   1576   1580   1641   2265   2269   2527
                2552   2655   2712   2798   2827   2984   3032   3082   3264
SWREG   000176  1075#  1321
```

L06

CROSS REFERENCE TABLE -- USER SYMBOLS

```
SW11  = 004000          1005#
TBL     001600          1534    1538#   1539    1540    1541    1542    1543    1544    1545    1546    1547    1548    1549
                        1550
TKS   = 177560          1180#   3320
TPADER  006374          2677    2807    2926#
TPCRLF  006560          1091    2714    2847    2980#   2988    3107    3283    3294    3327
TPPRER  006420          2694    2803    2934#
TRYSR   001014          1343    1359#
TSTGO   001666          1577    1580#
TSTMM   005120          1467    2592#
TSTREL  001320          1286    1452#
TSTRP   000770          1289    1291#
TSTSCP  001632          1568#   1668
TSTSIZ  001324          1453#
TST0    001702          1538    1596#
TST1    002134          1539    1676#
TST10   003446          1546    2132#
TST11   003774          1547    2257#
TST12   004046          1548    2298#   2367
TST13   004320          1549    2399#
TST2    002234          1540    1720#
TST3    002344          1541    1765#   1844
TST4    002612          1542    1853#
TST5    002722          1543    1901#
TST6    003070          1544    1989#
TST7    003224          1545    2041#
TYPCNT  000314          1172#   1279*   2689*   2812*   2853*   3040*   3067*
TYPDEC  006610          2681    2997#
TYPENB  000277          1146#   1149    2784*   2800*   2926    2934
TYPEOP  005470          2714#
TYPERR  006676          2816    3032#
TYPMEM  001152          1405#
TYPOCT  006730          3035    3041#
TYPSIZ  001130          1357    1360    1399#
TYPSTK  005316          2661#   2667
UPMM    007212          1387    2597    3116#
WRTMEM  000120          1049#   1771    1907    1996
SA    = 000001           393#    395     396#    398     399#    403     404#    408     409#    413     414#    419     420#
                         426     427#    432     433#    437     438#    441     442#    445     446#    449     450#    453
                         454#    457     458#    461     462#    465     466#    469     470#    474     475#    478     479#
                         482     483#    486     487#    490     491#    492     493#    500     501#    504     505#    508
                         509#    519     520#    523     524#    532     533#    536     537#    539     540#    545     546#
                         549     550#    553     554#    557     558#    565     566#    569     570#    573     574#    578
                         579#    583     584#    590     591#    596     597#    600     601#    605     606#    609#
SADERR  000301          1154#   1157    1619*   1740*   1804*   1869*   2788    2805
SAPTHD  000276          1122    1128#   1141
SCNTMM  005140          2596    2598#
SCPUOP  000426          1240#
SDEVCT  000410          1231#   1575*   2084*   2207*
SDOAGN  005542          2721    2729#
SENDAD  000156          1017    1068#   2727
SENV    000420          1236#   1324    1568    2821    2858
SENVM   000421          1237#   1342    1639    2968
SEOP    005440          2656    2664    2707#
SETABL  000420          1235#
SETEND  000450          1134    1264#
```

CROSS REFERENCE TABLE -- USER SYMBOLS

```
$FATAL  000402          1228#  2751*  2760*  2768   2852*  2855
$GTSIZ  007324          1410   1525   3179#
$HD   = 000002           996
$HIBTS  000276          1129#
$HIMAX  000334          1193#  1340*
$KSB  = 177562          1181#  3322
$MAOR1  000432          1253#
$MAOR2  000436          1257#
$MAOR3  000442          1260#
$MAOR4  000446          1263#
$MAIL   000400          1085   1130   1134   1226#  1296   1311
$MAMS1  000430          1247#
$MAMS2  000434          1255#
$MAMS3  000440          1258#
$MAMS4  000444          1261#
$MAXM   000336          1200#  1341*
$MBAOR  000300          1130#
$MSGAD  000414          1233#
$MSGLG  000416          1234#
$MSGTY  000400          1030*  1227#  2823*
$MTYP1  000431          1248#  1349
$MTYP2  000435          1256#
$MTYP3  000441          1259#
$MTYP4  000445          1262#  1345
$NWTST= 000001          1584#  1586   1671#  1673   1712#  1714   1754#  1756   1847#  1849   1887#  1889   1973#
                        1975   2030#  2032   2105#  2107   2229#  2231   2276#  2278   2369#  2371
$PASS   000406          1230#  1288   2711*  2718
$PASTM  000304          1132#
$PRERR  000300          1150#  1153   1278*  2785   2801   2818*  2904*
$RETMM  007154          3102#  3114
$SVPC = 000044          1015#  1020
$SWR  = 000000           996#  1005#  1597   1677   1721   1766   1854   1902   1990   2042   2133   2258   2299
                        2400
$SWREG  000422          1238#  1326
$TESTN  000404          1089   1137   1229#  1515*  1572   1596   1676   1720   1765   1853   1901   1989   2041
                        2132   2189   2197   2222   2257   2298   2399
STN   = 000014           986#   996   1584   1597#  1671   1677#  1712   1721#  1754   1766#  1847   1854#  1887
                        1902#  1973   1990#  2030   2042#  2105   2133#  2229   2258#  2276   2299#  2369   2400#
STPB  = 177566          1183#  2972*
STPCHR  006522          2959   2968#  3059
STPDEC  006614          2536   2683   2719   2999#
STPNUM  006766          3049   3054#
STPS  = 177564          1182#  2970
STPSTK  005306          2591   2657#
STSTM   000302          1131#
STYPE   006460          1405   2954#  2975   2981   3012   3050
$UNIT   000412          1083   1232#
$UNITM  000306          1133#
$USWR   000424          1239#
$Z    = 000362          1218#
$ZZ   = 007734          3332#
$SM   = 000200          2492#
.     = 007746          1003#  1008#  1015   1016#  1018#  1020#  1022#  1028#  1035#  1074#  1118   1119#  1121#
                        1123#  1141#  1145#  1149#  1153#  1157#  1159#  1161#  1166#  1168#  1171#  1218   1221#
                        1272#  1281   1534   1597   1649   1679   1722   1766   1854   1903   1992   2043   2134
                        2259   2299   2400   2519   2534#  2717#  2777   2977#  3110#  3268   3286#  3297#  3329#
```

```
                        3332#
.$X   = 000276          1118#   1123
```

```
DZKMA   MACY11 27(1006)  18-FEB-77  12:35  PAGE 80
DZKMAC.P11    18-FEB-77 12:30          CROSS REFERENCE TABLE -- MACRO NAMES

ERRLST    393#    395     398     403     408     413     419     426     432     437     441     445     449     453     457
           461     465     469     474     478     482     486     490     492     500     504     508     519     523     532
           536     539     545     549     553     557     565     569     573     578     583     590     596     600     605
MSG      1584#   1586    1671#   1673    1712#   1714    1754#   1756    1847#   1849    1887#   1889    1973#   1975    2030#
         2032    2105#   2107    2229#   2231    2276#   2278    2369#   2371
NEWTST    986#   1584    1671    1712    1754    1847    1887    1973    2030    2105    2229    2276    2369
PLCERR   1001#   1612    1615    1623    1792    1802    1818
STARS     986#   1013    1078    1081    1115    1117    1124    1190    1197    1224    1267    1269    1584    1595    1671
          1675   1712    1719    1754    1764    1847    1852    1887    1900    1973    1988    2030    2040    2105    2131
          2229   2256    2276    2297    2369    2398
SERRNM   1001#   1688    1742    1871    1911    1918    1945    2003    2078    2158    2174    2183    2328    2336    2463
          2475   2492    2905
SFATAL    394#    395     398     403     408     413     419     426     432     437     441     445     449     453     457
           461     465     469     474     478     482     486     490     492     500     504     508     519     523     532
           536     539     545     549     553     557     565     569     573     578     583     590     596     600     605
SFTERR   1001#   1304    1352    1464    1486    1654    2915    3111    3287
SSQERR   1001#   1598    1680    1723    1767    1855    1904    1993    2044    2135    2260    2300    2401
SSNEWT    986#   1584    1671    1712    1754    1847    1887    1973    2030    2105    2229    2276    2369
.MEADE    986#
.$ACTI    986#   1011
.$APTB    986#   1222
.$APTH    986#   1113


. ABS.   007746     000

 ERRORS DETECTED:  0
DEFAULT GLOBALS GENERATED:  0

DSKZ:DZKMAC.BIN,DSKZ:DZKMAC.SEQ/CRF/SOL=DSKZ:DZKMAC.P11
RUN-TIME: 7 8 .5 SECONDS
RUN-TIME RATIO: 82/16=4.8
CORE USED:  11K  (22 PAGES)
```