PDP11

MOS/CORE 0-123K EXCERCISER

CZKMAE0

AH-8851E-MC

FICHE 1 OF 1

DEC 1978

digital

MADE IN USA

```
    1  .REPT   0
    2
    3
    4
    5
    6
    7
    8
    9
   10
   11
   12
   13
   14
   15
   16
   17                        IDENTIFICATION
   18                        --------------
   19
   20
   21           PRODUCT CODE:            AC-8850E-MC
   22
   23           PRODUCT NAME:            CZKMAE0 MOS/CORE 0-124K EXER
   24
   25           DATE CREATED:            AUG, 1978
   26
   27           MAINTAINER:              DIAGNOSTIC GROUP
   28
   29
   30
   31           COPYRIGHT (C)     1975,1976,1977,1978
   32           DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.
   33
   34           THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY
   35           ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH
   36           THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE.  THIS
   37           SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED
   38           OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT
   39           FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE
   40           LICENSE TERMS.  TITLE TO AND OWNERSHIP OF THE SOFTWARE
   41           SHALL AT ALL TIMES REMAIN IN DEC.
   42
   43           THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE
   44           WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMIT-
   45           MENT BY DIGITAL EQUIPMENT CORPORATION.
   46
   47           DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY
   48           OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
   49           DEC.
```

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.
```

```
   161  [2.0]   REQUIREMENTS
   162
   163  [2.1]   EQUIPMENT
   164
   165          STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE
   166          AND FROM 4K TO 124K OF MEMORY.  PROGRAM WILL ALSO RUN ON THE
   167          TIM PROCESSOR AND ON 30K LSI SYSTEMS.
   168
   169
   170
   171  [2.2]   STORAGE
   172
   173          PROGRAM STORAGE - 0000 - 7744.  PROGRAM EXPANDS FOR ERROR
   174          HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR.
   175          (SEE SECTION 9. FOR DETAILS)
   176
   177
   178
   179  [3.0]   LOADING PROCEDURE
   180
   181          USE STANDARD PROCEDURE FOR PDP-11 ABSOLUTE BINARY FORMATTED TAPES.
   182
   183
   184
   185  [4.0]   STARTING PROCEDURE
   186
   187  [4.1]   SWITCH SETTINGS
   188
   189          SOFTWARE SWITCH REGISTER = LOCATION 176
   190
   191          BIT15(100000)   HALT ON ERROR
   192
   193          BIT14(040000)   LOOP ON TEST DEFINED BY SWITCH REGISTER BITS <3:0>
   194          BIT13(020000)   INHIBIT ERROR PRINTOUTS
   195          BIT12(010000)   ENABLE MEMORY MANAGEMENT (TESTING ABOVE 28K)
   196
   197          BIT11(004000)   ENABLE PARITY MODULES.
   198                          !"PARITY" WILL BE TYPED
   199          BIT10(002000)   HALT AFTER EACH SUBTEST
   200                          !PRESS CONTINUE TO DO NEXT SUBTEST
   201          BIT09(001000)   INHIBIT PROGRAM RELOCATION
   202                          !IF SET LOCATIONS 430-7776 WILL NOT BE
   203                          !TESTED.
   204
   205          BIT08(000400)   TYPE FIRST FAILING BIT IN EACH 4K BANK ONLY.
   206                          !THE TOTAL ERROR COUNT (UP TO 377) WILL
   207                          !BE SAVED IN THE ERROR HISTORY.
   208          BIT07(000200)   ENABLE LONG GALLOPING TEST.
   209                          !"GLP" WILL BE TYPED.
   210                          !CAUTION! INCREASES TEST TIME BY FACTOR OF 25.
   211          BIT06(000100)   INHIBIT MEMORY SIZING.
   212                          !THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:
   213                          (VALUES TO TEST 0-8K ARE SHOWN)
   214                          (LOWTWO=LOCATION 324)
   215
   216                          LOWTWO: 0               ;STORE BITS 17:16 OF LOW TEST ADDRESS
```

```
                        LOWADD: 0               ;STORE REST OF LOW TEST ADDRESS
                                                ;DO NOT ATTEMPT TO SET THE LOWER LIMIT
                                                ;AT OR ABOVE 160000 ON A 30K LSI SYSTEM.
                                                ;THE PROGRAM WILL ASSUME MEMORY MANAGEMENT
                                                ;MUST BE USED.
                        HIGHTWO: 0              ;STORE BITS 17:16 OF HIGH TEST ADDRESS
                        HIGHADD: 37776          ;STORE REST OF HIGH TEST ADDRESS

        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 ''CZKMAE'' 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 7) HAVE BEEN TESTED.
        ''RELOC''               ;THE DIAGNOSTIC RELOCATES TO HIGHEST
                                ;LOCATIONS UNDER TEST AND RUNS TST0-TST13 AGAIN.
```

```
273          'TST13 BNK 00''          ;TESTING BANK 00 IN TEST 13 (RELOCATED STATE.)
274                                   ;NOTE-ONLY BANK 00 IS TESTED IN THE RELOCATED STATE.
275
276          'END PASS #XX''          ;WHERE 'XX'' IS THE PASS NO.
277
278
279          ADDITIONAL PRINTOUTS
280          'NO PAR''                ;PRINTED IF PARITY SELECTED BUT NOT AVAILABLE.
281
282          'NO MNG''                ;PRINTED IF SWR BIT 12 IS SET AND NO MEMORY
283                                   ;MANAGEMENT AVAILABLE.
284
285
286
287  4.5     LONG GALLOP OPTION
288
289          NORMAL WORST CASE SR SETTING = 0000.  FOR LONG GALLOP
290          SR = 200.  LONG GALLOP OPTION SHOULD ONLY BE USED IF AN
291          MOS MEMORY PROBLEM IS SUSPECTED AND NO OTHER SUBTESTS
292          WILL FAIL.  THE TEST TIME IS INCREASED 25 TIMES.
293
294
295
296  [5.0]   PROGRAM HALTS (NORMAL+ ERROR)
297
298          THIS IS A LIST OF EXPECTED HALTS.  IF THE TEST HALTS
299          IN A LOCATION NOT IN THIS LIST AND IT IS LESS THAN 776, IT
300          MAY BE DUE TO A DEVICE INTERRUPTING.
301                  NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS
302          MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND
303          BY SUBTRACTING 500 FROM THE HALT PC AND ADDING THIS DIFFERENCE TO THE
304          CONTENTS OF SAVR6 [LOC. 350].
305
306                  PC              REASON              RECOVERY
307                  --              --------            ---------
308
309                  112             TRAP TO LOC. 4          EXAMINE R6, IT CONTAINS
310                                                         THE POINTER TO THE PC
311                                                         WHERE THE TRAP OCCURRED.
312
313                  146              POWER FAIL             POWER UP WILL RECOVER
314                                                         IF IN CORE MEMORY.
315
316                  1672            HALT AT END OF          PRESS CONTINUE TO GO TO
317                                  TEST SWITCH SET.        NEXT SUBTEST.
318
319                  6144            HALT ON ERROR           PRESS CONTINUE.
320                                  SWITCH SET.
321
322                  6230            CONTROL-C TYPED         PRESS CONTINUE TO RE-
323                                  OR FATAL ERROR          START TEST.
324                                  OCCURRED
325
326
327  [6.0]   ERRORS
328
```

```
[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.

```

```
553  ;ERROR # 41     ;[TST12] TEST SEQUENCE ERROR
554                  ;$TESTN SHOULD = 12
555                  ;       THE DIAGNOSTIC HAS BEEN CORRUPTED.
556
557  ;ERROR # 42     ;[TST12] WORST CASE CORE TEST DATA ERROR
558                  ;IF PASFLG=1  COMPARE GOOD AND BAD DATA FOR FAILING BITS.
559                  ;IF PASFLG=2  THE FAILING LOCATION WAS WRITTEN AND READ
560                  ;               WITH GOOD DATA,BUT FAILED READ CHECK
561                  ;               READING IN THE MIN. TO MAX DIRECTION.
562                  ;IF PASFLG=3  SAME CONDITIONS AS PASFLG=2 EXCEPT FAILED
563                  ;               DOING THE READ CHECK FROM MAX TO MIN DIRECTION.
564
565  ;ERROR # 43     ;[TST12] WORST CASE CORE TEST DATA ERROR
566                  ;       IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA WRITTEN
567                  ;AND READ IS COMPLEMENTED.
568
569  ;ERROR # 44     ;[TST13] TEST SEQUENCE ERROR
570                  ;$TESTN SHOOULD = 13
571                  ;       THE DIAGNOSTIC HAS BEEN CORRUPTED.
572
573  ;ERROR # 45     ;[TST13] WRITE RECOVERY TEST DATA ERROR
574                  ;IF PASFLG=0  COMPARE GOOD AND BAD DATA FOR FAILING BITS.
575                  ;IF PASFLG=77400  DATA ERROR FOUND WHILE DOING A SECOND READ CHECK.
576                  ;IF PASFLG=77402  DATA ERROR FOUND IN FAILING LOCATION AFTER
577                  ;               SMALL TEST PROGRAM RUN IN FAILING BANK.
578  ;ERROR # 46     ;[TST13] WRITE RECOVERY TEST DATA ERROR
579                  ;       DATA ERROR FOUNDJUST BEFORE THE SMALL TEST
580                  ;WAS TO BE RUN IN THE FAILING BANK. TO AVOID 'BLOWING'' UP
581                  ;WHEN THE SMALL TEST IS RUN TST13 IS ABORTED.
582
583  ;ERROR # 47     ;[TST13] WRITE RECOVERY TEST DATA ERROR
584                  ;       IDENTICAL TO ERROR #XXX EXCEPT THE DATA WRITTEN
585                  ;AND READ IS DIFFERENT.(177667).
586                  ;177667 IS THE COMPLEMENT OF ''JMP (R0)'' (110) WHICH IS
587                  ;THE ESCAPE FROM THE SMALL TEST PROGRAM RUN IN THE BANK
588                  ;UNDER TEST.
589
590  ;ERROR # 50     ;[PARERR] PARITY TRAP ERROR
591                  ; PARITY TRAP TO 114 OCCURRED.
592                  ;FOR THIS ERROR PRINTOUT THE ''GOOD DATA'' IS ACTUALLY
593                  ;THE FAILING PARITY MODULE UNIBUS ADDRESS.
594                  ; SAVLOC [352] CONTAINS THE PC WHERE THE TRAP OCCURRED.
595
596  ;ERROR # 51     ;[PARITY] PARITY TRAP FATAL ERROR
597                  ; A PARITY TRAP TO 114 OCCURRED, BUT NO PARITY MODULES COULD BE FOUND
598                  ;WITH AN ERROR BIT (BIT15) SET.
599
600  ;ERROR # 52     ;[NOMM] OPERATOR FATAL ERROR
601                  ;       TESTING ABOVE 28K WAS SELECTED, BUT NO MEMORY MANAGEMENT
602                  ;OPTION WAS FOUND.
603                  ;       RESET SWITCH OPTIONS AND RESTART AT 200.
604
605  ;ERROR # 53     ;[PARITY] OPERATOR FATAL ERROR
606                  ;       PARITY TESTING WAS SELECTED BUT NO PARITY MODULES
607                  ;WERE FOUND.
608                  ;       RESET SWITCH OPTIONS AND START AT 200.
```

```
609
610  .REPT   0
611
612
613  [6.3]   ERROR HISTORY
614
615          LOCATIONS IN MEMORY ARE SET ASIDE TO COLLECT A HISTORY
616          OF THE FAILING BITS IN A PARTICULAR MEMORY BANK.  THIS
617          DATA IS COLLECTED FOR EVERY ERROR REGARDLESS OF SWITCH
618          SETTINGS.
619
620          NORMALLY THE DATA IS OUTPUT AT THE END OF TESTING, BUT
621          IF CONTROL-C IS TYPED IT IS OUTPUT AT THE END OF THE
622          CURRENT TEST.
623
624          THE ERROR HISTORY IS INTENDED TO HIGHLIGHT IF THE ERRORS
625          ARE DUE TO 1 BIT FAILING OR ONLY ADDRESS ERRORS.
626
627
628
629          ERROR HISTORY FORMAT:
630
631
632          ERROR   BANK    COUNT
633          -----   ----    -----
634
635
636          WHERE:
637
638          ERROR   =       BIT THAT FAILED [NUMBER OF THE FAILING BIT IN DECIMAL I.E.
639                          0-15 WILL BE TYPED OUT OR THE WORDS "ADR ERR" OR "PAR ERR" WILL
640                          BE TYPED OUT IF ADDRESS ERROR OR PARITY ERROR WAS SEEN
641                          IN THE SPECIFIC BANK OF MEMORY
642          BANK    =       4K MEMORY BANK IN WHICH THIS FAILURE WAS SEEN
643                          A 0 FOR 0 TO 4K, A 1 FOR 4 TO 8K AND SO ON
644          COUNT   =       NUMBER OF TIMES THIS MEMORY BANK FAILED.
645                          (377 IS MAXIMUM FAILURE COUNT RECORDED.)
646  [6.4]   ERROR RECOVERY
647
648          IF THE PROGRAM IS HALTED AFTER REPORTING AN ERROR IT CAN EITHER
649          BE CONTINUED OR RESTARTED AT 200 OR 250 (SEE SEC 4.2). HOWEVER FOR
650          CPU'S THAT DESTROY CONTENTS OF REGISTERS AFTER COMING TO A HALT
651          THE PROGRAM SHOULD ONLY BE RESTARTED.
652
653
654  [7.0]   RESTRICTIONS
655
656          MEMORY UNDER TEST SHOULD BE CONTIGUOUS. FOR SYSTEMS HAVING NON-
657          CONTIGUOUS MEMORY THE MEMORY BOUNDARIES SHOULD BE DEFINED BY THE
658          OPERATOR. (CONTIGUOUS MEMORY IS DEFINED AS A MEMORY THAT CAN BE
659          BOTH READ AND WRITTEN IN CONSECUTIVE LOCATIONS.)
660
661
662
663
664  [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	NOT USED			ON 30K (LSI-11) SYSTEMS
			1	1600	772342	
8	32K-36K	200000-217776	2	2000	772344	
9	36K-40K	220000-237776	3	2200	772346	
10	40K-44K	240000-257776	4	2400	772350	
11	44K-48K	260000-277776	5	2600	772352	
12	48K-52K	300000-317776	6	3000	772354	
13	52K-56K	320000-337776	1	3200		
14	56K-60K	340000-357776	2	3400		
15	60K-64K	360000-377776	3	3600		
16	64K-68K	400000-417776	4	4000		
17	68K-72K	420000-437776	5	4200		
18	72K-76K	440000-457776	6	4400		
19	76K-80K	460000-477776	1	4600		
20	80K-84K	500000-517776	2	5000		
21	84K-88K	520000-537776	3	5200		
22	88K-92K	540000-557776	4	5400		
23	92K-96K	560000-577776	5	5600		
24	96K-100K	600000-617776	6	6000		
25	100K-104K	620000-637776	1	6200		
26	104K-108K	640000-657776	2	6400		
27	108K-112K	660000-677776	3	6600		
28	112K-116K	700000-717776	4	7000		
29	116K-120K	720000-737776	5	7200		
30	120K-124K	740000-757776	6	7400		
31	124K-128K	760000-777776	7	7600	772354	

NOTES:
1. THE PAR (PAGE ADDRESS REGISTER) CONTENTS ARE SHOWN IN A TEST THAT SELF SIZES.
   IF THE LIMITS OF TESTING ARE SET BY THE OPERATOR AND
   IF THE BANK IS ABOVE 28K PAR NO. 1 WILL BE SET TO THE

```
                BEGINNING PAGE.  FOR EXAMPLE IF THE TESTING WAS TO
                BEGIN WITH BANK 8 PAR NO. 1 WOULD EQUAL 2000, PAR 2
                WOULD EQUAL 2200 ETC.



[8.2]   EXECUTION TIME

        HERE ARE SOME TYPICAL EXECUTION TIMES.

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



[8.2]   PASS COUNT AND TEST NO. LOCATIONS

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

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


[8.4]   STACK POINTER

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

[8.5]   POWER FAIL

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



[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 ''CZKMAE'' TITLE

        2.  [TSTRP]  SAVE DATA FROM LOCATIONS 0-376
            INTO 7744-10314.

        3.  [TSTRP]  TEST LOCATIONS 0-376 BY WRITING AND
            READING 1'S AND 0'S.  NOTE THIS IS THE ONLY
            EXPLICIT TESTING OF THESE LOCATIONS.

        4.  [SLFSIZ]  SIZE MEMORY BY WRITING INTO SUCCEEDING
            MEMORY LOCATIONS UNTIL TIMEOUT TRAP TO 4 OCCURS,
            OR 30K BOUNDARY REACHED.
            ENABLE MEMORY MANAGEMENT AND SIZE MEMORY ABOVE
            28K.

        5.  [TYPSIZ]  TYPE MEMORY TEST LIMITS.

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

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

            IF GREATER THAN 4K UNDER TEST THE ABSOLUTE LOADER
            (300 ADDRESSES) IS APPENDED.  IF GREATER THAN 4K
            AND UNDER XXDP CHAIN MODE 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.
```

```
833
834          11. [TSTSCP]  COMES HERE AFTER EACH TEST AND IF
835              CNTRL-C TYPED THEN GO TO ERROR HISTORY PRINTOUT.
836              IF SR=2000 THEN HALT
837              IF SR=40000 THEN LOOP ON TEST DEFINED BY <3:0>
838              ELSE CONTINUE TO NEXT TEST.
839
840          12. [TST1-TST12]  EXECUTE TST1-TST12 EACH TIME
841              GOING TO STEP 9.
842
843          13. [TST13]  TEST 13 IS DIFFERENT FROM TESTS 0-12,
844              BECAUSE IT IS A SMALL PROGRAM ACTUALLY RUNNING
845              IN THE MEMORY UNDER TEST.  BEFORE THIS SMALL
846              PROGRAM IS STARTED 'TST13 BNK XX' IS TYPED.
847              THIS IS DONE IN CASE THE PROGRAM FAILS . THE
848              USER CAN THEN AT LEAST TELL WHICH BANK OF MEMORY
849              FAILED.
850
851          14. [RELOC]  THE PROGRAM RELOCATES TO HIGH MEMORY
852              TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG).
853                  WHERE 'ENDPRG' IS THE CONTENTS OF ENDSTK[306].
854              I.E THE LAST PROGRAM ADDRESS.  NOTE 'RELOC' IS
855              PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
856
857          15. TESTS 0-13 ARE RUN AS DESCRIBED ABOVE EXCEPT
858              ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED.
859
860          16. [RELOER]  RELOCATE THE PROGRAM BACK TO LOWER
861              MEMORY.
862
863          17. [LOWER]  IF CONTROL-C TYPED GO PRINT ERROR
864              HISTORY.
865
866          18.[TSTMM]  IF MEMORY MANAGEMENT SELECTED AND AVAILABLE,
867                  RUN TESTS 0-13 ON THE FIRST 24K SLICE ABOVE 28K.
868
869          19. [CONTMM]  CALL 'UPMM' TO UPDATE MEMORY MANAGEMENT
870              PAR REGISTERS TO POINT TO THE NEXT 24K SLICE OF
871              UPPER MEMORY.
872
873          20. [MAXADR]  REPEAT STEPS 18 + 19 UNTIL ALL
874              MEMORY ABOVE 28K IS TESTED.
875
876          21. [ENDPAS]  PRINT ERROR HISTORY OF FAILING BITS
877
878          22. [$EOP]  DISABLE PARITY MODULES.
879              PRINT 'END PASS #XX'
880
881
882
883  [9.2]   TEST TITLES
884
885          SEE THE TEST HEADINGS IN THE LISTING FOR DETAILS ON EACH TEST.
886
887                  TEST 0: TEST FOR PROPER BANK SELECTION
888                  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 ''CZKMAE'' 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.
```

```
945
946          THE EXAMPLE ASSUMES YOU ARE LOGGED INTO THE APT MONITOR.
947
948              READY
949
950              RUN APPLU
951              APT 11 PAPER TAPE PROGRAM LOAD UTILITY
952
953
954              COMMAND:  ED
955              PROGRAM NAME TO EDIT:  EXAMPL
956              DO YOU WANT TO LOAD A NEW REV OF THE PROGRAM(Y/N)? N
957              FIRST PASS RUN TIME IN SECONDS <110>:
958              LONGEST TEST TIME IN SECONDS <10>:
959              ADDITIONAL RUN TIME IN SECONDS <0>:
960              WHICH ETABLE DO YOU WISH TO EDIT? A
961              SOFTWARE ENVIRONMENT<000>:  1
962              ENVIRONMENTAL MODE<000>:  240
963              SWITCH 1 <000000>:
964              SWITCH 2 <000000>:
965              CPU OPTIONS<0000>:
966              MEMORY TYPE 1 <000>:
967              MAXIMUM ADDRESS<00000000>:
968              MEMORY TYPE 2 <000>:
969              MAXIMUM ADDRESS<00000000>:
970              MEMORY TYPE 3 <000>:
971              MAXIMUM ADDRESS<00000000>:
972              MEMORY TYPE 4 <000>:  1
973              MAXIMUM ADDRESS<00000000>:  17776
974              WHICH ETABLE DO YOU WISH TO EDIT?
975                  COMMAND: OFF
976
977
978  .ENDR
```

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

```
1022
1023
1024            000104                  .=104
1025                                    ; GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED.
1026    000104  005237  000400          BUSER:  INC     @#$MSGTY        ;TELL APT FATAL ERROR#000
1027    000110  000000                          HALT                    ;*ERROR* TRAP TO LOC. 4 OCCURRED.
1028    000112  000000                          HALT                    ;IN CASE CONTINUE PRESSED.
1029                                    ;114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS.  SETUP IN
1030                                    ;ROUTINE 'BEGIN'.
1031            000120                          .=120
1032
1033
1034                                            ;* WRITE MEMORY BACKGROUND
1035                                            ;* ----------------------
1036                                            ;*
1037                                            ;*      THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO
1038                                            ;*      THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES
1039                                            ;*      THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE
1040                                            ;*      HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE
1041                                            ;*      SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.
1042                                            ;*
1043
1044
1045    000120  010401          WRTMEM: MOV     R4,R1           ;SET R1 TO LOWEST LOCATION UNDER TEST
1046    000122  013700  000316          MOV     @#BAKPAT,R0     ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT
1047    000126  010021          2$:     MOV     R0,(R1)+        ;STARTING FROM THE LOWEST LOCATION WRITE THE
1048    000130  020105                  CMP     R1,R5           ;MEMORY TO BACK GROUND PATTERN
1049    000132  103775                  BLO     2$
1050    000134  000207                  RTS     PC              ;RETURN FROM THE SUBROUTINE
1051
1052
1053    000136  013706  000350  PWRUP: MOV     @#SAVR6,SP      ;RESTORE STACK POINTER
1054    000142  012700  006102          MOV     #PNTMES-BEGIN,R0
1055    000146  060600                  ADD     SP,R0           ;GET THE INDIRECT ADDRESS OF LOCATION TPCRLF
1056                                                            ;RELATIVE TO LOCATION OF DIAGNOSTIC IN THE CORE
1057    000150  004710                  JSR     PC,(R0)         ;GO TO THE TYPE ROUTINE AND TYPE CR, LF AND A 'P'
1058    000152  000120                  .ASCIZ  /P/
1059                                    .EVEN
1060
1061    000154  000411                  BR      START
1062
1063                                    ;* SERVICE XXDP/ACT11
1064    000156  004710  $ENDAD: JSR     PC,(R0)         ;RETURN TO ACT11/XXDP MONITOR
1065    000160  000240                  NOP                     ;IF QUICK VERIFY=RESET ELSE NOP
1066    000162  000240                  NOP                     ;IF QUICK VERIFY=CLR #-1 ELSE INC #0
1067    000164  000240                  NOP                     ;IF QUICK VERIFY=BR .-4 ELSE NOP
1068    000166  000430                  BR      RESTRT          ;REPEAT TEST UNDER ACT11/XXDP
1069
1070            000176                  .=176
1071    000176  000000          SWREG:  .WORD   0
1072
1073
1074                                    ;;*****************************************************************
1075                                    .SBTTL  START AND RESTART ROUTINES
1076                                    ;*      RESTART AT 200 TO CLEAR APT TABLES
1077                                    ;;*****************************************************************
```

```
1078  000200  013706  000350          START:  MOV     @#SAVR6,SP      ;SETUP STACK POINTER.
1079  000204  012703  000412                  MOV     #$UNIT,R3       ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1080  000210  005043                  1$:     CLR     -(R3)           ;CLEAR A MAILBOX LOCATION
1081  000212  022703  000400                  CMP     #$MAIL,R3       ;DONE?
1082  000216  001374                          BNE     1$              ;BRANCH IF NO
1083  000220  105737  000042                  TSTB    @#42            ;ACT11 MODE?
1084  000224  001011                          BNE     RESTRT          ;BRANCH IF YES
1085  000226  105737  000405                  TSTB    @#$TESTN+1              ;ARE WE RELOCATED?
1086  000232  100406                          BMI     RESTRT          ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
1087  000234  004767  006334                  JSR     PC,TPCRLF       ;PRINT TITLE
1088  000240  055103  046513  042501          .ASCIZ  /CZKMAE0/
1089  000246  000060
1090                                          .EVEN
1091
1092  000250  012704  007744          RESTRT: MOV     #ENDPRG,R4      ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
1093  000254  012703  000346                  MOV     #SAVR5,R3       ;CAUSE R3 TO POINT TO THE LOCATION SAVR5
1094  000260  012305                          MOV     (R3)+,R5        ;RESTORE R5
1095  000262  012306                          MOV     (R3)+,SP        ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1096  000264  010600                          MOV     SP,R0           ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1097  000266  012746  000340                  MOV     #340,-(SP)      ;SET HIGH PRIORITY FOR RTI
1098  000272  010046                          MOV     R0,-(SP)
1099  000274  000002                          RTI                     ;GO TO "START"-MAY BE RELOCATED.
1100                                                                  ;IF RELOCATED SEE LOCATION SAVR6 FOR START.
1101
1102
1103
1104
1105
1106
1107
1108
1109                                  .SBTTL  APT PARAMETER BLOCK
1110
1111                                  ;;**********************************************************************
1112                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
1113                                  ;;**********************************************************************
1114          000276                          .$X=.   ;;SAVE CURRENT LOCATION
1115          000024                  .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
1116  000024  000200                  200     ;;FOR APT START UP
1117          000044                  .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
1118  000044  000276                  $APTHDR ;;POINT TO APT HEADER BLOCK
1119          000276                  .=.$X   ;;RESET LOCATION COUNTER
1120                                  ;;**********************************************************************
1121                                  ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
1122                                  ;INTERFACE SPEC.
1123
1124  000276                          $APTHD:
1125  000276  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
1126  000300  000400                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
1127  000302  001440                  $TSTM:  .WORD   800.    ;;RUN TIM OF LONGEST TEST
1128  000304  002260                  $PASTM: .WORD   1200.   ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
1129  000306  000000                  $UNITM: .WORD           ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
1130  000310  000024                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
1131
1132
1133          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
                        .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
```

```
1190  000330  000000                HIGHTWO:        0                       ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
1191  000332  037776                HIGHADD:        37776                   ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1192                                  ;;*****************************************************************
1193
1194  000334  000000                $HIMAX: 0                               ;HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1195  000336  017776                $MAXM:  17776                           ;HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1196
1197  000340  000000                MAXMEM: .WORD   ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1198
1199  000342  000000                SAVMAX: .WORD
1200  000344  000000                SAVR4:  .WORD
1201  000346  000000                SAVR5:  .WORD
1202
1203                                  ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1204  000350  000500                SAVR6:  .WORD   BEGIN                   ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1205  000352  000000                PARMAP: 0               ;MAP OF PARITY MODULES UNDER TEST.
1206  000354  000000                SAVLOC: 0               ;TEST 6 STORES ERROR INFO HERE
1207  000356  000000                PARSP:  0       ;SAVE SP DURING PARITY ERROR TRAP.
1208  000360  000000                PARPS:  0               ;SAVE PSW DURING PARITY ERROR TRAP.
1209                                                        ;NOTE-PARSP +PARPS ARE NEEDED SINCE THERE IS
1210                                                        ;IS NOT ENOUGH ROOM ON THE STACK (500-452) AND
1211                                                        ;SO THE STACK MUST BE RESET IN THE PARERR ROUTINE.
1212                                                        ;IN THIS CRUDE FASHION.
1213
1214
1215                                  ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
```

```
                        .=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.
```

```
                                        ;;*****************************************************************
000450  177570                  SWR:    177570  ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER

        000500                          .=500
000500  010706                  BEGIN:  MOV     PC,SP           ;SET UP STACK POINTER TO EQUAL BEGIN ADDRESS

000502  005746                          TST     -(SP)
000504  010637  000350                  MOV     SP,@#SAVR6      ;SAVE SP FOR FUTURE USE
000510  012737  000070  000024          MOV     #PWRDN,@#24     ;PREPARE FOR ANY FUTURE POWER DOWN
000516  005037  000300                  CLR     @#$PRERR
000522  005037  000314                  CLR     @#TYPCNT
000526  012700  000114                  MOV     #114,R0         ;PREPARE TO SETUP PARITY TRAP VECTOR
000532  012710  005474                  MOV     #PARERR-.-6,(R0)
000536  060720                          ADD     PC,(R0)+        ;TO PARERR
000540  012710  000340                  MOV     #340,(R0)       ;AND PSW OF 340
000544  105737  000405                  TSTB    @#REL           ;IS THIS CODE RELOCATED?
000550  100002                          BPL     ONEPAS          ;BRANCH IF NO
000552  000167  000572                  JMP     TSTREL          ;THIS CODE IS RELOCATED SO GET TEST SIZE.

000556  005737  000406          ONEPAS: TST     @#$PASS         ;IS THIS THE FIRST PASS?
000562  001402                          BEQ     TSTRP           ;BRANCH IF YES (TEST TRAP CATCHER ADDRESSES)
000564  000167  000406                  JMP     SETSTK          ; GET THE TEST SIZE
000570  012704  007744          TSTRP:  MOV     #ENDPRG ,R4     ;LOAD R4 WITH THE ADDRESS OF THE END OF THE PROGRAM
000574  012700  000377                  MOV     #377,R0
000600  010037  000316                  MOV     R0,@#BAKPAT
000604  005001                          CLR     R1
000606  012124                  2$:     MOV     (R1)+,(R4)+     ;SAVE FROM 0000 TO BEGIN-30 AT END OF PROGRAM FOR NOW
000610  020127  000400                  CMP     R1,#$MAIL
000614  103774                          BLO     2$
000616  005741                  3$:     TST     -(R1)           ;PREPARE TO TEST THE TRAP VECTORS
000620  010011                  4$:     MOV     R0,(R1)         ;CHECK THE TRAP VECTORS FOR THE CAPABILITY
                                                                ;OF HOLDING 0'S & 1'S
000622  020011                          CMP     R0,(R1)         ;IS THE DATA OK?
000624  001403                          BEQ     6$              ;BRANCH IF YES

000626  004767  005322                  JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
000632  000001                          1                       ;******ERROR NUMBER 1******

000634  000300                  6$:     SWAB    R0
000636  001370                          BNE     4$
000640  005701                          TST     R1              ;IF WE HAVE NOT REACHED THE LOWEST MEMORY LOCATION
000642  001365                          BNE     3$              ;THEN REPEAT FROM 3$
000644  012701  000400                  MOV     #$MAIL,R1
000650  014441                  8$:     MOV     -(R4),-(R1)     ;RESTORE TRAP CATCHER ETC.
000652  005701                          TST     R1
000654  001375                          BNE     8$
000656  012700  000006          SETSWR: MOV     #6,R0
000662  012710  000340                  MOV     #340,(R0)       ;SET UP TIME OUT TRAP PSW
000666  012740  000700                  MOV     #4$,-(R0)       ;AND THE RETURN ADDRESS
000672  005777  177552                  TST     @SWR            ;DOES THE SWITCH REGISTER POINTED BY SWR EXIST ?
000676  000404                          BR      5$              ;BRANCH IF YES
000700  022626                  4$:     CMP     (SP)+,(SP)+     ;RESTORE THE STACK POINTER
000702  012737  000176  000450          MOV     #SWREG,@#SWR    ;AND PLACE THE ADDRESS OF THE SWITCH REGISTER
                                                                ;DESIGNED FOR THE COMPUTERS NOT HAVING HARDWARE
                                                                ;SWITCH REGISTER AND RUNNING STAND ALONE
000710  105737  000420          5$:     TSTB    @#$ENV          ;RUNNING UNDER APT?
```

```
1320  000714  001403                          BEQ     APTSIZ          ;BRANCH IF NO
1321  000716  012737  000422  000450          MOV     #$SWREG,@#SWR   ;SET SWR EQUAL TO APT SWITCH REGISTER.
1322
1323
1324
1325
1326                                  ;APTSIZ- THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE AND WHEN
1327                                  ;A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO GIVEN HIGH ADDRESS.
1328                                  ;       IF APT DEFINES SIZE THE LOW TEST ADDRESS MUST=00000.(DUE TO ETABLE FORMAT)
1329                                  ;FLOW;
1330                                  ;       IF BLOCK 4 (OR 3,2,1) TYPE NON ZERO THEN GET APT HIGH ADDRESS AND EXIT.
1331                                  ;       ELSE    SEND ERROR #3
1332                                  ;NOTE; THE MEMORY TYPE IS IGNORED SINCE ALL TESTS ARE RUN REGARDLESS OF MEMORY TYPE.
1333
1334  000724  012703  000340          APTSIZ: MOV     #MAXMEM,R3      ;POINT R3 TO MAXMEM.
1335  000730  013737  000330  000334          MOV     @#HIGHTWO,@#$HIMAX      ;IN CASE NO SELF SIZING DONE.
1336  000736  013737  000332  000336          MOV     @#HIGHADD,@#$MAXM       ;IN CASE NO SELF SIZING DONE.
1337  000744  105737  000421                  TSTB    @#$ENVM         ;DOES APT ALLOW SELF SIZING?
1338  000750  100021                          BPL     TRYSR           ;BRANCH IF YES
1339
1340  000752  012701  000451                  MOV     #$MTYP4+4,R1    ;POINT R1 TO BLOCK TYPE 4(+4)
1341  000756  162701  000004          1$:     SUB     #4,R1           ;POINT R1 TO NEXT BLOCK TYPE.
1342  000762  105711                          TSTB    (R1)            ;IS THE BLOCK TYPE NON ZERO?
1343  000764  001006                          BNE     2$              ;BRANCH IF YES (MEMORY EXISTS)
1344  000766  020127  000431                  CMP     R1,#$MTYP1      ;ALL APT BLOCK TYPES BEEN CHECKED?
1345  000772  101371                          BHI     1$              ;BRANCH IF NO
1346
1347  000774  004767  005154                  JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1348  001000  000002                          2                       ;******ERROR NUMBER 2******
1349
1350  001002  004767  006316          2$:     JSR     PC,GETADR       ;GO SET MAXIMUM APT ADDRESS INTO $MAXM + $HIMAX
1351  001006  004767  006312                  JSR     PC,GETADR       ;GO SET MAXIMUM APT ADDRESS INTO HIGHADD+HIGHTWO
1352  001012  000453          BRTPSZ: BR      TYPSIZ          ; TYPE THE SIZE OF MEMORY UNDER TEST
1353
1354  001014  032777  000100  177426  TRYSR:  BIT     #100,@SWR       ;USER DEFINED MEMORY TEST BOUNDARIES??
1355  001022  001047                          BNE     TYPSIZ          ;BRANCH IF YES (DON'T SIZE MEMORY)
1356
1357
1358
1359
1360
1361  001024  010401          SLFSIZ: MOV     R4,R1           ;SETUP R1 AND R4 TO THE LOWEST ADDRESS OF MEMORY
1362  001026  012710  001050          MOV     #4$,(R0)        ;SET UP RETURN ADDRESS FROM TIME OUT TRAP TO 4$
1363  001032  011111          2$:     MOV     (R1),(R1)       ;WRITE A MEMORY LOCATION INTO ITSELF AND TRAP IF NXM
1364  001034  062701  000002          ADD     #2,R1           ;ADD 2 TO THE ADDRESS POINTER
1365  001040  022701  170000          CMP     #170000,R1      ;CHECK IF BEYOND 30K MEMORY BOUNDARY
1366  001044  101372                  BHI     2$              ;KEEP ON SIZING UP THE MEMORY UNTIL NXM TRAP
1367                                                          ;(TIME OUT TRAP) IS ENCOUNTERED
1368  001046  000401                  BR      5$              ;OR 30K BOUNDARY REACHED
1369
1370  001050  022626          4$:     CMP     (SP)+,(SP)+     ;RESTORE THE STACK POINTER
1371  001052  004767  005776  5$:     JSR     PC,MEMMNG       ; SERVICE MEMORY MANAGEMENT IF IT IS AVAILABLE
1372                                                          ;AND IF IT HAS TO BE TESTED
1373  001056  105737  000276          TSTB    @#MMAVA         ;SEE IF MEMORY MANAGEMENT HAS TO BE TESTED
1374  001062  001416                  BEQ     12$             ;IF NO MEM. MANG. THEN GO TO 12$
1375  001064  012710  001076  6$:     MOV     #8$,(R0)        ;SET UP THE RETURN ADDRESS FROM TRAP TO 8$
```

```
1376  001070  012701  020000            MOV     #20000,R1       ;BEGIN CHECKING MEMORY ABOVE 28K
1377  001074  000756                    BR      2$
1378  001076  022626            8$:     CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
1379  001100  022701  160000            CMP     #160000,R1      ;IF R1 DID NOT READ ALL THE LOCATIONS POINTED BY
1380                                                            ;PAGE ADDRESS REGISTER 6 THEN IT HAS REACHED THE
1381                                                            ;MAXIMUM AVAILABLE MEMORY
1382  001104  001005                    BNE     12$             ;IN WHICH CASE GO TO 12$
1383  001106  013702  172352            MOV     @#172352,R2     ;PREPARE TO UPDATE MEMORY MANAGEMENT REGISTERS
1384  001112  004767  005742            JSR     PC,MMREG                ;OTHERWISE GO TO UPDATE MEM. MANG. REGISTERS
1385  001116  000762                    BR      6$
1386  001120  024341            12$:    CMP     -(R3),-(R1)     ;CAUSE R3 TO POINT TO LOCATION $MAXM AND R1
1387                                                            ;TO THE MAXIMUM AVAILABLE MEMORY
1388  001122  004767  006106            JSR     PC,PUTADR       ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
1389                                                            ;AT LOCATIONS $MAXM AND $HIMAX
1390  001126  024343                    CMP     -(R3),-(R3)     ;MAKE R3 POINT TO HIGHADD
1391  001130  004767  006100            JSR     PC,PUTADR       ;PLACE THE ADDRESS IN R1 AT LOCATIONS HIGHADD
1392                                                            ;AND HIGHTWO
1393  001134  005743                    TST     -(R3)
1394  001136  005043                    CLR     -(R3)           ;CLEAR THE LOCATION LOWADD
1395  001140  005043                    CLR     -(R3)           ;AND LOWTWO
1396  001142  012720  000104    TYPSIZ: MOV     #BUSER,(R0)+    ;SET UP VECTOR FOR ANY FUTURE TRAP
1397  001146  010403                    MOV     R4,R3           ;SET R3 TO POINT TO THE LOWEST AVAILABLE MEMORY
1398                                                            ;LOCATION
1399  001150  012701  000324            MOV     #LOWTWO,R1
1400  001154  004767  005402            JSR     PC,PCRLF        ;TYPE CR/LF
1401  001160  004767  005550            JSR     PC,OCTTYP       ;TYPE LOW TEST ADDRESSE (LOWTWO+LOWADD)
1402  001164  004767  005304    TYPMEM: JSR     PC,$TYPE        ; TYPE ''-''
1403  001170  000055                    .ASCIZ  /-/
1404                                     .EVEN
1405  001172  004767  005536            JSR     PC,OCTTYP       ;TYPE HIGHEST TEST ADDRESS (HIGHTWO+HIGHADD)
1406  001176  012703  000330    SETSTK: MOV     #HIGHTWO,R3     ;MAKE R3 POINT TO THE HIGH ORDER BITS OF TOP ADDRESS
1407  001202  004767  006132            JSR     PC,$GTSIZ       ; GET THE BITS 13-17 OF THE TOP ADDRESS
1408                                                            ;PLACED IN BITS 0-4 OF R2
1409  001206  010401                    MOV     R4,R1           ;SET R1 TO LOWEST TEST ADDRESS
1410
1411  001210  062704  000022    4$:     ADD     #18.,R4         ;APPEND THE ERROR STACK FOR THE MEMORY UNDER
1412                                                            ;TEST TO THE END OF THE PROGRAM
1413  001214  005302                    DEC     R2
1414  001216  002374                    BGE     4$
1415  001220  022737  167776  000336    CMP     #167776,@#$MAXM ;CHECK IF THIS IS A 30K SYSTEM
1416  001226  001002                    BNE     5$              ;BRANCH IF NOT
1417  001230  062704  000022            ADD     #18.,R4         ;SAVE ANOTHER BANKS WORTH OF ERROR STACK
1418  001234  010437  000310    5$:     MOV     R4,@#ENDSTK     ;SAVE THE ADDRESS OF THE END OF THE ERROR STACK
1419  001240  005021            6$:     CLR     (R1)+           ;CLEAR THE ERROR STACK
1420  001242  020104                    CMP     R1,R4
1421  001244  101775                    BLOS    6$
1422  001246  012737  157776  000340    MOV     #157776,@#MAXMEM                ;SET MAXMEM TO MAXIMUM VIRTUAL ADDRESS
1423  001254  005723                    TST     (R3)+           ;TESTING MEMORY MANAGEMENT?
1424  001256  001005                    BNE     SAVLDR          ;BRANCH IF YES (GO SAVE LOADERS AT TOP OF VIRTUAL MEMORY
1425  001260  021327  170000            CMP     (R3),#170000    ;IS THE VIRTUAL ADDRESS ABOVE 167776?
1426  001264  103002                    BHIS    SAVLDR          ;BRANCH IF YES (GO SAVE LOADERS)
1427  001266  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  001272  004767  006114    SAVLDR: JSR     PC,CLRMM        ; DISABLE THE MEMORY MANAGEMENT UNIT
```

```
1432  001276  005723                          TST     (R3)+           ;MAKE R3 TO POINT TO THE LOCATION MAXMEM
1433  001300  011305                          MOV     (R3),R5         ;R5 CONTAINS THE ADDRESS OF MAXIMUM AVAILABLE MEM.
1434
1435                                  ;IF ONLY 4K BEING TESTED DON'T SAVE LOADERS
1436
1437  001302  020527  017776                  CMP     R5,#17776       ;ONLY TESTING 4K MAX?
1438  001306  103416                          BLO     4$              ;BRANCH IF YES (DON'T SAVE LOADERS)
1439
1440  001310  162705  000276          3$:     SUB     #276,R5         ;PREPARE TO SAVE 300 BYTES OF THE LOADERS
1441  001314  005737  000042                  TST     @#42            ;IS THE PROGRAM RUNNING UNDER ACT OR XXDP ?
1442  001320  001406                          BEQ     2$              ;IF NOT THEN GO TO 2$
1443  001322  023737  000042  000046          CMP     @#42,@#46       ;ARE WE RUNNING UNDER XXDP CHAIN MODE?
1444  001330  001402                          BEQ     2$              ;BRANCH IF NO
1445  001332  162705  005674                  SUB     #<1502.*2>,R5   ;SAVE 1500. WORDS FOR XXDP CHAIN MODE
1446  001336  012524                  2$:     MOV     (R5)+,(R4)+     ;SAVE LOADER
1447  001340  020513                          CMP     R5,(R3)
1448  001342  101775                          BLOS    2$
1449  001344  012323                  4$:     MOV     (R3)+,(R3)+     ;SAVE THE CONTENTS OF LOCATION MAXMEM IN SAVMAX
1450  001346  010423                          MOV     R4,(R3)+        ;AND THE CONTENTS OF R4 AT SAVR4
1451
1452  001350  010537  000346          TSTREL: MOV     R5,@#SAVR5      ;SAVE HIGHEST VIRTUAL ADDRESS+2
1453  001354  004767  006032          TSTSIZ: JSR     PC,CLRMM        ;GO TO DISABLE MEMORY MANAGEMENT UNIT
1454  001360  005745                          TST     -(R5)           ;SET R5 BACK TO HIGHEST VIRTUAL ADDRESS
1455  001362  012703  000324          1$:     MOV     #LOWTWO,R3      ;PREPARE TO LOAD R4 AND R5 WITH THE MEMORY BOUNDRIES
1456  001366  005723                          TST     (R3)+           ;IF THE BITS 16,17 OF THE LOWEST LOCATION UNDER
1457                                                                  ;TEST ARE NON ZERO
1458  001370  001003                          BNE     2$              ;THEN GO TO 2$
1459  001372  021327  157776                  CMP     (R3),#157776    ;IF THE LOWEST LOCATION UNDER TEST IS HIGHER THAN
1460                                                                  ;157776 THEN GO TO TEST MEMORY MANAGEMENT
1461  001376  103411                          BLO     4$
1462  001400  032777  010000  177042  2$:     BIT     #10000,@SWR     ;IS MEMORY MANAGEMENT SELECTED?
1463  001406  001003                          BNE     3$              ;YES ALL IS WELL
1464  001410  004767  004540                  JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1465  001414  000003                          3                       ;******ERROR NUMBER 3******
1466
1467  001416  000167  003514          3$:     JMP     TSTMM           ;GO TO TEST MEMORY MANAGEMENT
1468  001422  020423                  4$:     CMP     R4,(R3)+        ;COMPARE TOP OF PROGRAM (WITH SAVED LOADERS) TO
1469                                                                  ;LOWEST LOCATION UNDER TEST
1470
1471  001424  103002                          BHIS    6$
1472  001426  016304  177776                  MOV     -2(R3),R4       ;ADJUST R4 TO POINT TO THE LOWEST LOCATION UNDER TEST
1473  001432  005723                  6$:     TST     (R3)+           ;IF BITS 16-17 OF HIGHEST LOCATION TO BE TESTED
1474  001434  001003                          BNE     8$              ;ARE NON ZERO THEN GO TO 8$
1475  001436  021305                          CMP     (R3),R5         ;OTHERWISE SEE IF THE HIGHEST LOCATION TO BE
1476                                                                  ;TESTED IS HIGHER THAN HIGHEST VIRTUAL ADDRESS
1477  001440  101001                          BHI     8$              ;IF SO THEN GO TO 8$
1478  001442  011305                          MOV     (R3),R5         ;MODIFY R5
1479  001444  105737  000405          8$:     TSTB    @#REL           ;ARE WE RELOCATED.?
1480  001450  100014                          BPL     10$             ;BRANCH IF NO
1481  001452  013704  000322                  MOV     @#RELBOT,R4     ;SET BOTTOM TEST ADDRESS WHEN RELOCATED.
1482  001456  020527  017776                  CMP     R5,#17776       ;ARE WE RELOCATED IN BANK 0?
1483  001462  103402                          BLO     9$              ;BRANCH IF YES
1484  001464  012705  017776                  MOV     #17776,R5       ;ELSE SET HIGH MEMORY UNDER TEST=4K
1485
1486  001470  020405                  9$:     CMP     R4,R5           ;IS LOW LIMIT LOWER THAN HIGH LIMIT?
1487  001472  103403                          BLO     10$             ;BRANCH IF YES
```

```
1488  001474  004767  004454          JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1489  001500  000004                  4                       ;******ERROR NUMBER 4******
1490
1491  001502  012703  000342  10$:    MOV     #SAVMAX,R3
1492  001506  011343                  MOV     (R3),-(R3)      ;RESTORE THE CONTENTS OF MAXMEM
1493  001510  062713  000002  MEMTST: ADD     #2,(R3)         ;MAKE THE CONTENTS OF MAXMEM = MAXIMUM AVAILABLE
1494                                                          ;MEMORY +2
1495  001514  005725                  TST     (R5)+           ;AND SET R5=MAX MEMORY+2
1496
1497                          ;CLEAR MEMORY UNDER TEST
1498
1499  001516  010500          CLRMEM: MOV     R5,R0           ;MOVE HIGH ADDRESS TO R0
1500  001520  005040          2$:     CLR     -(R0)           ;BEGIN CLEARING THE MEMORY FROM THE TOP
1501  001522  020004                  CMP     R0,R4           ;UNTIL THE BOTTOM IS REACHED
1502  001524  101375                  BHI     2$
1503  001526  012702  000001          MOV     #1,R2           ;SET R2 TO ENABLE PARITY MODULE CODE.
1504  001532  004767  005754          JSR     PC,PARITY       ;ENABLE PARITY IF WANTED AND AVAILABLE.
1505  001536  012702  000316          MOV     #BAKPAT,R2
1506  001542  012212                  MOV     (R2)+,(R2)      ;WRITE SWAPPED BAKPAT IN LOCATION SWAPAT
1507  001544  000312                  SWAB    (R2)
1508  001546  017702  176676          MOV     @SWR,R2         ;LOAD R2 WITH THE OPTIONS STORED AT $SWREG
1509  001552  042702  177760          BIC     #177760,R2      ;ONLY LEAVE THE LOWER 4 BITS OF $SWREG IN R2 TO GO TO
1510                                                          ;THE TEST # SPECIFIED [DEFAULT IS TEST#0]
1511
1512
1513
1514                          ;ENTER HERE FROM TSTSCP ROUTINE AT END OF SUBTEST
1515
1516  001556  005037  000306  CONT:   CLR     @#PASFLG        ;INIT SUBTEST PASS FLAG.
1517  001562  110237  000404          MOVB    R2,@#$TESTN     ;SET UP $TESTN WITH THE TEST NUMBER GOING
1518                                                          ;TO BE EXECUTED
1519  001566  010401  LOOP:   MOV     R4,R1           ;LOAD R1 WITH THE LOWEST LOCATION UNDER TEST
1520  001570  010400          MOV     R4,R0           ;PLACE THE ADDRESS OF THE LOWEST LOCATION UNDER
1521                                                  ;TEST IN R0
1522  001572  010403          MOV     R4,R3           ;AND IN R3
1523  001574  006302          ASL     R2
1524  001576  060702          ADD     PC,R2
1525  001600  066207  000004  ADD     TBL-.(R2),PC            ;GO TO THE TEST #
1526                                                  ;STORED IN BITS 0-3 OF SWITCH REGISTER
1527
1528
1529  001604  000102  TBL:    TST0-TBL                ;RELATIVE ADDRESS OF TEST # 0
1530  001606  000340          TST1-TBL                ;RELATIVE ADDRESS OF TEST # 1
1531  001610  000440          TST2-TBL                ;RELATIVE ADDRESS OF TEST # 2
1532  001612  000550          TST3-TBL                ;RELATIVE ADDRESS OF TEST # 3
1533  001614  001016          TST4-TBL                ;RELATIVE ADDRESS OF TEST # 4
1534  001616  001126          TST5-TBL                ;RELATIVE ADDRESS OF TEST # 5
1535  001620  001274          TST6-TBL                ;RELATIVE ADDRESS OF TEST # 6
1536  001622  001430          TST7-TBL                ;RELATIVE ADDRESS OF TEST # 7
1537  001624  001654          TST10-TBL               ;RELATIVE ADDRESS OF TEST # 10
1538  001626  002204          TST11-TBL               ;RELATIVE ADDRESS OF TEST # 11
1539  001630  002256          TST12-TBL               ;RELATIVE ADDRESS OF TEST # 12
1540  001632  002530          TST13-TBL               ;RELATIVE ADDRESS OF TEST # 13
1541  001634  003160          RELOC-TBL               ;RELATIVE ADDRESS OF ROUTINE 'RELOC'
1542
1543
```

```
1544
1545                                    ;R5 IS POINTING TO THE TOP OF THE MEMORY TO BE TESTED+2
1546                                    ;R4 & R0 ARE POINTING TO THE LOWEST ADDRESS OF MEMORY TO BE TESTED
```

```
CZKMA   MACY11 30A(1052)  18-SEP-78  11:56  PAGE 31                                                    SEQ 0031
CZKMAE.MAC    18-SEP-78 11:54          SCOPE ROUTINE

 1547                                 ;*      SCOPE ROUTINE
 1548                                 ;*      -------------
 1549                                 ;*
 1550                                 ;*
 1551                                 ;*          PROGRAM COMES TO THIS ROUTINE AFTER COMPLETION OF EACH TEST AND
 1552                                 ;*          IF CNTRL-C TYPED GOTO ERROR HISTORY TYPE ROUTINE.
 1553                                 ;*          IF SR= 2000 (BIT10) THEN HALT
 1554                                 ;*          IF SR= 40000 (BIT14) THEN LOOP ON TEST DEFINED BY SR BITS<3:0>
 1555                                 ;*          ELSE CONTINUE TO NEXT TEST.
 1556                                 ;*
 1557
 1558
 1559  001636  105737  000420         TSTSCP: TSTB    @#$ENV          ;ARE WE RUNNING UNDER APT?
 1560  001642  001002                         BNE     CNTSCP          ;IF SO THEN GO TO CNTSCP
 1561  001644  004767  006022                 JSR     PC,CHECKC       ;TEST FOR CONTROL-C AND IF TYPED GO
 1562                                                                  ;PRINT ERROR HISTORY AND HALT AT FATHLT.
 1563  001650  113702  000404         CNTSCP: MOVB    @#$TESTN,R2     ;PLACE THE TEST NUMBER IN THE LOWER BYTE OF R2
 1564                                                                  ;SINCE THERE ARE LESS THAN 377 TESTS UPPER BYTE
 1565                                                                  ;OF R2 WILL BE 0
 1566  001654  005237  000410                 INC     @#$DEVCT        ;TELL APT WE ARE STILL RUNNING OKAY
 1567  001660  032777  002000  176562         BIT     #2000,@SWR      ;IS THE PROGRAM GOING TO HALT AFTER EACH TEST?
 1568  001666  001401                         BEQ     TSTGO           ;IF NOT THEN GO TO 2$
 1569  001670  000000         SWHALT: HALT                            ;HALT AT END OF TEST SWITCH SET.
 1570
 1571  001672  032777  040000  176550 TSTGO:  BIT     #40000,@SWR     ;IS THE PROGRAM GOING TO LOOP ON TEST
 1572  001700  001332                         BNE     LOOP            ;IF SO THEN GO TO THE STARTING OF THE SAME TEST
 1573  001702  105202                         INCB    R2
 1574  001704  000724                         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
                                ;;*****************************************************************
001706  105737  000404  TST0:   TSTB    @#$TESTN        ;CHECK FOR PROPER TEST SEQUENCE
001712  001403            BEQ     .+10
001714  004767  004234    JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
001720  000005            5                       ;******ERROR NUMBER 5******

001722  012703  177777            MOV     #177777,R3
001726  010401          1$:     MOV     R4,R1           ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
001730  010310            MOV     R3,(R0)         ;SET ALL THE BITS AT (R0)
001732  020001          2$:     CMP     R0,R1           ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
001734  001417            BEQ     4$              ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
001736  005711            TST     (R1)            ;OTHERWISE CHECK (R1) FOR ALL 0'S
001740  001430            BEQ     5$
001742  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)

001744  001004            BNE     3$
001746  012767  000006  000042    MOV     #6,12$          ;*ERROR* SETUP ERROR NO. IN 12$
                                                ;******ERROR NUMBER #6******
001754  000403            BR      10$
001756                  3$:
001756  012767  000007  000032    MOV     #7,12$          ;*ERROR* SETUP ERROR NO. IN 12$
                                                ;******ERROR NUMBER #7******
001764  010046          10$:    MOV     R0,-(SP)        ;SAVE R0 ON STACK
001766  105237  000301            INCB    @#$ADERR        ;AN ADDRESSING ERROR IS SUSPECTED
001772  000407            BR      11$
001774  020311          4$:     CMP     R3,(R1)         ;CHECK (R1) FOR ALL 1'S
001776  001411            BEQ     5$
002000  012767  000010  000010    MOV     #10,12$         ;*ERROR* SETUP ERROR NO. IN 12$
                                                ;******ERROR NUMBER #10******
002006  010046            MOV     R0,-(SP)        ;SAVE R0 ON STACK
002010  010300            MOV     R3,R0
002012  004767  003600  11$:     JSR     PC,ERROR        ;GO TO THE ERROR SUBROUTINE
002016  000000  12$:     .WORD                   ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
002020  012600            MOV     (SP)+,R0        ;RESTORE R0

002022  013706  000350  5$:     MOV     @#SAVR6,SP      ;RESTORE THE STACK POINTER
002026  062701  020000            ADD     #20000,R1       ;CAUSE R1 TO POINT TO THE SAME CHIP
                                                ;LOCATION IN THE NEXT 4K BANK OF MEMORY
                                                ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
002032  020105            CMP     R1,R5           ;COMPARE R1 WITH THE HIGHEST MEMORY
                                                ;LOCATION WHICH IS STORED IN R5
002034  103736            BLO     2$              ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$

002036  105737  000421            TSTB    @#$ENVM         ;HAS APT INHIBITED SIZING?
```

```
1631  002042  100432                          BMI     8$              ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1632  002044  032777  000100  176376          BIT     #100,@SWR       ;HAS USER INHIBITED SIZING?
1633  002052  001026                          BNE     8$              ;BRANCH IF YES (DON'T TEST NON-EXISTENT MEMORY)
1634
1635  002054  020127  157776                  CMP     R1,#157776      ;SEE IF R1 HAS CROSSED 28K BOUNDRY OF VIRTUAL ADDRESS
1636  002060  101016                          BHI     6$              ;IN WHICH CASE GO TO 6$
1637                                                                  ;SHOULD BE LEFT AS IS FOR 30K SYSTEMS (WHICH USE 16K CHI
1638  002062  020137  000340                  CMP     R1,@#MAXMEM     ;IS R1 LOWER THAN THE MAXIMUM AVAILABLE
1639                                                                  ;MEMORY ?
1640  002066  103755                          BLO     5$              ;IF SO THEN GO TO 5$
1641  002070  012702  000006                  MOV     #6,R2           ;MAKE R2 POINT TO TRAP VECTOR+2 FOR NXM
1642  002074  012712  000340                  MOV     #340,(R2)       ;SET PSW TO 340
1643  002100  012742  177714                  MOV     #5$-.-6,-(R2)   ;SET UP RETURN ADDRESS FROM TRAP TO 5$
1644  002104  060712                          ADD     PC,(R2)
1645  002106  011111                          MOV     (R1),(R1)       ;TRY TO WRITE TO NON-EXISTENT MEMORY (SHOULD TRAP)
1646  002110  004767  004040                  JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
1647  002114  000011                          11                      ;******ERROR NUMBER 11******
1648
1649
1650  002116  012702  000004          6$:     MOV     #4,R2
1651  002122  012722  000006                  MOV     #6,(R2)+        ;RESTORE TRAP VECTOR
1652  002126  005012                          CLR     (R2)
1653  002130  005010                  8$:     CLR     (R0)
1654
1655  002132  062700  020000                  ADD     #20000,R0       ;CAUSE R0 TO POINT TO THE SAME CHIP
1656                                                                  ;LOCATION IN THE NEXT 4K MEMORY BANK
1657                                                                  ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R0
1658  002136  020005                          CMP     R0,R5           ;COMPARE R0 WITH THE HIGHEST MEMORY
1659                                                                  ;LOCATION WHICH IS STORED IN R5.
1660  002140  103672                          BLO     1$              ;IF R0 LESS THEN REPEAT THE TEST
1661  002142  000635                  ENDO:   BR      TSTSCP
1662
1663
```

```
                                        ;;*****************************************************************
                                        ;*TEST 1         CHECK DI/DO LINES
                                        ;*(1)    THIS TEST CHECKS THE DATI/DATO LINES BY SHIFTING
                                        ;*       A 1 IN THE WORD DIRECTION
                                        ;;*****************************************************************
002144  122737  000001  000404  TST1:   CMPB    #1,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE

002152  001403                         BEQ     .+10
002154  004767  003774                 JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
002160  000012                         12                      ;******ERROR NUMBER 12******

002162  012700  000001          1$:    MOV     #1,R0
002166  010002                         MOV     R0,R2           ;SET R2=1
002170  010011                 2$:     MOV     R0,(R1)         ;MOV 1 AT LOCATION (R1)
002172  020011                 3$:     CMP     R0,(R1)         ;COMPARE R1 WITH THE CONTENTS OF LOCATION (R1)
002174  001403                         BEQ     4$
002176  004767  003414                 JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
002202  000013                         13                      ;******ERROR NUMBER 13******

002204  005702                 4$:     TST     R2              ;ARE WE SHIFTING A 0 IN DATA DIRECTION?
002206  001406                         BEQ     5$              ;IF SO THEN GO TO 5$
002210  006300                         ASL     R0              ;SHIFT THE 1 BROUGHT IN AT 1$ IN
                                                               ;DATA DIRECTION
002212  103366                         BCC     2$              ;IF THE 1 HAS NOT BEEN SHIFTED THRU
                                                               ;THE 16 DATA BITS THEN REPEAT FROM 2$
002214  005002                         CLR     R2              ;INITIATE SHIFTING OF 0 IN DATA DIRECTION
002216  012700  177776                 MOV     #177776,R0
002222  000762                         BR      2$

002224  000261                 5$:     SEC                     ;SET C BIT
002226  006100                         ROL     R0              ;SHIFT A 0 16 TIMES IN DATA DIRECTION
002230  103757                         BCS     2$              ;IF THE 0 HAS NOT BEEN SHIFTED THRU
                                                               ;THE 16 DATA BITS THEN REPEAT FROM 2$
002232  062701  020000                 ADD     #20000,R1       ;OTHERWISE GO TO THE NEXT BANK OF
                                                               ;4K MEMORY AND REPEAT THE TEST
002236  020105                         CMP     R1,R5
002240  103750                         BLO     1$
002242  000737                 END1:   BR      END0
```

```
                                ;;*****************************************************************
                                ;*TEST 2         TEST MEMORY FOR HOLDING DATA AND BYTE SELECTION
                                        ;*(1)   THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
                                        ;*      OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
                                        ;*      OF BAKPAT AND READING IT
                                        ;*(2)   MEMORY IS WRITTEN USING A BYTE AT A TIME
                                        ;*(3)   STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
                                ;;*****************************************************************
002244  122737  000002  000404  TST2:  CMPB    #2,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE

002252  001403                        BEQ     .+10
002254  004767  003674                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
002260  000014                        14                      ;******ERROR NUMBER 14******

002262  013700  000316          1$:   MOV     @#BAKPAT,R0
002266  110021                        MOVB    R0,(R1)+
002270  113721  000317                MOVB    @#BAKPAT+1,(R1)+;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
002274  020105                        CMP     R1,R5
002276  103771                        BLO     1$

002300  020041          2$:   CMP     R0,-(R1)        ;TEST THE MEMORY TO SEE IF IT CONTAINS
                                                      ;THE WORD STORED IN BAKPAT
002302  001416                        BEQ     8$
002304  062701  000002                ADD     #2,R1
002310  123741  000317                CMPB    @#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
002314  001402                        BEQ     4$
002316  120041                        CMPB    R0,-(R1)        ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
002320  001002                        BNE     6$
002322  105237  000301          4$:   INCB    @#$ADERR        ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
002326  042701  000001          6$:   BIC     #1,R1           ;MAKE THE ADDRESS IN R1 EVEN
002332  004767  003260                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
002336  000015                        15                      ;******ERROR NUMBER 15******

002340  020104          8$:   CMP     R1,R4           ;KEEP ON TESTING THE MEMORY UNTIL
002342  101356                        BHI     2$              ;R1 EQUALS THE LOWEST ADDRESS
002344  000337  000316                SWAB    @#BAKPAT        ;CHANGE THE DATA PATTERN
002350  001744                        BEQ     1$              ;IF THE DATA PATTERN DOES NOT HAVE LOW
                                                              ; BYTE =0 THEN FALL THRU
002352  000733          END2: BR      END1

                                ;THE TEST LEAVES BAKPAT LOCATION THE SAME AS IT WAS IN THE BEGINNING
```

```
1747                                    ;;*****************************************************************
1748                                    ;*TEST 3         DUAL ADDRESS TEST A
1749
1750                                            ;*(1)   THIS TEST CHECKS FOR DUAL ADDRESSING PROBLEMS BY WRITING A
1751                                            ;*      BACK GROUND OF BAKPAT.
1752                                            ;*(2)   STARTING FROM THE LOWEST LOCATION IN THE BANK THE TEST WRITES A
1753                                            ;*      LOCATION WITH SWAPPED BAKPAT
1754                                            ;*(3)   READS THE MEMORY FOR PROPER CONTENTS
1755                                            ;*(4)   SHIFTS A 1 ALONG THE ADDRESS DIRECTION AND REPEATS STEPS 1-3
1756                                            ;*(5)   REPEATS STEP 1-4 FOR EACH 4K BANK
1757                                    ;;*****************************************************************
1758  002354  122737  000003  000404  TST3:   CMPB    #3,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE
1759  002362  001403                          BEQ     .+10
1760  002364  004767  003564                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1761  002370  000016                          16                      ;******ERROR NUMBER 16******
1762
1763  002372  005003                          CLR     R3
1764  002374  004737  000120          2$:     JSR     PC,@#WRTMEM     ; WRITE MEMORY WITH THE BACKGROUND STORED
1765                                                                  ;AT LOCATION BAKPAT
1766  002400  005002                  4$:     CLR     R2
1767  002402  050302                  6$:     BIS     R3,R2           ;MAKE R2 POINT TO THE MEMORY BANK POINTED BY R3
1768  002404  020204                          CMP     R2,R4           ;IF R2 IS LESS THAN R4
1769  002406  103465                          BLO     16$             ;THEN DO NOTHING
1770  002410  020205                          CMP     R2,R5           ;IF R2 IS HIGHER THAN THE HIGHEST LOCATION TO BE
1771  002412  103077                          BHIS    20$             ;TESTED THEN EXIT THE TEST
1772  002414  000312                          SWAB    (R2)            ;OTHERWISE WRITE THE COMPLEMENT OF BAKPAT IN
1773                                                                  ;THE LOCATION POINTED BY R2
1774  002416  005001                          CLR     R1
1775  002420  050301                  7$:     BIS     R3,R1
1776  002422  020104                          CMP     R1,R4           ;IF R1 IS POINTING TO A LOCATION LOWER THAN R4
1777  002424  103445                          BLO     12$             ;THEN GO TO 12$
1778  002426  020105                          CMP     R1,R5
1779  002430  103053                          BHIS    15$
1780  002432  020102                          CMP     R1,R2           ;CHECK THE MEMORY FOR CORRECT DATA
1781  002434  001431                          BEQ     10$
1782  002436  020011                          CMP     R0,(R1)         ;IF R1 IS NOT = TO R2 THEN (R1) SHOULD HAVE
1783                                                                  ;THE SAME WORD AS BAKPAT
1784  002440  001437                          BEQ     12$             ;IN WHICH CASE GO BACK TO 12$
1785  002442  012767  000017  000032          MOV     #17,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1786                                                                  ;******ERROR NUMBER #17******
1787  002450  010046                  8$:     MOV     R0,-(SP)        ;PLACE R0 ON THE STACK
1788  002452  000316                          SWAB    (SP)
1789  002454  022611                          CMP     (SP)+,(R1)      ;IF (R1) IS NOT = R0 THEN SEE IF IT IS SAME
1790                                                                  ;AS A SWAPPED R0
1791  002456  001003                          BNE     9$              ;IF NOT THEN A SUSPECTED DUAL ADDRESSING PROBLEM
1792                                                                  ;FOR THE BITS THAT ARE DIFFERENT IN R0 AND (R1)
1793                                                                  ;OTHERWISE THERE IS DUAL ADDRESSING FOR THE
1794                                                                  ;ENTIRE WORD
1795  002460  012767  000020  000014          MOV     #20,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1796                                                                  ;******ERROR NUMBER #20******
1797  002466  105237  000301          9$:     INCB    @#$ADERR        ;ADDRESSING PROBLEM IS DETECTED
1798  002472  010046                          MOV     R0,-(SP)        ;SAVE R0
1799  002474  010200                          MOV     R2,R0           ;SET R0=GOOD ADDRESS FOR ERROR REPORT
1800  002476  004767  003114                  JSR     PC,ERROR        ;GO TO THE ERROR SUBROUTINE
1801  002502  000000                  22$:    .WORD                   ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1802  002504  012600                          MOV     (SP)+,R0        ;RESTORE R0
```

```
1803  002506  010011                        MOV     R0,(R1)         ;RESTORE (R1)
1804  002510  020037  000316                CMP     R0,@#BAKPAT     ;IF THE CONTROL CAME HERE FROM 15$-2 THEN
1805  002514  001411                        BEQ     12$
1806  002516  000407                        BR      11$             ;RETURN TO 11$
1807  002520  000300                10$:    SWAB    R0              ;MAKE R0 SAME AS SWAPPED BAKPAT
1808  002522  020011                        CMP     R0,(R1)         ;IF R1 = R2 THEN (R1) SHOULD CONTAIN A WORD
1809                                                                ;EQUAL TO SWAPPED R0
1810  002524  001404                        BEQ     11$             ;IN WHICH CASE GO BACK TO 11$
1811  002526  012767  000021  177746        MOV     #21,22$         ;*ERROR* SETUP ERROR NO. IN 22$
1812                                                                ;******ERROR NUMBER #21******
1813  002534  000745                        BR      8$              ;AND GO TO 8$
1814  002536  000300                11$:    SWAB    R0              ;RESTORE R0 TO BAKPAT
1815  002540  040301                12$:    BIC     R3,R1           ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS IN R1
1816  002542  005701                        TST     R1              ;IF R1 IS 0 THEN PLACE A 1 IN R1
1817  002544  001001                        BNE     13$             ;OTHERWISE GO TO 13$
1818  002546  005201                        INC     R1
1819  002550  006101                13$:    ROL     R1
1820  002552  020127  020000                CMP     R1,#20000       ;IF R1 IS LESS THAN A 4K BOUNDRY
1821  002556  103720                        BLO     7$              ;THEN REPEAT FROM 7$
1822  002560  000312                15$:    SWAB    (R2)            ;RESTORE (R2) TO BAKPAT
1823  002562  040302                16$:    BIC     R3,R2           ;TAKE OUT THE BANK ADDRESS FROM THE ADDRESS
1824                                                                ;STORED IN R2
1825  002564  005702                        TST     R2              ;IF R2 = 0 THEN MOVE A 1 TO R2
1826  002566  001001                        BNE     18$             ;OTHERWISE GO TO 18$
1827  002570  005202                        INC     R2
1828  002572  006102                18$:    ROL     R2              ;SHIFT A ONE IN THE ADDRESS WORD
1829  002574  020227  020000                CMP     R2,#20000       ;IS THE ADDRESS IN R2 MORE THAN THE BOUNDRY
1830                                                                ;OF 4K
1831  002600  103700                        BLO     6$              ;IF NOT THEN GO TO 6$
1832  002602  060203                        ADD     R2,R3           ;OTHERWISE MAKE R3 POINT TO THE NEXT 4K BANK
1833  002604  020337  000340                CMP     R3,@#MAXMEM     ;IF R3 IS POINTING TO A BANK THAT IS LOWER
1834                                                                ;THAN MAXMEM
1835  002610  103673                        BLO     4$              ;THEN REPEAT FROM 4$
1836  002612  000337  000316        20$:    SWAB    @#BAKPAT
1837  002616  001656                        BEQ     TST3            ;REPEAT THE TEST WITH SWAPPED BAKPAT ONLY IF
1838                                                                ;THE LOWER BYTE OF BAKPAT IS 0
1839  002620  000654                END3:   BR      END2
```

```
1840                                    ;;*******************************************************************
1841                                    ;*TEST 4         DUAL ADDRESS TEST B
1842                                            ;*(1)   THIS TEST CHECKS FOR DUAL ADDRESSING BY WRITING
1843                                            ;*      AND READING THE ADDRESS IN THE LOCATION AND THEN
1844                                            ;*      WRITING AND READING ADDRESS COMPLEMENT
1845                                    ;;*******************************************************************
1846  002622  122737  000004  000404    TST4:   CMPB    #4,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE
1847  002630  001403                            BEQ     .+10
1848  002632  004767  003316                    JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1849  002636  000022                            22                      ;******ERROR NUMBER 22******
1850
1851  002640  005003                            CLR     R3
1852  002642  010100                    1$:     MOV     R1,R0
1853  002644  005703                            TST     R3              ;IF R3 IS NOT 0 THEN STORE THE ADDRESS
1854  002646  001401                            BEQ     2$              ;IN THE LOCATION
1855  002650  005100                            COM     R0              ;OTHERWISE STORE COMPLEMENT
1856  002652  010021                    2$:     MOV     R0,(R1)+        ;OF THE ADDRESS
1857  002654  020105                            CMP     R1,R5           ;UNTIL THE HIGHEST MEMORY LOCATION IS REACHED
1858  002656  103771                            BLO     1$
1859
1860  002660  020041                    3$:     CMP     R0,-(R1)        ;CHECK THE LOCATION FOR THE CORRECT CONTENTS
1861  002662  001405                            BEQ     4$
1862  002664  105237  000301                    INCB    @#$ADERR        ;THIS IS PROBABLY ADDRESS PROBLEM RATHER THAN
1863                                            ;BIT PROBLEM
1864  002670  004767  002722                    JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1865  002674  000023                            23                      ;******ERROR NUMBER 23******
1866
1867  002676  010100                    4$:     MOV     R1,R0
1868  002700  162700  000002                    SUB     #2,R0           ;CHECK THAT THE ADDRESS IS STORED AT
1869  002704  005703                            TST     R3              ;LOCATION IF R3 IS NOT 0
1870  002706  001401                            BEQ     5$              ;OTHERWISE CHECK FOR
1871  002710  005100                            COM     R0              ;ADDRESS COMPLEMENT
1872  002712  020104                    5$:     CMP     R1,R4
1873  002714  101361                            BHI     3$
1874  002716  112737  000001  000306            MOVB    #1,@#PASFLG     ;SET PASFLG FOR ERROR REPORT.
1875  002724  005103                            COM     R3              ;COMPLEMENT THE CONTENTS OF R3
1876  002726  001345                            BNE     1$              ;REPEAT TST3 IF R3, IS NON 0, ENABLING ADDRESS
1877                                            ;COMPLEMENT TO BE WRITTEN AND READ, OTHERWISE FALL THRU
1878  002730  000733                    END4:   BR      END3
1879
```

```
1880                                    ;;*****************************************************************
1881                                    ;*TEST 5         MARCHING 1'S AND 0'S
1882                                            ;*(1)   THIS TEST WRITES A BACK GROUND OF THE WORD STORED
1883                                            ;*      AT BAKPAT.
1884                                            ;*(2)   READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
1885                                            ;*      AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
1886                                            ;*      DIRECTION OF MEMORY LOCATIONS.
1887                                            ;*(3)   READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
1888                                            ;*      WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
1889                                            ;*      IN  MIN. TO MAX. DIRECTION
1890                                            ;*(4)   REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
1891                                            ;*(5)   REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION
1892
1893                                    ;;*****************************************************************
1894  002732  122737  000005  000404  TST5:   CMPB    #5,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE
1895
1896  002740  001403                          BEQ     .+10
1897  002742  004767  003206                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
1898  002746  000024                          24                      ;******ERROR NUMBER 24******
1899
1900  002750  004737  000120          1$:     JSR     PC,@#WRTMEM     ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
1901                                                                  ;WORD STORED IN BAKPAT
1902  002754  020041                  2$:     CMP     R0,-(R1)        ;READ THE CONTENTS OF LOCATION POINTED BY R1
1903  002756  001403                          BEQ     3$              ;TO SEE IF IT HAS THE SAME VALUE AS R0
1904  002760  004767  002632                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1905  002764  000025                          25                      ;******ERROR NUMBER 25******
1906
1907  002766  000300                  3$:     SWAB    R0
1908  002770  010011                          MOV     R0,(R1)         ;SWAP THE BYTES AT (R1)
1909  002772  021100                          CMP     (R1),R0         ;READ (R1) FOR CORRECT VALUE
1910  002774  001403                          BEQ     4$
1911  002776  004767  002614                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1912  003002  000026                          26                      ;******ERROR NUMBER 26******
1913
1914
1915  003004  000300                  4$:     SWAB    R0              ;SWAP THE BYTES OF THE REGISTER
1916                                                                  ;CONTAINING BACKGROUND PATTERN
1917  003006  001023                          BNE     9$              ;IF THE LOWER BYTE OF THE REGISTER
1918                                                                  ;IS NOT 0 THEN THE PROGRAM IS READING
1919                                                                  ;THE MEMORY TO CONTAIN A BACK GROUND OF
1920
1921                                                                  ;BAKPAT AND WRITING THE SWAPPED WORD
1922
1923                                                                  ;IN WHICH CASE GO TO 9$
1924
1925  003010  005703                  5$:     TST     R3              ;R3 WAS 0 WHEN THE PROGRAM ENTERED
1926                                                                  ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
1927                                                                  ;IF R3 EQUAL 0 THEN THE PROGRAM IS
1928                                                                  ;READING/WRITING MIN. TO MAX. OTHERWISE
1929                                                                  ;IT IS GOING IN MAX. TO MIN. DIRECTION
1930  003012  001023                          BNE     10$             ;IF R3 IS NOT CLEAR THEN GO TO 10$
1931  003014  062701  000002          6$:     ADD     #2,R1           ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
1932  003020  020105                          CMP     R1,R5           ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
1933                                                                  ;BE TESTED
1934  003022  103006                          BHIS    8$              ;IF R1>R5 THEN GO TO 8$ OTHERWISE
1935  003024  020011                  7$:     CMP     R0,(R1)         ;READ (R1) FOR THE CORRECT DATA
```

```
1936  003026  001757                      BEQ     3$              ; WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
1937                                                              ;AND REPEAT UNTIL R1 > R5
1938  003030  004767  002562              JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
1939  003034  000027                      27                      ;******ERROR NUMBER 27******
1940
1941  003036  000753                      BR      3$
1942  003040  105237  000306      8$:     INCB    @#PASFLG
1943  003044  000300                      SWAB    R0
1944  003046  001742                      BEQ     2$              ;IF THE LOWER BYTE OF R0 IS ALL 0'S
1945                                                              ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
1946                                                              ;AND READING BAKPAT GOING FROM MAX. TO MIN.[PASFLG=4]
1947  003050  005103                      COM     R3              ;OTHERWISE CLEAR R0
1948  003052  010401                      MOV     R4,R1           ;PUT THE LOWEST TESTING ADDRESS IN R1
1949  003054  000763                      BR      7$              ;AND BEGIN READING 0'S, WRITING 1'S AND
1950                                                              ;READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]
1951
1952  003056  005703              9$:     TST     R3              ;IF R3 IS NON 0, I.E. PASFLG=3
1953  003060  001353                      BNE     5$              ;THEN READ BAKPAT, WRITE
1954                                                              ;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
1955                                                              ;IN MIN. TO MAX. DIRECTION
1956  003062  020104              10$:    CMP     R1,R4           ;OTHERWISE TEST IS PROCEEDING IN MAX. TO
1957                                                              ;MIN. DIRECTION.
1958  003064  101333                      BHI     2$              ;KEEP ON LOOPING UNTIL R1=R4
1959  003066  105237  000306              INCB    @#PASFLG
1960  003072  000300                      SWAB    R0
1961  003074  001753                      BEQ     7$              ;IF R0 SWAPPED HAS LOWER BYTE=0
1962                                                              ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
1963                                                              ;AND READ BAKPAT GOING FROM MIN. TO MAX.
1964  003076  000714              END5:   BR      END4
1965
```

```
                                ;;*****************************************************************
                                ;*TEST 6         CELLS' VOLATILITY TEST
                                ;
                                        ;*(1)   THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
                                        ;*(2)   WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
                                        ;*      AND THEN INCREMENTS PASFLG
                                        ;*(3)   IT THEN READS/SWAPS BYTES/WRITES A LOCATION X FOR
                                        ;*      OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
                                        ;*(4)   REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
                                        ;*(5)   IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
                                        ;*      BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
                                        ;*      SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
                                        ;*(6)   REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
                                        ;*      BAKPAT INSTEAD OF BAKPAT.
                                ;;*****************************************************************
003100  122737  000006  000404  TST6:  CMPB    #6,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE
003106  001403                        BEQ     .+10
003110  004767  003040                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
003114  000030                        30                      ;******ERROR NUMBER 30******
003116  004737  000120          RPT6:  JSR     PC,@#WRTMEM     ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
                                                              ;WORD STORED AT LOCATION BAKPAT
003122  005037  000306                CLR     @#PASFLG
003126  010403                  1$:   MOV     R4,R3           ;SET R3
003130  010401                  2$:   MOV     R4,R1           ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
003132  020011                  3$:   CMP     R0,(R1)         ;CHECK (R1) FOR CORRECT DATA
003134  001403                        BEQ     4$
003136  004767  002454                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
003142  000031                        31                      ;******ERROR NUMBER 31******
003144  062701  000002            4$:   ADD     #2,R1           ;INCREMENT R1 BY 2
003150  020105                        CMP     R1,R5           ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
003152  103767                        BLO     3$
003154  132737  000001  000306        BITB    #1,@#PASFLG     ;CHECK TO SEE IF PASFLG=0 OR 2
003162  001002                        BNE     5$
003164  105237  000306                INCB    @#PASFLG        ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
003170  020305                  5$:   CMP     R3,R5           ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
003172  103012                        BHIS    7$
003174  012702  037776                MOV     #37776,R2       ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
003200  000313                  6$:   SWAB    (R3)
003202  005302                        DEC     R2
003204  001375                        BNE     6$
003206  010337  000354                MOV     R3,@#SAVLOC     ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
003212  062703  020000                ADD     #20000,R3       ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
                                                              ;R3 TO POINT TO A LOCATION IN THE NEXT
                                                              ;4K BANK OF MEMORY
003216  000744                        BR      2$
003220  105237  000306            7$:   INCB    @#PASFLG        ;MAKE PASFLG=2
003224  000337  000316                SWAB    @#BAKPAT        ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
003230  001732                        BEQ     RPT6            ;THEN GO BACK TO THE LOCATION RPT6
003232  000721                  END6: BR      END5
```

```
                                ;;*****************************************************************
                                ;*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
                                ;;*****************************************************************
003234  122737  000007  000404  TST7:   CMPB    #7,@#$TESTN     ;CHECK FOR PROPER TEST SEQUENCE

003242  001403                        BEQ     .+10
003244  004767  002704                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
003250  000032                        32                      ;******ERROR NUMBER 32******

003252  005037  000306          2$:   CLR     @#PASFLG
003256  010337  000304                MOV     R3,@#LOWBNK     ;LOWBNK CONTAINS ADDRESS OF THE LOWEST LOCATION
                                                              ;IN THE 4K BANK THAT CAN BE TESTED
003262  010302                        MOV     R3,R2
003264  052702  017777                BIS     #17777,R2       ;R2 CONTAINS THE ADDRESS OF THE TOP OF THE BANK
003270  005202                        INC     R2              ;ADD 1 TO POINT IT TO NEXT BANK
003272  001402                        BEQ     3$              ;BRANCH IF ZERO (IT MUST BE A 30K SYSTEM)
003274  020502                        CMP     R5,R2
003276  103001                        BHIS    4$              ;IF R2 IS GREATER THAN R5 THEN GO TO 4$
003300  010502                3$:     MOV     R5,R2           ;NOW R2 CONTAINS THE ADDRESS OF THE HIGHEST LOCATION
                                                              ;THAT CAN BE TESTED
003302  010337  000302          4$:   MOV     R3,@#STRTDI     ;LOAD STRTDI WITH THE STARTING ADDRESS OF THE
                                                              ;DIAGONAL
003306  013701  000304                MOV     @#LOWBNK,R1     ;R1 IS NOW POINTING TO THE LOWEST LOCATION IN THE 4K
                                                              ;BANK
003312  013700  000316          6$:   MOV     @#BAKPAT,R0     ;STORE THE CONTENTS OF BAKPAT IN R0
003316  020103                        CMP     R1,R3           ;IS R1 POINTING TO A LOCATION IN THE DIAGONAL ?
003320  001010                        BNE     10$             ;IF NOT THEN GO TO 10$
003322  062703  000002                ADD     #2,R3           ;THE FOLLOWING CODE IS USED TO PLACE THE
003326  032703  000176                BIT     #176,R3         ;ADDRESS OF THE NEXT LOCATION IN THE DIAGONAL
003332  001402                        BEQ     8$              ;IN R3
003334  062703  000200                ADD     #200,R3         ;
003340  000300                  8$:   SWAB    R0              ;DIAGONAL WILL CONTAIN SWAPPED BACKGROUND PATTERN
003342  132737  000001  000306  10$:   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
003350  001001                        BNE     12$             ;IF IT IS BEING READ ONLY THEN GO TO 12$
003352  010011                        MOV     R0,(R1)         ;OTHERWISE WRITE THE MEMORY WITH THE CONTENTS
                                                              ;OF R0
003354  020011                 12$:   CMP     R0,(R1)         ;CHECK THE LOCATION POINTED BY R1 TO CONTAIN
                                                              ;PROPER DATA
003356  001403                        BEQ     14$             ;IF IT IS OK THEN GO TO 14$
003360  004767  002232                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
003364  000033                        33                      ;******ERROR NUMBER 33******

003366  062701  000002          14$:  ADD     #2,R1           ;CAUSE R1 TO POINT TO THE NEXT MEMORY LOCATION
003372  020102                        CMP     R1,R2           ;IS IT THE END OF THE BANK ?
003374  103746                        BLO     6$              ;IF NOT THEN GO TO 6$
003376  005237  000410          16$:  INC     @#$DEVCT                ;TELL APT WE ARE STIL RUNNING OKAY
```

```
2077  003402  105237  000306              INCB    @#PASFLG
2078  003406  013703  000302              MOV     @#STRTDI,R3     ;LOAD R3 WITH THE STARTING ADDRESS OF THE DIAGONAL
2079  003412  132737  000001  000306      BITB    #1,@#PASFLG     ;HAS THE READ OF THE MEMORY BEEN DONE ?
2080  003420  001330                      BNE     4$              ;IF NOT THEN GO TO 4$
2081  003422  005723                      TST     (R3)+           ;ADD 2 TO THE STARTING ADDRESS OF THE DIAGONAL
2082  003424  020302                      CMP     R3,R2           ;AND UNLESS THE END OF THE BANK IS REACHED
2083  003426  103003                      BHIS    18$             ;
2084  003430  105737  000306              TSTB    @#PASFLG        ;OR THE DIAGONAL HAS BEEN ROTATED 64 TIMES
2085  003434  100322                      BPL     4$              ;REPEAT FROM 4$
2086  003436  013703  000304      18$:    MOV     @#LOWBNK,R3     ;MAKE R3 POINT TO THE LOWEST LOCATION IN THE
2087                                                              ;IN THE BANK UNDER TEST
2088  003442  000337  000316              SWAB    @#BAKPAT
2089  003446  001715                      BEQ     4$              ;AND IF THE TEST HAS NOT BEEN PERFORMED WITH THE
2090                                                              ;SWAPPED BACK GROUND PATTERN THEN GO TO 4$
2091  003450  010203                      MOV     R2,R3           ;MAKE THE PRESENT HIGH BOUNDRY AS THE NEXT
2092                                                              ;LOW BOUNDRY
2093  003452  020205                      CMP     R2,R5           ;UNLESS THE PRESENT HIGH BOUNDRY IS ALSO THE
2094                                                              ;HIGH BOUNDRY FOR THE MEMORY UNDER TEST
2095  003454  103676                      BLO     2$
2096  003456  000665              END7:   BR      END6
```

```
                                ;;*****************************************************************
                                ;*TEST 10        READ RECOVERY GALLOPING TEST/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

                                ;;*****************************************************************
003460  122737  000010  000404  TST10:  CMPB    #10,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE

003466  001403                        BEQ     .+10
003470  004767  002460                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
003474  000034                        34                      ;******ERROR NUMBER 34******

003476  010402                        MOV     R4,R2           ;SET R2 TO THE LOWEST MEMORY UNDER TEST
003500  052702  017776        RPT10:  BIS     #17776,R2       ;MAKE R2 POINT TO THE HIGHEST LOCATION IN THE 4K
                                                              ;BANK FOR WHICH GALLOPING WILL BE PERFOMED
003504  062702  000002        GALLOP: ADD     #2,R2           ;INCREMENT R2 BY 2
003510  001402                        BEQ     1$              ;BR IF IT WENT TO 0 (IT MUST BE A 30K SYSTEM)
003512  020205                        CMP     R2,R5           ;IF THE HIGH BOUNDRY OF THE TEST IS HIGHER THAN
003514  101401                        BLOS    2$
003516  010502                1$:     MOV     R5,R2           ;THE MAXIMUM ALLOWED ADDRESS THEN ADJUST R2
003520  005046                2$:     CLR     -(SP)
003522  010200                        MOV     R2,R0
003524  013740  000316        4$:     MOV     @#BAKPAT,-(R0)  ;WRITE THE MEMORY UNDER TEST WITH A BACKGROUND OF
                                                              ;BAKPAT
003530  020003                        CMP     R0,R3
003532  101374                        BHI     4$
003534  010301                6$:     MOV     R3,R1           ;R3 AND R1 ARE POINTING TO THE LOWEST LOCATION THAT
                                                              ;CAN BE TESTED IN THIS BLOCK
003536  023710  000316                CMP     @#BAKPAT,(R0)   ;BEFORE STARTING THE GALLOPING TEST FOR LOCATION
                                                              ;(R0) CHECK IT
003542  001410                        BEQ     8$              ;CONTINUE IF OK
003544  010001                        MOV     R0,R1           ;OTHERWISE PREPARE TO REPORT THE ERROR
003546  013700  000316                MOV     @#BAKPAT,R0
003552  004767  002040                JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
003556  000035                        35                      ;******ERROR NUMBER 35******
```

```
2153
2154  003560  010011                          MOV     R0,(R1;         ;RESTORE THE CONTENTS OF (R1)
2155  003562  010100                          MOV     R1,R0           ;RESTORE R0
2156
2157  003564  000310                  8$:     SWAB    (R0)
2158  003566  031011                  10$:    BIT     (R0),(R1)       ;CHECK TO SEE THAT NONE OF THE BITS SET
2159                                                                  ;IN (R0) ARE SET IN (R1) AND VICE VERSA
2160  003570  020001                          CMP     R0,R1           ;THE ONLY EXCEPTION TO THIS WILL BE WHEN R0=R1
2161
2162  003572  001412                          BEQ     12$
2163  003574  021137  000316                  CMP     (R1),@#BAKPAT   ;CHECK THAT (R1) HAS BAKPAT IN IT
2164  003600  001407                          BEQ     12$
2165  003602  010046                          MOV     R0,-(SP)        ;SAVE R0 ON STACK
2166  003604  013700  000316                  MOV     @#BAKPAT,R0     ;PLACE THE PATTERN WORD IN R0
2167  003610  004767  002002                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2168  003614  000036                          36                      ;******ERROR NUMBER 36******
2169
2170  003616  012600                          MOV     (SP)+,R0        ;RESTORE R0
2171  003620  021037  000320          12$:    CMP     (R0),@#SWAPAT   ;CHECK THAT (R0) HAS SWAPPED BAKPAT IN IT
2172  003624  001412                          BEQ     14$
2173  003626  010146                          MOV     R1,-(SP)        ;SAVE R1 ON THE STACK
2174  003630  010001                          MOV     R0,R1           ;MAKE R1 POINT TO THE FAILING LOCATION
2175  003632  013700  000320                  MOV     @#SWAPAT,R0     ;LOAD R0 WITH THE EXPECTED RESULT IN (R1)
2176  003636  004767  001754                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2177  003642  000037                          37                      ;******ERROR NUMBER 37******
2178
2179  003644  010011                          MOV     R0,(R1)         ;RECOVER (R1) FROM THE ERROR
2180  003646  010100                          MOV     R1,R0           ;RESTORE R0
2181  003650  012601                          MOV     (SP)+,R1        ;AND RESTORE R1
2182  003652  122737  000011  000404  14$:    CMPB    #11,@#$TESTN    ;IS THE PROGRAM EXECUTING TEST # 11 ?
2183  003660  001402                          BEQ     16$             ;IF SO THEN GO TO 16$
2184  003662  062701  000176                  ADD     #176,R1
2185  003666  062701  000002          16$:    ADD     #2,R1           ;MAKE R1 POINT TO THE NEXT ADJACENT CELL
2186  003672  020102                          CMP     R1,R2           ;AND IF R1 HAS NOT REACHED THE END OF THE BOUNDRY
2187  003674  103734                          BLO     10$             ;THEN REPEAT FROM 10$
2188  003676  000320                          SWAB    (R0)+           ;RESTORE THE LOCATION FOR WHICH THE GALLOPING TEST
2189                                                                  ;WAS BEING PERFORMED
2190  003700  122737  000011  000404          CMPB    #11,@#$TESTN    ;IS IT TEST 11 ?
2191  003706  001407                          BEQ     17$             ;IF SO THEN GO TO 17$
2192  003710  005723                          TST     (R3)+           ;OTHERWISE INCREMENT R3 BY 2
2193  003712  062716  000002                  ADD     #2,(SP)         ;FOR EVERY ROW/COLUMN TESTED ADD 2
2194  003716  105716                          TSTB    (SP)
2195  003720  100002                          BPL     17$             ;UNTIL (SP) IS 200
2196  003722  161603                          SUB     (SP),R3         ;SUBTRACT 200 FROM R3
2197  003724  005016                          CLR     (SP)
2198  003726  032700  000177          17$:    BIT     #177,R0         ;AT A 64TH CALL BOUNDRY?
2199  003732  001002                          BNE     18$             ;BRANCH IF NO
2200  003734  005237  000410                  INC     @#$DEVCT        ;TELL APT WE ARE STILL RUNNING
2201  003740  020002                  18$:    CMP     R0,R2           ;IF R0 HAS NOT REACHED THE END OF THE BOUNDRY
2202  003742  103674                          BLO     6$              ;THEN REPEAT FROM 6$
2203  003744  162603                          SUB     (SP)+,R3        ;RESTORE SP AND R3
2204  003746  000337  000320                  SWAB    @#SWAPAT
2205  003752  000337  000316                  SWAB    @#BAKPAT
2206  003756  001660                          BEQ     2$              ;IF THE LOWER BYTE OF BAKPAT IS 0 THEN REPEAT FROM 2$
2207  003760  010203                          MOV     R2,R3           ;OTHERWISE MAKE THE PRESENT HIGH BOUNDRY AS THE
2208                                                                  ;NEXT LOW BOUNDRY
```

```
2209  003762  020205                          CMP     R2,R5
2210  003764  001410                          BEQ     END10           ;IF PREVIOUS HIGH BOUNDRY WAS THE END OF THE
2211                                                                  ;TEST BOUNDRY THEN EXIT THE TEST
2212  003766  032702  017776                   BIT     #17776,R2       ;WAS IT A 4K BOUNDRY ?
2213  003772  001025                          BNE     RPT11           ;IF NOT THEN WE WERE PERFORMING TEST 11 WITH LONG
2214                                                                  ;GALLOPING TEST DISABLED
2215  003774  122737  000011  000404          CMPB    #11,@#$TESTN    ;IF IT IS TEST # 11 THEN GO TO REPEAT TEST 11
2216  004002  001421                          BEQ     RPT11
2217  004004  000635                          BR      RPT10           ;OTHERWISE REPEAT TEST 10
2218  004006  000623                  END10:  BR      END7
2219
2220
2221
```

```
                                ;;*****************************************************************
                                ;*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 IT '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

                                ;;*****************************************************************
004010  122737  000011  000404  TST11:  CMPB    #11,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE

004016  001403                        BEQ     .+10
004020  004767  002130                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
004024  000040                        40                      ;******ERROR NUMBER 40******

004026  010402                        MOV     R4,R2           ;MAKE R2 TO POINT TO THE LOWEST LOCATION
                                                              ;UNDER TEST
004030  105777  174414                TSTB    @SWR            ;LONG GALLOP ENABLED?
004034  100004                        BPL     RPT11           ;BRANCH IF NO
004036  004767  002540                JSR     PC,PNTMES       ;TYPE ''GLP''
004042  046107  000120                .ASCIZ  /GLP/
004046  105777  174376        RPT11:  TSTB    @SWR            ;LONG GALLOPING ENABLED?
004052  100612                        BMI     RPT10           ;BRANCH IF YES
                                                              ;TO RPT10
004054  052702  000176                BIS     #176,R2         ;OTHERWISE SET THE LOW ORDER BITS OF THE ADDRESS
                                                              ;TO GET THE HIGH BOUNDRY

004060  000611                        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.
                                ;;*****************************************************************
004062  122737  000012  000404  TST12: CMPB    #12,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE
004070  001403                        BEQ     .+10
004072  004767  002056                JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
004076  000041                        41                      ;******ERROR NUMBER 41******

004100  012702  000002                MOV     #2,R2           ;PREPARE TO TAKE THE EXCLUSIVE OR OF ADDRESS BITS 1
004104  012703  000400                MOV     #400,R3         ;AND 8
004110  112737  000001  000306  1$:   MOVB    #1,@#PASFLG     ;INITIALIZE THE COUNTER FOR THE SUBTEST
004116  010401                2$:     MOV     R4,R1           ;PLACE THE STARTING ADDRESS OF MEMORY UNDER
                                                                ;TEST IN R1
004120  013700  000316        4$:     MOV     @#BAKPAT,R0
004124  030201                        BIT     R2,R1           ;CHECK TO SEE IF ADDRESS BIT STORED IN R2 IS SET
004126  001004                        BNE     8$              ;IF IT IS SET THEN GO TO 8$
004130  030301                        BIT     R3,R1           ;CHECK TO SEE IF ADDRESS BIT POINTED BY R3 IS SET
004132  001404                        BEQ     12$             ;IF IT IS NOT SET THEN GO TO 12$
004134  005100                6$:     COM     R0              ;COME HERE ONLY IF EXCLUSIVE OR OF ADDRESS BITS
                                                                ;POINTED BY R2 & POINTED BY R3 = 1 IN WHICH
                                                                ;CASE PREPARE TO WRITE THE LOCATION
                                                                ;WITH A COMPLEMENT OF LOCATIONS NOT MEETING
                                                                ;THIS CONDITION
004136  000402                        BR      12$
004140  030301                8$:     BIT     R3,R1           ;COME HERE IF ADDRESS BIT POINTED BY R2 IS 1 AND
                                                                ;CHECK ADDRESS BIT POINTED BY R3
004142  001774                        BEQ     6$              ;IF ADDRESS BIT POINTED BY R3 IS 0 THEN GO TO 6$
004144  132737  000002  000306  12$:  BITB    #2,@#PASFLG     ;IS IT 2ND OR 3RD PASS OF THE SUBTEST ?
004152  001001                        BNE     14$             ;IF SO THEN READ THE MEMORY
```

```
2318  004154  010011                          MOV     R0,(R1)         ;OTHERWISE WRITE THE MEMORY BFORE READING IT
2319  004156  020011                  14$:    CMP     R0,(R1)         ;READ THE MEMORY FOR CORRECT CONTENTS
2320  004160  001403                          BEQ     16$
2321  004162  004767  001430                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2322  004166  000042                          42                      ;******ERROR NUMBER 42******
2323
2324  004170  012746  000002          16$:    MOV     #2,-(SP)
2325  004174  005100                  18$:    COM     R0
2326  004176  005111                          COM     (R1)
2327  004200  020011                          CMP     R0,(R1)         ;READ THE MEMORY AGAIN
2328  004202  001404                          BEQ     19$
2329  004204  004767  001406                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2330  004210  000043                          43                      ;******ERROR NUMBER 43******
2331
2332  004212  010011                          MOV     R0,(R1)         ;RESTORE THE LOCATION (R1)
2333  004214  005316                  19$:    DEC     (SP)
2334  004216  001366                          BNE     18$             ;EXECUTE THE CODE FROM 18$ TWICE
2335  004220  005726                          TST     (SP)+           ;RESTORE THE STACK POINTER
2336  004222  122737  000003  000306          CMPB    #3,@#PASFLG     ;IS IT THE 3RD PASS OF THE SUBTEST ?
2337  004230  001412                          BEQ     20$             ;IF SO THEN GO TO 20$
2338  004232  062701  000002                  ADD     #2,R1           ;IN FIRST 2 PASSES THE PROGRAM PROCEEDS IN
2339                                                                  ;MIN. TO MAX. DIRECTION
2340  004236  020105                          CMP     R1,R5           ;HAVE WE REACHED THE MAX. ADDRESS UNDER TEST ?
2341  004240  103727                          BLO     4$              ;IF NOT THEN REPEAT FROM 4$
2342  004242  105237  000306                  INCB    @#PASFLG
2343  004246  122737  000002  000306          CMPB    #2,@#PASFLG     ;IF IT IS THE 2ND PASS OF THE SUBTEST
2344  004254  001720                          BEQ     2$              ;THEN REPEAT FROM 2$
2345  004256  162701  000002          20$:    SUB     #2,R1           ;OTHERWISE EXECUTE THE TEST IN MAX. TO MIN.
2346                                                                  ;DIRECTION
2347  004262  020104                          CMP     R1,R4           ;HAVE WE REACHED THE MIN. ADDRESS UNDER TEST ?
2348  004264  103315                          BHIS    4$              ;IF NOT THEN REPEAT FROM 4$
2349  004266  012702  020000                  MOV     #20000,R2       ;PREPARE TO CHECK THE MEMORY WITH THE XOR OF
2350                                                                  ;ADDRESS BITS 8 AND 13
2351  004272  105237  000307                  INCB    @#PASFLG+1      ;THE SUB TEST HAS CHECKED THE XOR ONE KIND
2352  004276  123727  000307  000002          CMPB    @#PASFLG+1,#2   ;HAS TWO XOR COMBINATIONS BEEN CHECKED ?
2353  004304  103701                          BLO     1$              ;IF NOT THEN GO TO 1$
2354  004306  101004                          BHI     22$             ;IF ALL THREE HAVE BEEN CHECKED THEN GO TO 22$
2355  004310  012702  000010                  MOV     #10,R2          ;IF IT IS THE 2ND XOR COMBINATION THEN CHECK
2356  004314  006303                          ASL     R3              ;FOR ADDRESS BITS 3 & 8
2357  004316  000674                          BR      1$
2358  004320  005137  000316          22$:    COM     @#BAKPAT        ;IF THE TEST WAS NOT PERFORMED WITH THE SWAPPED
2359  004324  105737  000316                  TSTB    @#BAKPAT
2360  004330  001654                          BEQ     TST12           ;BAKPAT THEN RE-EXECUTE THE TEST
2361  004332  000625                  END12:  BR      END10
```

```
                        ;;*****************************************************************
                        ;*TEST 13       WRITE RECOVERY TEST
                        ;*      THIS TEST DIFFERS FROM 0-12 IN THAT IT CONSISTS OF A SMALL TEST PROGRAM
                        ;*      ACTUALLY RUNNING IN THE 4K BANK UNDER TEST.
                        ;*              THE PROGRAM IS SELF MODIFYING AND MAY BE DIFFICULT TO DEBUG.
                        ;*      TO AID IN THE DEBUG, BEFORE A A BANK IS ENTERED 'TST13 BANK XX''
                        ;*      IS TYPED. THIS WILL ALLOW THE USER TO AT LEAST SEE WHICH MEMORY
                        ;*      BANK FAILED.
                        ;*      THE TEST CONSISTS OF 1/2 OF THE BANK STORED WITH 'MOV R2,-(PC)''
                        ;*      AND THE OTHER 1/2 CONTAINING ''177667''. ''177667'' IS THE COMPLEMENT
                        ;*      OF ''JMP (R0)'' INSTRUCTION.
                        ;*      R2 CONTAINS ''COM -(R1)'' INSTRUCTION ON ENTRY TO THE BANK AND R1 CONTAINS
                        ;*      THE HIGHEST TEST ADDRESS IN THAT BANK. THE HIGHEST TEST ADDRESS IS
                        ;*      USUALLY ON 4K BOUNDARIES. WHEN TESTING BANK 0 RELOCATED,HOWEVER
                        ;*      R1 CONTAINS THE FIRST FREE TEST ADDRESS BELOW THE DIAGNOSTIC.
                        ;*      IF YOU UNDERSTAND THIS SO FAR THE REST IS EASY.
                        ;*              THE TEST EXECUTION IS AS FOLLOWS:
                        ;*              1. THE 'MOV R2,-(PC)'' INSTRUCTION EXECUTES STORING
                        ;*                THE CONTENTS OF R2 IN THE ADDRESS IT VACATED (DUE TO -(PC).
                        ;*              2. SINCE R2 CONTAINS A ''COM -(R1)'' INSTRUCTION IT COMPLEMENTS
                        ;*                THE HIGHEST ADDRESS UNDER TEST. THIS ADDRESS CONTAINED
                        ;*                ''177667'' SO AFTER THE COM -(R1) IT EQUALS 110
                        ;*                CLEVERLY THIS IS THE ''JMP (R0)'' INSTRUCTION.
                        ;*              3. THIS SEQUENCE CONTINUES UNTIL THE 'MOV R2,-(PC) INSTRUCTIONS
                        ;*                REACH THE MIDDLE OF THE TEST BANK. THEN THE ''JMP (R0)'' INSTRUCTION IS
                        ;*                AND EXECUTED. R0 CONTAINED THE RETURN ADDRESS BACK
                        ;*                TO TEST 13.
                        ;*              4. THESE STEPS ARE REPEATED FOR EACH BANK UNDER TEST.
                        ;*
                        ;;*****************************************************************
004334  122737  000013  000404  TST13:  CMPB    #13,@#$TESTN    ;CHECK FOR PROPER TEST SEQUENCE
004342  001403                          BEQ     .+10
004344  004767  001604                  JSR     PC,SEQERR       ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
004350  000044                          44                      ;******ERROR NUMBER 44*******

004352  012702  010247          1$:     MOV     #10247,R2       ;PLACE THE OP CODE OF INSTRUCTION MOV R2,-(PC)
                                                                ;IN R2.
004356  012700  177667                  MOV     #177667,R0      ;PLACE THE COMPLEMENT OF THE INSTRUCTION
                                                                ;JMP (R0) IN R0
                        ;INSURE LOWEST TEST ADDRESS TO END OF 4K SEGMENT IS MULTIPLE OF 2
                        ;SINCE THE TEST STORES 'MOV R2,-(PC) IN 1/2 AND 177667 IN THE OTHER 1/2

004362  010546          2$:     MOV     R5,-(SP)        ;SAVE R5
004364  010446                  MOV     R4,-(SP)        ;STORE LOWEST ADDRESS ON STACK
004366  000241          29$:    CLC
004370  006005                  ROR     R5              ;MAKE POSITIVE BYTE COUNT OF HIGH ADDRESS
004372  006004                  ROR     R4              ;DO SAME FOR LOWEST ADDRESS
004374  160405                  SUB     R4,R5           ;GET DIFFERENCE OF LOWEST ADDRESS AND HIGHEST
004376  006005                  ROR     R5              ;IF DIFFERENCE IS ODD THEN R4 IS AT LOWEST ADDRESS
004400  103002                  BCC     30$             ;BRANCH IF R4 IS AT LOWEST TEST ADDRESS.
004402  062716  000002          ADD     #2,(SP)         ;INCREASE LOWEST TEST ADDRESS BY 2
004406  012604          30$:    MOV     (SP)+,R4        ;RESTORE R4 (POSSIBLY INCREASED BY 2 FROM ENTRY)
004410  012605                  MOV     (SP)+,R5        ;RESTORE HIGHEST TEST ADDRESS
004412  010403                  MOV     R4,R3           ;PLACE THE LOWEST LOCATION UNDER TEST
                                                        ;IN R3
004414  000406                  BR      28$             ;LEAVE LOW BITS OF R3 ALONE FIRST TIME IN CASE BANK 0
```

```
2418  004416  042703  017776          3$:     BIC     #17776,R3       ;CAUSE R3 TO POINT TO THE LOWEST LOCATION
2419                                                                    ;IN THE 4K BANK UNDER TEST
2420  004422  001507                          BEQ     14$             ;IF ADDRESS WENT TO 0, IT MUST BE A 30K SYSTEM
2421  004424  105737  000405                  TSTB    @#REL           ;ARE WE RELOCATED?
2422  004430  100504                          BMI     14$             ;BRANCH IF YES-TEST BANK0 ONLY-
2423  004432  020305                  28$:    CMP     R3,R5           ;IF R3 IS HIGHER THAN THE HIGHEST LOCATION
2424  004434  103102                          BHIS    14$             ;UNDER TEST THEN EXIT
2425                                  ;IF R5 LESS THAN 20000 THEN WE ARE TESTING BANK0 RELOCATED IN BANK0
2426  004436  020527  020000                  CMP     R5,#20000       ;IS HIGHEST TEST ADDRESS BELOW 4K?
2427  004442  103002                          BHIS    31$             ;BRANCH IF NO
2428  004444  010501                          MOV     R5,R1           ;SET R1 TO HIGHEST TEST ADDRESS IN BANK0
2429  004446  000407                          BR      32$
2430
2431  004450  010301                  31$:    MOV     R3,R1           ;SET R1 TO LOWEST CURRENT TEST ADDRESS
2432  004452  052701  017777                  BIS     #17777,R1       ;SET LOW ORDER ADDRESS BITS
2433  004456  005201                          INC     R1              ;CAUSE R1 TO POINT TO THE HIGHEST LOCATION+2
2434                                                                    ;OF THE 4K BANK BEING POINTED BY R3
2435  004460  001402                          BEQ     32$             ;BRANCH IF R1 WENT TO 0 (WHICH MIGHT
2436                                                                    ;HAVE HAPPENED IF TESTING A 30K LSI SYSTEM)
2437  004462  020105                          CMP     R1,R5           ;COMPARE R1 TO HIGHEST ADDRESS UNDER TEST
2438  004464  101401                          BLOS    33$             ;BRANCH IF WITHIN RANGE
2439  004466  010501                  32$:    MOV     R5,R1           ;SET R1 TO THE MAXIMUM AVAILABLE MEMORY
2440
2441  004470  132737  000001  000306  33$:    BITB    #1,@#PASFLG     ;IS THE LOWEST BIT OF LOCATION PASFLG
2442  004476  001101                          BNE     16$             ;SET? IN WHICH CASE BACK GROUND HAS
2443                                                                    ;ALREADY BEEN WRITTEN AND WRITE RECOVERY
2444                                                                    ;TEST IS BEING PERFORMED
2445
2446  004500  020304                  4$:     CMP     R3,R4           ;OTHERWISE WRITE THE BACKGROUND
2447  004502  103430                          BLO     8$              ;DEFINED AT STEP 3.
2448  004504  105737  000307                  TSTB    @#PASFLG+1      ;IS THE TEST JUST DOING READ, I.E.
2449  004510  001002                          BNE     6$              ;IS THE PASFLG+1 LOCATION NON ZERO? IF SO
2450                                                                    ;THEN GO TO 6$
2451  004512  012713  010247                  MOV     #10247,(R3)     ;WRITE THE LOCATION WITH THE OP CODE FOR MOV R2,-(PC)
2452  004516  020213                  6$:     CMP     R2,(R3)         ;READ (R3) TO CONTAIN CORRECT DATA
2453  004520  001421                          BEQ     8$
2454  004522  010046                          MOV     R0,-(SP)        ;SAVE R0
2455  004524  010146                          MOV     R1,-(SP)        ;AND R1 ON THE STACK
2456  004526  010301                          MOV     R3,R1
2457  004530  010200                          MOV     R2,R0           ;SET R0= GOOD DATA FOR ERROR PRINTOUT
2458  004532  004767  001060                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2459  004536  000045                          45                      ;******ERROR NUMBER 45******
2460
2461  004540  012601                          MOV     (SP)+,R1        ;RESTORE R1
2462  004542  012600                          MOV     (SP)+,R0        ;AND R0
2463  004544  105737  000306                  TSTB    @#PASFLG        ;IF PASFLG IS 0 AND THE MEMORY DOES NOT HAVE
2464                                                                    ;THE PROPER DATA THEN WE DON'T WANT TO GO AND
2465                                                                    ;EXECUTE THE INSTRUCTIONS STORED IN MEMORY UNDER
2466                                                                    ;TEST
2467  004550  001005                          BNE     8$              ;BRANCH IF PASFLG NOT =0
2468
2469  004552  010200                          MOV     R2,R0   ;SAVE FOR ERROR REPORT
2470  004554  004767  001036                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2471  004560  000046                          46                      ;******ERROR NUMBER 46******
2472
2473  004562  000663                          BR      END12           ;ABORT TST 13.
```

```
2474
2475  004564  062703  000002          8$:     ADD     #2,R3           ;INCREMENT R3 BY 2
2476  004570  162701  000002                  SUB     #2,R1           ;DECREMENT R1 BY 2
2477  004574  020105                          CMP     R1,R5           ;WRITE THE BACKGROUND DEFINED AT STEP 4.
2478  004576  103014                          BHIS    12$
2479  004600  020103                          CMP     R1,R3           ;HAS STORING THE 177667 REACHED WHERE 'MOV R2,-(PC) IS?
2480  004602  103405                          BLO     10$             ;BRANCH IF YES DON'T DESTROY THE MOV R2,-(PC) IS.
2481  004604  105737  000307                  TSTB    @#PASFLG+1      ;IS THE THE READ ONLY CHECK PASS?
2482  004610  001002                          BNE     10$             ;BRANCH IF YES
2483  004612  012711  177667                  MOV     #177667,(R1)    ;WRITE THE LOCATION WITH THE COMPLEMENT OF THE
2484                                                                  ;OP CODE JMP (R0)
2485  004616  020011                  10$:    CMP     R0,(R1)         ;READ R1 TO CONTAIN CORRECT DATA
2486  004620  001403                          BEQ     12$
2487  004622  004767  000770                  JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2488  004626  000047                          47                      ;******ERROR NUMBER 47******
2489
2490  004630  020301                  12$:    CMP     R3,R1           ;IF WE HAVE NOT REACHED THE MIDDLE OF 4K BANK
2491  004632  103722                          BLO     4$              ;THEN REPEAT FROM 4$
2492
2493                                  ;RETURN HERE AFTER PROGRAM RUN IN BANK UNDER TEST
2494
2495  004634  062703  020000          13$:    ADD     #20000,R3       ;OTHERWISE GO TO THE NEXT 4K BANK
2496  004640  000666                          BR      3$
2497
2498  004642  122737  000001  000306  14$:    CMPB    #1,@#PASFLG     ;THE PROGRAM CONTROL COMES HERE AS FOLLOWS
2499                                                                  ;1-PASFLG=0, PROGRAM HAS JUST COMPLETED A
2500                                                                  ;  WRITE/READ CYCLE FOR THE BACK GROUND
2501                                                                  ;  AND WANTS TO BEGIN THE WRITE RECOVERY TEST
2502  004650  001440                          BEQ     24$             ;2-PASFLG=1, PROGRAM HAS JUST COMPLETED
2503                                                                  ;  THE WRITE RECOVERY TEST AND WANTS TO
2504                                                                  ;  READ MEMORY FOR CORRECT DATA
2505  004652  103627                          BLO     END12           ;3-PASFLG=2, PROGRAM HAS CORRECTLY READ THE
2506                                                                  ;  MEMORY AND WANTS TO GO THE NEXT TEST.
2507
2508  004654  105137  000307                  COMB    @#PASFLG+1      ;ENTER HERE WITH PASFLG=0, ON THE FIRST ENTRY
2509                                                                  ;ENABLE READ ONLY FOR THE MEMORY AND ON THE SECOND
2510                                                                  ;ENTRY DISABLE READ ONLY
2511  004660  001240                          BNE     2$
2512  004662  012702  005141                  MOV     #5141,R2        ;PLACE THE OP CODE FOR INSTRUCTION COM -(R1)
2513                                                                  ;IN R2
2514  004666  012700  177740                  MOV     #13$-.-6,R0     ;PLACE THE RETURN ADDRESS IN R0 AS 13$
2515  004672  060700                          ADD     PC,R0           ;THUS WHEN THE READ RECOVERY TEST REACHES
2516                                                                  ;THE MIDDLE OF THE 4K MEMORY THEN THE
2517                                                                  ;INSTRUCTION EXECUTED WILL BE JMP (R0)
2518                                                                  ;BRANCHINGING THE PROGRAM TO 13$
2519  004674  105237  000306          15$:    INCB    @#PASFLG        ;INCREMENT PASFLG BY 1.
2520  004700  000630                          BR      2$
2521
2522  004702  032777  000020  173540  16$:    BIT     #20,@SWR        ;HAS THE PRINTOUTS BEEN SUPRESSED ?
2523  004710  001017                          BNE     18$             ;IF SO THEN GO TO 18$
2524  004712  105737  000042                  TSTB    @#42            ;IS THE PROGRAM RUNNING UNDER ACT?
2525  004716  001014                          BNE     18$             ;BRANCH IF YES
2526  004720  004767  001656                  JSR     PC,PNTMES       ; TYPE THE BANK UNDER TEST
2527  004724  051524  030524  020063          .ASCIZ  /TST13 BANK/
2528  004732  040502  045516     000
2529          004740                          .EVEN
```

```
2530  004740  004767  002466          JSR     PC,GETBNK       ;GET BANK NO. UNDER TEST INTO DECWRD FOR PRINT.
2531  004744  004767  001660          JSR     PC,$TPDEC       ;TYPE BANK NO. UNDER TEST
2532
2533  004750  000113          18$:    JMP     (R3)            ;BEGIN EXECUTING MOV R2,-(PC) ,COM -(R1) SEQUENCE IN TES
2534
2535
2536  004752  105137  000307  24$:    COMB    @#PASFLG+1
2537  004756  012700  000110          MOV     #110,R0         ;PLACE THE OP CODE FOR JMP (R0) IN R0
2538  004762  000744                  BR      15$             ; READ THE MEMORY FOR CORRECT DATA AFTER
2539                                                          ;INCREMEMTING PASFLG TO 2
2540
2541                          ;TST13 EXITS VIA END12.
2542
```

```
2543  004764  012737  000377  000316  RELOC:  MOV     #377,@#BAKPAT
2544  004772  105737  000276                  TSTB    @#MMAV;
2545  004776  001065                          BNE     CONTMM          ;IS THE MEMORY MANAGEMENT BEING TESTED ?
2546                                                                  ;IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2547  005000  032777  001000  173442          BIT     #1000,@SWR      ;MEMORY MANAGEMENT
2548  005006  001046                          BNE     CKDONE          ;RELOCATION WANTED?
2549  005010  105737  000405                  TSTB    @#REL           ;BRANCH IF NO
2550  005014  100420                          BMI     RELOER          ;IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
2551  005016  112737  000200  000405          MOVB    #200,@#REL      ; PLACE THE PROGRAM BACK IN LOWER CORE
2552                                                                  ;OTHERWISE PREPARE TO RELOCATE
2553                                  ;RELOCATE THE DIAGNOSTIC TO HIGHEST AVAILABLE MEMORY
2554
2555
2556  005024  004767  001552                  JSR     PC,PNTMES       ;TYPE 'RELOC'
2557  005030  042522  047514  000103          .ASCIZ  /RELOC/
2558                                          .EVEN
2559  005036  013705  000340                  MOV     @#MAXMEM,R5     ;PREPARE TO LOAD THE PROGRAM IN THE HIGHEST
2560                                                                  ;AVAILABLE MEMORY
2561  005042  014445          2$:     MOV     -(R4),-(R5)     ;RELOCATE THE PROGRAM
2562  005044  020427  000430                  CMP     R4,#BEGIN-50    ;NEITHER RELOCATE NOR TEST LOCATIONS LOWER THAN BEGIN-50
2563  005050  101374                          BHI     2$
2564  005052  000165  000050                  JMP     50(R5)
2565
2566                                  ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
2567
2568
2569  005056  013705  000346  RELOER: MOV     @#SAVR5,R5      ;RESTORE R5
2570  005062  105737  000405                  TSTB    @#REL           ;IS DIAGNOSTIC IN RELOCATED STATE?
2571  005066  100016                          BPL     CKDONE          ;BRANCH IF NO
2572
2573  005070  012704  000430                  MOV     #BEGIN-50,R4    ;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
2574  005074  012524          2$:     MOV     (R5)+,(R4)+
2575  005076  020537  000340                  CMP     R5,@#MAXMEM
2576  005102  103774                          BLO     2$
2577  005104  105037  000405                  CLRB    @#REL
2578  005110  010537  000346                  MOV     R5,@#SAVR5      ;SAVE R5
2579  005114  012706  000500                  MOV     #BEGIN,SP       ;RESET STACK TO LOWER MEMORY
2580  005120  010637  000350                  MOV     SP,@#SAVR6      ;'BEGIN' USES THIS TO RESET THE STACK.
2581  005124  000137  005130  CKDONE: JMP     @#LOWER         ;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
2582
2583
2584
2585  005130  105737  000315  LOWER:  TSTB    @#SAVKBB        ;HERE DUE TO ^C TYPED?
2586  005134  001073                          BNE     $TPSTK          ;BRANCH IF YES (TYPE ERROR STACK)
2587  005136  004767  001712  TSTMM:  JSR     PC,MEMMNG       ; SET THE REGISTERS IF THE MEMORY MANAGEMENT
2588                                                                  ;IS AVAILABLE
2589  005142  105737  000276                  TSTB    @#MMAVA         ;IS MEM. MANAG. AVAILABLE ?
2590  005146  001462                          BEQ     ENDPAS          ;BRANCH IF NO
2591  005150  000402                          BR      $CNTMM          ;BEGIN TESTING ABOVE 28K
2592  005152  004767  002050  CONTMM: JSR     PC,UPMM         ;GO TO UPDATE MEM. MANAG. REGISTERS
2593  005156  012703  000324  $CNTMM: MOV     #LOWTWO,R3      ;MAKE R3 POINT TO THE LOCATION LOWTWO
2594  005162  004767  002154          JSR     PC,GETSIZ       ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
2595                                                                  ;OF THE LOWEST ADDRESS UNDER TEST
2596  005166  012704  020000                  MOV     #20000,R4       ;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
2597                                                                  ;POINTED BY PAGE ADDRESS REGISTER 1 (PAR1)
2598  005172  020237  172342                  CMP     R2,@#172342     ;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
```

```
2599
2600  005176  103405                  BLO     2$              ;PAR1 ?
2601  005200  050104                  BIS     R1,R4           ;IF SO THEN GO 2$
2602                                                          ;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
2603  005202  162702  000200          SUB     #200,R2         ;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
2604  005206  004767  001646          JSR     PC,MMREG        ; SET MEM. MANAG. REGISTERS
2605  005212  004767  002124  2$:     JSR     PC,GETSIZ       ;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
2606                                                          ;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
2607                                                          ;OF LOCATION HIGHADD IN R1
2608  005216  004767  000020          JSR     PC,MAXADR       ; GET THE ADDRESS OF MAX. MEM. UNDER TEST
2609  005222  010005                  MOV     R0,R5
2610  005224  004767  002112          JSR     PC,GETSIZ       ; PREPARE TO SET UP LOCATION MAXMEM
2611  005230  004767  000006          JSR     PC,MAXADR       ; GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
2612  005234  010013                  MOV     R0,(R3)         ;AND STORE INTO 'MAXMEM'
2613  005236  000167  174254          JMP     CLRMEM          ;GO TEST A 24K SLICE ABOVE 28K.
2614
2615
2616                                  ;MAXADR - SUBROUTINE TO GET CURRENT 24K SLICE OF MEMORY ADDRESSES ABOVE 28K.
2617                                  ;REGISTERS;
2618                                  ;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
2619                                  ;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
2620                                  ;R2= PAR BLOCK NO. CURRENTLY UNDER TEST.
2621
2622  005242  010046          MAXADR: MOV     R0,-(SP)        ;PUT MAXIMUM AVAILABLE ADDRESS ON STACK
2623  005244  012700  172356          MOV     #172356,R0      ;R0=PAR7 UNIBUS ADDRESS
2624                                  ;**BEGIN LOOP TO FIND PAR ADDRESS UNDER TEST
2625  005250  162716  020000  2$:     SUB     #20000,(SP)     ;DECREMENT VIRTUAL ADDRESS BY 4K
2626  005254  050116                  BIS     R1,(SP)         ;SET BITS 11:0 TO MAXIMUM VIRTUAL TEST ADDRESS
2627  005256  020240                  CMP     R2,-(R0)        ;DOES CURRENT PAR= TEST BLOCK NO.?
2628  005260  001410                  BEQ     3$              ;BRANCH IF YES
2629  005262  020027  172340          CMP     R0,#172340      ;ARE WE AT PAR0?
2630  005266  101370                  BHI     2$              ;NO KEEP TRYING
2631                                  ;**END LOOP TO FIND PAR ADDRESS UNDER TEST
2632  005270  005720                  TST     (R0)+           ;SET TO PAR CURRENT
2633  005272  021002                  CMP     (R0),R2 ;IS THE PAR BLOCK UNDER TEST GTR THAN ALLOWED?
2634  005274  003006                  BGT     4$              ;BRANCH IF YES (FALL INTO ENDPAS)
2635  005276  012716  157776          MOV     #157776,(SP)    ;EXIT WITH MAXADR= 28K SEGMENT TEST SIZE
2636  005302  012600          3$:     MOV     (SP)+,R0        ;SET R0 TO MAXIMUM VIRTUAL TEST ADDRESS
2637  005304  062700  000002          ADD     #2,R0           ;MAKE MAXIMUM MEMORY+2
2638  005310  000207                  RTS     PC              ;AND EXIT MAXADR ROUTINE
2639
2640  005312  022626          4$:     CMP     (SP)+,(SP)+     ;FIXUP STACK
2641                                                          ;AND FALL THRU TO ENDPAS.
```

```
                                        ;* TYPE ROUTINE FOR ERROR STACK
                                        ;* ---------------------------
                                        ;*
                                        ;*      THIS ROUTINE IS USED TO DETERMINE IF TYPE OUT OF THE ERROR STACK
                                        ;*      FOR ONLY THE FAILING BITS IS REQUIRED OR NOT


005314  032777  000020  173126  ENDPAS: BIT     #20,@SWR        ;ARE WE GOING TO TYPE THE ERROR STACK AND END OF PASS?
005322  001055                          BNE     $EOP            ;IF NOT THEN GO TO $EOP
005324  012746  177777          $TPSTK: MOV     #-1,-(SP)       ;THE PROGRAM HAS REACHED THE END AND ERROR
                                                                ;STACK AND END OF PASS WILL BE TYPED OUT
005330  012701  007744                  MOV     #ENDPRG,R1      ;PLACE THE STARTING ADDRESS OF THE ERROR STACK
                                                                ;FOR 0 TO 4K MEMORY IN R1
005334  012703  000376          TYPSTK: MOV     #376,R3
005340  005216                          INC     (SP)            ;IF WE HAVE GONE THRU THE ENTIRE
005342  020137  000310                  CMP     R1,@#ENDSTK     ;HAS THE END OF THE ERROR STACK BEEN REACHED ?
005346  103043                          BHIS    $EOP            ;THEN GO TO TYPE END OF PASS
005350  112702  000022                  MOVB    #18.,R2
005354  105302          RETSTK: DECB    R2      ;IF ALL 16 BITS OF THIS BANK HAVE BEEN CHECKED.
005356  002766                          BLT     TYPSTK          ;BEEN CHECKED FOR ERROR THEN SEE IF THERE
                                                                ;IS ANY MORE 4K MEMORY BANK
005360  105721                          TSTB    (R1)+           ;OTHERWISE CHECK THE BYTE STORED AT (R1)
005362  001774                          BEQ     RETSTK          ;IF IT IS 0 WE WILL NOT TYPE IT
005364  020227  000020                  CMP     R2,#16.         ;IS THE POINTER POINTING TO ERROR STACK BYTE
                                                                ;MEANT FOR COLLECTING ADDRESS FAILURES FOR
                                                                ;THE SPECIFIC MEMORY BANK
005370  103404                          BLO     2$              ;IF NOT THEN GO TO TYPE BIT NUMBER
005372  101026                          BHI     PARFL           ;IF IT IS POINTING TO THE STACK LOCATION INTENDED
                                                                ;TO COLLECT PARITY FAILURES THEN GO TO PARFL
005374  004767  001010                  JSR     PC,TPADER       ;OTHERWISE TYPE ''ADDRESS ERROR''
005400  000404                          BR      FAILNM
005402  010237  000312          2$:     MOV     R2,@#DECWRD     ;PREPARE TO TYPE THE NUMBER OF THE FAILING BIT
                                                                ;IN DECIMAL
005406  004767  001212                  JSR     PC,TYPDEC       ;GO TO TYPE THE BIT NUMBER IN DECIMAL
005412  011637  000312          FAILNM: MOV     (SP),@#DECWRD   ;PREPARE TO TYPE THE PAGE NUMBER
005416  004767  001206                  JSR     PC,$TPDEC       ;IN DECIMAL
005422  005043                          CLR     -(R3)
005424  114113                          MOVB    -(R1),(R3)      ;PREPARE TO PRINTOUT THE NUMBER OF TIMES THIS
                                                                ;FAILURE OCCURED
005426  105021                          CLRB    (R1)+           ;CLEAR THE ERROR STACK
005430  005043                          CLR     -(R3)
005432  105237  000314                  INCB    @#TYPCNT        ;ENABLE THE TYPE OUT OF 1 WORDS
005436  004767  001326                  JSR     PC,RPTOCT       ; TYPE THE 4K BANK AND THE NUMBER OF TIMES
                                                                ;THIS FAILURE WAS SEEN
005442  012703  000376                  MOV     #376,R3         ;RESET SCRATCH STACK FOR EACH BIT PRINTED.
005446  000742                          BR      RETSTK
005450  004767  000760          PARFL:  JSR     PC,TPPRER       ; TYPE 'PAR ERR''
005454  000756                          BR      FAILNM
```

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

005456  005002                  $EOP:   CLR     R2              ;SET R2= PARITY MODULE DISABLE CODE
005460  004767  002026                  JSR     PC,PARITY       ;GO DISABLE PARITY MODULES IF SELECTED.
005464  105737  000315                  TSTB    @#SAVKBB        ;CONTROL-C TYPED?
005470  001046                          BNE     CTLC            ;BRANCH IF YES-RESTORE LOADERS AND HALT-
005472  005237  000406                  INC     @#$PASS         ;INCREMENT PASS COUNT
005476  032777  000040  172744          BIT     #40,@SWR        ;"END PASS #XX" PRINTOUT WANTED?
005504  001015                          BNE     ACT11           ;BRANCH IF NO
005506  004767  001062          TYPEOP: JSR     PC,TPCRLF       ; TYPE CR, LF, AND "END PASS #"
005512  047105  020104  040520          .ASCIZ  /END PASS #/
005520  051523  021440     000
        005526                          .EVEN
005526  013737  000406  000312          MOV     @#$PASS,@#DECWRD        ;GET PASS COUNT
005534  004767  001070                  JSR     PC,$TPDEC               ;TYPE IT
005540  013700  000042          ACT11:  MOV     @#42,R0         ;GET THE MONITOR ADDRESS
005544  001405                          BEQ     $DOAGN          ;IF NONE
005546  004767  000012                  JSR     PC,RLODER       ;RESTORE XXDP MONITOR
005552  000005                          RESET           ;RETURN TO ACT11 MONITOR.


                                        ;* SERVICE XXDP/ACT11
005554  000137  000156                  JMP     @#$ENDAD        ;JUMP TO ACT SERVICE

005560  000137  000250          $DOAGN: JMP     @#RESTRT        ;REPEAT TEST IF NOT UNDER ACT11/XXDP

005564  004767  001622          RLODER: JSR     PC,CLRMM        ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
005570  013704  000344                  MOV     @#SAVR4,R4      ;RESTORE R4 WITH SAVR4
005574  014445                  4$:     MOV     -(R4),-(R5)     ;RESTORE LOADERS
005576  020437  000310                  CMP     R4,@#ENDSTK
005602  101374                          BHI     4$
005604  000207                          RTS     PC              ;RETURN FROM RLODER CALL

                                        ;CONTROL C HANDLER

005606  004767  177752          CTLC:   JSR     PC,RLODER       ;RESTORE ABS LOADER
005612  000167  000402                  JMP     APTHLT          ;IF NOT APT HALT AT FATHLT
```

```
                                        ;* ERROR HANDLING ROUTINE
                                        ;* ----------·----------
                                        ;*
                                        ;*      PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
                                        ;*      ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
005616  017637  000000  000402  ERROR:  MOV     @(SP),@#$FATAL  ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
005624  010346                  1$:     MOV     R3,-(SP)        ;SAVE R3
005626  010046                          MOV     R0,-(SP)        ;AND R0 ON THE STACK

                                ;SETUP BANK NO. IN FATAL FOR APT

005630  010103                          MOV     R1,R3           ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
005632  004767  001574                  JSR     PC,GETBNK       ;GET BANK NO. UNDER TEST INTO PBNK
005636  013703  000312                  MOV     @#PBNK,R3       ;GET BANK UNDER TEST
005642  110337  000403                  MOVB    R3,@#$FATAL+1   ;STORE FAILING BANK NO. FOR APT

                                ;

005646  010346                          MOV     R3,-(SP)        ;TEMPORARILY STORE R3
005650  012703  000376                  MOV     #376,R3         ;MAKE R3 AS THE STACK POINTER
005654  013743  000306                  MOV     @#PASFLG,-(R3)  ;OUTPUT THE WORD STORED AT
005660  005043                  2$:     CLR     -(R3)
005662  113713  000402                  MOVB    @#$FATAL,(R3)   ;PUT ERROR NO. ON ERROR STACK
005666  016643  000006                  MOV     6(SP),-(R3)     ;PLACE THE RETURN PC AT (R3)
005672  011143                          MOV     (R1),-(R3)      ;PLACE BAD DATA,
005674  010043                          MOV     R0,-(R3)        ;AND GOOD DATA ON THE STACK
005676  005043                          CLR     -(R3)
005700  016313  000004                  MOV     4(R3),(R3)      ;TAKE THE
005704  040013                          BIC     R0,(R3)         ;EXCLUSIVE OR OF GOOD AND BAD DATA
005706  046300  000004                  BIC     4(R3),R0        ;TO FIND THE BITS THAT FAILED
005712  050013                          BIS     R0,(R3)         ;AND PLACE IT ON THE STACK
005714  012700  002000                  MOV     #ENDPRG-.-24.,R0;THIS CODE BRINGS THE RELATIVE ADDRESS
005720  060700                          ADD     PC,R0           ;OF THE STARTING OF THE ERROR STACK
005722  062700  000022          6$:     ADD     #18.,R0         ;FOR THE SPECIFIC 4K BANK
005726  005316                          DEC     (SP)
005730  002374                          BGE     6$
005732  005726                          TST     (SP)+           ;RESTORE THE STACK POINTER

005734  105037  000277          ERRTYP: CLRB    @#TYPENB        ;DISABLE ANY TYPE OUT
005740  105737  000300          1$:     TSTB    @#$PRERR        ;IF THIS IS PARITY PROBLEM
005744  001007                          BNE     3$              ;THEN GO TO 3$
005746  105720                          TSTB    (R0)+           ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
005750  105737  000301                  TSTB    @#$ADERR        ;IF THIS IS ADDRESSING PROBLEM
005754  001003                          BNE     3$              ;THEN GO TO 3$
005756  105720                          TSTB    (R0)+           ;INCREMENT THE POINTER R0 BY 1
005760  005713                  2$:     TST     (R3)            ;IS BIT 15 OF (R3) SET?
005762  100015                          BPL     4$              ;IF NOT THEN GO TO 4$
005764  122710  000377          3$:     CMPB    #377,(R0)       ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
005770  001401                          BEQ     5$              ;IF SO DON'T BUMP ERROR COUNT
005772  105210                          INCB    (R0)            ;INCREMENT THE ERROR COUNTER BY 1
005774  122710  000001          5$:     CMPB    #1,(R0)         ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
006000  001404                          BEQ     7$              ;BRANCH IF NO
006002  032777  000400  172440          BIT     #400,@SWR       ;STOP ERROR PRINTOUT AFTER 1 WANTED?
006010  001002                          BNE     4$              ;BRANCH IF YES (DON'T TYPE ERROR)
```

```
2795  006012  105237  000277          7$:     INCB    @#TYPENB        ;ENABLE THE TYPE OUT ROUTINE
2796  006016  105737  000300          4$:     TSTB    @#$PRERR        ;PARITY ERROR?
2797  006022  001403                          BEQ     9$              ;BRANCH IF NO
2798  006024  004767  000404                  JSR     PC,TPPRER       ;ELSE TYPE 'PAR ERR'
2799  006030  000411                          BR      8$              ;AND DON'T TEST INDIVIDUAL BIT FAILURES.
2800  006032  105737  000301          9$:     TSTB    @#$ADERR        ;ADDRESS ERROR?
2801  006036  001403                          BEQ     6$              ;BRANCH IF NO
2802  006040  004767  000344                  JSR     PC,TPADERR      ;PRINT 'ADR ERR'
2803  006044  000403                          BR      8$
2804  006046  105720                  6$:     TSTB    (R0)+           ;POINT TO NEXT ENTRY IN ERROR STACK
2805  006050  006313                          ASL     (R3)            ;IS THERE STILL AN ERROR BIT SET IN ERROR.
2806  006052  001342                          BNE     2$              ;BR IF YES - KEEP FILLING ERROR STACK
2807  006054  112737  000006  000314  8$:     MOVB    #6,@#TYPCNT     ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
2808                                                                  ;THE STACK POINTED BY R3
2809  006062  004767  001146                  JSR     PC,PUTADR       ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
2810                                                                  ;AT LOCATIONS (R3) AND (R3-2)
2811  006066  004767  000620                  JSR     PC,TYPERR       ;TYPE ERROR STACK (7 WORDS)
2812
2813  006072  005037  000300          10$:    CLR     @#$PRERR        ;CLEAR ADDRESS/PARITY ERROR FLAGS
2814  006076  012600                          MOV     (SP)+,R0        ;RESTORE R0
2815  006100  012603                          MOV     (SP)+,R3        ;AND R3
2816  006102  105737  000420          FNDERR: TSTB    @#$ENV          ;ARE WE RUNNING UNDER APT?
2817  006106  001404                          BEQ     2$              ;IF NOT THEN TEST FOR HALT
2818  006110  012737  000001  000400          MOV     #1,@#$MSGTY     ;OTHERWISE INFORM THE APT
2819  006116  000443                          BR      FATHLT          ;GOTO FATHLT AND WAIT FOR APT.
2820
2821  006120  010246                  2$:     MOV     R2,-(SP)        ;SAVE R2 TEMP
2822  006122  005777  172322                  TST     @SWR            ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
2823                                                                  ;ON ERROR
2824  006126  100405                          BMI     4$              ;IF SO THEN HALT ON ERROR
2825                                  ;CHECK FOR CONTROL-C KEY
2826
2827  006130  004767  001536                  JSR     PC,CHECKC       ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
2828                                                                  ;AND HALT AT FATHLT.
2829  006134  105737  000042          7$:     TSTB    @#42            ;ARE WE RUNNING UNDER ACT?
2830  006140  001401                          BEQ     6$              ;BRANCH IF NO
2831
2832  006142  000000                  4$:     HALT                    ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
2833                                                                  ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
2834                                                                  ;THE WORD STORED IN R0
2835  006144  012602                  6$:     MOV     (SP)+,R2        ;RESTORE R2
2836  006146  062716  000002                  ADD     #2,(SP)         ;RESTORE THE RETURN ADDRESS
2837  006152  000207                          RTS     PC              ;RETURN FROM THE SUBROUTINE
2838
2839
2840
2841  006154                          FATERR:
2842  006154  004767  000414          SEQERR: JSR     PC,TPCRLF       ;TYPE 'ERROR #'
2843  006160  051105  047522  020122          .ASCIZ  /ERROR #/
2844  006166  000043
2845                                          .EVEN
2846
2847  006170  017637  000000  000402          MOV     @(SP),@#$FATAL  ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
2848  006176  105237  000314                  INCB    @#TYPCNT        ;TELL $TPNUM TO TYPE 1 WORD
2849  006202  012703  000376                  MOV     #376,R3         ;$TPNUM USES R3 AS STACK
2850  006206  013743  000402                  MOV     @#$FATAL,-(R3)  ;PUT ERROR NO. ON STACK
```

```
2851  006212  005743                  TST     -(R3)           ;$TPNUM REQUIRES THIS
2852  006214  004767  000560          JSR     PC,FAT%P        ;TYPE ERROR NO.
2853  006220  105737  000420  APTHLT: TSTB    @#$ENV  ;RUNNING UNDER APT?
2854  006224  001326                  BNE     FNDERR          ;BRANCH IF YES
2855  006226  000000          FATHLT: HALT                    ;FATAL ERROR OR ^C HALT.
2856  006230  000137  000250          JMP     @#RESTRT                ;RESTART TST BUT DON'T CLEAR PASS COUNT
2857                                                          ;IN CASE ^C RESTART.
2858
2859
2860                          ;PARERR
2861                          ;       PARITY TRAP HANDLER
2862                          ;COME HERE FROM A TRAP TO 114.
2863                          ;       THIS ROUTINE SEARCHES THE AVAILABLE PARITY MODULES AND IF ONE
2864                          ;HAS A PARITY ERROR BIT SET THE GET THE PARITY ERROR ADDRESS
2865                          ;AND CALL THE "ERROR" ROUTINE TO PRINT ERROR MESSAGE.
2866                          ;IF NO PARITY ERROR BITS CAN BE FOUND A FATAL ERROR IS DONE.
2867                          ;
2868                          ;REGISTER US AGE.
2869                          ;R0= HOLDS PARITY MODULE ADDRESSES
2870                          ;R1= GETS ERROR ADDRESS FOR "ERROR" CALL.
2871
2872  006234  012637  000356  PARERR: MOV     (SP)+,@#PARSP   ;SET PARSP TO RETURN ADDRESS
2873  006240  011637  000360          MOV     (SP),@#PARPS    ;SAVE PSW FOR RETURN
2874  006244  013706  000350          MOV     @#SAVR6,SP      ;AND RESET THE SP SINCE NOT ENOUGH STACK ROOM
2875                                                          ;TO COMPLETE THE ERROR SERVICE ROUTINE.
2876  006250  010067  000130          MOV     R0,SAVR0        ;SAVE R0 DURING PARITY SERVICE
2877  006254  010167  000126          MOV     R1,SAVR1        ;SAVE R1 DURING PARITY SERVICE
2878  006260  013701  000352          MOV     @#PARMAP,R1     ;GET PARITY AVAILABLE MAP
2879  006264  012700  172100          MOV     #172100,R0      ;R0= FIRST PARITY ADDRESS.
2880
2881  006270  005701                  TST     R1              ;ANY PARITY MODULES AVAILABLE?
2882  006272  001441                  BEQ     4$              ;BR IF NO -FATAL ERROR-
2883  006274  006001          1$:     ROR     R1              ;SHIFT PARITY MAP BIT INTO C BIT.
2884  006276  103005                  BCC     2$              ;BRANCH IF THIS PARITY MODULE NOT AVAILABLE.
2885  006300  005710                  TST     (R0)            ;PARITY MODULE ERROR BIT SET?
2886  006302  100406                  BMI     3$              ;BRANCH IF YES -CALL "ERROR" ROUTINE
2887  006304  020027  172136          CMP     R0,#172136      ;DONE ALL PARITY MODULES?
2888  006310  002032                  BGE     4$              ;BR IF YES- GO TO FATAL ERROR CALL-
2889  006312  062700  000002  2$:     ADD     #2,R0   ;POINT TO NEXT PARITY ADDRESS
2890  006316  000766                  BR      1$              ;AND KEEP TRYING
2891  006320  042710  100000  3$:     BIC     #100000,(R0)    ;CLEAR PARITY ERROR BIT.
2892  006324  011001                  MOV     (R0),R1         ;GET PARITY MODULE CSR
2893  006326  006101                  ROL     R1              ;SHIFT ERROR ADDRESS BITS 11-5 INTO 15-9
2894  006330  006101                  ROL     R1
2895  006332  006101                  ROL     R1
2896  006334  006101                  ROL     R1
2897  006336  042701  000777          BIC     #777,R1 ;SAVE ERROR ADDRESS ONLY
2898  006342  105237  000300          INCB    @#$PRERR        ;TELL "ERROR" PARITY ERROR CALL.
2899  006346  004767  177244          JSR     PC,ERROR        ;*ERROR* REPORT ERROR MESSAGE
2900  006352  000050                  50                      ;******ERROR NUMBER 50******
2901
2902  006354  016700  000024          MOV     SAVR0,R0        ;RESTORE R0
2903  006360  016701  000022          MOV     SAVR1,R1        ;RESTORE R1
2904  006364  013746  000360          MOV     @#PARPS,-(SP)   ;SET RETURN PSW ON STACK
2905  006370  013746  000356          MOV     @#PARSP,-(SP)   ;AND SET RETURN ADDRESS ON STACK
2906  006374  000002                  RTI                     ;RETURN TO TEST WHERE PARITY TRAP OCCURRED.
```

```
                                ;COME HERE IF NO PARITY ERROR FLAG FOUND SET
006376                          4$:
006376  004767  177552                JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
006402  000051                        51                      ;******ERROR NUMBER 51******

                                ;R0+R1 ARE SAVED HERE FOR PARITY TRAP DUE TO INSUFFICIENT
                                ;STACK SPACE BETWEEN 500-450.
006404  000000                  SAVR0:  0               ;SAVE R0 DURING PARITY TRAP SERVICE
006406  000000                  SAVR1:  0       ;SAVE R1 DURING PARITY TRAP SERVICE

006410  105737  000277        TPADER: TSTB    @#TYPENB        ;TYPE ERROR?
006414  001406                        BEQ     1$              ;BRANCH IF NO
006416  004767  000160                JSR     PC,PNTMES       ; TYPE CR, LF AND ''ADR ER''
006422  042101  020122  051105        .ASCIZ  /ADR ERR/
006430  000122
                                        .EVEN
006432  000207                  1$:     RTS     PC

006434  105737  000277        TPPRER: TSTB    @#TYPENB        ;ERROR PRINTOUTS ALLOWED?
006440  001406                        BEQ     1$              ;BRANCH IF NO
006442  004767  000134                JSR     PC,PNTMES       ;GO TO TYPE CR, LF AND 'PAR ERR''
006446  040520  020122  051105        .ASCIZ  /PAR ERR/
006454  000122
                                        .EVEN
006456  000207                  1$:     RTS     PC
```

```
2935
2936
2937                                    ;* TYPE OUT ROUTINE
2938                                    ;* ----------------
2939                                    ;*
2940                                    ;*      THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
2941                                    ;*
2942
2943  006460  010146                NOTYP:  MOV     R1,-(SP)
2944  006462  016601  000002               MOV     2(SP),R1
2945  006466  105721                4$:     TSTB    (R1)+           ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
2946  006470  001376                        BNE     4$              ;PREPARE TO RETURN
2947  006472  000412                        BR      RETTYP
2948  006474  010146                $TYPE:  MOV     R1,-(SP)        ;SAVE R1
2949  006476  010046                        MOV     R0,-(SP)        ;AND R0 ON TiE STACK
2950  006500  016601  000004               MOV     4(SP),R1        ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
2951  006504  112100                2$:     MOVB    (R1)+,R0        ;PLACE THE BYTE TO BE TYPED IN R0
2952  006506  001403                        BEQ     4$              ;IF IT IS END OF MESSAGE THEN GO TO 4$
2953  006510  004767  000022               JSR     PC,$TPCHR       ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
2954  006514  000773                        BR      2$
2955  006516  012600                4$:     MOV     (SP)+,R0        ;RESTORE R0
2956  006520  005201                RETTYP: INC     R1              ;CAUSE R1 TO
2957  006522  042701  000001               BIC     #1,R1           ;POINT TO EVEN ADDRESS
2958  006526  010166  000002               MOV     R1,2(SP)        ;MODIFY THE RETURN ADDRESS
2959  006532  012601                        MOV     (SP)+,R1        ;RESTORE R1
2960  006534  000416                        BR      EXTYP           ;AND RETURN VIA RTS PC
2961
2962  006536  132737  000040  000421  $TPCHR: BITB   #40,@#$ENVM     ;HAVE TYPE OUTS BEEN DISABLED?
2963  006544  001005                        BNE     4$              ;IF SO THEN RETURN FROM THE SUBROUTINE
2964  006546  105737  177564         2$:     TSTB    @#$TPS          ;WAIT HERE
2965  006552  100375                        BPL     2$              ;UNTIL THE PRINTER IS READY
2966  006554  110037  177566               MOVB    R0,@#$TPB       ;LOAD DATA TO BE TYPED INTO DATA REG.
2967  006560  000404                4$:     BR      EXTYP           ;RETURN
2968
2969  006562  004767  177706        PCRLF:  JSR     PC,$TYPE
2970  006566  005015     000               .ASCIZ  <15><12>        ;CR/LF
2971          006572                       .EVEN
2972  006572  000207                EXTYP:  RTS     PC      ;RETURN
2973
2974  006574  004767  177762        TPCRLF: JSR     PC,PCRLF        ;TYPE CR/LF
2975  006600  000735                        BR      $TYPE           ;NOW GO TO TYPE THE REST OF THE MESSAGE
2976
2977
2978  006602  032777  000020  171640  PNTMES: BIT     #20,@SWR        ;PRINTOUTS ALLOWED?
2979  006610  001323                        BNE     NOTYP           ;BRANCH IF NO
2980  006612  123737  000042  000046        CMPB    @#42,@#46       ;RUNNING UNDER ACT 11?
2981  006620  001717                        BEQ     NOTYP           ;BRANCH IF YES -NOT PRINTOUT-
2982  006622  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
                                        ;*

006624  004767  177732          TYPDEC: JSR     PC,PCRLF        ;TYPE CR/LF

006630  005046                  $TPDEC: CLR     -(SP)
006632  013746  000312                  MOV     @#DECWRD,-(SP)  ;GET THE WORD THAT HAS TO BE CONVERTED TO A
                                                                ;DECIMAL NUMBER
006636  162716  000012          2$:     SUB     #10.,(SP)
006642  002403                          BLT     4$              ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
                                                                ;GO TO 4$
006644  005266  000002                  INC     2(SP)           ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
006650  000772                          BR      2$              ;AND RETURN TO 2$
006652  062716  000012          4$:     ADD     #10.,(SP)
006656  052716  000060                  BIS     #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
006662  112667  000020                  MOVB    (SP)+,6$-2      ;PLACE THE 1'S DIGIT TO BE TYPED
006666  052716  000060                  BIS     #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
006672  112667  000007                  MOVB    (SP)+,6$-3      ;PLACE THE 10'S DIGIT TO BE TYPED
006676  004767  177572                  JSR     PC,$TYPE        ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
                                                                ;3 SPACES
006702  020040  030040  000060          .ASCIZ  /   00/
                                        .EVEN
006710  000207          6$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
```

```
3011
3012
3013                                    ;* OCTAL TYPE OUT ROUTINE
3014                                    ;* ---------------------
3015                                    ;*
3016                                    ;*      THIS ROUTINE IS USED TO TYPE OUT THE OCTAL VALUES
3017                                    ;*      CONTROL SHOULD COME TO THIS ROUTINE WITH R3 POINTING TO
3018                                    ;*      THE LOW ORDER BITS (I.E. BITS 0-15) OF THE ADDRESS TO
3019                                    ;*      BE TYPED WHERE AS R3-2 SHOULD CONTAIN THE HIGH ORDER BITS
3020                                    ;*      (I.E. BITS 16 & 17). CONTENTS OF LOCATION R3-1 AND R0 ARE
3021                                    ;*      DESTROYED BY THIS SUBROUTINE
3022                                    ;*      BYTE TYPCNT SHOULD BE SET TO THE NUMBER OF WORDS THAT HAVE
3023                                    ;*      TO BE TYPED.
3024                                    ;*
3025
3026  006712  032777  020000  171530  TYPERR: BIT     #20000,@SWR     ;ERROR PRINTOUT WANTED?
3027  006720  001054                          BNE     OCTXT           ;BRANCH IF NO
3028  006722  004767  177634                  JSR     PC,PCRLF        ;TYPE CR/LF
3029  006726  004767  000012                  JSR     PC,TYPOCT       ;TYPE OCTAL NO.
3030  006732  000447                          BR      OCTXT           ;RETURN VIA RTS PC
3031  006734  012123                  OCTTYP: MOV     (R1)+,(R3)+     ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
3032                                                                  ;BY R3
3033  006736  012113                          MOV     (R1)+,(R3)      ;AND NOW PLACE THE LOW ORDER BITS
3034  006740  105237  000314                  INCB    @#TYPCNT        ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3035  006744  052743  000004          TYPOCT: BIS     #4,-(R3)
3036  006750  106113                  2$:     ROLB    (R3)
3037  006752  103376                          BCC     2$
3038  006754  005000                          CLR     R0
3039  006756  106113                          ROLB    (R3)            ;GET BITS 17 & 16 INTO R0
3040  006760  006100                          ROL     R0
3041  006762  106113                          ROLB    (R3)
3042  006764  006100                          ROL     R0
3043  006766  000405                          BR      $TPNUM
3044  006770  004767  177500          RPTOCT: JSR     PC,$TYPE        ; TYPE 3 SPACES
3045  006774  020040  000040                  .ASCIZ  /   /
3046                                          .EVEN
3047  007000  005000                  FATYP:  CLR     R0
3048  007002  012723  000006          $TPNUM: MOV     #6,(R3)+        ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3049  007006  000241                  4$:     CLC
3050  007010  006113                          ROL     (R3)
3051  007012  006100                          ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3052  007014  052700  000060                  BIS     #60,R0          ;OR THE CONTENTS OF R0 WITH AN ASCII 0
3053  007020  004767  177512                  JSR     PC,$TPCHR       ; TYPE THE OCTAL NUMBER STORED IN R0
3054  007024  005000                          CLR     R0
3055  007026  006113                          ROL     (R3)
3056  007030  006100                          ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3057  007032  006113                          ROL     (R3)
3058  007034  006100                          ROL     R0              ;PLACE THE CARRY FROM (R3) IN R0
3059  007036  105363  177776                  DECB    -2(R3)          ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3060  007042  001361                          BNE     4$              ;THEN REPEAT FROM 4$
3061  007044  105337  000314                  DECB    @#TYPCNT        ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
3062  007050  001347                          BNE     RPTOCT          ;TYPED THEN REPEAT FROM RPTOCT
3063  007052  000207                  OCTXT:  RTS     PC
```

```
3064
3065                                    ;* ROUTINE TO SET UP MEMORY MANAGEMENT REGISTERS
3066                                    ;* ----------------------------------------------
3067                                    ;*
3068                                    ;*      PROGRAM CONTROL COMES HERE TO DETERMINE IF THE MEMORY MANAGEMENT
3069                                    ;*      IS AVAILABLE OR NOT, AND IF IT IS AVAILABLE THEN WHETHER
3070                                    ;*      THE MEMORY ABOVE 28K IS REQUIRED TO BE TESTED OR NOT.
3071                                    ;*
3072
3073  007054  012702  001400        MEMMNG: MOV     #1400,R2
3074  007060  105037  000276        MMREG:  CLRB    @#MMAVA         ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
3075                                                                  ;THAT MEM. MANAG. IS AVAILABLE FOR TESTING
3076  007064  032777  010000  171356        BIT     #10000,@SWR     ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
3077  007072  001441                        BEQ     RETMM           ;IF NOT THEN RETURN FROM THE SUBROUTINE
3078  007074  012700  000004                MOV     #4,R0           ;PREPARE TO SETUP TIME OUT VECTOR
3079  007100  012720  007200                MOV     #NOMM,(R0)+     ;RETURN ADDRESS TO NOMM
3080  007104  012710  000340                MOV     #340,(R0)       ;AND WITH A PSW OF 340
3081  007110  005037  177572                CLR     @#SR0           ;TRY TO REACH MEM. MANAG. SR0
3082  007114  105237  000276                INCB    @#MMAVA         ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
3083                                                                  ;BYTE
3084  007120  012701  172340                MOV     #172340,R1      ;R1 IS POINTING TO PAR0
3085  007124  005021                        CLR     (R1)+           ;PAR0 WILL POINT TO BANK 0
3086  007126  062702  000200        2$:     ADD     #200,R2
3087  007132  010221                        MOV     R2,(R1)+        ;SETUP PAR1-PAR6
3088  007134  020127  172356                CMP     R1,#172356
3089  007140  103772                        BLO     2$
3090  007142  012711  007600                MOV     #7600,(R1)      ;PAR7 IS POINTING TO THE I/O PAGE
3091  007146  012701  172300                MOV     #172300,R1
3092  007152  012721  077406        4$:     MOV     #77406,(R1)+    ;SETUP PDR0-PDR7
3093  007156  020127  172316                CMP     R1,#172316
3094  007162  101773                        BLOS    4$
3095  007164  005237  177572                INC     @#SR0           ;ENABLE MEM. MANAG.
3096  007170  005010                $RETMM: CLR     (R0)            ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
3097  007172  012740  000104                MOV     #BUSER,-(R0)
3098  007176  000207                RETMM:  RTS     PC
3099
3100  007200  022626                NOMM:   CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
3101  007202  004767  177366                JSR     PC,TPCRLF       ;TYPE 'NO MEMORY MANAGEMENT MESSAGE
3102  007206  047516  046440  043516        .ASCIZ  /NO MNG/
3103  007214     000
3104           007216                        .EVEN
3105  007216  004767  176732                JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
3106  007222  000052                        52                      ;******ERROR NUMBER 52******
3107
3108  007224  000761                        BR      $RETMM          ; RESTORE TIME OUT TRAP VECTOR
3109
3110  007226  013702  172354        UPMM:   MOV     @#172354,R2     ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
3111  007232  000712                        BR      MMREG
```

```
3112
3113                                    ;* 18 BIT ADDRESS GENERATOR
3114                                    ;* ------------------------
3115                                    ;*
3116                                    ;*      THIS SUBROUTINE IS USED TO PLACE THE ADDRESS STORED IN R1
3117                                    ;*      IN THE LOCATION POINTED BY R3. THE ADDRESS IN R1 IS CONVERTED
3118                                    ;*      TO AN 18 BIT ADDRESS ONLY IF MEM. MANAG. IS AVILABLE IN WHICH
3119                                    ;*      CASE THE HIGH ORDER BITS OF THE ADDRESS ARE PLACED IN LOCATION
3120                                    ;*      POINTED BY R3-2
3121                                    ;*
3122
3123  007234  005063  177776   PUTADR: CLR     -2(R3)
3124  007240  010113                   MOV     R1,(R3)         ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
3125  007242  105737  000276           TSTB    @#MMAVA         ;IS THE MEM. MANAG. AVAILABLE ?
3126  007246  001425                   BEQ     6$              ;IF NOT THEN RETURN FROM THE SUBROUTINE
3127  007250  010146                   MOV     R1,-(SP)        ;SAVE R1
3128  007252  042701  017777           BIC     #17777,R1       ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
3129  007256  040113                   BIC     R1,(R3)         ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
3130  007260  052701  004000           BIS     #4000,R1        ;PREPARE TO SHIFT R1 BY 12 PLACES
3131  007264  006001           2$:     ROR     R1
3132  007266  103376                   BCC     2$              ;GET THE NUMBER OF PAR IN R1
3133  007270  062701  172340           ADD     #172340,R1      ;GET THE ADDRESS OF PAR IN R1
3134  007274  011101                   MOV     (R1),R1         ;LOAD R1 WITH THE CONTENTS OF PAR
3135  007276  052701  010000           BIS     #10000,R1
3136  007302  006101           4$:     ROL     R1
3137  007304  103376                   BCC     4$              ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
3138  007306  006101                   ROL     R1
3139  007310  006143                   ROL     -(R3)           ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
3140  007312  006101                   ROL     R1
3141  007314  006123                   ROL     (R3)+           ;PLACE BIT 16 OF THE ADDRESS
3142  007316  050113                   BIS     R1,(R3)         ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
3143  007320  012601                   MOV     (SP)+,R1        ;RESTORE R1
3144  007322  000207           6$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
3145
3146                                    ;* GET ADDRESS FROM THE APT MAILBOX
3147                                    ;* --------------------------------
3148                                    ;*
3149                                    ;*      THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
3150                                    ;*      PLACE IT IN THE LOCATION USED BY THE PROGRAM TO  DEFINE THE
3151                                    ;*      MEMORY BOUNDRIES.
3152                                    ;*      PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
3153                                    ;*      ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
3154                                    ;*      THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
3155                                    ;*      TO BE PLACED
3156                                    ;*
3157
3158  007324  016143  000001   GETADR: MOV     1(R1),-(R3)     ;PLACE THE LOW ORDER BITS OF THE ADDRESS
3159  007330  005043                   CLR     -(R3)           ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
3160                                                            ;HAVE TO BE PLACED
3161  007332  116113  177777           MOVB    -1(R1),(R3)     ;PLACE BITS 16 & 17
3162  007336  000207           2$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
```

```
3163
3164                                    ;* CONVERT 18 BIT ADDRESS TO THE PAR FORM
3165                                    ;* ---------------------------------------
3166                                    ;*
3167                                    ;*      THIS SUBROUTINE IS USED TO CONVERT 18 BIT ADDRESS STORED IN
3168                                    ;*      LOCATIONS POINTED BY R3 AND R3+2 TO THE FORM IT WILL BE STORED
3169                                    ;*      IN A PAR. THE RESULT IS LEFT IN R2. R1 IS LOADED WITH BITS
3170                                    ;*      0-12 OF THE ADDRESS AND R0 WITH 160000
3171                                    ;*
3172
3173  007340  005046          $GTSIZ: CLR     -(SP)           ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3174                                                          ;0-4 OF R2
3175
3176  007342  012301          GETSIZ: MOV     (R3)+,R1
3177  007344  011302                  MOV     (R3),R2         ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3178  007346  042702  017777          BIC     #17777,R2       ;CLEAR ADDRESS BITS 0-12
3179  007352  052702  000040  2$:     BIS     #40,R2
3180  007356  006001          4$:     ROR     R1
3181  007360  006002                  ROR     R2              ;ROTATE R1 AND R2 7 TIMES
3182  007362  103375                  BCC     4$
3183  007364  005716                  TST     (SP)            ;IF RETURN PC IS ZERO THEN IT MUST BE THE
3184  007366  001004                  BNE     6$              ;FLAG THAT WAS SET AT $GTSIZ
3185  007370  005726                  TST     (SP)+           ;POP THE FLAG OFF STACK
3186  007372  052702  000100          BIS     #100,R2         ;KEEP ROTATING
3187  007376  000767                  BR      4$
3188  007400  012301          6$:     MOV     (R3)+,R1        ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3189  007402  012700  160000          MOV     #160000,R0
3190  007406  040001                  BIC     R0,R1           ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3191  007410  000207                  RTS     PC              ;RETURN FROM THE SUBRORNE
3192
3193
3194
3195                                    ;* SUBROUTINE TO DISABLE MEMORY MANAGEMENT
3196                                    ;* ---------------------------------------
3197                                    ;*
3198                                    ;*      THIS SUBROUTINE IS CALLED TO DISABLE THE MEMORY MANAGEMENT
3199                                    ;*      UNIT
3200
3201  007412  105737  000276  CLRMM:   TSTB    @#MMAVA         ;WAS THE MEMORY MANAGEMENT ENABLED ?
3202  007416  001404                  BEQ     1$              ;IF NOT THEN GO TO 1$
3203  007420  005037  177572          CLR     @#SR0           ;DISABLE THE MEMORY MANAGEMENT
3204  007424  105037  000276          CLRB    @#MMAVA         ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
3205  007430  000207          1$:     RTS     PC              ;RETURN FROM THE SUBROUTINE
3206
3207
3208                                    ;* GET BANK NO. UNDER TEST
3209                                    ;       CALLED BY ERRTYP AND TST13 TO GET BANK NO. UNDER TEST INTO PBNK.
3210                                    ;REGISTERS
3211                                    ;R0=POINTER TO PAR UNDER TEST
3212                                    ;R3=VIRTUAL ADDRESS ON ENTRY
3213                                    ;R0+R3 ARE RESTORED ON EXIT.
3214
3215  007432  010046          GETBNK: MOV     R0,-(SP)        ;SAVE R0
3216  007434  010346                  MOV     R3,-(SP)        ;SAVE R3
3217  007436  042703  017777          BIC     #17777,R3       ;SAVE ONLY VIRTUAL BANK BITS
3218  007442  052703  010000          BIS     #10000,R3               ;SETUP R3 SHIFT BIT
```

```
                                        CLC
                                1$:     ROR     R3              ;SHIFT A BANK BIT
                                        BCC     1$              ;UNTIL IN BITS <2:0> OF R3
                                        TSTB    @#MMAVA         ;MEMORY MANAGEMENT UNDER TEST?
                                        BEQ     2$              ;NO EXIT

                                ;GET PAR ADDRESS AND PHYSICAL BANK NO.
                                        ASL     R3              ;MAKE R3 PAR ADDRESS OFFSET.
                                        ADD     #172340,R3      ;MAKE FULL PAR ADDRESS.
                                        MOV     (R3),R0         ;GET PAR CONTENTS
                                        ASL     R0
                                        SWAB    R0              ;SHIFT BANK BITS TO BITS <7:0>
                                        MOVB    R0,R3           ;SET R3 TO PHYSICAL BANK NO.
                                2$:     MOV     R3,@#PBNK       ;STORE PHYSICAL BANK NO.
                                        MOV     (SP)+,R3        ;RESTORE R3
                                        MOV     (SP)+,R0        ;RESTORE R0
                                        RTS     PC              ;RETURN TO CALLER


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

                                ;CALL IS
                                ;       MOV     #1,R2   ;ENABLE CODE
                                ;       JSR     PC,PARITY


                                PARITY: BIT     #4000,@SWR      ;PARITY TEST WANTED?
                                        BEQ     6$                      ;BRANCH IF NO

                                        MOV     #4,R0           ;POINT R0 TO BUS TIMEOUT ADDRESS.
                                        MOV     #5$-.-6,(R0)    ;SET RETURN FROM TIMEOUT TRAP TO 5$
                                        ADD     PC,(R0)         ;IN THE CURRENT BANK.
                                1$:     CLR     @#PARMAP        ;CLEAR PARITY MAP HOLDER.
                                        MOV     #172140,R1      ;SET R1 TO LAST PARITY MODULE ADDRESS+2
                                        MOV     #100000,R3      ;SET R3 TO PARMAP AVAILABLE CODE BEGIN.
                                2$:     MOV     R2,-(R1)        ;ENABLE A PARITY MODULE+TRAP IF NOT AVAILABLE.
                                        BIS     R3,@#PARMAP     ;NO TRAP TO 5$, SO SET PARITY AVAILABLE.
                                        CLC
                                3$:     ROR     R3              ;SETUP NEXT PARMAP BIT
                                        BCC     2$              ;BRANCH IF NOT DONE ALL PARITY ADDRESSES.
                                        MOV     #BUSER,(R0)     ;RESET BUS TIMEOUT TRAP VECTOR
                                        TST     R2              ;IS THIS A DISABLE CALL?
                                        BEQ     6$              ;BRANCH IF YES (EXIT)
```

```
3275  007574  005737  000352            TST     @#PARMAP        ;WERE ANY PARITY MODULES FOUND?
3276  007600  001011                    BNE     4$              ;BRANCH IF YES
3277  007602  004767  176766            JSR     PC,TPCRLF       ;PRINT 'NO PAR''
3278  007606  047516  050040  051101    .ASCIZ  /NO PAR/
3279  007614     000
3280          007616                    .EVEN
3281  007616  004767  176332            JSR     PC,FATERR       ;*ERROR* REPORT  ERROR MESSAGE AND HALT AT FATHLT
3282  007622  000053                    53                      ;******ERROR NUMBER 53******
3283
3284
3285  007624  152737  000040  000405  4$:     BISB    #40,@#REL       ;SET PARITY UNDER TEST FLAG
3286  007632  012737  000376  000316          MOV     #376,@#BAKPAT   ;SET BACKGROUND PATTERN TO
3287                                          ;WORST CASE PARITY CODE.
3288  007640  004767  176730            JSR     PC,TPCRLF       ;PRINT 'TST PARITY''
3289  007644  040520  044522  054524    .ASCIZ  /PARITY/
3290  007652     000
3291          007654                    .EVEN
3292  007654  000405                    BR      EXITC           ;AND EXIT VIA RTS PC
3293
3294                          ;GET HERE IF PARITY ADDRESS TIMED OUT TO LOC. 4
3295
```

```
3296  007656  022626                  5$:     CMP     (SP)+,(SP)+     ;RESET STACK FROM TRAP
3297  007660  000737                          BR      3$              ;KEEP TRYING PARITY ADDRESES.
3298
3299  007662  142737  000040  000405  6$:     BICB    #40,@#REL       ;CLEAR PARITY TESTING FLAG
3300  007670                          EXITC:
3301  007670  000207                  7$:     RTS     PC              ;RETURN TO CALLER
3302
3303
3304
3305
3306                                  ;CHECKC
3307                                  ;       THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
3308                                  ;TEST OR IN THE ERROR TYPE ROUTINE.
3309                                  ;IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
3310                                  ;RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
3311                                  ;FINALLY IT HALTS AT FATHLT.
3312
3313  007672  105037  000315          CHECKC: CLRB    @#SAVKBB        ;INIT CONTROL-C FLAG.
3314  007676  105737  177560                  TSTB    @#TKS           ;ANY CHAR. TYPED?
3315  007702  100372                          BPL     EXITC           ;BR IF NO-EXIT VIA RTS PC-
3316  007704  113702  177562                  MOVB    @#$KBB,R2       ;GET THE CHAR TYPED.
3317  007710  042702  000200                  BIC     #200,R2 ;CLEAR THE PARITY BIT.
3318  007714  122702  000003                  CMPB    #3,R2           ;IS IT CONTROL-C?
3319  007720  001363                          BNE     EXITC           ;BRANCH IF NO -EXIT VIA RTS PC-
3320  007722  110237  000315                  MOVB    R2,@#SAVKBB     ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
3321  007726  004767  176642                  JSR     PC,TPCRLF       ;PRINT ''^C''
3322  007732  041536     000                  .ASCIZ  /^C/
3323          007736                          .EVEN
3324  007736  000167  175114                  JMP     RELOER          ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
3325
3326          007744                  .=7744
3327  007744  000000                  ENDPRG: 0               ;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
3328                                                          ;STACK.FOR EACH 4K BANK 18. BYTES ARE SAVED.
3329                                                          ;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
3330                                                          ;AFTER THE ERROR STACK.
3331                                                          ;FOR 4K MEMORY SIZE THEN PROGRAM=7744+22=7776
3332          000001                          .END
```

```
ABASE = 000000           1220
ACDW1 = 000000           1220
ACDW2 = 000000           1220
ACPUOP= 000000           1220    1235
ACT11   005540           2708    2715#
ADDW0 = 000000           1220
ADDW1 = 000000           1220
ADDW10= 000000           1220
ADDW11= 000000           1220
ADDW12= 000000           1220
ADDW13= 000000           1220
ADDW14= 000000           1220
ADDW15= 000000           1220
ADDW2 = 000000           1220
ADDW3 = 000000           1220
ADDW4 = 000000           1220
ADDW5 = 000000           1220
ADDW6 = 000000           1220
ADDW7 = 000000           1220
ADDW8 = 000000           1220
ADDW9 = 000000           1220
ADEVCT= 000000           1220    1226
ADEVM = 000000           1220
AENV  = 000000           1220    1231
AENVM = 000000           1220    1232
AFATAL= 000000           1220    1223
AMADR1= 000000           1220    1248
AMADR2= 000000           1220    1252
AMADR3= 000000           1220    1255
AMADR4= 000000           1220    1258
AMAMS1= 000000           1220    1242
AMAMS2= 000000           1220    1250
AMAMS3= 000000           1220    1253
AMAMS4= 000000           1220    1256
AMSGAD= 000000           1220    1228
AMSGLG= 000000           1220    1229
AMSGTY= 000000           1220    1222
AMTYP1= 000000           1220    1243
AMTYP2= 000000           1220    1251
AMTYP3= 000000           1220    1254
AMTYP4= 000000           1220    1257
APASS = 000000           1220    1225
APRIOR= 000000           1220
APTHLT  006220           2736    2853#
APTSIZ  000724           1320    1334#
ASWREG= 000000           1220    1233
ATESTN= 000000           1220    1224
AUNIT = 000000           1220    1227
AUSWR = 000000           1220    1234
AVECT1= 000000           1220
AVECT2= 000000           1220
BAKPAT  000316           1046    1180#   1288*   1505    1719    1721    1729    1740*   1804    1836*   2018*   2053    2088*
                         2140    2146    2150    2163    2166    2205*   2302    2358*   2359    2543*   3286*
BEGIN   000500           1054    1183    1204    1268#   2562    2573    2579
BRTPSZ  001012           1352#
BUSER   000104           1026#   1396    3097    3272
```

```
CHECKC  007672            1561    2827    3313#
CKDONE  005124            2548    2571    2581#
CLRMEM  001516            1499#   2613
CLRMM   007412            1431    1453    2726    3201#
CNTSCP  001650            1560    1563#
CONT    001556            1516#   1574
CONTMM  005152            2545    2592#
CTLC    005606            2705    2735#
DECWRD  000312            1166#   1167    2674*   2677*   2713*   2994
ENDPAS  005314            2590    2650#
ENDPRG  007744            1092    1286    2654    2772    3327#
ENDSTK  000310            1163#   1164    1418*   2658    2729
END0    002142            1661#   1703
END1    002242            1703#   1743
END10   004006            2210    2218#   2361
END12   004332            2361#   2473    2505
END2    002352            1743#   1839
END3    002620            1839#   1878
END4    002730            1878#   1964
END5    003076            1964#   2020
END6    003232            2020#   2096
END7    003456            2096#   2218
ERROR   005616            1618    1681    1735    1800    1864    1904    1911    1938    1996    2070    2151    2167    2176
                          2321    2329    2458    2470    2487    2746#   2899
ERRTYP  005734            2779#
EXITC   007670            3292    3300#   3315    3319
EXTYP   006572            2960    2967    2972#
FAILNM  005412            2673    2677#   2690
FATERR  006154            1299    1347    1464    1488    1646    2841#   2910    3105    3281
FATHLT  006226            2819    2855#
FATYP   007000            2852    3047#
FNDERR  006102            2816#   2854
GALLOP  003504            2133#   2268
GETADR  007324            1350    1351    3158#
GETBNK  007432            2530    2753    3215#
GETSIZ  007342            2594    2605    2610    3176#
HIGHAD  000332            1191#   1336
HIGHTW  000330            1190#   1335    1406
LOOP    001566            1519#   1572
LOWADD  000326            1188#
LOWBNK  000304            1156#   1157    2039*   2051    2086
LOWER   005130            2581    2585#
LOWTWO  000324            1187#   1399    1455    2593
M       =  000200         1001#   2487
MAXADR  005242            2608    2611    2622#
MAXMEM  000340            1197#   1334    1422*   1638    1833    2559    2575
MEMMNG  007054            1371    2587    3073#
MEMTST  001510            1493#
MMAVA   000276            1138#   1141    1373    2544    2589    3074*   3082*   3125    3201    3204*   3222
MMREG   007060            1384    2604    3074#   3111
N       =  000054         1001#   1299    1302#   1347    1350#   1464    1467#   1488    1491#   1589    1592#   1603    1605#
                          1606    1609#   1614    1616#   1646    1649#   1673    1676#   1681    1684#   1716    1719#   1735
                          1738#   1760    1763#   1785    1787#   1795    1797#   1811    1813#   1848    1851#   1864    1867#
                          1897    1900#   1904    1907#   1911    1914#   1938    1941#   1986    1989#   1996    1999#   2035
                          2038#   2070    2073#   2127    2130#   2151    2154#   2167    2170#   2176    2179#   2253    2256#
                          2293    2296#   2321    2324#   2329    2332#   2394    2397#   2458    2461#   2470    2473#   2487
```

```
                    2490#  2899   2902#  2909   2913#  3105   3108#  3281   3284#
NOMM     007200     3079   3100#
NOTYP    006460     2943#  2979   2981
OCTTYP   006734     1401   1405   3031#
OCTXT    007052     3027   3030   3063#
ONEPAS   000556     1280   1283#
PARERR   006234     1276   2872#
PARFL    005450     2670   2689#
PARITY   007512     1504   2703   3258#
PARMAP   000352     1205#  2878   3264*  3268*  3275
PARPS    000360     1208#  2873*  2904
PARSP    000356     1207#  2872*  2905
PASFLG   000306     1158#  1162   1516*  1874*  1942*  1959*  1991*  2002   2004*  2017*  2038*  2061   2077*
                    2079   2084   2299*  2316   2336   2342*  2343   2351*  2352   2441   2448   2463   2481
                    2498   2508*  2519*  2536*  2761
PBNK     000312     1165#  2754   3232*
PCRLF    006562     1400   2969#  2974   2991   3028
PNTMES   006602     1054   2260   2526   2556   2922   2930   2978#
PUTADR   007234     1388   1391   2809   3123#
PWRDN    000070     1019#  1272
PWRUP    000136     1019   1053#
REL    = 000405     1133#  1279   1479   2421   2549   2551*  2570   2577*  3285*  3299*
RELBOT   000322     1183#  1481
RELOC    004764     1541   2543#
RELOER   005056     2550   2569#  3324
RESTRT   000250     1068   1084   1086   1092#  2724   2856
RETMM    007176     3077   3098#
RETSTK   005354     2661#  2665   2688
RETTYP   006520     2947   2956#
RLODER   005564     2717   2726#  2735
RPTOCT   006770     2685   3044#  3062
RPT10    003500     2131#  2217   2263
RPT11    004046     2213   2216   2259   2262#
RPT6     003116     1989#  2019
SAVKBB   000315     1170#  2585   2704   3313*  3320*
SAVLDR   001272     1424   1426   1431#
SAVLOC   000354     1206#  2012*
SAVMAX   000342     1199#  1491
SAVR0    006404     2876*  2902   2916#
SAVR1    006406     2877*  2903   2917#
SAVR4    000344     1200#  2727
SAVR5    000346     1093   1201#  1452*  2569   2578*
SAVR6    000350     1053   1078   1204#  1271*  1622   2580*  2874
SCOPE  = 000240     1002#
SEQERR   006154     1589   1673   1716   1760   1848   1897   1986   2035   2127   2253   2293   2394   2842#
SETSTK   001176     1285   1406#
SETSWR   000656     1310#
SLFSIZ   001024     1361#
SR0    = 177572     1179#  3081*  3095*  3203*
START    000200     1061   1078#
STRTDI   000302     1154#  1155   2049*  2078
SWAPAT   000320     1182#  2171   2175   2204*
SWHALT   001670     1569#
SWR      000450     1265#  1313   1316*  1321*  1354   1462   1508   1567   1571   1632   2258   2262   2522
                    2547   2650   2707   2793   2822   2978   3026   3076   3258
SWREG    000176     1071#  1316
```

```
SW11   = 004000          1001#
TBL      001604          1525    1529#   1530    1531    1532    1533    1534    1535    1536    1537    1538    1539    1540
                         1541
TKS    = 177560          1175#   3314
TPADER   006410          2672    2802    2920#
TPCRLF   006574          1087    2709    2842    2974#   2982    3101    3277    3288    3321
TPPRER   006434          2689    2798    2928#
TRYSR    001014          1338    1354#
TSTGO    001672          1568    1571#
TSTMM    005136          1467    2587#
TSTREL   001350          1281    1452#
TSTRP    000570          1284    1286#
TSTSCP   001636          1559#   1661
TSTSIZ   001354          1453#
TST0     001706          1529    1587#
TST1     002144          1530    1669#
TST10    003460          1537    2124#
TST11    004010          1538    2250#
TST12    004062          1539    2291#   2360
TST13    004334          1540    2392#
TST2     002244          1531    1713#
TST3     002354          1532    1758#   1837
TST4     002622          1533    1846#
TST5     002732          1534    1894#
TST6     003100          1535    1982#
TST7     003234          1536    2032#
TYPCNT   000314          1168#   1274*   2684*   2807*   2848*   3034*   3061*
TYPDEC   006624          2676    2991#
TYPENB   000277          1142#   1145    2779*   2795*   2920    2928
TYPEOP   005506          2709#
TYPERR   006712          2811    3026#
TYPMEM   001164          1402#
TYPOCT   006744          3029    3035#
TYPSIZ   001142          1352    1355    1396#
TYPSTK   005334          2656#   2662
UPMM     007226          2592    3110#
WRTMEM   000120          1045#   1764    1900    1989
$A     = 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#
$ADERR   000301          1150#   1153    1610*   1733*   1797*   1862*   2783    2800
$APTHD   000276          1118    1124#   1137
$CNTMM   005156          2591    2593#
$CPUOP   000426          1235#
$DEVCT   000410          1226#   1566*   2076*   2200*
$DOAGN   005560          2716    2724#
$ENDAD   000156          1013    1064#   2722
$ENV     000420          1231#   1319    1559    2816    2853
$ENVM    000421          1232#   1337    1630    2962
$EOP     005456          2651    2659    2702#
$ETABL   000420          1230#
$ETEND   000450          1130    1259#
```

```
$FATAL  000402          1223#  2746*  2755*  2763   2847*  2850
$GTSIZ  007340          1407   3173#
$HD   = 000002           992
$HIBTS  000276          1125#
$HIMAX  000334          1194#  1335*
$KBB  = 177562          1176#  3316
$MADR1  000432          1248#
$MADR2  000436          1252#
$MADR3  000442          1255#
$MADR4  000446          1258#
$MAIL   000400          1081   1126   1130   1221#  1291   1306
$MAMS1  000430          1242#
$MAMS2  000434          1250#
$MAMS3  000440          1253#
$MAMS4  000444          1256#
$MAXM   000336          1195#  1336*  1415
$MBADR  000300          1126#
$MSGAD  000414          1228#
$MSGLG  000416          1229#
$MSGTY  000400          1026*  1222#  2818*
$MTYP1  000431          1243#  1344
$MTYP2  000435          1251#
$MTYP3  000441          1254#
$MTYP4  000445          1257#  1340
$NWTST= 000001          1575#  1577   1664#  1666   1705#  1707   1747#  1749   1840#  1842   1880#  1882   1966#
                        1968   2021#  2023   2097#  2099   2222#  2224   2269#  2271   2362#  2364
$PASS   000406          1225#  1283   2706*  2713
$PASTM  000304          1128#
$PRERR  000300          1146#  1149   1273*  2780   2796   2813*  2898*
$RETMM  007170          3096#  3108
$SVPC = 000044          1011#  1016
$SWR  = 000000           992#  1001#  1588   1670   1714   1759   1847   1895   1983   2033   2125   2251   2292
                        2393
$SWREG  000422          1233#  1321
$TESTN  000404          1085   1133   1224#  1517*  1563   1587   1669   1713   1758   1846   1894   1982   2032
                        2124   2182   2190   2215   2250   2291   2392
$TN   = 000014           982#   992   1575   1588#  1664   1670#  1705   1714#  1747   1759#  1840   1847#  1880
                        1895#  1966   1983#  2021   2033#  2097   2125#  2222   2251#  2269   2292#  2362   2393#
$TPB  = 177566          1178#  2966*
$TPCHR  006536          2953   2962#  3053
$TPDEC  006630          2531   2678   2714   2993#
$TPNUM  007002          3043   3048#
$TPS  = 177564          1177#  2964
$TPSTK  005324          2586   2652#
$TSTM   000302          1127#
$TYPE   006474          1402   2948#  2969   2975   3006   3044
$UNIT   000412          1079   1227#
$UNITM  000306          1129#
$USWR   000424          1234#
$Z    = 000362          1213#
$ZZ   = 007742          3326#
$$M   = 000200          2487#
.     = 007746           999#  1004#  1011   1012#  1014#  1016#  1018#  1024#  1031#  1070#  1114   1115#  1117#
                        1119#  1137#  1141#  1145#  1149#  1153#  1155#  1157#  1162#  1164#  1167#  1213   1216#
                        1267#  1276   1525   1588   1643   1672   1715   1759   1847   1896   1985   2034   2126
                        2252   2292   2393   2514   2529#  2712#  2772   2971#  3104#  3262   3280#  3291#  3323#
```

```
                        3326#
.$X    = 000276         1114#   1119
```

```
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      1575#  1577   1664#  1666   1705#  1707   1747#  1749   1840#  1842   1880#  1882   1966#  1968   2021#
         2023   2097#  2099   2222#  2224   2269#  2271   2362#  2364
NEWTST    982#  1575   1664   1705   1747   1840   1880   1966   2021   2097   2222   2269   2362
PLCERR    997#  1603   1606   1614   1785   1795   1811
STARS     982#  1009   1074   1077   1111   1113   1120   1185   1192   1219   1262   1264   1575   1586   1664
         1668   1705   1712   1747   1757   1840   1845   1880   1893   1966   1981   2021   2031   2097   2123
         2222   2249   2269   2290   2362   2391
$ERRNM    997#  1681   1735   1864   1904   1911   1938   1996   2070   2151   2167   2176   2321   2329   2458
         2470   2487   2899
$FATAL    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
$FTERR    997#  1299   1347   1464   1488   1646   2909   3105   3281
$SQERR    997#  1589   1673   1716   1760   1848   1897   1986   2035   2127   2253   2293   2394
$$NEWT    982#  1575   1664   1705   1747   1840   1880   1966   2021   2097   2222   2269   2362
.HEADE    982#
.$ACT1    982#  1007
.$APTB    982#  1217
.$APTH    982#  1109


. ABS.  007746     000


 ERRORS DETECTED:  0

 CZKMAE,CZKMAE.SEQ/CRF/SOL/NL:TOC=CZKMAE.MAC
 RUN-TIME: 8 9 .6 SECONDS
 RUN-TIME RATIO: 74/19=3.8
 CORE USED:  11K  (21 PAGES)
```