

# MF11S-K

MEMORY DIAGNOSTIC  
MD-11-DZMML-B

EP-DZMML-B-DL-A  
COPYRIGHT © 73-77  
FICHE 1 OF 1

OCT 1977  
**digital**  
MADE IN USA

This microfiche card contains a grid of 48 frames of diagnostic data, arranged in 8 rows and 6 columns. Each frame displays a different set of test results, including memory addresses, data values, and status indicators. The text is small and dense, typical of microfiche format. The frames are separated by thin white lines, and the overall layout is organized and systematic.

B01

EOF1DZDJBESEG

00010000

771026

PDP10 411

E HORIDZMMLBSEG

00010000

771026

POP10 411 SEG 0002

DZMML

MACY11 30(1046)

05-AUG-77 12:59

PAGE 1

DZMMLB.P11 05-AUG-77 12:57

.REPT 0

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZMML-B-D  
PRODUCT NAME: MF11S-K MEMORY DIAGNOSTIC  
DATE CREATED: MAY 1 , 1977  
DATE MODIFIED: AUG 1 , 1977  
MAINTAINER: DIAGNOSTIC GROUP  
AUTHOR: JAMES P. RYAN

COPYRIGHT (C) 1977  
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

CONTENTS  
-----

|       |  |
|-------|--|
| 1.0   | ABSTRACT                                 |
| 1.1   | GETTING STARTED                          |
| 2.0   | REQUIREMENTS                             |
| 2.1   | EQUIPMENT                                |
| 2.2   | STORAGE                                  |
| 3.0   | LOADING PROCEDURE                        |
| 4.0   | STARTING PROCEDURE                       |
| 4.1   | SWITCH SETTINGS                          |
| 4.2   | CONTROL-C FUNCTION                       |
| 4.3   | STARTING ADDRESS =200                    |
|       | RESTART ADDRESS =250                     |
| 4.4   | PROGRAM AND/OR OPERATOR ACTION           |
| 5.0   | PROGRAM HALTS (NORMAL + ERROR)           |
| 6.0   | ERRORS                                   |
| 6.1   | ERROR MESSAGE FORMAT.                    |
| 6.2   | ERROR DICTIONARY                         |
| 6.3   | ERROR HISTORY                            |
| 6.4   | ERROR RECOVERY                           |
| 6.5   | MOS CHIP CROSS REFERENCE TABLE           |
| 7.0   | RESTRICTIONS                             |
| 8.0   | MISCELLANEOUS                            |
| 8.1   | ADDRESS/BANK RANGES IN OCTAL AND DECIMAL |
| 8.2   | EXECUTION TIME                           |
| 8.3   | PASS COUNT AND TEST NO. LOCATIONS        |
| 8.4   | STACK POINTER                            |
| 8.6   | POWER FAIL                               |
| 9.0   | PROGRAM DESCRIPTION                      |
| 9.1   | NARRATIVE FLOW CHART                     |
| 9.2   | TEST TITLES AND TEST TIMES               |
| 10.0  | RXDP & ACT11 & APT OPERATION             |
| 11.01 | ABSTRACT                                 |

THIS DIAGNOSTIC WILL TEST 0 - 124K OF MF115-K MEMORY ON ANY PDP-11 FAMILY COMPUTER THAT DOES NOT PERFORM A DATIP CYCLE BEFORE A DATO CYCLE FOR ALL INST-RUCTIONS.

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

REGISTER = LOCATION 174.

### [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 ADDRESS OF THE CONTROL AND STATUS REGISTER (CSR) AND THE MEMORY LIMITS IT CONTROLS WILL BE PRINTED. IF TWO CONTROLLERS ARE PRESENT, BOTH CSR ADDRESSES WILL BE TYPED AS WELL AS BOTH MEMORY LIMITS.
- D. SEE SECTION 4.4 FOR REST OF PRINTOUTS EXPECTED.
- E. "END PASS #01" WILL BE TYPED. THIS IS A QUICK VERIFY PASS AND THE TEST IS NOT COMPLETE UNTIL "END PASS #02" IS TYPED.
- F. TO HALT THE TEST, TYPE CONTROL-C. THIS WILL INSURE THE PROGRAM IS RELOCATED BACK TO LOWER MEMORY AND THAT THE MEMORY HAS BEEN PURGED OF ANY DOUBLE ERRORS FORCED DURING TESTING. 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 <4:0>         |
| BIT13(020000) | INHIBIT ERROR PRINTOUTS                       |
| BIT12(010000) | DISABLE TESTING ABOVE 28K (MEMORY MANAGEMENT) |
| BIT11(004000) | ENABLE PRINTOUTS OF SINGLE ARRAY ERRORS       |
| BIT10(002000) | HALT AFTER EACH SUBTEST                       |
| BIT09(001000) | INHIBIT PROGRAM RELOCATION                    |
| BIT08(000400) | TYPE FIRST FAILING BIT ERROR PER 4K.          |
| BIT07(000200) | ENABLE XOR PRINTOUT FOR ARRAY TESTS           |
| BIT06(000100) | INHIBIT MEMORY SIZING                         |
| BIT05(000040) | INHIBIT "END PASS #XX" AND "RELOC" PRINTOUTS  |
| BIT04         | TO CLEAR MEMORY OF DBE'S(NO TESTS RUN)        |
| BIT03-BIT00   | BEGINNING TEST NUMBER.                        |

### [2.0] REQUIREMENTS

#### [2.1] EQUIPMENT

STANDARD 11 FAMILY COMPUTER WITH A CONSOLE OUTPUT DEVICE

AND FROM 32K TO 124K OF MS11K MEMORY (WITH EXCEPTION IN 1.0) .

## [2.2] STORAGE

PROGRAM STORAGE - 0000 - 14200. PROGRAM EXPANDS FOR ERROR HISTORY AND TO SAVE ABSOLUTE LOADER OR XXDP CHAIN MONITOR. (SEE SECTION 9. FOR DETAILS)

## [3.0] LOADING PROCEDURE

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

## [4.0] STARTING PROCEDURE

### [4.1] SWITCH SETTINGS

SOFTWARE SWITCH REGISTER = LOCATION 176

BIT15(100000) HALT ON ERROR (SEE 5.0)

BIT14(040000) LOOP ON TEST DEFINED BY SWITCH REGISTER BITS <4:0>  
BIT13(020000) INHIBIT ERROR PRINTOUTS AND ERROR HISTORY PRINTOUTS.  
ERROR HISTORY CAN STILL BE OBTAINED BY TYPING ↑C.  
BIT12(010000) DISABLE TESTING ABOVE 28K. (MEMORY MANAGEMENT)

BIT11(004000) ENABLE PRINTOUTS OF SINGLE ERRORS ON ARRAY TESTS  
(DISABLES ECC FOR ARRAY TESTS)

BIT10(002000) HALT AFTER EACH SUBTEST  
!PRESS CONTINUE TO DO NEXT SUBTEST  
BIT09(001000) INHIBIT PROGRAM RELOCATION  
!IF SET LOCATIONS 430-7776 WILL NOT BE  
!TESTED.

BIT08(000400) TYPE FIRST UNCORRECTABLE ERROR IN EACH 4K BANK ONLY.  
!THE TOTAL ERROR COUNT (UP TO 377) WILL  
!BE SAVED IN THE ERROR HISTORY.

BIT07(000200) ENABLE XOR PRINTOUT FOR ARRAY TESTS  
BIT06(000100) INHIBIT MEMORY SIZING.  
!THE MEMORY LIMITS MUST BE SETUP IN THE FOLLOWING LOCATIONS:  
(VALUES TO TEST 0-28K ARE SHOWN)  
(LOWTWO=LOCATION 332)

LOWTWO: 0 ;STORE BITS 17:16 OF LOW TEST ADDRESS  
LOWADD: 0 ;STORE REST OF LOW TEST ADDRESS  
HIGHTWO: 0 ;STORE BITS 17:16 OF HIGH TEST ADDRESS  
HIGHADD: 157776 ;STORE REST OF HIGH TEST ADDRESS  
NOTE: HIGHADD MUST BE SET TO A 4K BOUNDARY. (E.G. 37776)  
AND LOCATION LDFLG MUST = 1 IF THE PROGRAM  
IS RESIDING IN THE MS11K MEMORY.

BIT05(000040) INHIBIT "END PASS #XX" AND "RELOC" PRINTOUTS  
 BIT04 SCANS MEMORY WITH ECCDIS. NO TESTS RUN.  
 TERMINATE AFTER PASS#1 WITH ↑C.  
 BIT03-BIT00 NUMBER OF TEST (0-17) TO RUN FIRST.  
 !NORMALLY USED WITH BIT14 (LOOP ON TEST)

## [4.2] CONTROL-C FUNCTION

CONTROL C [↑C] AFTER COMPLETION OF THE CURRENT TEST.  
 THE ERROR HISTORY (SEE SEC. 6.3) WILL BE  
 TYPED. THE PROGRAM WILL LOOP IN LOWER MEMORY.  
 HALT AND RESTART AT 250 IF DESIRED.

IN ORDER TO COMPLETELY TEST THE ERROR CORRECTING  
 LOGIC, DOUBLE ERRORS ARE FORCED INTO MEMORY ON  
 CERTAIN TESTS. TO EXIT FROM THE PROGRAM AND HALT  
 EXECUTION, ↑C MUST BE TYPED AND THE TEST  
 PERMITTED TO CLEAN UP THE MEMORY UNDER TEST. IF  
 NOT, THERE EXISTS THE POSSIBILITY OF UNCORRECT-  
 ABLE ERRORS REMAINING IN THE MEMORY.

[4.3] STARTING ADDRESS = 200  
RESTART ADDRESS = 250 OR 200

RESTART AT 200 CLEARS PASS COUNT (\$PASS) AND PRINTS "DZMML-A" 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.

"NO RELOC-SBE IN LOC 0-500" ;RELOCATION IS INHIBITED BECAUSE  
 ;SINGLE ERRORS IN THIS AREA WILL CORRUPT  
 ;PROGRAM WHEN RELOCATED.

"RELOC" ;THE DIAGNOSTIC RELOCATES TO HIGHEST  
 ;BANK UNDER TEST. AND RUNS TST0-TST13 AGAIN.

"END PASS #XX" ;WHERE "XX" IS THE PASS NO.

"TST7-NO ERROR FREE LOC IN SLICE AT XXXXXX" THE LAST PART OF  
 TEST 7 CHECKS FOR PROPER 1K ADDRESSES IN

THE CSR AFTER A DOUBLE ERROR. THIS MESSAGE INDICATES THAT THIS SLICE COULD NOT BE TESTED AND IT IS PROCEEDING TO THE NEXT 1K SLICE.

## ADDITIONAL PRINTOUTS

"NO MNG"

;PRINTED IF TESTING ABOVE 28K IS ENABLED  
;(BIT 12 OF SWR) AND MEMORY MANAGEMENT IS  
;NOT AVAILABLE.

## [5.0] PROGRAM HALTS (NORMAL+ ERROR)

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

NOTE THE HALT AT END OF SUBTEST AND HALT ON ERROR HALT LOCATIONS MAY BE RELOCATED. THE ACTUAL LOCATIONS THEY ARE IN CAN BE FOUND BY SUBTRACTING 500 FROM 2362[SWHALT] AND ADDING THIS DIFFERENCE TO THE CONTENTS OF SAVR6 [LOC. 352].

| PC<br>--- | REASON<br>-----                         | RECOVERY<br>-----  |
|-----------|---|--|
| 112       | TRAP TO LOC. 4                          | EXAMINE R6, IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED. (SEE NOTE)<br>THE PROGRAM WILL LOOP AT LOC BUSER UNTIL MANUALLY HALTED. CHECK THE LOCATION POINTED TO BY MINMEM(324) TO INSURE THAT NO DBE'S REMAIN IN THE TESTED AREA. |
| 100       | POWER FAIL                              | POWER UP WILL RECOVER IF IN CORE MEMORY.   |
| 2364      | HALT AT END OF TEST SWITCH SET.         | PRESS CONTINUE TO GO TO NEXT SUBTEST.  |
| 13624     | HALT ON ERROR                           | RESTART AT 250   |
| 13712     | CONTROL-C TYPED OR FATAL ERROR OCCURRED | RESTART AT 250   |

NOTE: ANY OF THE ABOVE HALT CONDITIONS THAT PERTAIN TO

APT ACTUALLY RESULT IN A TIGHT LOOP SO THAT THE  
ERROR INFORMATION CAN BE RECOVERED. CONSULT LISTING  
FOR MORE DETAIL, IF NECESSARY.

## [6.0] ERRORS

ANY REFERENCE TO 'ERROR' IN THIS DOCUMENT INDICATES AN ERROR  
AS DEFINED BY THE PROGRAM AT EXECUTION TIME. NORMALLY, IT IS  
AN UNCORRECTABLE ERROR UNLESS ERROR CORRECTING IS DISABLED  
THEN IT MEANS ALL ERRORS.

## [6.1] ERROR MESSAGE FORMAT

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

"ADDR GOOD BAD PC ERR # PASFLG XOR "

"ADR ERR" WILL BE PRINTED PRIOR IF AN ADDRESSING ERROR IS SUSPECTED.

## WHERE:

ADDR = FAILING MEMORY LOCATION  
GOOD = GOOD DATA [DATA THAT WAS EXPECTED]  
BAD = BAD DATA [DATA THAT WAS FOUND]  
PC = PROGRAM COUNTER AT ERROR CALL.  
ERR # = 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)  
XOR = EXCLUSIVE 'ORING' OF GOOD VS. BAD (BITS IN ERROR).  
AVAILABLE ONLY IF SINGLE ERRORS ARE TO BE PRINTED  
OUT VIA SWREG BIT 07 HIGH.

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

## (FATAL ERRORS)

"ERROR #XXXXXX" WILL BE TYPED WHERE "XXXXXX" IS  
THE ERROR NUMBER. THE DIAGNOSTIC WILL LOOP AT FATHLT UNTIL MANUALLY  
HALTED. 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 LOOPS 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. IF THE ERROR IS SUCH THAT THE TESTING  
CAN BE CONTINUED AFTER THE ERROR PRINTOUT, IT WILL DO SO.  
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. BAKPAT HAS THE VALUE = 377, AND  
SWAPAT HAS THE VALUE = 177400.

.ENDR

```
;ERROR # 0      ;[BUSER] BUS ERROR TRAP TO LOC. 4 OCCURRED
                ;      THIS ERROR IS NOT PRINTED AND IS FOR "APT" USE.

;ERROR # 1      ;[BEGIN]FATAL LOAD ERROR
                ;DOUBLE ERROR EXISTS IN AREA
                ;OF MEMORY THAT PROGRAM RESIDES

;ERROR # 2      ;[TSTRP]FATAL DATA ERROR
                ;LOCATIONS 0000-430 FAILED 1'S + 0'S TEST.
                ;RD = GOOD DATA
                ;R1 = ADDRESS OF FAILING LOCATION.

;ERROR # 3      ;[APTSIZ] APT FATAL ERROR
                ;APT MEMORY TABLES NOT SETUP CORRECTLY.
                ;CHECK LOCATIONS $MAMS1 [430] TO $MADR4[446]
                ;, FOR CORRECT MEMORY SIZE DATA.

;ERROR # 4      ;[TSTSIZ] OPERATOR FATAL ERROR
                ;SELECTED MEMORY SIZE GREATER THAN 28K, BUT
                ;SR BIT12 (10000) SET.
                ;CLEAR BIT12 AND RESTART AT 200.

;ERROR # 5      ;[TSTSIZ] OPERATOR FATAL ERROR
                ;LOWEST SELECTED TEST LIMIT IS HIGHER THAN
                ;HIGHEST TEST LIMIT. SET LOCATIONS "LOWTWO"[332]
                ;TO "HIGHADD" [340] CORRECTLY AND RESTART
                ;AT 200.

;ERROR # 6      ;[CLRMEM] TEST SELECTION FATAL ERROR
                ;TEST SELECTED FOR LOOPING IS HIGHER
                ;THAN HIGHEST EXISTING TEST NUMBER.

;ERROR # 7      ;[TSTO] TEST SEQUENCE ERROR
                ;TSTO HAS BEEN ENTERED OUT OF SEQUENCE
                ;TESTN SHOULD = 00
```

```
      ;THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 10  ;[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 # 11  ;[TST0] ADDRESS AND DATA ERROR
              ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA
              ;WRITTEN INTO THE FAILING LOCATION WAS IN
              ;ERROR ALSO.

;ERROR # 12  ;[TST0] DATA ERROR
              ;IF BAD DATA = 0000 COULD BE AN ADDRESSING
              ;ERROR , ELSE COMPARE GOOD AND BAD DATA FOR FAILING BITS.

;ERROR # 13  ;[TST1] TEST SEQUENCE ERROR
              ;$TEST [404] SHOULD = 01
              ; THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 14  ;[TST1] DATA ERROR LOWER WORD (SHIFTING 1)
              ;COMPARE GOOD AND BAD PRINTED DATA, FAILING
              ;DATA BITS MAY SHORTED OR SWAPPED.

;ERROR # 15  ;[TST1] DATA ERROR UPPER WORD
              ;COMPARE GOOD AND BAD PRINTED DATA, FAILING
              ;DATA BITS MAY BE SHORTED OR SWAPPED

;ERROR # 16  ;[TST1] DATA ERROR LOWER WORD (SHIFTING 0)

;ERROR # 17  ;[TST1] DATA ERROR UPPER WORD (SHIFTING 0)

;ERROR # 20  ;[TST2] TEST SEQUENCE ERROR
              ;$TESTN [404] SHOULD = 02
              ; THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 21  ;[TST2] ADDRESS BIT TEST
              ;COMPARE GOOD VS. BAD DATA
              ;SUSPECT BYTE ADDRESSING PROBLEM

;ERROR # 22  ;[TST2] ADDRESS BIT TEST
              ;COMPLEMENTING BYTE DATA
              ;SUSPECT BYTE ADDRESSING

;ERROR # 23  ;[TST2] ADDRESS BIT TEST
              ;ADDRESS BIT DID NOT GET ASSERTED
              ;ADDRESSING PROBLEM

;ERROR # 24  ;[TST2] ADDRESS BIT TEST
              ;DATA DID NOT GET COMPLEMENTED
              ;ADDRESSING PROBLEM ?

;ERROR # 25  ;[TST3] TEST SEQUENCE ERROR
              ;$TESTN [404] SHOULD = 03
              ; THE DIAGNOSTIC HAS BEEN CORRUPTED.
```

K01

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-9  
DZMLB.P11 05-AUG-77 12:57

SEQ 0011

```
;ERROR # 26      ;[TST3] BYTE ADDRESSING TEST  
                ;COMPARE GOOD VS. BAD  
                ;ADDRESSING PROBLEM  
  
;ERROR # 27      ;[TST3] BYTE ADDRESSING TEST  
                ;DATA DID NOT GET COMPLEMENTED  
                ;COMPARE GOOD VS. BAD  
  
;ERROR # 30      ;[TST4] TEST SEQUENCE ERROR  
                ;STESTN [404] SHOULD = 04.  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 31      ;[TST4] CSR ACCESS ERROR  
                ;TRAP TO 4 OCCURRED  
                ;CSR DID NOT RESPOND  
  
;ERROR # 32      ;[TST4] CSR BIT ERROR  
                ;BIT FAILURE IN CSR  
                ;COMPARE GOOD VS. BAD  
                ;IF BAD DATA HAS SBE OR DBE  
                ;SET, SUSPECT OTHER THAN CSR  
  
;ERROR # 33      ;[TST5] TEST SEQUENCE ERROR  
                ;STESTN [404] SHOULD = 05  
                ; THE DIAGNOSTIC HAS BEEN CORRUPTED.  
  
;ERROR # 34      ;[TST5] ECC INH ERROR  
                ;ECC WAS DISABLED BUT LOW WORD  
                ;OF DATA WAS CORRECTED  
  
;ERROR # 35      ;[TST5] ECC INH ERROR  
                ;ECC WAS DISABLED BUT HIGH WORD  
                ;OF DATA WAS CORRECTED  
  
;ERROR # 36      ;[TST5] ECC ERROR  
                ;ECC WAS ENEBLED, BUT...  
                ;LOW WORD WAS NOT CORRECTED  
  
;ERROR # 37      ;[TST5] ECC ERROR  
                ;SINGLE ERROR BIT WAS NOT SET  
                ;AS RESULT OF PREVIOUS TEST  
  
;ERROR # 40      ;[TST5] ECC ERROR  
                ;ECC WAS ENABLED, BUT...  
                ;HIGH WORD WAS NOT CORRECTED  
  
;ERROR # 41      ;[TST5] ECC ERROR  
                ;SINGLE ERROR BIT WAS NOT SET  
                ;AS RESULT OF PREVIOUS TEST  
  
;ERROR # 42      ;[TST6] TETS SEQUENCE ERROR  
                ;STESTN [404] SHOULD = 06  
                ; DIAGNOSTIC HAS BEEN CORRUPTED  
  
;ERROR # 43      ;TST6] WRITE BYTE TO CLEAR SBE
```

```
                ;WRITING INTO BYTE WITH SINGLE ERROR  
                ;DID NOT RESULT IN CLEARING THE SBE.  
;ERROR # 44     ;[TST6] WRITE BYTE TO CLEAR SBE  
                ;DATA COMPARE ERROR  
;ERROR # 45     ;[TST6] WRITE BYTE TO CLEAR SBE  
                ;WRITING INTO BYTE WITH NO SINGLE ERROR  
                ;CLEARED ERROR IN THE BYTE WITH THE ERROR.  
;ERROR # 46     ;[TST7] TEST SEQUENCE ERROR  
                ;STESTN SHOULD = 07  
                ;      DIAGNOSTIC HAS BEEN CORRUPTED  
;ERROR # 47     ;[TST7] DOUBLE ERROR STATUS  
                ;DOUBLE ERROR BIT (BIT 15)  
                ;NOT SET AFTER DBE  
;ERROR # 50     ;[TST7] DOUBLE ERROR STATUS  
                ;TRAP THROUGH LOCATION 114  
                ;DID NOT OCCUR AFTER DBE  
                ;DISREGARD GOOD AND BAD DATA  
;ERROR # 51     ;[TST7] DOUBLE ERROR STATUS  
                ;CSR DID NOT CONTAIN CORRECT  
                ;1K BANK POINTING TO ERROR  
                ;ADDR = ADDR WITH DBLE ERROR  
                ;GOOD DATA = 1K BANK CONTAINING DBE  
                ;BAD DATA = BANK POINTED TO BY CSR BITS 11-5  
                ;POSSIBLE MARGINAL SINGLE ERROR CAUSED  
                ;DOUBLE ERROR NOT TO BE SET.  
;ERROR # 52     ;[TST10] TESTS SEQUENCE ERROR  
                ;STESTN SHOULD = 10  
                ;      DIAGNOSTIC HAS BEEN CORRUPTED  
;ERROR # 53     ;[TST10] DATIP WRITE INHIBIT ON DBE  
                ;'INC' INSTRUCTION DID NOT WRITE  
                ;INHIBIT ON LOWER WORD WITH DBE  
;ERROR # 54     ;[TST10] DATIP WRITE INHIBIT ON DBE  
                ;'INC' INSTRUCTION DID NOT WRITE  
                ;INHIBIT ON UPPER WORD WITH DBE  
;ERROR # 55     ;[TST11] TEST SEQUENCE ERROR  
                ;STESTN SHOULD = 11  
                ;      DIAGNOSTIC HAS BEEN CORRUPTED  
;ERROR # 56     ;[TST11] INHIBIT WRITE BYTE ON DBE  
                ;WRITE BYTE DID NOT GET INHIBITED  
                ;ON LOWER WORD WITH DBE  
;ERROR # 57     ;[TST11] INHIBIT WRITE BYTE ON DBE  
                ;WRITE BYTE DID NOT GET INHIBITED  
                ;ON UPPER WORD WITH DBE
```

;ERROR # 60 :[TST12] TEST SEQUENCE ERROR  
;STESTN SHOULD = 12  
;DIAGNOSTIC HAS BEEN CORRUPTED

;ERROR # 61 :[TST12] WORD WRITE INHIBIT ON DBE  
;WORD WRITE DID NOT GET INHIBITED  
;ON LOWER WORD WITH DBE

;ERROR # 62 :[TST12] WORD WRITE INHIBIT ON DBE  
;WORD WRITE DID NOT GET INHIBITED  
;ON UPPER WORD WITH DBE

;ERROR # 63 :[TST13] TEST SEQUENCE ERROR  
;STESTN SHOULD = 13  
;DIAGNOSTIC HAS BEEN CORRUPTED

;ERROR # 64 :[TST13] DUAL ADDRESSING ERROR  
;IF PASFLG = 0 THEN FAILING LOCATION  
;AND FAILING DATA ARE DUAL ADDRESS

;ERROR # 65 :[TST14] TEST SEQUENCE ERROR  
;STESTN SHOULD = 14  
;DIAGNOSTIC HAS BEEN CORRUPTED

;ERROR # 66 :[TST14] ADDRESS OR DATA ERROR  
;IF "ADD ERR" NOT PRINTED THEN THE  
;BYTE SELECT CIRCUITRY PROBABLY FAILED

;ERROR # 67 :[TST15] TEST SEQUENCE ERROR  
;STESTN SHOULD = 15  
;DIAGNOSTIC HAS BEEN CORRUPTED

;ERROR # 70 :[TST15] CHECK BIT WRITE ERROR  
;CHECK BITS SHOULD BE 4000  
;FROM WRITE OPERATION

;ERROR # 71 :[TST15] CHECK BIT DATA ERROR  
;MARCHING MIN TO MAX  
;CHECK BITS SHOULD BE 3740

;ERROR # 72 :[TST15] CHECKBIT DATA ERROR  
;MARCHING MIN TO MAX  
;CHECKBITS SHOULD BE 4000

;ERROR # 73 :[TST15] CHECK BIT DATA ERROR  
;MARCHING MAX TO MIN  
;CHECKBITS SHOULD BE 3740

;ERROR # 74 :[TST15] CHECKBIT DATA ERROR  
;MARCHING MIN TO MAX  
;CHECKBITS SHOULD BE 4000

;ERROR # 75 :[TST16] TEST SEQUENCE ERROR  
;STESTN SHOULD = 16  
;DIAGNOSTIC HAS BEEN CORRUPTED

```

;ERROR # 76      ;[TST16] DATA ERROR
                  ;DATA WRITE OR READ ERROR.

;ERROR # 77      ;[TST16] 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 # 100     ;[TST16] MARCHING 1'S AND 0'S DATA ERROR
                  ;IDENTICAL TO PREVIOUS ERROR EXCEPT THE DATA IS
                  ;CHECKED IMMEDIATELY AFTER BEING WRITTEN.

;ERROR # 101     ;[TST17] TEST SEQUENCE ERROR
                  ;STESTN SHOULD = 17
                  ;THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 102     ;[TST17] 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 # 103     ;[RELOC] FATAL RELOCATION ERROR
                  ;A DOUBLE ERROR EXISTS IN THE AREA
                  ;IN WHICH THE PROGRAM WANTS TO RELOCATE
                  ;RELOCATION ABORTED.

```

.REPT 0

### [6.3] ERROR HISTORY

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

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

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

ERROR HISTORY FORMAT:

ERROR    BANK    COUNT  
 -----

WHERE:

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

[6.4] ERROR RECOVERY

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

[6.5] MOS CHIP CROSS REFERENCE

IN THE FOLLOWING MATRIX, THERE ARE TWO "E" NUMBERS UNDER THE APPROPRIATE DATA BIT FOR EACH 8K WORTH OF ADDRESSING. THESE ADDRESSES ARE RELATIVE TO THE BASE ADDRESS OF THE SPECIFIC ARRAY BOARD BEING REFERENCED.  
 THE UPPER "E" NUMBER SHOULD BE USED IF THE LEAST SIGNIFICANT OCTAL DIGIT OF THE ADDRESS IS EITHER A 2 OR A 6 (BIT 1 = 1). THE LOWER "E" NUMBER SHOULD BE USED IF THE LEAST SIG OCTAL DIGIT IS A 0 OR A 4 (BIT 1 = 0). FOR EXAMPLE, IF THE BAD ADDRESS IS 100404 AND THE XOR OF GOOD VS BAD IS 100, THE SUSPECT MOS CHIP IS "E 167". THE BINARY BITS IN THE OCTAL XOR PRINTOUT REFERENCE DATA BITS 15-0 ON THE CHART HEADER.

|         | 15 | 14 | 13 | 12 | 11 | 10 | 9  | 8  | 7   | 6   | 5   | 4   | 3   | 2   | 1   | 0  |
|---------|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|----|
| 140000- | 88 | 77 | 67 | 56 | 46 | 35 | 24 | 12 | 184 | 172 | 160 | 148 | 135 | 122 | 110 | 97 |
| 177776  | 84 | 73 | 63 | 52 | 42 | 31 | 20 | 8  | 180 | 168 | 156 | 144 | 131 | 118 | 106 | 93 |
| 100000- | 87 | 76 | 66 | 55 | 45 | 34 | 23 | 11 | 183 | 171 | 159 | 147 | 134 | 121 | 109 | 96 |
| 137776  | 83 | 72 | 62 | 51 | 41 | 30 | 19 | 7  | 179 | 167 | 155 | 143 | 130 | 117 | 105 | 92 |

|        |    |    |    |    |    |    |    |    |     |     |     |     |     |     |     |    |
|--------|----|----|----|----|----|----|----|----|-----|-----|-----|-----|-----|-----|-----|----|
| 40000- | 86 | 75 | 65 | 54 | 44 | 33 | 22 | 10 | 182 | 170 | 158 | 146 | 133 | 120 | 108 | 95 |
| 77776  | 82 | 71 | 61 | 50 | 40 | 29 | 18 | 6  | 178 | 166 | 154 | 142 | 129 | 116 | 104 | 91 |
| 00000- | 85 | 74 | 64 | 53 | 43 | 32 | 21 | 9  | 181 | 169 | 157 | 145 | 132 | 119 | 107 | 94 |
| 37776  | 81 | 70 | 60 | 49 | 39 | 28 | 17 | 5  | 177 | 165 | 153 | 141 | 128 | 115 | 103 | 90 |

## [7.0] RESTRICTIONS

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

IT IS POSSIBLE TO CONFIGURE A SYSTEM USING TWO (2) MF115-K CONTROLLERS EACH WITH ITS OWN CSR. THE FIRST CSR WILL BE AT 172136 AND THE SECOND AT 172134. IN THE EVENT OF TWO CSR'S BOTH SYSTEMS WILL BE TESTED BEFORE A COMPLETE PASS IS ACKNOWLEDGED.

## [8.0] MISCELLANEOUS

## [8.1] ADDRESS/BANK RANGES IN OCTAL AND DECIMAL

THIS REFERENCE TABLE CROSS REFERENCES THE MEMORY BANK NO.5, THE RANGE AND THE PAR USED WHEN MEMORY MANAGEMENT IS ENABLED. IT IS ALSO USEFUL TO SHOW STARTING ADDRESSES IN A PARTICULAR 4K BANK.

| BANK NO. | DECIMAL RANGE | OCTAL RANGE   | [PAGE ADDRESS REGISTER] USED/CONTENT | UNIBUS ADDRESS |
|----------|---------------|---------------|--------------------------------------|----------------|
| 0        | 0 - 4K        | 000000-017776 | 0 0000                               | 772340         |
| 1        | 4K - 8K       | 020000-037776 | 1 0200                               | 772342         |
| 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 | 2 1600                               | 772344         |
| 8        | 32K-36K       | 200000-217776 | 3 2000                               | 772346         |
| 9        | 36K-40K       | 220000-237776 | 4 2200                               | 772350         |



|    |           |               |   |      |        |
|----|-----------|---------------|---|------|--------|
| 10 | 40K-44K   | 240000-257776 | 5 | 2400 | 772352 |
| 11 | 44K-48K   | 260000-277776 | 6 | 2600 | 772354 |
| 12 | 48K-52K   | 300000-317776 | 2 | 3000 |        |
| 13 | 52K-56K   | 320000-337776 | 3 | 3200 |        |
| 14 | 56K-60K   | 340000-357776 | 4 | 3400 |        |
| 15 | 60K-64K   | 360000-377776 | 5 | 3600 |        |
| 16 | 64K-68K   | 400000-417776 | 6 | 4000 |        |
| 17 | 68K-72K   | 420000-437776 | 2 | 4200 |        |
| 18 | 72K-76K   | 440000-457776 | 3 | 4400 |        |
| 19 | 76K-80K   | 460000-477776 | 4 | 4600 |        |
| 20 | 80K-84K   | 500000-517776 | 5 | 5000 |        |
| 21 | 84K-88K   | 520000-537776 | 6 | 5200 |        |
| 22 | 88K-92K   | 540000-557776 | 2 | 5400 |        |
| 23 | 92K-96K   | 560000-577776 | 3 | 5600 |        |
| 24 | 96K-100K  | 600000-617776 | 4 | 6000 |        |
| 25 | 100K-104K | 620000-637776 | 5 | 6200 |        |
| 26 | 104K-108K | 640000-657776 | 6 | 6400 |        |
| 27 | 108K-112K | 660000-677776 | 2 | 6600 |        |
| 28 | 112K-116K | 700000-717776 | 3 | 7000 |        |
| 29 | 116K-120K | 720000-737776 | 4 | 7200 |        |
| 30 | 120K-124K | 740000-757776 | 5 | 7400 |        |
| 31 | 124K-128K | 760000-777776 | 7 | 7600 | 772356 |

## NOTES:

- 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. 3 WOULD EQUAL 2000, PAR 4 WOULD EQUAL 2200 ETC.

## [8.2] EXECUTION TIME

HERE ARE SOME TYPICAL EXECUTION TIMES.

32K (PASS 1 + PASS 2) = 3 MIN 40 SEC.  
64K (PASS 1 + PASS 2) = 5 MIN 10 SEC.

## [8.3] PASS COUNT AND TEST NO. LOCATIONS

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

\$TESTN [404] = CURRENT TEST NO.

DURING EXECUTION, THE CURRENT TEST # WILL BE AVAILABLE IN THE HARDWARE DISPLAY REG (177570) OR THE SOFTWARE DISPLAY REG (174).

## [8.4] STACK POINTER

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

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

## [8.5] POWER FAIL

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

## [9.0] PROGRAM DESCRIPTION

## [9.1] NARRATIVE FLOW CHART

THE TEST IS LOADED INTO LOCATIONS 0000 - 14216 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 "DZMML-A" TITLE
2. [TSTRP] SAVE DATA FROM LOCATIONS 0-376
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 SETTING THE DIAGNOSTIC CHECK BIT IN THE MS11K CONTROL AND STATUS REGISTER (172136) AND LOOKING FOR MEMORY THAT RESPONDS BY LOADING CHECK BITS INTO THE CSR (BITS 11-5).
5. [TYPsiz] TYPE MEMORY TEST LIMITS AND THE ADDRESS OF THE CSR CONTROLLING THE MEMORY UNDER TEST.
6. [SETSTK] SPACE IS SAVED AT THE END OF THE TEST FOR AN ERROR HISTORY. FOR EACH 4K BANK 18 BYTES ARE SAVED IN THE FOLLOWING FORMAT:

```

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

```

THE ABSOLUTE LOADER (300 ADDRESSES) IS APPENDED AND IF IN XXDP CHAIN MODE, 5674 ADDRESSES ARE APPENDED TO THE TEST. THIS SAVES THE XXDP MONITOR AND ALLOWS THE LOCATIONS OCCUPIED BY XXDP TO BE TESTED.

7. [CLRMEM] INITIALIZE BAKPAT AND SWAPAT
8. [CONT] CLEAR CHUNK OF MEMORY UNDER TEST AND DISPATCH CONTROL TO THE APPROPRIATE TEST.
9. [TST0] EXECUTE TEST 0. SEE SECTION 10 FOR TEST DESCRIPTIONS.
10. [TSTSCP] COMES HERE AFTER EACH TEST AND IF CNTRL-C TYPED THEN GO TO ERROR HISTORY PRINTOUT. IF SR=2000 THEN HALT IF SR=40000 THEN LOOP ON TEST DEFINED BY <3:0> ELSE CONTINUE TO NEXT TEST.
11. [TST1-TST17] EXECUTE TST1-TST17 EACH TIME GOING TO STEP 9. \*\*\* SEE NOTE BELOW
12. [RELOC] THE PROGRAM RELOCATES TO 2ND 16K(IF NECESSARY AND IF NO SINGLE ERRORS EXIST IN LOCATIONS 0-500) TO TEST THE LOCATIONS IT OCCUPIES. (430-ENDPRG). WHERE "ENDPRG" IS THE CONTENTS OF ENDSTK(314). I.E THE LAST PROGRAM ADDRESS. NOTE "RELOC" IS PRINTED JUST PRIOR TO THE ACTUAL RELOCATION.
13. TESTS 0-17 ARE RUN AS DESCRIBED ABOVE EXCEPT ONLY BANK 0 LOCATIONS 430-ENDPRG ARE TESTED. \*\*\* SEE NOTE BELOW
14. [RELOER] RELOCATE THE PROGRAM BACK TO LOWER MEMORY.
15. [LOWER] IF CONTROL-C TYPED GO PRINT ERROR HISTORY.
16. [TSTM] IF MEMORY MANAGEMENT SELECTED AND AVAILABLE, RUN TESTS 0-17 ON THE FIRST 20K SLICE ABOVE 28K.
17. [CONTMM] CALL "UPMM" TO UPDATE MEMORY MANAGEMENT PAR REGISTERS TO POINT TO THE NEXT 20K SLICE OF

## UPPER MEMORY.

18. [MAXADR] REPEAT STEPS 18 + 19 UNTIL ALL MEMORY ABOVE 28K IS TESTED.
19. [ENDPAS] PRINT ERROR HISTORY OF FAILING BITS
20. [SEOP]  
PRINT "END PASS #XX"

\*\*\* NOTE: THERE EXISTS TWO DIFFERENT EXECUTION MODES. THEY ARE PASS #1 AND PASS #2 AND ALL SUCCESSIVE PASSES. THEY ARE DESCRIBED BELOW.

PASS 1 - QUICK VERIFY - ONE ITERATION ONLY OF ECC TESTS (TEST5-12) AND NO EXECUTION OF TEST20 AND TEST 21.

PASS 2 AND ABOVE - COMPLETE EXECUTION OF ALL TESTS (TEST0-21) THE FIRST COMPLETE PASS IS ACTUALLY PASS 2.

## [9.2] TEST TITLES AND TEST TIMES PER 16K

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

|   |        |
|---|--------|
| TEST 0: TEST FOR PROPER BANK SELECTION          | <1 SEC |
| TEST 1: CHECK DATI/DATO LINES                   | <1 SEC |
| TEST 2: TEST ADDRESS LINES FOR UNIQUENESS       | <1 SEC |
| TEST 3: TEST BYTE ADDRESSING                    | <1 SEC |
| TEST 4: MS11K STATUS REGISTER TESTS             | <1 SEC |
| TEST 5: SINGLE ERROR TESTS                      | 15 SEC |
| TEST 6: TEST THAT WRITE BYTE CLEARS SBE         | 7 SEC  |
| TEST 7: DOUBLE BIT ERROR- STATUS & TRAP         | 15 SEC |
| TEST 10: DBE- WRITE INH ON DATIP CYCLE          | 1 SEC  |
| TEST 11: DBE- WRITE INH ON WRITE BYTE           | 15 SEC |
| TEST 12: DBE- WRITE INH ON WORD WRITE           | 15 SEC |
| TEST 13: DUAL ADDRESS TEST                      | 2 SEC  |
| TEST 14: TEST MEMORY FOR HOLDING 1'S AND 0'S    | 2 SEC  |
| TEST 15: MARCHING 1'S AND 0'S IN CHECKBIT CHIPS | 3 SEC  |
| TEST 16: MARCHING 1'S AND 0'S IN DATA CHIPS     | 2 SEC  |
| TEST 17: CELLS VOLATILITY TEST (REFRESH)        | 4 SEC  |

## [10.0] RXDP &amp; ACT11 &amp; APT OPERATION

## RXDP CHAIN MODE

OPERATION IS IDENTICAL TO STAND ALONE EXCEPT:

1. NO "DZMML-A" TITLE IS PRINTED.
2. THE PROGRAM ALWAYS HALTS ON ERROR.

3. AT THE END OF TEST (SENDAD) 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 (SENDAD) CONTROL IS RETURNED TO THE ACT11 MONITOR VIA LOCATION 42.

APT

-----  
OPERATION IS SIMILAR TO STAND ALONE EXCEPT:

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

## APT MANAGER INFORMATION

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

THE EXAMPLE ASSUMES YOU ARE LOGGED INTO THE APT MONITOR.

READY

```
RUN APPLU
APT 11 PAPER TAPE PROGRAM LOAD UTILITY
```

THE FOLLOWING COMMANDS ARE VALID

```
ED      EDIT A PROGRAM
LI      LIST A PROGRAM
```

```
COMMAND: ED
PROGRAM NAME TO EDIT: EXAMPL
DO YOU WANT TO LOAD A NEW REV OF THE PROGRAM(Y/N)? N
FIRST PASS RUN TIME IN SECONDS <110>:
LONGEST TEST TIME IN SECONDS <10>:
ADDITIONAL RUN TIME IN SECONDS <0>:
WHICH ETABLE DO YOU WISH TO EDIT? A
SOFTWARE ENVIRONMENT<000>: 1
```

ENVIRONMENTAL MODE<000>: 240  
SWITCH 1 <000000>:  
SWITCH 2 <000000>:  
CPU OPTIONS<0000>:  
MEMORY TYPE 1 <000>:  
MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 2 <000>:  
MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 3 <000>:  
MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 4 <000>: 1  
MAXIMUM ADDRESS<00000000>: 17776  
WHICH ETABLE DO YOU WISH TO EDIT?  
COMMAND: OFF

.ENDR  
.ABS  
.NLIST MD,MC,CND

1164  
1169  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
1170  
1171  
1172  
1173  
1205  
1206  
1214  
1215  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
(1)  
(2)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
1229

.LIST ME,BIN,SEQ,LOC  
.TITLE DZMML  
;\*COPYRIGHT (C) FEBRUARY 1977  
;\*DIGITAL EQUIPMENT CORP.  
;\*MAYNARD, MASS. 01754  
\*  
;\*PROGRAM BY JIM RYAN  
\*  
\*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC  
\*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.  
\*  
\$SWR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

;;TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS

000240  
000042  
000042 000000  
000046 000156  
000052 040000  
000044

SCOPE =NOP  
.=42  
.WORD 0 ;FOR ACT/XXDP  
.SBTTL ACT11 HOOKS  
\*\*\*\*\*  
;HOOKS REQUIRED BY ACT11  
\$SVPC=. ;SAVE PC  
.=46  
\$ENDAD ;;1)SET LOC.46 TO ADDRESS OF \$ENDAD IN .SEOP  
.=52  
.WORD 40000 ;;2)SET LOC.52 TO 40000  
.= \$SVPC ;; RESTORE PC



# K02

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-22  
 DZMMLB.P11 05-AUG-77 12:57 ACT11 HOOKS

SEQ 0024

1240  
1241  
1242  
1243  
1244  
1245  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298

000120

;114 AND 116 ARE RESERVED FOR PARITY TRAP VECTORS.USED IN TST7.  
 .=120

;\* WRITE MEMORY BACKGROUND

-----  
 ;\*  
 ;\*  
 ;\* THIS ROUTINE IS USED TO WRITE THE MEMORY BACKGROUND TO  
 ;\* THE VALUE STORED AT LOCATION BAKPAT. THE ROUTINE ASSUMES  
 ;\* THAT R4 IS POINTING TO THE LOWEST LOCATION AND R5 TO THE  
 ;\* HIGHEST LOCATION TO BE WRITTEN. THE PROGRAM LEAVES THE  
 ;\* SUBROUTINE WITH R0 CONTAINING THE CONTENTS OF BAKPAT.  
 ;\*

WRTMEM: MOV R4,R1 ;SET R1 TO LOWEST LOCATION UNDER TEST  
 MOV @#BAKPAT,R0 ;LOAD R0 WITH THE CONTENTS OF LOCATION BAKPAT  
 2\$: MOV R0,(R1)+ ;STARTING FROM THE LOWEST LOCATION WRITE THE  
 CMP R1,R5 ;MEMORY TO BACK GROUND PATTERN  
 BLO 2\$  
 RTS PC ;RETURN FROM THE SUBROUTINE

PWRUP: MOV @#SAVR6,SP ;RESTORE STACK POINTER  
 MOV @#PNTMES-BEGIN,R0 ;  
 ADD SP,R0 ;GET THE INDIRECT ADDRESS OF LOCATION  
 ;TPCRLF RELATIVE TO LOCATION OF THE  
 ;DIAGNOSTIC IN MEMORY AND GO TO THE  
 JSR PC,(R0) ;TYPE ROUTINE AND TYPE CR, LF & A "P".  
 .ASCIZ /P/  
 .EVEN  
 BR START

;\* SERVICE XXDP/ACT11  
 \$ENDAD: JSR PC,(R0) ;RETURN TO ACT11/XXDP MONITOR  
 NOP ;IF QUICK VERIFY=RESET ELSE NOP  
 NOP ;IF QUICK VERIFY=CLR #-1 ELSE INC #0  
 NOP ;IF QUICK VERIFY=BR .-4 ELSE NOP  
 BR RESTRT ;REPEAT TEST UNDER ACT11/XXDP

REL: .BYTE 0 ;TELLS IF WE'RE RELOC

.=174  
 DSPREG: .WORD 0  
 SWREG: .WORD 0

\*\*\*\*\*  
 ;SBTTL START AND RESTART ROUTINES  
 ;\* RESTART AT 200 TO CLEAR APT TABLES  
 ;\*  
 ;\*\*\*\*\*

START: CLR @#CSRADR ;JUST IN CASE  
 MOV @#SAVR6,SP ;SETUP STACK POINTER.  
 MOV @#SUNIT,R3 ;CLEAR THE APT MAILBOX FROM \$MAIL TO \$DEVCT

010401 000312  
 013700  
 010021  
 020105  
 103775  
 000207  
 013706 000346  
 012700 013740  
 060600  
 004710  
 000120  
 000411  
 004710  
 000240  
 000240  
 000240  
 000430  
 000  
 000174  
 000000  
 000000  
 005077 002524  
 013706 000346  
 012703 000412



START AND RESTART ROUTINES

```

1299 000214 005043          1S: CLR      -(R3)          ;CLEAR A MAILBOX LOCATION
1300 000216 022703 000400    CMP      #SMAIL,R3      ;DONE?
1301 000222 001374          BNE      1S             ;BRANCH IF NO
1302 000224 105737 000042    TSTB    @#42           ;ACT11 MODE?
1303 000230 001007          BNE      RESTRT        ;BRANCH IF YES
1304 000232 105737 000170    TSTB    @#REL         ;ARE WE RELOCATED?
1305 000236 100404          BMI      RESTRT        ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
1306 000240 004767 015472    JSR     PC,PRTITL     ;PRINT "DZMML-A" TITLE
1307 000244 000240          NOP                     ;SO WE FALL THRU TO RESTART
1308 000246 000240          NOP                     ;DITTO
1309
1310 000250 012703 000344    RESTRT: MOV     #SAVR5,R3 ;POINT R3 TO THE LOC SAVR5
1311 000254 012305          MOV     (R3)+,R5      ;RESTORE R5
1312 000256 012306          MOV     (R3)+,SP      ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1313 000260 010600          MOV     SP,R0         ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1314 000262 012746 000340    MOV     #340,-(SP)    ;SET HIGH PRIORITY FOR RTI
1315 000266 010046          MOV     R0,-(SP)
1316 000270 000002          RTI                    ;GO TO "START"-MAY BE RELOCATED.
1317                                     ;IF RELOCATED SEE LOCATION SAVR6 FOR START.
1318
1319
1320
1321
1322
1323
1324
1325
1326

```

.SBTTL APT PARAMETER BLOCK

```

(1)
(2)
(1)
(2)
(1) 000272
(1) 000024 000024
(1) 000024 000200
(1) 000044 000044
(1) 000044 000272
(1) 000272
(2)
(1)
(1)
(1)
(1) 000272
(1) 000272 000000
(1) 000274 000400
(1) 000276 000031
(1) 000300 000156
(1) 000302 000000
(1) 000304 000024
1327
1328
1329
1330 000272
1331
1332
1333 000273

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

$APTHD:
$SHIBTS: .WORD 0 ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
$MBADR: .WORD $MAIL ;ADDRESS OF APT MAILBOX (BITS 0-15)
$STMT: .WORD 25 ;RUN TIM OF LONGEST TEST
$PASTM: .WORD 110. ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
$UNITM: .WORD ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
.WORD SETEND-$MAIL/2 ;LENGTH MAILBOX-ETABLE(WORDS)

MMAVA: .= $APTHD ;THIS BYTE IS USED TO DETERMINE IF MEMORY
;MANAGEMENT IS AVAILABLE OR NOT

.= MMAVA+1

```

|      |        |        |  |  |   |
|------|--------|--------|--|--|---|
| 1334 | 000273 |        | TYPENB:  |  | ; THIS BYTE IS USED TO DETERMINE IF THE<br>; TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT  |
| 1335 |        |        |  |  |   |
| 1336 |        |        |  |  |   |
| 1337 | 000274 | 000274 | SPRERR: . =TYPENB+1                                  |  | ; THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND<br>; A PARITY ERROR   |
| 1338 | 000274 |        |  |  |   |
| 1339 |        |        |  |  |   |
| 1340 |        |        |  |  |   |
| 1341 | 000275 | 000275 | SADERR: . =SPRERR+1                                  |  | ; THIS BYTE IS USED TO DETERMINE IF THE<br>; PROGRAM HAS ENCOUNTERED ADDRESS ERROR  |
| 1342 | 000275 |        |  |  |   |
| 1343 |        |        |  |  |   |
| 1344 |        |        |  |  |   |
| 1345 | 000276 | 000276 | STRTDI: . =SADERR+1                                  |  |   |
| 1346 | 000276 |        |  |  |   |
| 1347 | 000300 | 000300 | LOWBNK: . =STRTDI+2                                  |  |   |
| 1348 | 000300 |        |  |  |   |
| 1349 | 000302 | 000302 | PASFLG: . =LOWBNK+2                                  |  | ; LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR<br>; THE SPECIFIC TEST WHEREAS THE UPPER BYTE<br>; HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES |
| 1350 | 000302 |        |  |  |   |
| 1351 |        |        |  |  |   |
| 1352 |        |        |  |  |   |
| 1353 |        |        |  |  |   |
| 1354 | 000304 | 000304 | ENDSTK: . =PASFLG+2                                  |  |   |
| 1355 | 000304 |        |  |  |   |
| 1356 | 000306 | 000306 | PBNK: . =ENDSTK+2                                    |  | ; HOLDS BANK UNDER TEST FOR "TST BNK XX" PRINTOUT.  |
| 1357 | 000306 |        | DECWRD:  |  |   |
| 1358 | 000306 |        |  |  |   |
| 1359 | 000310 | 000310 | TYPCNT: . =DECWRD+2                                  |  | ; THIS BYTE DETERMINES THE NUMBER OF WORDS<br>; TO BE TYPED   |
| 1360 | 000310 | 000    | SAVKBB: .BYTE 0                                      |  | ; THIS LOCATION IS USED TO SAVE THE CHARACTER<br>; HIT BY THE OPERATOR  |
| 1361 | 000311 | 000    |  |  | ; ALSO IS USED AS TEMP IN ROUTINE \$GTSIZ.  |
| 1362 |        |        |  |  |   |
| 1363 |        |        |  |  |   |
| 1364 |        |        |  |  |   |
| 1365 |        |        | .EVEN  |  |   |
| 1366 |        |        |  |  |   |
| 1367 |        |        |  |  |   |
| 1368 |        | 177560 | TKS= 177560  |  |   |
| 1369 |        | 177562 | \$KBB= 177562  |  |   |
| 1370 |        | 177564 | \$TPS= 177564  |  |   |
| 1371 |        | 177566 | \$TPB= 177566  |  |   |
| 1372 |        | 177572 | SRO= 177572  |  |   |
| 1373 | 000312 | 000377 | BAKPAT: .WORD 377                                    |  | ; BACKGROUND PATTERN WRITTEN TO MEMORY.   |
| 1374 |        |        |  |  |   |
| 1375 | 000314 | 000000 | SWAPAT: .WORD  |  |   |
| 1376 | 000316 | 000430 | RELBOT: BEGIN-50                                     |  | ; HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.   |
| 1377 | 000320 | 100000 | MINMEM: .WORD 100000                                 |  | ; FIRST ADDR FOR ECC IN CASE OF APT   |
| 1378 |        |        |  |  |   |
| 1379 |        |        |  |  |   |
| 1380 |        |        | :: *****   |  |   |
| 1381 | 000322 | 000000 | ; LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERATOR |  |   |
| 1382 | 000324 | 000000 | LOWTWO: 0  |  | ; HOLDS BITS 17:16 OF LOW TEST ADDRESS  |
| 1383 |        |        | LOWADD: 0  |  | ; HOLDS BITS 15:0 OF LOW TEST ADDRESS   |
| 1384 | 000326 | 000000 | HIGHTWO: 0   |  | ; HOLDS BITS 17:16 OF HIGH TEST ADDRESS   |
| 1385 | 000330 | 037776 | HIGHADD: 37776                                       |  | ; HOLDS BITS 15:0 OF HIGH TEST ADDRESS  |
| 1386 |        |        | :: *****   |  |   |
| 1387 |        |        |  |  |   |
| 1388 | 000332 | 000000 | \$HIMAX: 0   |  | ; HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY  |
| 1389 | 000334 | 017776 | \$MAXM: 17776  |  | ; HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY   |

```

1390
1391 000336 000000          MAXMEM: .WORD          ;MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1392
1393 000340 000000          SAVMAX: .WORD
1394 000342 000000          SAVR4:  .WORD
1395 000344 000000          SAVR5:  .WORD
1396
1397          ;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.
1398 000346 000500          SAVR6:  .WORD  BEGIN          ;CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1399 000350 000000          SAVLOC: 0          ;TEST 17 STORES ERROR INFO HERE
1400          ;
1401          ;GET HERE IF AN ILLEGAL TRAP TO LOC. 4 OCCURRED
1402          ;
1403 000352 012737 000001 000400  BUSER:  MOV      #1,2#MSGTY          ;TELL APT FATAL ERROR #000
1404 000360 000774          BR      BUSER          ;LOOP SO APT CAN GET INFO
1411
1412
1413          ;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT
    
```

1415 000400  
1416  
(1)  
(2)  
(1)  
(1) 000400  
(1) 000400 000000  
(1) 000402 000000  
(1) 000404 000000  
(1) 000406 000000  
(1) 000410 000000  
(1) 000412 000000  
(1) 000414 000000  
(1) 000416 000000  
(1) 000420  
(1) 000420 000  
(1) 000421 000  
(1) 000422 000000  
(1) 000424 000000  
(1) 000426 000000  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1)  
(1) 000430 000  
(1) 000431 000  
(1)  
(1)  
(1)  
(1) 000432 000000  
(1)  
(1) 000434 000  
(1) 000435 000  
(1) 000436 000000  
(1) 000440 000  
(1) 000441 000  
(1) 000442 000000  
(1) 000444 000  
(1) 000445 000  
(1) 000446 000000  
(1) 000450

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

1417  
1418  
1419  
1420  
1421 000500  
1422 000500 010706  
1423 000502 005746  
1424 000504 012704 016132  
1425 000510 010637 000346  
1426 000514 012737 000044 000024  
1427 000522 004767 014560

```
      ;*****  
      .SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.  
      .=500  
      BEGIN: MOV PC, SP ; SET UP STACK POINTER  
            TST -(SP) ; TO EQUAL BEGIN ADDRESS  
            MOV #ENDPROG, R4 ; PUT END OF PROG ADDRESS INTO R4  
            MOV SP, @SAVR6 ; SAVE SP FOR FUTURE USE  
            MOV #PWRDN, @24 ; PREPARE FOR FUTURE POWER DOWN  
            JSR PC, CLRM ; CLEAR MEM MGMT.
```

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

|      |        |        |        |         |        |                         |  |
|------|--------|--------|--------|---------|--------|-------------------------|--|
| 1428 | 000526 | 005037 | 000310 |         | CLR    | 2#TYP CNT               |  |
| 1429 | 000532 | 105737 | 000170 |         | TSTB   | 2#REL                   | :RELOCATED ?                                     |
| 1430 | 000536 | 100002 |        |         | BPL    | 4\$                     | :BR IF NOT                                       |
| 1431 | 000540 | 000167 | 001344 |         | JMP    | TSTSIZ                  |  |
| 1432 | 000544 | 005067 | 002146 | 4\$:    | CLR    | LDFLG                   |  |
| 1433 | 000550 | 012737 | 000566 | 000004  | MOV    | 87\$,2#4                | :PREPARE FOR NON-EXISTANT                        |
| 1434 | 000556 | 012737 | 000014 | 177746  | MOV    | 814,2#177746            | :11/60 CACHE CONTROL REG                         |
| 1435 | 000564 | 000402 |        |         | BR     | 6\$                     | :BR IF ALL OK                                    |
| 1436 | 000566 | 062706 | 000004 |         | ADD    | 84,SP                   | :RECOVER FROM TRAP                               |
| 1437 | 000572 | 012701 | 100000 |         | MOV    | 810000,R1               | :2ND 16K   |
| 1438 | 000576 | 005021 |        |         | CLR    | (R1)+                   | :CLEAR FIRST LOC                                 |
| 1439 | 000600 | 005011 |        |         | CLR    | (R1)                    | :CLEAR SECOND LOC                                |
| 1440 | 000602 | 012777 | 020004 | 002120  | MOV    | 820004,2CSRADR          | :DIAG BIT  |
| 1441 | 000610 | 005741 |        |         | TST    | -(R1)                   | :READ THE LOC TO GET CHECKBITS IF ANY            |
| 1442 | 000612 | 032777 | 007740 | 002110  | BIT    | 87740,2CSRADR           | :IF SET WE'RE IN MS11K                           |
| 1443 | 000620 | 001007 |        |         | BNE    | 5\$                     | :BR IF IN MEM UNDER TEST                         |
| 1444 | 000622 | 042767 | 020000 | 002060  | BIC    | 820000,ECCDIS           | :JUST IN CASE WE ARE TESTING                     |
| 1445 | 000630 | 042767 | 020000 | 002054  | BIC    | 820000,DIAGA            | :THE SECOND CSR (172134)                         |
| 1446 | 000636 | 000533 |        |         | BR     | SETSWR                  | :BR IF NOT IN MEM UNDER TEST                     |
| 1447 | 000640 | 005077 | 002064 |         | CLR    | 2CSRADR                 | :CLEAR CSR                                       |
| 1448 | 000644 | 012767 | 000001 | 002044  | MOV    | 81,LDFLG                | :INDICATE PROG IN MEM UNDER TEST                 |
| 1449 | 000652 | 052767 | 020000 | 002030  | BIS    | 820000,ECCDIS           | :DISABLE ECCINH IN LOW 16K                       |
| 1450 | 000660 | 052767 | 020000 | 002024  | BIS    | 820000,DIAGA            | :SAME FOR DIAG CHECK                             |
| 1451 | 000666 | 005001 |        |         | CLR    | R1                      |  |
| 1452 | 000670 | 005000 |        |         | CLR    | R0                      | :IT IS, CHECK FOR ERRORS                         |
| 1453 | 000672 | 005067 | 002030 |         | CLR    | INHREL                  |  |
| 1454 | 000676 | 005077 | 002026 |         | CLR    | 2CSRADR                 | :DON'T WANT TO TRAP                              |
| 1455 | 000702 | 005711 |        |         | TST    | (R1)                    | :READ THE LOC                                    |
| 1456 | 000704 | 032777 | 100000 | 002016  | BIT    | 810000,2CSRADR          | :CHECK FOR DOUBLE ERROR                          |
| 1457 | 000712 | 001403 |        |         | BEQ    | 8\$                     |  |
| 1458 | 000714 | 004767 | 013274 |         | JSR    | PC,FATERR               | :*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 000720 | 000001 |        |         | 1      |                         | :*****ERROR NUMBER 1*****                        |
| (1)  |        |        |        |         |        |                         |  |
| 1459 | 000722 | 005767 | 002000 |         | TST    | INHREL                  | :ANY ERRORS FOUND YET ?                          |
| 1460 | 000726 | 001012 |        |         | BNE    | 2\$                     | :YES, DON'T LOOK ANY FURTHER                     |
| 1461 | 000730 | 020127 | 000500 |         | CMP    | R1,#BEGIN               |  |
| 1462 | 000734 | 101007 |        |         | BHI    | 2\$                     |  |
| 1463 | 000736 | 032777 | 000020 | 001764  | BIT    | 820,2CSRADR             | :LOOKING FOR SBE'S IN 0-500                      |
| 1464 | 000744 | 001403 |        |         | BEQ    | 2\$                     | :BR IF NO ERRORS                                 |
| 1465 | 000746 | 012767 | 000001 | 001752  | MOV    | 81,INHREL               | :INHIBIT RELOCATION                              |
| 1466 | 000754 | 022701 | 016132 |         | CMP    | 8ENDPROG,R1             | :END OF PROGRAM YET?                             |
| 1467 | 000760 | 003403 |        |         | BLE    | 3\$                     | :BR IF YES                                       |
| 1468 | 000762 | 062701 | 000004 |         | ADD    | 84,R1                   | :NO POINT TO NEXT LOCATION                       |
| 1469 | 000766 | 000743 |        |         | BR     | 1\$                     | :KEY READING                                     |
| 1470 | 000770 | 005767 | 001732 |         | TST    | INHREL                  | :RELOCATION ALLOWED ?                            |
| 1471 | 000774 | 001415 |        |         | BEQ    | ONEPAS                  | :BR IF YES                                       |
| 1472 | 000776 | 004767 | 013430 |         | JSR    | PC,TPCRLF               | :PRINT ROUTINE                                   |
| 1473 | 001002 | 047516 | 051040 | 046105  | .ASCIZ | /NO RELOC-SBE IN 0-500/ |  |
|      | 001010 | 041517 | 051455 | 042502  |        |                         |  |
|      | 001016 | 044440 | 020116 | 026460  |        |                         |  |
|      | 001024 | 030065 | 000060 |         |        |                         |  |
| 1474 |        |        |        |         | .EVEN  |                         |  |
| 1475 | 001030 | 005000 |        |         | CLR    | R0                      | :INITIALIZE POINTER                              |
| 1476 | 001032 | 005737 | 000404 | ONEPAS: | TST    | 2#TESTN                 | :IS THIS THE FIRST PASS                          |
| 1477 | 001036 | 001402 |        |         | BEQ    | TSTRP                   | :BR IF YES (TEST TRAP CATCHERS)                  |
| 1478 | 001040 | 000167 | 000062 |         | JMP    | SETSWR                  | :SET UP SWREG                                    |

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

```

1479 001044 012704 016132 TSTRP: MOV #ENDPROG,R4 ;ADDRESS OF END OF PROGRAM
1480 001050 012700 000377 MOV #377,R0 ;
1481 001054 005001 CLR R1 ;POINT TO ADDRESS 0
1482 001056 012124 1S: MOV (R1)+,(R4)+ ;SAVE 0000 TO BEGIN-
1483 001060 020127 000400 CMP R1,#SMAIL ;BEGINNING OF ETABLE
1484 001064 103774 BLO 1S ;BR IF NOT DONE
1485 001066 005741 3S: TST -(R1) ;ADJUST POINTER TO TRAP VECTORS
1486 001070 010011 4S: MOV R0,(R1) ;WRITE 1'S AND 0'S
1487 001072 020011 CMP R0,(R1) ;NOW COMPARE
1488 001074 001403 BEQ 5S ;BR IF OK
1489 001076 004767 013112 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 001102 000002 2 ;*****ERROR NUMBER 2*****
1490
1491 001104 000300 5S: SWAB R0 ;
1492 001106 001370 BNE 4S ;BR IF BOTH BYTES NOT TESTED
1493 001110 005701 TST R1 ;HAVE WE REACHED THE BOTTOM YET
1494 001112 001365 BNE 3S ;BRANCH IF NO
1495 001114 012701 000400 MOV #SMAIL,R1 ;APT, RESTORE TO SMAIL
1496 001120 014441 6S: MOV -(R4),-(R1) ;HERE!
1497 001122 005701 TST R1 ;FINISHED?
1498 001124 001375 BNE 6S ;BR IF NOT
1499 001126 005077 001576 SETSWR: CLR @CSRADR ;CLEAR CSR
1500 001132 012700 000006 MOV #6,R0 ;ADDRESS OF PSW FOR TIMEOUT TRAP
1501 001136 012710 000340 MOV #340,(R0) ;SET UP TIME OUT TRAP PSW
1502 001142 012740 001170 MOV #2S,-(R0) ;AND RETURN ADDRESS
1503 001146 105737 000420 TSTB @SENV ;RUNNING UNDER APT
1504 001152 001403 BEQ 1S ;IF NOT, SET UP ADDRESS FOR SWR
1505 001154 012737 000422 002732 MOV #SSWREG,@SWR ;OTHERWISE, PREPARE TO USE APT SWR
1506 001162 005777 001544 1S: TST @SWR ;DOES SWITCH REG POINTED TO BY SWR EXIST
1507 001166 000410 BR APTSIZ ;YES, GO TO APTSIZ
1508 001170 062706 000004 2S: ADD #4,SP ;RESTORE STACK POINTER
1509 001174 012737 000176 002732 MOV #SWREG,@SWR ;PLACE ADDRESS OF SWITCH REG DESIGNED
1510 001202 012737 000174 002734 MOV #DSPREG,@DISPLAY ;FOR COMPUTERS NOT HAVING HARDWARE
1511 ;SWITCH REG AND RUNNING STAND ALONE.
1512 ;R4=# END PROG ON EXIT
1513
1514
1515
1516
1517 ;APTSIZ-THIS ROUTINE WILL SEARCH THE APT MEMORY ETABLE
1518 ;AND WHEN A NON ZERO TYPE IS FOUND WILL SETUP TO TEST TO
1519 ;GIVEN HIGH ADDRESS. IF APT DEFINES SIZE LO ADDRESS MUST -00000
1520
1521 001210 012703 000336 APTSIZ: MOV #MAXMEM,R3 ;POINT R3 TO MAXMEM
1522 001214 013737 000326 000332 MOV @#HIGHTWO,@#SHIMAX ;IN CASE NO SELF SIZING DONE
1523 001222 013737 000330 000334 MOV @#HIGHADD,@#SMAXM ;
1524 001230 105737 000421 TSTB @SENV ;DOES APT ALLOW SELF SIZING
1525 001234 100021 BPL TRYR ;BR IF YES
1526 001236 012701 000451 MOV #SMTYP4+4,R1 ;POINT R1 TO BLOCK TYPE 4+4
1527 001242 162701 000004 1S: SUB #4,R1 ;POINT TO NEXT BLOCK TYPE
1528 001246 105711 TSTB (R1) ;IS IT NON-ZERO
1529 001250 001006 BNE 2S ;BR IF YES (MEMORY EXISTS)
1530 001252 020127 000431 CMP R1,#SMTYP1 ;ALL APT BLOCK TYPES BEEN CHECKED
1531 001256 101371 BHI 1S ;BR IF NO
1532 001260 004767 012730 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
    
```

# E03

DZMLB.P11 MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-29  
 05-AUG-77 12:57

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

SEG 0031

```

(1) 001264 000003          3          ;*****ERROR NUMBER 3*****
(1)
1533
1534 001266 004767 013720      2$:   JSR   PC,GETADR      ;SET MAX APT ADDR INTO SMAXM +$HIMAX
1535 001272 004767 013714      JSR   PC,GETADR      ;ALSO INTO HIGHADD + HIGHTWO
1536 001276 000564          BRTPSZ: BR          TYPsiz      ;TYPE SIZE OF MEM UNDER TEST
1537
1538 001300 032777 000100 001424 TRYSR: BIT   #100,JSWR      ;USER DEFINED BOUNDARIES
1539 001306 001160          BNE          TYPsiz      ;BR IF YES (DON'T SIZE MEM)
1540
1541 001310 005067 001400      SLFSIZ: CLR   MSYES      ;INITIALIZE MSYES
1542 001314 012703 000324      MOV   #LOWADD,R3     ;GET READY FOR LOW TEST ADDRESS
1543 001320 005001          CLR   R1             ;FIRST TEST LOC (0)
1544 001322 012710 001514      MOV   #4$, (R0)      ;SET UP RETURN FROM TIMEOUT TRAP
1545 001326 005077 001376      1$:   CLR   @CSRADR      ;
1546 001332 011146          MOV   (R1),-(SP)     ;SAVE THE CONTENTS OF TEST LOC.
1547 001334 016146 000002      MOV   2(R1),-(SP)   ;
1548 001340 005011          CLR   (R1)          ;TO SET CHECK BITS OF MS11K
1549 001342 005061 000002      CLR   2(R1)         ;
1550 001346 016777 001340 001354 MOV   DIAGA,@CSRADR ;DIAGNOSTIC BIT
1551 001354 005711          TST   (R1)          ;READ TO GET CHECK BITS
1552 001356 017702 001346      MOV   @CSRADR,R2    ;GET CSR CONTENTS
1553 001362 005077 001342      CLR   @CSRADR      ;
1554 001366 012661 000002      MOV   (SP)+,2(R1)   ;RESTORE THE HI WORD
1555 001372 012611          MOV   (SP)+,(R1)   ;RESTORE THE LO WORD
1556 001374 042702 170037      BIC   #170037,R2    ;ONLY CARE ABOUT 11-5
1557 001400 005702          TST   R2            ;
1558 001402 001440          BEQ   3$            ;BR IF NO MS11K
1559 001404 005767 001304      TST   MSYES         ;MS11K-FIRST SUCCESSFUL TEST
1560 001410 001032          BNE   22$          ;NO, BRANCH
1561 001412 020127 100000      CMP   R1,#100000    ;IF THIS IS FIRST READABLE LOC
1562 001416 001015          BNE   11$          ;THEN FIRST ADDRESS IS 0000, ELSE BR
1563 001420 105737 000272      TSTB  @MMAVA        ;MEM MGMT AVAIL ?
1564 001424 001012          BNE   11$          ;YES, GO PUT ADDR AWAY
1565 001426 005767 001264      TST   LDFLG         ;RUNNING IN MEM UNDER TEST
1566 001432 001407          BEQ   11$          ;BR IF NOT
1567 001434 005037 000324      CLR   @LOWADD       ;CLEAR THESE LOCATIONS
1568 001440 005037 000322      CLR   @LOWTWO       ;INDICATE SOME MS11K FOUND
1569 001444 005267 001244      INC   MSYES         ;CONTINUE TO FIND HIGH LIMIT
1570 001450 000404          BR    2$           ;PLACE ADDR AWAY (START ADDR)
1571 001452 004767 013444      11$: JSR   PC,PUTADR     ;INDICATE SOME MS11 FOUND
1572 001456 005267 001232      INC   MSYES         ;FIRST TEST ADDRESS
1573 001462 010137 000320      2$:   MOV   R1,@MINMEM ;SAVE FOR RECOVERY FROM RELOC
1574 001466 010167 001230      MOV   R1,SAVMIN     ;POINT TO $MAXMEM
1575 001472 062703 000012      ADD   #12,R3        ;NEXT 4K BOUNDARY
1576 001476 062701 020000      22$: ADD   #20000,R1   ;GO TEST IT
1577 001502 000711          BR    1$           ;FOUND ANY MS11K YET
1578 001504 005767 001204      3$:   TST   MSYES         ;BR IF NO
1579 001510 001772          BEQ   22$          ;
1580 001512 000430          BR    7$           ;
1581 001514 062706 000004      4$:   ADD   #4,SP        ;RESTORE STACK POINTER AFTER TRAP
1582 001520 004767 013212      JSR   PC,MEMMNG     ;SERVICE MM IF AVAILABLE
1583 001524 105737 000272      TSTB  @MMAVA        ;SEE IF MM HAS TO BE TESTED
1584 001530 001421          BEQ   7$           ;BR IF NO MM
1585 001532 012710 001544      5$:   MOV   #6$, (R0)    ;SET UP RETURN ADDRESS FROM TRAP
1586 001536 012701 040000      MOV   #40000,R1     ;BEGIN CHECKING ABOVE 28K

```

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

|      |        |        |        |        |        |               |         |  |
|------|--------|--------|--------|--------|--------|---------------|---------|--|
| 1587 | 001542 | 000671 |        |        | BR     | 1\$           |         |  |
| 1588 | 001544 | 062706 | 000004 |        | ADD    | #4,SP         | 6\$:    | :RESTORE STACK POINTER                   |
| 1589 | 001550 | 022701 | 160000 |        | CMP    | #160000,R1    |         | :IF R1 DID NOT READ THE HIGHEST          |
| 1590 |        |        |        |        |        |               |         | :LOCATION POINTED TO BY PAR6,            |
| 1591 |        |        |        |        |        |               |         | :IT HAS REACHED MAX MS11K.               |
| 1592 | 001554 | 001007 |        |        | BNE    | 7\$           |         | :GO TO 7\$ IF DONE                       |
| 1593 | 001556 | 022737 | 007400 | 172352 | CMP    | #7400,#172352 |         | :CHECK PAR5 FOR MAX                      |
| 1594 | 001564 | 001403 |        |        | BEQ    | 7\$           |         | :BR IF DONE                              |
| 1595 | 001566 | 004767 | 013322 |        | JSR    | PC,UPMM       |         | :NOT DONE UPDATE MEM MGMT.               |
| 1596 | 001572 | 000757 |        |        | BR     | 5\$           |         |  |
| 1597 |        |        |        |        |        |               |         |  |
| 1598 |        |        |        |        |        |               |         |  |
| 1599 | 001574 | 024341 |        |        | CMP    | -(R3),-(R1)   | 7\$:    | :CAUSE R3 TO POINT TO SMAXM AND          |
| 1600 |        |        |        |        |        |               |         | :R1 TO MAX AVAILABLE MEMORY              |
| 1601 | 001576 | 020127 | 037776 |        | CMP    | R1,#37776     |         | :MEANS WE MOVED INTO PAR2                |
| 1602 | 001602 | 001014 |        |        | BNE    | 8\$           |         | :IF WE DID WE MUST ADJUST ADDRESS        |
| 1603 | 001604 | 005721 |        |        | TST    | (R1)+         |         | :RESTORE R1                              |
| 1604 | 001606 | 004767 | 013310 |        | JSR    | PC,PUTADR     |         | :PUT AWAY ORIGINAL ADDRESS               |
| 1605 | 001612 | 162713 | 000002 |        | SUB    | #2,(R3)       |         | :ADJUST BITS 15-0                        |
| 1606 | 001616 | 162703 | 000004 |        | SUB    | #4,R3         |         | :POINT TO HIGHADD                        |
| 1607 | 001622 | 004767 | 013274 |        | JSR    | PC,PUTADR     |         | :PUT IT AWAY                             |
| 1608 | 001626 | 162713 | 000002 |        | SUB    | #2,(R3)       |         | :ADJUST                                  |
| 1609 | 001632 | 000406 |        |        | BR     | TYPSIZ        |         | :AND CONTINUE TO TYPE THE SIZE           |
| 1610 | 001634 | 004767 | 013262 |        | JSR    | PC,PUTADR     | 8\$:    | :PLACE ADDR IN R1 AT LOCATIONS           |
| 1611 |        |        |        |        |        |               |         | :SMAXM AND SHIMAX                        |
| 1612 | 001640 | 162703 | 000004 |        | SUB    | #4,R3         |         | :POINT TO HIGHADD                        |
| 1613 | 001644 | 004767 | 013252 |        | JSR    | PC,PUTADR     |         | :PLACE ADDR IN R1 AT HIGHADD & HIGHTWO   |
| 1614 |        |        |        |        |        |               |         |  |
| 1615 |        |        |        |        |        |               |         |  |
| 1616 | 001650 | 012710 | 000352 |        | MOV    | #BUSER,(R0)   | TYPSIZ: | :SET UP VECTOR FOR FUTURE TRAP           |
| 1617 | 001654 | 005737 | 000406 |        | TST    | #SPASS        |         | :ONLY CARE ABOUT FIRST PASS              |
| 1618 | 001660 | 001035 |        |        | BNE    | SETSTK        |         | :BR IF NOT                               |
| 1619 | 001662 | 004767 | 012544 |        | JSR    | PC,TPCRLF     |         | :PRINT ROUTINE                           |
| 1620 | 001666 | 051503 | 020122 | 042101 | .ASCIZ | /CSR ADDR = / |         |  |
| 1621 | 001674 | 051104 | 036440 | 000040 |        |               |         |  |
| 1622 |        |        |        |        | .EVEN  |               |         |  |
| 1623 | 001702 | 012703 | 002664 |        | MOV    | #DATABUF,R3   |         | :NEED A LOCATION TO PRINT FROM           |
| 1624 | 001706 | 005023 |        |        | CLR    | (R3)+         |         | :CLEAR HIGH ORDER BITS                   |
| 1625 | 001710 | 016713 | 001014 |        | MOV    | CSRADR,(R3)   |         | :ADDRESS TO BE TYPED                     |
| 1626 | 001714 | 105237 | 000310 |        | INCB   | #TYPCNT       |         | :ONE WORD TO BE TYPED                    |
| 1627 | 001720 | 004767 | 012656 |        | JSR    | PC,TYPOCT     |         | :TYPE THE WORD                           |
| 1628 | 001724 | 010403 |        |        | MOV    | R4,R3         |         | :POINT R3 TO LOWEST AVAIL MEM            |
| 1629 | 001726 | 012701 | 000322 |        | MOV    | #LOWTWO,R1    |         |  |
| 1630 | 001732 | 004767 | 012462 |        | JSR    | PC,PCRLF      |         | :TYPE <CR><LF>                           |
| 1631 | 001736 | 004767 | 012630 |        | JSR    | PC,OCTTYP     |         | :TYPE LOW TEST ADDRESS                   |
| 1632 |        |        |        |        |        |               |         | :(LOWTWO + LOWADD)                       |
| 1633 | 001742 | 004767 | 012364 |        | JSR    | PC,STYPE      | TYPMEM: | :TYPE "--"                               |
| 1634 | 001746 | 000055 |        |        | .ASCIZ | 1-1           |         |  |
| 1635 |        |        |        |        | .EVEN  |               |         |  |
| 1636 | 001750 | 004767 | 012616 |        | JSR    | PC,OCTTYP     |         | :TYPE HIGHEST TEST ADDRESS               |
| 1637 |        |        |        |        |        |               |         | :(HIGHTWO + HIGHADD)                     |
| 1638 | 001754 | 012703 | 000326 |        | MOV    | #HIGHTWO,R3   | SETSTK: | :POINT R3 TO HIGH ORDER BITS OF HIGH MEM |
| 1639 | 001760 | 004767 | 013242 |        | JSR    | PC,SGTSIZ     |         | :GET THE BITS 13-17 ON TOP ADDRESS       |
| 1640 |        |        |        |        |        |               |         | :PLACED IN BITS 0-4 OF R2                |
| 1641 | 001764 | 010401 |        |        | MOV    | R4,R1         |         | :#ENDPROG                                |
| 1642 | 001766 | 062704 | 000022 |        | ADD    | #18.,R4       | 4\$:    | :APPEND THE ERROR STACK FOR MEMORY       |







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

```

1740 002432 005040          SS:  CLR  -(R0)          ;BEGIN CLEARING FROM TOP
1741 002434 020004          CMP   R0,R4          ;TO BOTTOM
1742 002436 101375          BHI  SS              ;
1743 002440 005077 000264  CLR   @CSRADR        ;RESTORE CSR
1744
1745                          ;ENTER HERE FROM TSTSCP ROUTINE AT END OF EACH TEST
1746
1747 002444 005037 000302  CLR   @#PASFLG       ;INIT SOFTEST PASS FLAG
1748 002450 110237 000404  MOVB  R2,@#STESTN    ;SET UP $STESTN WITH THE TEST
1749 002454 110277 000254  MOVB  R2,@DISPLAY    ;NUMBER TO BE EXECUTED & DISPLAYED
1750
1751 002460 005077 000244          LOOP: CLR   @CSRADR ;
1752 002464 032777 004000 000240 BIT   #4000,@$WR     ;ENABLE PRINTOUT OF SBE'S(INH ECC)
1753 002472 001403          BEQ   1$            ;BR IF NO
1754 002474 016777 000210 000226 MOV   ECCDIS,@CSRADR ;YES, INHIBIT ECC
1755 002502 010401          1$:  MOV   R4,R1        ;LOWEST ADDRESS UNDER TEST
1756 002504 010246          MOV   R2,-(SP)      ;SAVE R2
1757 002506 012703 000376          MOV   #376,R3      ;POINT R3 TO SCRATCH STACK
1758 002512 004767 012404          JSR   PC,$UTADR    ;GEN 18 BIT ADDRESS FROM R1
1759                          ;AND STORE IT IN (R3) AND (R3-2)
1760 002516 005743          TST  -(R3)         ;POINT R3 TO HIGH ORDER BITS
1761 002520 004767 012502          JSR   PC,$GTSIZ    ;PLACE BITS 13-17 OF ADDRESS BITS
1762                          ;INTO 0-4 OF R2.
1763 002524 010400          MOV   R4,R0        ;PLACE ADDRESS OF LOWEST LOC
1764                          ;UNDER TEST IN R0
1765 002526 010401          MOV   R4,R1        ;AND INTO R1
1766 002530 010403          MOV   R4,R3        ;AND INTO R3
1767 002532 012602          MOV   (SP)+,R2     ;RESTORE R2
1768 002534 006302          ASL  R2            ;DOUBLE IT
1769 002536 060702          ADD  PC,R2
1770 002540 066207 000004          ADD  TBL-,(R2),PC ;GO TO TEST # STORED IN BITS
1771                          ;3-0 OF SWITCH REG.
1772 002544 000172          TBL: TST0-TBL        ;RELATIVE ADDRESS OF TEST 0
1773 002546 000326          TST1-TBL        ;RELATIVE ADDRESS OF TEST 1
1774 002550 000620          TST2-TBL        ;RELATIVE ADDRESS OF TEST 2
1775 002552 001000          TST3-TBL        ;RELATIVE ADDRESS OF TEST 3
1776 002554 001232          TST4-TBL        ;RELATIVE ADDRESS OF TEST 4
1777 002556 001406          TST5-TBL        ;RELATIVE ADDRESS OF TEST 5
1778 002560 002056          TST6-TBL        ;RELATIVE ADDRESS OF TEST 6
1779 002562 002776          TST7-TBL        ;RELATIVE ADDRESS OF TEST 7
1780 002564 003772          TST10-TBL       ;RELATIVE ADDRESS OF TEST 10
1781 002566 004420          TST11-TBL       ;RELATIVE ADDRESS OF TEST 11
1782 002570 005052          TST12-TBL       ;RELATIVE ADDRESS OF TEST 12
1783 002572 005512          TST13-TBL       ;RELATIVE ADDRESS OF TEST 13
1784 002574 005624          TST14-TBL       ;RELATIVE ADDRESS OF TEST 14
1785 002576 005736          TST15-TBL       ;RELATIVE ADDRESS OF TEST 15
1786 002600 006570          TST16-TBL       ;RELATIVE ADDRESS OF TEST 16
1787 002602 006740          TST17-TBL       ;RELATIVE ADDRESS OF TEST 17
1788 002604 007076          RELOC-TBL       ;RELATIVE ADDRESS OF RELOC
1789
1790                          ;SCOPE ROUTINE
1791                          ;PROG COMES HERE AFTER EACH TEST AND
1792                          ;IF CTRL C GO TO ERROR HISTORY TYPEOUT.
1793                          ;IF SR=2000 (BIT 10) THEN HALT
1794                          ;IF SR=40000 (BIT 14) THEN LOOP ON TEST
1795                          ;DEFINED BY SR BITS <3:0>, ELSE GO ON
    
```

# J03

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-34  
 DZMMLB.P11 05-AUG-77 12:57 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0036

```

1796
1797
1798 002606 005077 000116 TSTSCP: CLR ;
1799 002612 105737 000420 TSTB ;
1800 002616 001002 BNE ;RUNNING UNDER APT?
1801 002620 004767 013230 JSR CNTSCP ;IF SO GO TO CNTSCP
1802 PC,CHECKC ;TEST FOR CTRL-C AND IF TYPED
1803 002624 113702 000404 CNTSCP: MOVB ;GO TO ERROR HISTORY AND HALT
1804 002630 005237 000410 INC ;PUT TEST NUMBER IN R2 LO BYTE
1805 002634 032777 002000 000070 BIT ;TELL APT EVERYTHING'S OK
1806 002642 001401 BEQ #2000,JSWR ;HALT AFTER EACH TEST?
1807 002644 000000 SWHALT: HALT TSTGO ;BRANCH IF NOT HALTING
1808 002646 032777 040000 000056 TSTGO: BIT #40000,JSWR ;LOOP ON TEST DESIRED?
1809 002654 001301 BNE LOOP ;YES, GO START SAME TEST
1810 002656 105202 INCB R2 ;
1811 002660 000167 177536 JMP CONT ;GO CONTINUE EXECUTING NEXT TEST
1812
1813 002664 000000 000000 DATBUF: .WORD 0,0
1814 002670 000000 000000 TSTDAT: .WORD 0,0
1815 002674 000000 000000 SBEMSK: .WORD 0,0
1816 002700 000000 000000 DBEMSK: .WORD 0,0
1817 002704 000000 000000 DATSAV: .WORD 0,0
1818 002710 000002 ECCDIS: .WORD 2
1819 002712 000004 DIAGA: .WORD 4
1820 002714 000000 MSYES: .WORD 0
1821 002716 000000 LDFLG: .WORD 0
1822 002720 000000 BASE: .WORD 0 ;OFFSET FOR RELOC
1823 002722 100000 SAVMIN: .WORD 100000 ;TO SAVE MINMEM FROM RELOC
1824 002724 000000 ERRFLG: .WORD 0 ;FLAG TO INDICATE FIRST ERROR FOR HEADER
1825 002726 000000 INHREL: .WORD 0 ;FLAG TO INDICATE NO RELOCATION ALLOWED
1826 002730 172136 CSRADR: 172136 ;DEFAULT CSR ADDRESS
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836 002732 177570 SWR: 177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
1837 002734 177570 DISPLAY: 177570 ;CHANGES TO DSPREG IF NO HARDWARE DISPLAY REG
1838
1839 ;*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
1840 ;* OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
1841 ;* LOCATION UNDER TEST
1842 ;*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
1843 ;* 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
1844 ;* LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
1845 ;* [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
1846
1847 002736 122737 000000 000404 TST0: CMPB #0,JSSTESTN ;CHECK FOR PROPER TEST SEQUENCE
1848 002744 001403 BEQ +10
1849 002746 004767 011242 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002752 000007 7 ;*****ERROR NUMBER 7*****
(1)
    
```

# K03

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-35  
 DZMMLB.P11 05-AUG-77 12:57 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0037

|      |        |        |               |       |       |               |   |
|------|--------|--------|---------------|-------|-------|---------------|---|
| 1850 | 002754 | 012703 | 177777        |       | MOV   | #177777,R3    |   |
| 1851 | 002760 | 010401 |               | 15:   | MOV   | R4,R1         | ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST |
| 1852 | 002762 | 010310 |               |       | MOV   | R3,(R0)       | ;SET ALL THE BITS AT (R0)                             |
| 1853 | 002764 | 020001 |               | 25:   | CMP   | R0,R1         | ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1     |
| 1854 | 002766 | 001417 |               |       | BEQ   | 45            | ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)              |
| 1855 | 002770 | 005711 |               |       | TST   | (R1)          | ;OTHERWISE CHECK (R1) FOR ALL 0'S                     |
| 1856 | 002772 | 001430 |               |       | BEQ   | 55            |   |
| 1857 | 002774 | 020311 |               |       | CMP   | R3,(R1)       | ;IF R1 IS NOT EQUAL TO R0 AND (R1)                    |
| 1858 |        |        |               |       |       |               | ;DOES NOT CONTAIN ALL 0'S THEN                        |
| 1859 |        |        |               |       |       |               | ;CHECK TO SEE IF (R0) = (R1)                          |
| 1860 | 002776 | 001004 |               |       | BNE   | 35            |   |
| 1861 | 003000 | 012767 | 000010 000042 |       | MOV   | #10,125       | ;*****ERROR NUMBER 10*****                            |
| (1)  |        |        |               |       |       |               |   |
| 1862 | 003006 | 000403 |               |       | BR    | 105           |   |
| 1863 | 003010 |        |               | 35:   |       |               |   |
| (1)  | 003010 | 012767 | 000011 000032 |       | MOV   | #11,125       | ;*****ERROR NUMBER 11*****                            |
| (1)  |        |        |               |       |       |               |   |
| 1864 | 003016 | 010046 |               | 105:  | MOV   | R0,-(SP)      | ;SAVE R0 ON STACK                                     |
| 1865 | 003020 | 105237 | 000275        |       | INCB  | #5ADERR       | ;AN ADDRESSING ERROR IS SUSPECTED                     |
| 1866 | 003024 | 000407 |               |       | BR    | 115           |   |
| 1867 | 003026 | 020311 |               | 45:   | CMP   | R3,(R1)       | ;CHECK (R1) FOR ALL 1'S                               |
| 1868 | 003030 | 001411 |               |       | BEQ   | 55            |   |
| 1869 | 003032 | 012767 | 000012 000010 |       | MOV   | #12,125       | ;*****ERROR NUMBER 12*****                            |
| (1)  |        |        |               |       |       |               |   |
| 1870 | 003040 | 010046 |               |       | MOV   | R0,-(SP)      | ;SAVE R0 ON STACK                                     |
| 1871 | 003042 | 010300 |               |       | MOV   | R3,R0         |   |
| 1872 | 003044 | 004767 | 010470        | 115:  | JSR   | PC,ERROR      | ;GO TO THE ERROR SUBROUTINE                           |
| 1873 | 003050 | 000000 |               | 125:  | .WORD |               | ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE      |
| 1874 | 003052 | 012600 |               |       | MOV   | (SP)+,R0      | ;RESTORE R0   |
| 1875 |        |        |               |       |       |               |   |
| 1876 | 003054 | 013706 | 000346        | 55:   | MOV   | #5SAVR6,SP    | ;RESTORE THE STACK POINTER                            |
| 1877 | 003060 | 062701 | 020000        |       | ADD   | #20000,R1     | ;CAUSE R1 TO POINT TO THE SAME CHIP                   |
| 1878 |        |        |               |       |       |               | ;LOCATION IN THE NEXT 4K BANK OF MEMORY               |
| 1879 |        |        |               |       |       |               | ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1         |
| 1880 | 003064 | 020105 |               |       | CMP   | R1,R5         | ;COMPARE R1 WITH THE HIGHEST MEMORY                   |
| 1881 |        |        |               |       |       |               | ;LOCATION WHICH IS STORED IN R5                       |
| 1882 | 003066 | 103736 |               |       | BLO   | 25            | ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 25      |
| 1883 |        |        |               |       |       |               |   |
| 1884 | 003070 | 000646 |               | END0: | BR    | TSTSCP        |   |
| 1885 |        |        |               |       |       |               | ;INITIAL DATA TEST                                    |
| 1886 |        |        |               |       |       |               | THIS TEST CHECKS THE DI/DO LINES BY                   |
| 1887 |        |        |               |       |       |               | SHIFTING A 1 THROUGH THE WORD.                        |
| 1888 |        |        |               |       |       |               |   |
| 1889 | 003072 | 122737 | 000001 000404 | TST1: | CMPB  | #1,#STESTN    | ;CHECK FOR PROPER SEQUENCE                            |
| 1890 | 003100 | 001403 |               |       | BEQ   | +10           | ;BR IF OK   |
| 1891 | 003102 | 004767 | 011106        |       | JSR   | PC,SEQERR     | ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT      |
| (1)  | 003106 | 000013 |               |       | 13    |               | ;*****ERROR NUMBER 13*****                            |
| (1)  |        |        |               |       |       |               |   |
| 1892 | 003110 | 013701 | 000320        |       | MOV   | #MINMEM,R1    | ;GET TEST ADDRESS                                     |
| 1893 | 003114 | 012767 | 000001 177542 | 15:   | MOV   | #1,DATBUF     | ;SET THE FIRST TEST BIT                               |
| 1894 | 003122 | 005067 | 177540        |       | CLR   | DATBUF+2      | ;CLEAR 2ND WORD                                       |
| 1895 | 003126 | 016711 | 177532        | 25:   | MOV   | DATBUF,(R1)   | ;WRITE TEST WORD 1                                    |
| 1896 | 003132 | 016761 | 177530 000002 |       | MOV   | DATBUF+2,(R1) | ;AND TEST WORD 2                                      |
| 1897 | 003140 | 026711 | 177520        |       | CMP   | DATBUF,(R1)   | ;NOW READ THEM  |
| 1898 | 003144 | 001405 |               |       | BEQ   | 45            | ;BR IF FIRST 16 OK                                    |
| 1899 | 003146 | 016700 | 177512        |       | MOV   | DATBUF,R0     | ;GET GOOD DATA FOR ERROR MSG                          |

## L03

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-36  
 DZMMLB.P11 05-AUG-77 12:57 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0038

```

1900 003152 004767 010362      JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
      (1) 003156 000014      14          ;*****ERROR NUMBER 14*****
      (1)
1901
1902 003160 026761 177502 000002 4$:  CMP    DATBUF+2,2(R1) ;NOW READ SECOND WORD
1903 003166 001405      BEQ    5$           ;BR IF OK
1904 003170 016700 177472      MOV    DATBUF+2,R0   ;GOOD DATA FOR ERROR MSG
1905 003174 004767 010340      JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
      (1) 003200 000015      15          ;*****ERROR NUMBER 15*****
      (1)
1906
1907
1908 003202 005767 177460      5$:  TST    DATBUF+2     ;HAS LAST BIT BEEN TESTED ?
1909 003206 100404      BMI    6$           ;MINUS MEANS BIT 31
1910 003210 004567 010140      JSR    R5,DASHL     ;NO, SHIFT TEST BIT LEFT
1911 003214 002664      .WORD  DATBUF       ;IN DATBUF
1912 003216 000743      BR     2$           ;GO WRITE NEW TEST DATA
1913
1914
1915 003220 012767 177776 177436 6$:  MOV    #177776,DATBUF ;NOW GOING TO SHIFT A 0 IN DATA DIRECTION
1916 003226 012767 177777 177432      MOV    #-1,DATBUF+2 ;PUT A 0 IN BIT 0
1917 003234 012702 002664      MOV    #DATBUF,R2   ;AND 1'S IN ALL OTHERS
1918 003240 066702 177454      ADD    BASE,R2      ;LOC OF WRITE DATA
1919 003244 016711 177414      66$:  MOV    DATBUF,(R1)   ;OFFSET IT IN CASE OF RELOC
1920 003250 016761 177412 000002      MOV    DATBUF+2,2(R1) ;WRITE THE DATA
1921 003256 026711 177402      CMP    DATBUF,(R1)  ;2 WORDS WORTH
1922 003262 001405      BEQ    7$           ;NOW READ FIRST WORD
1923 003264 016700 177374      MOV    DATBUF,R0    ;BR IF OK
1924 003270 004767 010244      JSR    PC,ERROR      ;GOOD DATA
      (1) 003274 000016      16          ;*ERROR* REPORT ERROR MESSAGE
      (1)          ;*****ERROR NUMBER 16*****
1925
1926 003276 026761 177364 000002 7$:  CMP    DATBUF+2,2(R1) ;NOW, READ SECOND WORD
1927 003304 001405      BEQ    8$           ;BR IF OK
1928 003306 016700 177354      MOV    DATBUF+2,R0   ;GOOD DATA
1929 003312 004767 010222      JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
      (1) 003316 000017      17          ;*****ERROR NUMBER 17*****
      (1)
1930
1931 003320 022762 077777 000002 8$:  CMP    #77777,2(R2)  ;TESTED BIT 31 YET?
1932 003326 001410      BEQ    10$          ;BR IF YES, WE'RE DONE
1933 003330 006162 000002      ROL    2(R2)        ;SHIFT DATBUF
1934 003334 006112      ROL    (R2)         ;AND DATBUF +2
1935 003336 103403      BCS    9$           ;
1936 003340 005362 000002      DEC    2(R2)        ;
1937 003344 000261      SEC    ;
1938 003346 000736      9$:  BR     66$         ;SET CARRY FOR NEXT ROL
1939 003350 062701 020000      10$:  ADD    #20000,R1   ;KEEP GOING
1940 003354 020105      CMP    R1,R5        ;NEXT 4K BANK
1941 003356 103656      BLO    1$           ;COMPARE AGAINST HIGHEST VIRTUAL MEM
1942
1943 003360 000167 177222      END1: JMP    TSTSCP
1944
1945
1946
1947 ;TEST2 THIS TEST CHECKS TO SEE THAT EACH ADDRESS
; BIT IN EACH 4K BANK CAN BE ASSERTED UNIQUELY.

```



# N03

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-38

SEQ 0040

DZMMLB.P11 05-AUG-77 12:57

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

|      |        |        |        |        |       |      |            |  |  |  |  |  |
|------|--------|--------|--------|--------|-------|------|------------|--|--|--|--|--|
| 1994 |        |        |        |        |       |      |            |  |  |  |  |  |
| 1995 | 003544 | 122737 | 000003 | 000404 | TST3: | CMPB | #3,#STESTN |  |  |  |  | ;CHECK FOR PROPER SEQUENCE NUMBER                |
| 1996 | 003552 | 001403 |        |        |       | BEQ  | 1\$        |  |  |  |  | ;OK BRANCH                                       |
| 1997 | 003554 | 004767 | 010434 |        |       | JSR  | PC,SEQERR  |  |  |  |  | ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 003560 | 000025 |        |        |       | 25   |            |  |  |  |  | ;*****ERROR NUMBER 25*****                       |
| (1)  |        |        |        |        |       |      |            |  |  |  |  |  |
| 1998 |        |        |        |        |       |      |            |  |  |  |  |  |
| 1999 | 003562 | 010446 |        |        | 1\$:  | MOV  | R4,-(SP)   |  |  |  |  | ;SAVE R4   |
| 2000 | 003564 | 013702 | 000320 |        |       | MOV  | #MINMEM,R2 |  |  |  |  | ;MINMEM IS LOWEST ADDRESS                        |
| 2001 | 003570 | 010203 |        |        |       | MOV  | R2,R3      |  |  |  |  | ;PUT IT IN R3 ALSO                               |
| 2002 | 003572 | 062702 | 000004 |        |       | ADD  | #4,R2      |  |  |  |  | ;POINT R2 TO LAST BYTE +1                        |
| 2003 | 003576 | 012713 | 177777 |        |       | MOV  | #-1,(R3)   |  |  |  |  | ;WRITE ALL ONES IN                               |
| 2004 | 003602 | 012763 | 177777 | 000002 |       | MOV  | #-1,2(R3)  |  |  |  |  | ;THE 4 TEST BYTES                                |
| 2005 | 003610 | 105013 |        |        | 2\$:  | CLRB | (R3)       |  |  |  |  | ;CLEAR A BYTE                                    |
| 2006 | 003612 | 013701 | 000320 |        |       | MOV  | #MINMEM,R1 |  |  |  |  | ;INITIALIZE R1 FOR EACH PASS                     |
| 2007 | 003616 | 020201 |        |        | 3\$:  | CMP  | R2,R1      |  |  |  |  | ;IF EQUAL, JUST READ LAST BYTE                   |
| 2008 | 003620 | 001456 |        |        |       | BEQ  | 5\$        |  |  |  |  | ;BR IF EQUAL                                     |
| 2009 | 003622 | 020301 |        |        |       | CMP  | R3,R1      |  |  |  |  | ;IS THIS THE BYTE OF ZEROS                       |
| 2010 | 003624 | 001022 |        |        |       | BNE  | 4\$        |  |  |  |  | ;BR IF NOT                                       |
| 2011 | 003626 | 122711 | 000000 |        |       | CMPB | #0,(R1)    |  |  |  |  | ;IT IS, COMPARE FOR ZEROS                        |
| 2012 | 003632 | 001415 |        |        |       | BEQ  | 6\$        |  |  |  |  |  |
| 2013 | 003634 | 012700 | 000377 |        |       | MOV  | #377,RO    |  |  |  |  | ;SET UP LO BYTE                                  |
| 2014 | 003640 | 032701 | 000001 |        |       | BIT  | #1,R1      |  |  |  |  | ;WHICH BYTE ARE WE TESTING ?                     |
| 2015 | 003644 | 001001 |        |        |       | BNE  | 9\$        |  |  |  |  | ;LO BYTE RO IS OK                                |
| 2016 | 003646 | 000300 |        |        |       | SWAB | RO         |  |  |  |  | ;HIGH BYTE, MAKE RO RIGHT                        |
| 2017 | 003650 | 010146 |        |        | 9\$:  | MOV  | R1,-(SP)   |  |  |  |  | ;SAVE R1   |
| 2018 | 003652 | 042701 | 000001 |        |       | BIC  | #1,R1      |  |  |  |  | ;FOR PRINTOUT                                    |
| 2019 | 003656 | 004767 | 007656 |        |       | JSR  | PC,ERROR   |  |  |  |  | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 003662 | 000026 |        |        |       | 26   |            |  |  |  |  | ;*****ERROR NUMBER 26*****                       |
| (1)  |        |        |        |        |       |      |            |  |  |  |  |  |
| 2020 | 003664 | 012601 |        |        |       | MOV  | (SP)+,R1   |  |  |  |  | ;RESTORE R1 FOR FURTHER TESTING                  |
| 2021 | 003666 | 005201 |        |        | 6\$:  | INC  | R1         |  |  |  |  | ;NEXT BYTE                                       |
| 2022 | 003670 | 000752 |        |        |       | BR   | 3\$        |  |  |  |  | ;RETURN  |
| 2023 | 003672 | 122711 | 177777 |        | 4\$:  | CMPB | #-1,(R1)   |  |  |  |  | ;ITS NOT THE BYTE OF 0'S, READ 1'S               |
| 2024 | 003676 | 001425 |        |        |       | BEQ  | 7\$        |  |  |  |  |  |
| 2025 | 003700 | 010304 |        |        |       | MOV  | R3,R4      |  |  |  |  | ;GET ADDRESS                                     |
| 2026 | 003702 | 042704 | 177775 |        |       | BIC  | #177775,R4 |  |  |  |  | ;TO FIND BYTE                                    |
| 2027 | 003706 | 030104 |        |        |       | BIT  | R1,R4      |  |  |  |  | ;IS IT SET IN TEST ADDRESS ?                     |
| 2028 | 003710 | 001003 |        |        |       | BNE  | 12\$       |  |  |  |  | ;NO, BRANCH CAUSE IT'S NOT IN SAME WORD          |
| 2029 | 003712 | 012700 | 177777 |        |       | MOV  | #-1,RO     |  |  |  |  | ;DATA SHOULD BE ALL ONES                         |
| 2030 | 003716 | 000406 |        |        |       | BR   | 10\$       |  |  |  |  | ;GO REPORT ERROR                                 |
| 2031 | 003720 | 012700 | 000377 |        | 12\$: | MOV  | #377,RO    |  |  |  |  | ;IT'S IN SAME WORD, BUT WHICH BYTE ?             |
| 2032 | 003724 | 032701 | 000001 |        |       | BIT  | #1,R1      |  |  |  |  | ;CHECK FOR LO BYTE                               |
| 2033 | 003730 | 001401 |        |        |       | BEQ  | 10\$       |  |  |  |  | ;NO, LEAVE RO ALONE                              |
| 2034 | 003732 | 000300 |        |        |       | SWAB | RO         |  |  |  |  | ;SWAP BYTES                                      |
| 2035 | 003734 | 010146 |        |        | 10\$: | MOV  | R1,-(SP)   |  |  |  |  | ;SAVE R1   |
| 2036 | 003736 | 042701 | 000001 |        |       | BIC  | #1,R1      |  |  |  |  | ;FOR PRINTOUT                                    |
| 2037 | 003742 | 004767 | 007572 |        |       | JSR  | PC,ERROR   |  |  |  |  | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 003746 | 000027 |        |        |       | 27   |            |  |  |  |  | ;*****ERROR NUMBER 27*****                       |
| (1)  |        |        |        |        |       |      |            |  |  |  |  |  |
| 2038 | 003750 | 012601 |        |        |       | MOV  | (SP)+,R1   |  |  |  |  | ;RESTORE R1                                      |
| 2039 | 003752 | 005201 |        |        | 7\$:  | INC  | R1         |  |  |  |  | ;MOVE TO NEXT BYTE                               |
| 2040 | 003754 | 000720 |        |        |       | BR   | 3\$        |  |  |  |  |  |
| 2041 | 003756 | 112713 | 177777 |        | 5\$:  | MOVB | #-1,(R3)   |  |  |  |  | ;RESTORE 1'S TO BYTE JUST TESTED                 |
| 2042 | 003762 | 005203 |        |        |       | INC  | R3         |  |  |  |  | ;INC TO NEXT BYTE                                |
| 2043 | 003764 | 020302 |        |        |       | CMP  | R3,R2      |  |  |  |  | ;WAS THAT JUST THE LAST ONE?                     |



```

2044 003766 001310          BNE      2$          ;BR IF NO
2045 003770 012604          MOV      (SP)+,R4    ;RESTORE R4
2046 003772 000167 176610  END3:    JMP      TSTSCP
2047
2048
2049
2050
2051
2052
2053
2054 003776 122737 000004 000404  TST4:  CMPB    #4,2*STESTN ;TEST FOR PROPER TEST SEQUENCE
2055 004004 001403          BEQ      1$
2056 004006 004767 010202          JSR      PC,SEQERR  ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004012 000030          JSR      30          ;*****ERROR NUMBER 30*****
(1)
2057
2058 004014 012702 004042          1$:     MOV      #2$,R2
2059 004020 066702 176674          ADD     BASE,R2    ;OFFSET FOR RELOC
2060 004024 010237 000004          MOV     R2,2*4
2061 004030 012701 172136          MOV     #172136,R1 ;FOR ERROR PRINTOUT
2062 004034 005777 176670          TST    2*CSRADR   ;TRY ACCESSING CSR
2063 004040 000403          BR      3$          ;IF WE GOT HERE ALL IS WELL
2064 004042
(1) 004042 004767 010146          2$:     JSR      PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004046 000031          JSR      31          ;*****ERROR NUMBER 31*****
(1)
2065
2066 004050 012700 000001          3$:     MOV      #1,R0    ;SET FIRST TEST BIT
2067 004054 012737 000352 000004  MOV     #BUSER,2*4 ;SET UP FOR FUTURE TRAPS
2068 004062 005767 176630          4$:     TST     LDFLG   ;RUNNING IN MEM UNDER TEST?
2069 004066 001004          BNE     5$          ;YES, BRANCH TO 5$
2070 004070 032700 050020          BIT     #50020,R0 ;NO, CHECK FOR UNUSED BITS
2071 004074 001017          BNE     7$
2072 004076 000403          BR      8$
2073 004100 032700 050026          5$:     BIT     #50026,R0
2074 004104 001013          BNE     7$
2075 004106 010077 176616          8$:     MOV     R0,2*CSRADR ;WRITE THE TEST BIT
2076 004112 017702 176612          MOV     2*CSRADR,R2 ;GET CSR CONTENTS
2077 004116 042702 000020          BIC     #20,R2     ;SINGLE ERRORS WILL CONFUSE US
2078 004122 020002          CMP     R0,R2     ;READ AND COMPARE
2079 004124 001403          BEQ     7$
2080 004126 004767 007406          JSR     PC,ERROR  ;*ERROR* REPORT ERROR MESSAGE
(1) 004132 000032          JSR     32          ;*****ERROR NUMBER 32*****
(1)
2081
2082 004134 006300          7$:     ASL     R0        ;SHIFT TO NEXT BIT
2083 004136 103351          BCC     4$          ;CONTINUE TESTING THRU BIT 15
2084 004140 012737 000352 000004  MOV     #BUSER,2*4 ;FOR ANY FUTURE TRAPS
2085 004146 000167 176434  END4:    JMP      TSTSCP
2086
2087
2088
2089
2090
2091
2092

```

```

; SINGLE BIT ERROR TEST
; THIS TEST FIRST CHECKS FOR UNCORRECTED DATA
; WITH THE INH ECC BIT SET, CHECKS THAT THE SINGLE
; ERROR INDICATE OCCURRED AND FINALLY, WITH ECC
; ENABLED, CHECKS FOR CORRECTED DATA.

```

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

|      |        |        |        |        |       |       |                    |   |
|------|--------|--------|--------|--------|-------|-------|--------------------|---|
| 2093 | 004152 | 122737 | 000005 | 000404 | TST5: | CMPB  | #5, #STESTN        | ;CHECK FOR PROPER TEST SEQUENCE                   |
| 2094 | 004160 | 001403 |        |        |       | BEQ   | 15                 |   |
| 2095 | 004162 | 004767 | 010026 |        |       | JSR   | PC, SEGERR         | ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 004166 | 000033 |        |        |       | 33    |                    | ; *****ERROR NUMBER 33*****                       |
| (1)  |        |        |        |        |       |       |                    |   |
| 2096 | 004170 | 005077 | 176534 |        | 15:   | CLR   | ACSRADR            |   |
| 2097 | 004174 | 012767 | 000001 | 176462 | T5A:  | MOV   | #1, DATBUF         | ; INITIAL DATA                                    |
| 2098 | 004202 | 005067 | 176460 |        |       | CLR   | DATBUF+2           | ; 32 BITS WORTH                                   |
| 2099 | 004206 | 012767 | 000001 | 176460 | 20\$: | MOV   | #1, SBEMSK         | ; INITIAL ERROR MASK                              |
| 2100 | 004204 | 005067 | 176456 |        |       | CLR   | SBEMSK+2           | ; 32 BITS WORTH                                   |
| 2101 | 004200 | 016767 | 176440 | 176442 | 30\$: | MOV   | DATBUF, TSTDAT     |   |
| 2102 | 004226 | 016767 | 176434 | 176436 |       | MOV   | DATBUF+2, TSTDAT+2 | ; TO SAVE ORIG DATA                               |
| 2103 | 004234 | 105737 | 000302 |        |       | TSTB  | #PASFLG            | ; COMP DATA ON SECOND PASS ONLY                   |
| 2104 | 004240 | 001404 |        |        |       | BEQ   | 40\$               | ; BR IF FIRST PASS                                |
| 2105 | 004242 | 005167 | 176422 |        |       | COM   | TSTDAT             | ; SECOND PASS, COMP BOTH WORDS                    |
| 2106 | 004246 | 005167 | 176420 |        |       | COM   | TSTDAT+2           |   |
| 2107 | 004252 | 016702 | 176412 |        | 40\$: | MOV   | TSTDAT, R2         |   |
| 2108 | 004256 | 016703 | 176410 |        |       | MOV   | TSTDAT+2, R3       |   |
| 2109 | 004262 | 012767 | 002670 | 006522 |       | MOV   | #TSTDAT, SOURCE    | ; SET UP ADDRESS FOR CHKGEN                       |
| 2110 | 004270 | 004767 | 006570 |        |       | JSR   | PC, CHKGEN         | ; GEN CHECKBITS ON TSTDAT                         |
| 2111 | 004274 | 004567 | 007104 |        |       | JSR   | RS, BITCOM         | ; BIT COMPLEMENT ROUTINE                          |
| 2112 | 004300 | 002674 |        |        |       | .WORD | SBEMSK             | ; MASK  |
| 2113 | 004302 | 002670 |        |        |       | .WORD | TSTDAT             | ; DATA  |
| 2114 | 004304 | 013701 | 000320 |        | 50\$: | MOV   | #MINMEM, R1        | ; FIRST TEST ADDRESS                              |
| 2115 | 004310 | 016777 | 176374 | 176412 |       | MOV   | ECCDIS, ACSRADR    | ; FORCE VALIDATE WITH ECC DISABLED                |
| 2116 | 004316 | 016721 | 176346 |        |       | MOV   | TSTDAT, (R1)+      | ; WRITE FIRST 16 BITS                             |
| 2117 | 004322 | 016700 | 006466 |        |       | MOV   | CHECK, R0          | ; GET CHECKBITS FROM CHKGEN                       |
| 2118 | 004326 | 056700 | 176360 |        |       | BIS   | DIAGA, R0          | ; SET DIAGNOSTIC CHECK A                          |
| 2119 | 004332 | 010077 | 176372 |        |       | MOV   | R0, ACSRADR        | ; LOAD CSR WITH IMAGE IN R0                       |
| 2120 | 004336 | 016711 | 176330 |        |       | MOV   | TSTDAT+2, (R1)     | ; WRITE SECOND 16 BITS AND                        |
| 2121 |        |        |        |        |       |       |                    | ; CHECK BITS. WE NOW HAVE CHECKBITS               |
| 2122 |        |        |        |        |       |       |                    | ; GENERATED ON DATBUF AND DATA WITH               |
| 2123 |        |        |        |        |       |       |                    | ; ONE BIT IN ERROR (AS PER SBEMSK).               |
| 2124 | 004342 | 016777 | 176342 | 176360 |       | MOV   | ECCDIS, ACSRADR    | ; DISABLE ERROR CORRECTING                        |
| 2125 | 004350 | 016700 | 176314 |        |       | MOV   | TSTDAT, R0         | ; SET UP GOOD DATA FOR ERROR REPT.                |
| 2126 | 004354 | 020041 |        |        |       | CMP   | R0, -(R1)          | ; READ THE LOW WORD                               |
| 2127 | 004356 | 001403 |        |        |       | BEQ   | 54\$               | ; BR IF OK  |
| 2128 | 004360 | 004767 | 007154 |        |       | JSR   | PC, ERROR          | ; *ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004364 | 000034 |        |        |       | 34    |                    | ; *****ERROR NUMBER 34*****                       |
| (1)  |        |        |        |        |       |       |                    |   |
| 2129 | 004366 | 016700 | 176300 |        | 55\$: | MOV   | TSTDAT+2, R0       | ; HIGH WORD                                       |
| 2130 | 004372 | 062701 | 000002 |        |       | ADD   | #2, R1             | ; POINT R1 TO SECOND 16 BITS                      |
| 2131 | 004376 | 020011 |        |        |       | CMP   | R0, (R1)           | ; READ THE HIGH WORD                              |
| 2132 | 004400 | 001403 |        |        |       | BEQ   | 56\$               | ; BR IF OK  |
| 2133 | 004402 | 004767 | 007132 |        |       | JSR   | PC, ERROR          | ; *ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004406 | 000035 |        |        |       | 35    |                    | ; *****ERROR NUMBER 35*****                       |
| (1)  |        |        |        |        |       |       |                    |   |
| 2134 | 004410 | 005077 | 176314 |        | 56\$: | CLR   | ACSRADR            | ; ENABLE ECC                                      |
| 2135 | 004414 | 010200 |        |        |       | MOV   | R2, R0             | ; THIS IS ORIGINAL DATA                           |
| 2136 | 004416 | 020041 |        |        |       | CMP   | R0, -(R1)          | ; SEE IF ITS BEEN CORRECTED                       |
| 2137 | 004420 | 001403 |        |        |       | BEQ   | 57\$               | ; IT SHOULD HAVE BEEN                             |
| 2138 | 004422 | 004767 | 007112 |        |       | JSR   | PC, ERROR          | ; *ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004426 | 000036 |        |        |       | 36    |                    | ; *****ERROR NUMBER 36*****                       |
| (1)  |        |        |        |        |       |       |                    |   |
| 2139 | 004430 | 032777 | 000020 | 176272 | 57\$: | BIT   | #20, ACSRADR       | ; CHECK IF SBE INDICATE SET                       |
| 2140 | 004436 | 001011 |        |        |       | BNE   | 58\$               | ; BR IF IT IS SET                                 |

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

|      |        |        |        |        |       |       |             |  |
|------|--------|--------|--------|--------|-------|-------|-------------|--|
| 2141 | 004440 | 010146 |        |        |       | MOV   | R1, -(SP)   | :SAVE R1   |
| 2142 | 004442 | 016701 | 176262 |        |       | MOV   | CSRADR, R1  | :GET CSR ADDRESS                                 |
| 2143 | 004446 | 012700 | 000020 |        |       | MOV   | #20, R0     | :READY R0 FOR PRINTOUT                           |
| 2144 | 004452 | 004767 | 007062 |        |       | JSR   | PC, ERROR   | :*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004456 | 000037 |        |        |       | 37    |             | :*****ERROR NUMBER 37*****                       |
| (1)  |        |        |        |        |       |       |             |  |
| 2145 | 004460 | 012601 |        |        |       | MOV   | (SP)+, R1   | :RESTORE R1                                      |
| 2146 | 004462 | 005077 | 176242 | 58\$:  |       | CLR   | CSRADR      | :RESTORE CSR                                     |
| 2147 | 004466 | 010300 |        |        |       | MOV   | R3, R0      | :HIGH WORD                                       |
| 2148 | 004470 | 062701 | 000002 |        |       | ADD   | #2, R1      | :POINT TO HIGH WORD                              |
| 2149 | 004474 | 020011 |        |        |       | CMP   | R0, (R1)    | :SEE IF ITS BEEN CORRECTED                       |
| 2150 | 004476 | 001403 |        |        |       | BEQ   | 59\$        | :BR IF OK  |
| 2151 | 004500 | 004767 | 007034 |        |       | JSR   | PC, ERROR   | :*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004504 | 000040 |        |        |       | 40    |             | :*****ERROR NUMBER 40*****                       |
| (1)  |        |        |        |        |       |       |             |  |
| 2152 | 004506 | 032777 | 000020 | 176214 | 59\$: | BIT   | #20, CSRADR | :SBE BIT SET ?                                   |
| 2153 | 004514 | 001010 |        |        |       | BNE   | 60\$        | :BR IF YES                                       |
| 2154 | 004516 | 010146 |        |        |       | MOV   | R1, -(SP)   | :SAVE R1   |
| 2155 | 004520 | 016701 | 176204 |        |       | MOV   | CSRADR, R1  | :GET ADDRESS OF CSR BEING TESTED                 |
| 2156 | 004524 | 012700 | 000020 |        |       | MOV   | #20, R0     | :FOR PRINTOUT                                    |
| 2157 | 004530 | 004767 | 007004 |        |       | JSR   | PC, ERROR   | :*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 004534 | 000041 |        |        |       | 41    |             | :*****ERROR NUMBER 41*****                       |
| (1)  |        |        |        |        |       |       |             |  |
| 2158 | 004536 | 005737 | 000406 |        | 60\$: | TST   | CSRADR      | :Q V ONLY ONE ITERATION                          |
| 2159 | 004542 | 001425 |        |        |       | BEQ   | ENDS        | :TEST FOR LAST MASK BIT                          |
| 2160 | 004544 | 005767 | 176126 |        |       | TST   | SBEMSK+2    | :MINUS MEANS BIT 31                              |
| 2161 | 004550 | 100404 |        |        |       | BMI   | 65\$        | :SHIFT LEFT ROUTINE FOR 32 BITS                  |
| 2162 | 004552 | 004567 | 006576 |        |       | JSR   | R5, DASHL   | :WORD TO BE SHIFTED                              |
| 2163 | 004556 | 002674 |        |        |       | .WORD | SBEMSK      |  |
| 2164 | 004560 | 000617 |        |        |       | BR    | 30\$        |  |
| 2165 | 004562 | 005767 | 176100 |        | 65\$: | TST   | DATBUF+2    | :LAST DATA BIT ?                                 |
| 2166 | 004566 | 100404 |        |        |       | BMI   | 70\$        | :WHICH IS BIT 31                                 |
| 2167 | 004570 | 004567 | 006560 |        |       | JSR   | R5, DASHL   | :SHIFT IT  |
| 2168 | 004574 | 002664 |        |        |       | .WORD | DATBUF      |  |
| 2169 | 004576 | 000603 |        |        |       | BR    | 20\$        |  |
| 2170 | 004600 | 105737 | 000302 |        | 70\$: | TSTB  | CSRADR      | :FIRST OR SECOND PASS ?                          |
| 2171 | 004604 | 001004 |        |        |       | BNE   | ENDS        | :NON ZERO MEANS WE'RE DONE                       |
| 2172 | 004606 | 105237 | 000302 |        |       | INCB  | CSRADR      | :NOT DONE, GO DO SECOND PASS                     |
| 2173 | 004612 | 000167 | 177356 |        |       | JMP   | TSA         |  |
| 2174 | 004616 | 000167 | 175764 |        | ENDS: | JMP   | TSTSCP      |  |
| 2175 |        |        |        |        |       |       |             | :SINGLE BIT ERROR TEST TO INSURE THAT A WRITE    |
| 2176 |        |        |        |        |       |       |             | :BYTE CLEARS SINGLE BIT ERRORS.                  |
| 2177 |        |        |        |        |       |       |             |  |
| 2178 | 004622 | 122737 | 000006 | 000404 | TST6: | CMPB  | #6, CSRADR  | :CHECK FOR PROPER TEST SEQUENCE                  |
| 2179 | 004630 | 001403 |        |        |       | BEQ   | 1\$         |  |
| 2180 | 004632 | 004767 | 007356 |        |       | JSR   | PC, SEQERR  | :*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 004636 | 000042 |        |        |       | 42    |             | :*****ERROR NUMBER 42*****                       |
| (1)  |        |        |        |        |       |       |             |  |
| 2181 | 004640 | 010701 |        |        | 1\$:  | MOV   | PC, R1      | :GET PC  |
| 2182 | 004642 | 012700 | 005534 |        |       | MOV   | #END6, R0   | :END OF THIS TEST                                |
| 2183 | 004646 | 066700 | 176046 |        |       | ADD   | BASE, R0    | :FOR OFFSET                                      |
| 2184 | 004652 | 005711 |        |        | 2\$:  | TST   | (R1)        | :READ A LOC                                      |
| 2185 | 004654 | 032777 | 000020 | 176046 |       | BIT   | #20, CSRADR | :LOOKING FOR SBE'S IN THIS TEST                  |
| 2186 | 004662 | 001403 |        |        |       | BEQ   | 3\$         | :BR IF NO ERROR                                  |
| 2187 | 004664 | 005267 | 000650 |        |       | INC   | T6FLG       | :INDICATE WITH FLAG                              |
| 2188 | 004670 | 000404 |        |        |       | BR    | 4\$         | :AND BRANCH                                      |

|      |        |        |        |        |      |       |                   |  |
|------|--------|--------|--------|--------|------|-------|-------------------|--|
| 2189 | 004672 | 062701 | 000004 |        | 3S:  | ADD   | #4,R1             | : MOVE TO NEXT WORD                    |
| 2190 | 004676 | 020100 |        |        |      | CMP   | R1,R0             | : END OF TEST 6 YET ?                  |
| 2191 | 004700 | 103764 |        |        |      | BLO   | 2S                | : NO, BRANCH                           |
| 2192 | 004702 | 004467 | 006420 |        | 4S:  | JSR   | R4,SAVOT4         | : SAVE REGS R0 TO R4                   |
| 2193 | 004706 | 005077 | 176016 |        |      | CLR   | ACSRADR           | : CLEAR CSR                            |
| 2194 | 004712 | 012767 | 000001 | 175744 |      | MOV   | #1,DATBUF         | : INITIAL DATA                         |
| 2195 | 004720 | 005067 | 175742 |        |      | CLR   | DATBUF+2          | : 32 BITS WORTH                        |
| 2196 | 004724 | 012767 | 000001 | 175742 | T6A: | MOV   | #1,SBEMSK         | : INITIAL ERROR MASK                   |
| 2197 | 004732 | 005067 | 175740 |        |      | CLR   | SBEMSK+2          | : 32 BITS WORTH                        |
| 2198 | 004736 | 016767 | 175722 | 175724 | T6B: | MOV   | DATBUF,TSTDAT     | : SAVE ORIGINAL DATA                   |
| 2199 | 004744 | 016767 | 175716 | 175720 |      | MOV   | DATBUF+2,TSTDAT+2 | : BOTH WORDS                           |
| 2200 | 004752 | 016767 | 175706 | 175724 |      | MOV   | DATBUF,DATSAV     | : IN CASE PROG HAS SBE                 |
| 2201 | 004760 | 016767 | 175702 | 175720 |      | MOV   | DATBUF+2,DATSAV+2 | : DITTO                                |
| 2202 | 004766 | 012767 | 002670 | 006016 |      | MOV   | #TSTDAT,SOURCE    | : NEED ADDRESS FOR CHKGEN              |
| 2203 | 004774 | 004767 | 006064 |        |      | JSR   | PC,CHKGEN         | : GENERATE CHECK BITS                  |
| 2204 | 005000 | 004567 | 006400 |        |      | JSR   | R5,BITCOM         | : BIT COM ROUTINE                      |
| 2205 | 005004 | 002674 |        |        |      | .WORD | SBEMSK            | : MASK FOR COMPLEMENTING               |
| 2206 | 005006 | 002670 |        |        |      | .WORD | TSTDAT            | : WORD TO BE COMPLEMENTED              |
| 2207 | 005010 | 013701 | 000320 |        | 30S: | MOV   | #MINMEM,R1        | : FIRST TEST ADDRESS                   |
| 2208 | 005014 | 010104 |        |        |      | MOV   | R1,R4             | : PUT IT IN R4 ALSO                    |
| 2209 | 005016 | 016777 | 175666 | 175704 |      | MOV   | ECCDIS,ACSRADR    | : INHIBIT ECC                          |
| 2210 | 005024 | 016724 | 175640 |        |      | MOV   | TSTDAT,(R4)+      | : WRITE 16 BITS                        |
| 2211 | 005030 | 016703 | 005760 |        |      | MOV   | CHECK,R3          | : GET CHECKBITS                        |
| 2212 | 005034 | 056703 | 175652 |        |      | BIS   | DIAGA,R3          | : SET DIAGNOSTIC A BIT                 |
| 2213 | 005040 | 010377 | 175664 |        |      | MOV   | R3,ACSRADR        | : LOAD CSR WITH NEW IMAGE              |
| 2214 | 005044 | 016714 | 175622 |        |      | MOV   | TSTDAT+2,(R4)     | : WRITE HIGH WORD+CHECKBITS            |
| 2215 | 005050 | 005077 | 175654 |        |      | CLR   | ACSRADR           | : IT'S DANGEROUS IF WE DON'T           |
| 2216 | 005054 | 012702 | 002674 |        |      | MOV   | #SBEMSK,R2        | : ADDRESS OF ERROR MASK                |
| 2217 | 005060 | 066702 | 175634 |        |      | ADD   | BASE,R2           |  |
| 2218 | 005064 | 162704 | 000002 |        |      | SUB   | #2,R4             | : ADJUST ADDRESS                       |
| 2219 | 005070 | 112714 | 177777 |        | 40S: | MOVB  | #-1,(R4)          | : WRITE A BYTE OF 1'S                  |
| 2220 | 005074 | 132712 | 177777 |        |      | BITB  | #-1,(R2)          |  |
| 2221 | 005100 | 001476 |        |        |      | BEQ   | 60S               |  |
| 2222 | 005102 | 005767 | 000432 |        |      | TST   | T6FLG             | : SINGLE ERRORS IN TST6 AREA ?         |
| 2223 | 005106 | 001016 |        |        |      | BNE   | 45S               | : BR IF YES                            |
| 2224 | 005110 | 005077 | 175614 |        |      | CLR   | ACSRADR           | : CLEAR CSR                            |
| 2225 | 005114 | 105714 |        |        |      | TSTB  | (R4)              | : READ                                 |
| 2226 | 005116 | 032777 | 000020 | 175604 |      | BIT   | #20,ACSRADR       | : SINGLE ERROR INDICATE SET ?          |
| 2227 | 005124 | 001453 |        |        |      | BEQ   | 50S               |  |
| 2228 | 005126 | 016701 | 175576 |        |      | MOV   | CSRADR,R1         | : FOR PRINTOUT                         |
| 2229 | 005132 | 005000 |        |        |      | CLR   | R0                |  |
| 2230 | 005134 | 004767 | 006400 |        |      | JSR   | PC,ERROR          | : #ERROR* REPORT ERROR MESSAGE         |
| (1)  | 005140 | 000043 |        |        |      | 43    |                   | : *****ERROR NUMBER 43*****            |
| (1)  |        |        |        |        |      |       |                   |  |
| 2231 | 005142 | 000444 |        |        | 45S: | BR    | 50S               | : CONTINUE                             |
| 2232 | 005144 | 010446 |        |        |      | MOV   | R4,-(SP)          | : NEED A SCRATCH LOC                   |
| 2233 | 005146 | 163716 | 000320 |        |      | SUB   | #MINMEM,(SP)      | : TO FIND BYTE OFFSET                  |
| 2234 | 005152 | 012703 | 002704 |        |      | MOV   | #DATSAV,R3        | : GET ADDRESS                          |
| 2235 | 005156 | 066703 | 175536 |        |      | ADD   | BASE,R3           | : OFFSET FOR RELOC                     |
| 2236 | 005162 | 062603 |        |        |      | ADD   | (SP)+,R3          | : ADD BYTE COUNT                       |
| 2237 | 005164 | 112713 | 177777 |        |      | MOVB  | #-1,(R3)          | : WRITE BYTE OF 1'S LIKE TEST LOCATION |
| 2238 | 005170 | 016777 | 175516 | 175532 |      | MOV   | DIAGA,ACSRADR     | : DIAG BIT                             |
| 2239 | 005176 | 105714 |        |        |      | TSTB  | (R4)              | : READ THE BYTE TO GET THE CHECKBITS   |
| 2240 |        |        |        |        |      |       |                   |  |
| 2241 | 005200 | 017703 | 175524 |        |      | MOV   | ACSRADR,R3        | : GET CHECKBITS                        |
| 2242 | 005204 | 042703 | 170037 |        |      | BIC   | #170037,R3        | : CLEAR OTHER BITS                     |

# F04

DZML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-43  
 DZMLB.P11 05-AUG-77 12:57

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

SEQ 0045

|      |        |        |        |        |      |                 |                                  |
|------|--------|--------|--------|--------|------|-----------------|----------------------------------|
| 2243 | 005210 | 005077 | 175514 |        | CLR  | @CSRADR         | ;                                |
| 2244 | 005214 | 012767 | 002704 | 005570 | MOV  | @DATSAV, SOURCE | : FOR CHKGEN                     |
| 2245 | 005222 | 004767 | 005636 |        | JSR  | PC, CHKGEN      | : GENERATE CHECKBITS             |
| 2246 | 005226 | 020367 | 005562 |        | CMP  | R3, CHECK       | : COMPARE ACTUAL VS. EXPECTED    |
| 2247 | 005232 | 001410 |        |        | BEQ  | 50\$            | : BR IF OK                       |
| 2248 | 005234 | 010300 |        |        | MOV  | R3, R0          | : GOOD DATA                      |
| 2249 | 005236 | 012701 | 013014 |        | MOV  | @CHECK, R1      | : ADDRESS OF CHECKBITS           |
| 2250 | 005242 | 066701 | 175452 |        | ADD  | BASE, R1        | : PLUS OFFSET                    |
| 2251 | 005246 | 004767 | 006266 |        | JSR  | PC, ERROR       | : ERROR ROUTINE                  |
| 2252 | 005252 | 000043 |        |        | 43   |                 |                                  |
| 2253 | 005254 | 122714 | 177777 | 50\$:  | CMP# | #-1, (R4)       | : CHECK DATA                     |
| 2254 | 005260 | 001461 |        |        | BEQ  | 70\$            | : BR IF OK                       |
| 2255 | 005262 | 010401 |        |        | MOV  | R4, R1          | : GET R4                         |
| 2256 | 005264 | 042701 | 000001 |        | BIC  | #1, R1          | : TO MAKE EVEN ADDR FOR PRINTOUT |
| 2257 | 005270 | 004767 | 006244 |        | JSR  | PC, ERROR       | : *ERROR* REPORT ERROR MESSAGE   |
| (1)  | 005274 | 000044 |        |        | 44   |                 | : *****ERROR NUMBER 44*****      |
| (1)  |        |        |        |        |      |                 |                                  |
| 2258 | 005276 | 005767 | 000236 | 60\$:  | TST  | T6FLG           | : SINGLE ERRORS IN TST6 AREA ?   |
| 2259 | 005302 | 001011 |        |        | BNE  | 65\$            | : BR IF YES                      |
| 2260 | 005304 | 105714 |        |        | TSTB | (R4)            | : READ THE BYTE                  |
| 2261 | 005306 | 032777 | 000020 | 175414 | BIT  | #20, @CSRADR    | : SBE ERROR BIT SET ?            |
| 2262 | 005314 | 001357 |        |        | BNE  | 50\$            | : SHOULD BE SET, BR IF OK        |
| 2263 | 005316 | 004767 | 006216 |        | JSR  | PC, ERROR       | : *ERROR* REPORT ERROR MESSAGE   |
| (1)  | 005322 | 000045 |        |        | 45   |                 | : *****ERROR NUMBER 45*****      |
| (1)  |        |        |        |        |      |                 |                                  |
| 2264 | 005324 | 000437 |        |        | BR   | 70\$            | : CONTINUE                       |
| 2265 | 005326 | 010446 |        | 65\$:  | MOV  | R4, -(SP)       | : SAVE R4                        |
| 2266 | 005330 | 163716 | 000320 |        | SUB  | @MINMEM, (SP)   | : FOR BYTE OFFSET                |
| 2267 | 005334 | 012703 | 002704 |        | MOV  | @DATSAV, R3     | : ADDRESS OF DATA                |
| 2268 | 005340 | 066703 | 175354 |        | ADD  | BASE, R3        | : FOR RELOC                      |
| 2269 | 005344 | 062603 |        |        | ADD  | (SP)+, R3       | : BYTE OFFSET                    |
| 2270 | 005346 | 112713 | 177777 |        | MOVB | #-1, (R3)       | : WRITE A BYTE OF 1'S            |
| 2271 | 005352 | 016777 | 175334 | 175350 | MOV  | DIAGA, @CSRADR  | : DIAG BIT                       |
| 2272 | 005360 | 105714 |        |        | TSTB | (R4)            | : READ THE BYTE                  |
| 2273 | 005362 | 017746 | 175342 |        | MOV  | @CSRADR, -(SP)  | : GETCONTENTS                    |
| 2274 | 005366 | 005077 | 175336 |        | CLR  | @CSRADR         | : CLEAR THE CSR                  |
| 2275 | 005372 | 042716 | 170037 |        | BIC  | #170037, (SP)   | : ONLY WANT CHECKBITS            |
| 2276 | 005376 | 012767 | 002704 | 005406 | MOV  | @DATSAV, SOURCE | : FOR CHKGEN                     |
| 2277 | 005404 | 004767 | 005454 |        | JSR  | PC, CHKGEN      | : GENERATE CHECKBITS ON DATA     |
| 2278 | 005410 | 022667 | 005400 |        | CMP  | (SP)+, CHECK    | : COMPARE ACTUAL VS. EXPECTED    |
| 2279 | 005414 | 001403 |        |        | BEQ  | 70\$            | : BR IF OK                       |
| 2280 | 005416 | 004767 | 006116 |        | JSR  | PC, ERROR       | : ERROR ROUTINE                  |
| 2281 | 005422 | 000045 |        |        | 45   |                 |                                  |
| 2282 | 005424 | 132712 | 177777 | 70\$:  | BITB | #-1, (R2)       | : CHECK FOR LAST BYTE            |
| 2283 | 005430 | 001012 |        |        | BNE  | 80\$            | :                                |
| 2284 | 005432 | 005202 |        |        | INC  | R2              |                                  |
| 2285 | 005434 | 005204 |        |        | INC  | R4              | : MOVE TO NEXT BYTE              |
| 2286 | 005436 | 013701 | 000320 |        | MOV  | @MINMEM, R1     | : FIRST TEST ADDRESS             |
| 2287 | 005442 | 032704 | 000002 |        | BIT  | #2, R4          | : TEST FOR LOWER WORD            |
| 2288 | 005446 | 001610 |        |        | BEQ  | 40\$            | : BR IF IT'S LOW 16 BITS         |
| 2289 | 005450 | 062701 | 000002 |        | ADD  | #2, R1          | : ADJUST POINTER FOR ERROR REPT. |
| 2290 | 005454 | 000605 |        |        | BR   | 40\$            |                                  |
| 2291 | 005456 | 005737 | 000406 | 80\$:  | TST  | @\$PASS         | : FIRST PASS ?                   |
| 2292 | 005462 | 001420 |        |        | BEQ  | 100\$           | : BR IF YES                      |
| 2293 | 005464 | 005767 | 175206 |        | TST  | SBEMSK+2        | : LAST ERROR BIT ?               |
| 2294 | 005470 | 100405 |        |        | BMI  | 90\$            | : MINUS MEANS BIT 31             |

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

```

2295 005472 004567 005656      JSR    R5,DASHL      ;DOUBLE ARITHMETIC SHIFT LEFT
2296 005476 002674              .WORD    SBEMSK      ;ON THE ERROR MASK
2297 005500 000167 177232      JMP     T6B
2298 005504 005767 175156      90$:   TST    DATBUF+2  ;LAST DATA BIT ?
2299 005510 100405              BMI     100$         ;MINUS = BIT 31
2300 005512 004567 005636      JSR    R5,DASHL
2301 005516 002664              .WORD    DATBUF      ;SHIFT IT LEFT
2302 005520 000167 177200      JMP     T6A
2303 005524 004767 005610      100$:  JSR    PC,RSTOT4   ;RESTORE REGS R0 TO R4
2304 005530 005067 000004      CLR    T6FLG        ;FOR NEXT TIME
2305 005534 000167 175046      END6:  JMP     TSTSCP
2306 005540 000000      T6FLG: .WORD    0
2307
2308
2309
2310
2311 005542 122737 000007 000404  TST7:  CMPB   07,00$TESTN
2312 005550 001403              BEQ
2313 005552 004767 006436      JSR    PC,SEGERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 005556 000046              46          ;*****ERROR NUMBER 46*****
(1)
2314 005560 005077 175144      1$:   CLR    0CSRADR      ;CLEAR CSR
2315 005564 013701 000320      MOV    0$MINMEM,R1
2316 005570 004467 005532      JSR    R4,SAVOT4   ;SAVE REGS R0 TO R4
2317 005574 005067 175064      10$:  CLR    DATBUF      ;MAKE INITIAL DATA
2318 005600 005067 175062      CLR    DATBUF+2    ;ALL ZEROS
2319 005604 012767 000001 175062  20$:  MOV    01,SBEMSK   ;INITIAL SINGLE ERROR MASK
2320 005612 005067 175060      CLR    SBEMSK+2    ;SECOND WORD
2321 005616 012767 000001 175054  30$:  MOV    01,DBEMSK   ;INITIAL DOUBLE ERROR MASK
2322 005624 005067 175052      CLR    DBEMSK+2    ;32 BITS HERE ALSO
2323 005630 016767 175030 175032  35$:  MOV    DATBUF,TSTDAT
2324 005636 016767 175024 175026      MOV    DATBUF+2,TSTDAT+2
2325 005644 105737 000302      TSTB   0$PASFLG   ;NO COMPLEMENTING FIRST PASS
2326 005650 001404              BEQ    40$
2327 005652 005167 175012      COM    TSTDAT      ;COMP FIRST WORD
2328 005656 005167 175010      COM    TSTDAT+2    ;SECOND WORD
2329 005662 005077 175042      40$:  CLR    0CSRADR
2330 005666 026767 175002 175004      CMP    SBEMSK,DBEMSK ;CAN'T HAVE THE SAME ERROR BIT SET
2331 005674 001004              BNE    45$         ;IN BOTH MASKS
2332 005676 026767 174774 174776      CMP    SBEMSK+2,DBEMSK+2 ;COULD BE EQUAL IN SECOND WORD
2333 005704 001502              BEQ    65$         ;GO MAKE THEM NOT EQUAL
2334 005706 012767 002670 005076  45$:  MOV    0TSTDAT,SOURCE ;SOURCE ADDRESS FOR CHKGEN
2335 005714 004767 005144      JSR    PC,CHKGEN   ;GO GENERATE CHECK BITS
2336 005720 004567 005460      JSR    R5,BITCOM   ;FORCE SINGLE ERROR
2337 005724 002674              .WORD    SBEMSK    ;AS PER SBEMSK
2338 005726 002670              .WORD    TSTDAT    ;ON DATA IN TSTDAT
2339 005730 004567 005450      JSR    R5,BITCOM   ;FORCING DOUBLE ERROR
2340 005734 002700              .WORD    DBEMSK    ;THIS MASK INDICATES SECOND BIT IN ERROR
2341 005736 002670              .WORD    TSTDAT    ;SAME DATA WORD
2342 005740 016704 005050      MOV    CHECK,R4   ;GET CHECKBITS
2343 005744 056704 174742      BIS    DIAGA,R4   ;SET DIAGNOSTIC A BIT
2344 005750 016777 174734 174752      MOV    ECCDIS,0CSRADR ;INHIBIT ECC
2345 005756 016721 174706      MOV    TSTDAT,(R1)+ ;WRITE 16 BITS
2346 005762 010477 174742      MOV    R4,0CSRADR ;LOAD CSR
2347 005766 016711 174700      MOV    TSTDAT+2,(R1) ;WRITE HIGH WORD
2348 005772 005077 174732      CLR    0CSRADR    ;CLEAR CSR AGAIN

```

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

|      |        |        |        |        |       |                 |                               |
|------|--------|--------|--------|--------|-------|-----------------|-------------------------------|
| 2349 | 005776 | 162701 | 000002 |        | SUB   | #2,R1           | :ADJUST TEST ADDRESS          |
| 2350 | 006002 | 005711 |        |        | TST   | (R1)            | :READ THE LOCATION            |
| 2351 | 006004 | 032777 | 100000 | 174716 | BIT   | #100000,ACSRADR | :LOOK FOR DBE STATUS BIT      |
| 2352 | 006012 | 001011 |        |        | BNE   | 50\$            | :IT SHOULD BE SET             |
| 2353 | 006014 | 010146 |        |        | MOV   | R1, -(SP)       | :SAVE R1                      |
| 2354 | 006016 | 016701 | 174706 |        | MOV   | CSRADR, R1      | :GET CSR ADDRESS              |
| 2355 | 006022 | 012700 | 100000 |        | MOV   | #100000, R0     | :SHOW BIT 15 SET              |
| 2356 | 006026 | 004767 | 005506 |        | JSR   | PC, ERROR       | :*ERROR* REPORT ERROR MESSAGE |
| (1)  | 006032 | 000047 |        |        | 47    |                 | :*****ERROR NUMBER 47*****    |
| (1)  |        |        |        |        |       |                 |                               |
| 2357 | 006034 | 012601 |        |        | MOV   | (SP)+, R1       | :RESTORE R1                   |
| 2358 | 006036 | 012702 | 006100 | 50\$:  | MOV   | #60\$, R2       | :SET UP FOR BUS PBL           |
| 2359 | 006042 | 066702 | 174652 |        | ADD   | BASE, R2        |                               |
| 2360 | 006046 | 010237 | 000114 |        | MOV   | R2, #114        |                               |
| 2361 | 006052 | 052777 | 000001 | 174650 | BIS   | #1, ACSRADR     | :SET DOUBLE ERROR INDICATE    |
| 2362 |        |        |        |        |       |                 |                               |
| 2363 | 006060 | 005711 |        |        | TST   | (R1)            | :READ THE TEST LOCATION AGAIN |
| 2364 | 006062 | 005077 | 174642 |        | CLR   | ACSRADR         | :CLEAR STATUS REG             |
| 2365 | 006066 | 005000 |        |        | CLR   | R0              | :CLEAR R0                     |
| 2366 | 006070 | 004767 | 005444 |        | JSR   | PC, ERROR       | :*ERROR* REPORT ERROR MESSAGE |
| (1)  | 006074 | 000050 |        |        | 50    |                 | :*****ERROR NUMBER 50*****    |
| (1)  |        |        |        |        |       |                 |                               |
| 2367 | 006076 | 000402 |        |        | BR    | 61\$            | :IF ERROR, DON'T ADJUST STACK |
| 2368 | 006100 | 062706 | 000004 | 60\$:  | ADD   | #4, SP          | :ADJUST STACK FROM TRAP       |
| 2369 | 006104 | 005737 | 000406 | 61\$:  | TST   | A#PASS          | :FIRST PASS ?                 |
| 2370 | 006110 | 001424 |        |        | BEQ   | 80\$            | :BR IF YES                    |
| 2371 | 006112 | 005767 | 174564 | 65\$:  | TST   | DBEMSK+2        | :CHECK MASK FOR LAST BIT      |
| 2372 | 006116 | 100404 |        |        | BMI   | 70\$            | :MINUS = BIT31                |
| 2373 | 006120 | 004567 | 005230 |        | JSR   | R5, DASHL       | :DOUBLE SHIFT                 |
| 2374 | 006124 | 002700 |        |        | .WORD | DBEMSK          |                               |
| 2375 | 006126 | 000640 |        |        | BR    | 35\$            |                               |
| 2376 | 006130 | 005767 | 174542 | 70\$:  | TST   | SBEMSK+2        | :CHECK SINGLE ERROR MASK TOO  |
| 2377 | 006134 | 100404 |        |        | BMI   | 75\$            | :BR IF DONE                   |
| 2378 | 006136 | 004567 | 005212 |        | JSR   | R5, DASHL       | :SHIFT                        |
| 2379 | 006142 | 002674 |        |        | .WORD | SBEMSK          |                               |
| 2380 | 006144 | 000624 |        |        | BR    | 30\$            |                               |
| 2381 | 006146 | 105737 | 000302 | 75\$:  | TSTB  | A#PASFLG        | :FIRST PASS                   |
| 2382 | 006152 | 001003 |        |        | BNE   | 80\$            | :NON ZERO MEANS WE'RE DONE    |
| 2383 | 006154 | 105237 | 000302 |        | INCB  | A#PASFLG        | :FIRST PASS, NOT DONE         |
| 2384 | 006160 | 000605 |        |        | BR    | 10\$            | :KEEP GOING                   |
| 2385 | 006162 | 005737 | 000406 | 80\$:  | TST   | A#PASS          | :CHECKING PASS AGAIN          |
| 2386 | 006166 | 001406 |        |        | BEQ   | 82\$            | :CHECK 1K ADDR FUNCTION ON    |
| 2387 |        |        |        |        |       |                 | :FIRST PASS ONLY              |
| 2388 | 006170 | 016777 | 174514 | 174532 | MOV   | ECCDIS, ACSRADR | :OTHERWISE DISABLE ECC        |
| 2389 | 006176 | 005021 |        |        | CLR   | (R1)+           | :CLEAR THE DBE'S              |
| 2390 | 006200 | 005011 |        |        | CLR   | (R1)            |                               |
| 2391 | 006202 | 000545 |        |        | BR    | 90\$            | :AND EXIT                     |
| 2392 | 006204 | 005077 | 174520 | 82\$:  | CLR   | ACSRADR         | :CLEAR CSR                    |
| 2393 | 006210 | 005002 |        |        | CLR   | R2              | :OUR COUNTER                  |
| 2394 | 006212 | 005711 |        |        | TST   | (R1)            | :READ THE LOCATION            |
| 2395 | 006214 | 017703 | 174510 |        | MOV   | ACSRADR, R3     | :MOV CHECKBITS INTO R3        |
| 2396 | 006220 | 005077 | 174504 |        | CLR   | ACSRADR         | :CLEAR CSR AGAIN              |
| 2397 | 006224 | 042703 | 170037 |        | BIC   | #170037, R3     | :ONLY WANT CHECKBITS          |
| 2398 | 006230 | 006303 |        |        | ASL   | R3              | :POSITION THEM                |
| 2399 | 006232 | 006303 |        |        | ASL   | R3              | :INTO LOW BYTE                |
| 2400 | 006234 | 006303 |        |        | ASL   | R3              |                               |

```

2401 006236 000303          SWAB      R3          ;HERE
2402 006240 004767 007142    JSR      PC,GET1K    ;GET THE 1K BANK
2403 006244 020300          CMP      R3,R0      ;BETTER BE THE SAME
2404 006246 001413          BEQ      85$        ;BR IF OK
2405 006250 010146          MOV      R1,-(SP)    ;SAVE R1 FOR NOW
2406 006252 016777 174432 174450  MOV      ECCDIS,ACSRADR ;DISABLE ECC
2407 006260 010311          MOV      R3,(R1)    ;PUT BAD DATA TO BE PRINTED INTO R1
2408 006262 005077 174442    CLR      ACSRADR    ;RESTORE CSR
2409 006266 004767 005246    JSR      PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
(1) 006272 000051          51          ;*****ERROR NUMBER 51*****
(1)
2410 006274 012601          MOV      (SP)+,R1    ;RESTORE R1
2411 006276 016777 174406 174424 85$:  MOV      ECCDIS,ACSRADR ;DISABLE ERROR CORRECTING
2412 006304 005021          CLR      (R1)+      ;TO CLEAR ANY DOUBLE ERRORS
2413 006306 005011          CLR      (R1)
2414 006310 005077 174414    CLR      ACSRADR    ;RESTORE CSR
2415 006314 042701 003777    BIC      #3777,R1    ;STRIP AWAY TO LAST 1K
2416 006320 062701 004000    ADD      #4000,R1    ;MOVE UP TO NEXT 1K
2417 006324 020105          CMP      R1,R5      ;OVER TOP YET ?
2418 006326 103073          BHIS     90$        ;BR IF YES
2419 006330 004767 007142    JSR      PC,TSTADD   ;CHECK FOR ERROR FREE LOC
2420 006334 005703          TST      R3          ;ZERO MEANS ERROR FREE
2421 006336 001456          BEQ      86$        ;BR IF OK
2422 006340 005202          INC      R2          ;COUNT ONE WORD CHECKED
2423 006342 020227 001000    CMP      R2,#1000   ;TESTED WHOLE 1K SLICE YET ?
2424 006346 103447          BLO      87$        ;BR IF NOT
2425 006350 004767 006056    JSR      PC,TPCRLF   ;PRINT ROUTINE
2426 006354 051524 033524 047055  .ASCIZ  /TST7-NO ERROR FREE LOC IN 1K SLICE AT /
      006362 020117 051105 047522
      006370 020122 051106 042505
      006376 046040 041517 044440
      006404 020116 045461 051440
      006412 044514 042503 040440
      006420 020124      000
2427
2428 006424 042701 003776          .EVEN
2429 006430 012703 006534          BIC      #3776,R1
2430 006434 066703 174260          MOV      #SAV7+2,R3 ;WANT STARTING ADDRESS OF 1K
2431 006440 010146          ADD      BASE,R3    ;SCRATCH LOC FOR ADDRESS
2432 006442 004767 006454          MOV      R1,-(SP)   ;FOR OFFSET
2433 006446 105277 171636          JSR      PC,PUTADR  ;SAVE R1
2434 006452 005741          INCB     ATYPCNT    ;PUT ADDRESS IN SAV7 & SAV7+2
2435 006454 004767 006122          TST      -(R1)      ;PRINT ONE WORD
2436 006460 005002          JSR      PC,TYPOCT  ;ADJUST FOR PRINTOUT
2437 006462 012601          CLR      R2          ;PRINT IT HERE
2438 006464 000704          MOV      (SP)+,R1   ;RESTORE R2 FOR NEXT SLICE
2439 006466 062701 000004          BR       85$        ;RESTORE R1
2440 006472 000716          BR       85$        ;GO LOOK FOR GOOD LOC
2441 006474 016721 174170          ADD      #4,R1      ;NEXT WORD
2442 006500 010477 174224          BR       86$        ;GO CHECK IT
2443 006504 016711 174162          MOV      TSTDAT,(R1)+ ;WRITE FIRST WORD
2444 006510 162701 000002          MOV      R4,ACSRADR ;CSR IMAGE WITH BAD CHECKBITS
2445 006514 000633          MOV      TSTDAT+2,(R1) ;WRITE 2ND WORD + CHECKBITS
2446 006516 005077 174206          SUB      #2,R1      ;ADJUST R1
2447 006522 004767 004612          BR       82$        ;CONTINUE TESTING
2448 006526 000167 174054          CLR      ACSRADR   ;CLEAR CSR TO NORMAL
      000167 174054          JSR      PC,RSTOT4  ;RESTORE THE REGS 0 TO 4
      000167 174054          JMP      TSTSCP
    
```



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

```

2449 006532 000000 000000 SAV7: .WORD 0,0
2450
2451 ; THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
2452 ; BIT ERRORS DURING A DATIP OPERATION BY USE
2453 ; OF AN 'INC' INSTRUCTION.
2454
2455 006536 122737 000010 000404 TST10: CMPB #10, @#STESTN
2456 006544 001403 BEQ 1$
2457 006546 004767 005442 JSR PC, SEGERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 006552 000052 52 ; *****ERROR NUMBER 52*****
(1)
2458 006554 004467 004546 1$: JSR R4, SAVOT4 ; SAVE REGS R0 TO R4
2459 006560 005067 174100 10$: CLR DATBUF ; INITIAL DATA
2460 006564 005067 174076 CLR DATBUF+2 ; 2 WORDS WORTH
2461 006570 012767 000001 174076 20$: MOV #1, SBEMSK ; INITIAL ERROR MASK
2462 006576 005067 174074 CLR SBEMSK+2 ;
2463 006602 012767 000001 174070 30$: MOV #1, DBEMSK ; DOUBLE ERROR MASK
2464 006610 005067 174066 CLR DBEMSK+2 ; 2 WORDS
2465 006614 016767 174044 174046 35$: MOV DATBUF, TSTDAT ; PRESERVE ORIG DATA
2466 006622 016767 174040 174042 MOV DATBUF+2, TSTDAT+2 ;
2467 006630 105737 000302 TSTB @#PASFLG ; SECOND PASS YET ?
2468 006634 001404 BEQ 40$ ; BR IF NO
2469 006636 005167 174026 COM TSTDAT ; COMPL DATA ON SECOND PASS
2470 006642 005167 174024 COM TSTDAT+2 ;
2471 006646 005077 174056 40$: CLR @CSRADR ;
2472 006652 026767 174022 174014 CMP DBEMSK, SBEMSK ; CHECK FOR SAME MASKS
2473 006660 001004 BNE 45$ ; BR IF OK
2474 006662 026767 174014 174006 CMP DBEMSK+2, SBEMSK+2 ;
2475 006670 001476 BEQ 70$ ; BR IF THEY'RE EQUAL
2476 006672 012767 002670 004112 45$: MOV @TSTDAT, SOURCE ; SET UP ADDRESS FOR CHKGEN
2477 006700 004767 004160 JSR PC, CHKGEN ; GENERATE CHECK BITS
2478 006704 004567 004474 JSR R5, BITCOM ; COMPL ROUTINE
2479 006710 002674 .WORD SBEMSK ; MASK
2480 006712 002670 .WORD TSTDAT ; DATA
2481 006714 004567 004464 JSR R5, BITCOM ; TO MAKE DOUBLE ERROR
2482 006720 002700 .WORD DBEMSK
2483 006722 002670 .WORD TSTDAT
2484 006724 016702 004064 MOV CHECK, R2 ; GET CHECKBITS
2485 006730 056702 173756 BIS DIAGA, R2 ; SET DIAGNOSTIC A BIT
2486 006734 013701 000320 50$: MOV @#MINMEM, R1 ; TEST ADDRESS
2487 006740 016777 173744 173762 MOV ECCDIS, @CSRADR ; INHIBIT ECC
2488 006746 016721 173716 MOV TSTDAT, (R1)+ ; WRITE FIRST 16 BITS
2489 006752 010277 173752 MOV R2, @CSRADR ; LOAD CSR WITH IMAGE FROM R2
2490 006756 016711 173710 MOV TSTDAT+2, (R1) ; SECOND 16 BITS+CHECKBITS
2491 006762 105067 004064 CLRB UPPFLG ; INDICATE LOWER WORD
2492 006766 013703 000320 MOV @#MINMEM, R3 ; TEST ADDRESS
2493 006772 005077 173732 55$: CLR @CSRADR ; CLEAR IT
2494 006776 005223 INC (R3)+ ; INC INSTRUCTION
2495 007000 026741 173664 CMP TSTDAT, -(R1) ; CHECK FOR UNCHANGED DATA
2496 007004 001405 BEQ 60$ ; SHOULD BE UNCHANGED
2497 007006 016700 173656 MOV TSTDAT, R0 ; FOR PRINTOUT
2498 007012 004767 004522 JSR PC, ERROR ; *ERROR* REPORT ERROR MESSAGE
(1) 007016 000053 53 ; *****ERROR NUMBER 53*****
(1)
2499 007020 062701 000002 60$: ADD #2, R1 ; POINT TO UPPER WORD
2500 007024 026711 173642 CMP TSTDAT+2, (R1) ; READ IT
    
```

# K04

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-48  
 DZMMLB.P11 05-AUG-77 12:57

SEQ 0050

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

|      |        |        |        |        |        |  |       |                   |  |
|------|--------|--------|--------|--------|--------|--|-------|-------------------|--|
| 2501 | 007030 | 001434 |        |        |        |  | BEQ   | 80\$              | ;BR IF UNCHANGED                                 |
| 2502 | 007032 | 016700 | 173634 |        |        |  | MOV   | TSTDAT+2,R0       |  |
| 2503 | 007036 | 004767 | 004476 |        |        |  | JSR   | PC,ERROR          | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 007042 | 000054 |        |        |        |  | 54    |                   | ;*****ERROR NUMBER 54*****                       |
| (1)  |        |        |        |        |        |  |       |                   |  |
| 2504 | 007044 | 105767 | 004002 | 65\$:  |        |  | TSTB  | UPPFLG            | ;LOWER WORD                                      |
| 2505 | 007050 | 001003 |        |        |        |  | BNE   | 66\$              | ;BR IF NO  |
| 2506 | 007052 | 105267 | 003774 |        |        |  | INCB  | UPPFLG            |  |
| 2507 | 007056 | 000745 |        |        |        |  | BR    | 55\$              |  |
| 2508 | 007060 | 005737 | 000406 | 66\$:  |        |  | TST   | 2#SPASS           | ;CHECK PASS #                                    |
| 2509 | 007064 | 001424 |        |        |        |  | BEQ   | 90\$              | ;BR IF FIRST                                     |
| 2510 | 007066 | 005767 | 173610 | 70\$:  |        |  | TST   | DBEMSK+2          | ;LAST BIT IN MASK ?                              |
| 2511 | 007072 | 100404 |        |        |        |  | BMI   | 75\$              | ;BR IF BIT 31                                    |
| 2512 | 007074 | 004567 | 004254 |        |        |  | JSR   | R5,DASHL          | ;SHIFT ROUTINE                                   |
| 2513 | 007100 | 002700 |        |        |        |  | .WORD | DBEMSK            |  |
| 2514 | 007102 | 000644 |        |        |        |  | BR    | 35\$              |  |
| 2515 | 007104 | 005767 | 173566 | 75\$:  |        |  | TST   | SBEMSK+2          | ;LAST BIT IN SINGLE ERROR MASK ?                 |
| 2516 | 007110 | 100404 |        |        |        |  | BMI   | 80\$              | ;BR IF YES                                       |
| 2517 | 007112 | 004567 | 004236 |        |        |  | JSR   | R5,DASHL          | ;SHIFT   |
| 2518 | 007116 | 002674 |        |        |        |  | .WORD | SBEMSK            |  |
| 2519 |        |        |        |        |        |  |       |                   |  |
| 2520 | 007120 | 000630 |        |        |        |  | BR    | 30\$              |  |
| 2521 | 007122 | 105737 | 000302 | 80\$:  |        |  | TSTB  | 2#PASFLG          | ;WHICH PASS                                      |
| 2522 | 007126 | 001003 |        |        |        |  | BNE   | 90\$              | ;BR IF WE'RE DONE                                |
| 2523 | 007130 | 105237 | 000302 |        |        |  | INCB  | 2#PASFLG          | ;INDICATE SECOND PASS COMING                     |
| 2524 | 007134 | 000611 |        |        |        |  | BR    | 10\$              | ;GO DO IT!                                       |
| 2525 | 007136 | 016777 | 173546 | 173564 | 90\$:  |  | MOV   | ECCDIS,2CSRADR    | ;INHIBIT ECC TO CLEAR DBE'S                      |
| 2526 | 007144 | 005011 |        |        |        |  | CLR   | (R1)              |  |
| 2527 | 007146 | 005041 |        |        |        |  | CLR   | -(R1)             |  |
| 2528 | 007150 | 005077 | 173554 |        |        |  | CLR   | 2CSRADR           | ;RESTORE CSR TO NORMAL                           |
| 2529 | 007154 | 004767 | 004160 |        |        |  | JSR   | PC,RSTOT4         | ;RESTORE THE REGS                                |
| 2530 | 007160 | 000167 | 173422 | END10: |        |  | JMP   | TSTSCP            |  |
| 2531 |        |        |        |        |        |  |       |                   |  |
| 2532 |        |        |        |        |        |  |       |                   | ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.    |
| 2533 |        |        |        |        |        |  |       |                   | ;CHECKS FOR UNCORRECTED DATA.                    |
| 2534 |        |        |        |        |        |  |       |                   |  |
| 2535 | 007164 | 122737 | 000011 | 000404 | TST11: |  | CMPB  | 11,2#STESTN       |  |
| 2536 | 007172 | 001403 |        |        |        |  | BEQ   | 1\$               |  |
| 2537 | 007174 | 004767 | 005014 |        |        |  | JSR   | PC,SEQERR         | ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 007200 | 000055 |        |        |        |  | 55    |                   | ;*****ERROR NUMBER 55*****                       |
| (1)  |        |        |        |        |        |  |       |                   |  |
| 2538 | 007202 | 004467 | 004120 | 1\$:   |        |  | JSR   | R4,SAVOT4         | ;SAVE REGS R0 TO R4                              |
| 2539 | 007206 | 005067 | 173452 | 10\$:  |        |  | CLR   | DATBUF            | ;INITIAL DATA                                    |
| 2540 | 007212 | 005067 | 173450 |        |        |  | CLR   | DATBUF+2          | ;32 BITS WORTH                                   |
| 2541 | 007216 | 012767 | 000001 | 173450 | 20\$:  |  | MOV   | 1,SBEMSK          | ;SINGLE ERROR MASK                               |
| 2542 | 007224 | 005067 | 173446 |        |        |  | CLR   | SBEMSK+2          |  |
| 2543 | 007230 | 012767 | 000001 | 173442 | 30\$:  |  | MOV   | 1,DBEMSK          | ;DOUBLE ERROR MASK                               |
| 2544 | 007236 | 005067 | 173440 |        |        |  | CLR   | DBEMSK+2          |  |
| 2545 | 007242 | 016767 | 173416 | 173420 | 35\$:  |  | MOV   | DATBUF,TSTDAT     | ;PRESERVE ORIG DATA                              |
| 2546 | 007250 | 016767 | 173412 | 173414 |        |  | MOV   | DATBUF+2,TSTDAT+2 |  |
| 2547 | 007256 | 105737 | 000302 |        |        |  | TSTB  | 2#PASFLG          | ;WHICH PASS ?                                    |
| 2548 | 007262 | 001404 |        |        |        |  | BEQ   | 40\$              | ;FIRST PASS, NO COMPLEMENTING                    |
| 2549 | 007264 | 005167 | 173400 |        |        |  | COM   | TSTDAT            |  |
| 2550 | 007270 | 005167 | 173376 |        |        |  | COM   | TSTDAT+2          | ;SECOND PASS, COMPLEMENT TSTDAT                  |
| 2551 | 007274 | 005077 | 173430 | 40\$:  |        |  | CLR   | 2CSRADR           |  |
| 2552 | 007300 | 026767 | 173370 | 173372 |        |  | CMP   | SBEMSK,DBEMSK     | ;CHECK FOR SAME MASKS                            |

|      |        |        |        |        |        |       |                    |  |                                   |
|------|--------|--------|--------|--------|--------|-------|--------------------|--|-----------------------------------|
| 2553 | 007306 | 001004 |        |        |        | BNE   | 45\$               |  | ;BR IF NOT EQUAL                  |
| 2554 | 007310 | 026767 | 173362 | 173364 |        | CMP   | SBEMSK+2, DBEMSK+2 |  | ;SECOND WORD ALSO                 |
| 2555 | 007316 | 001473 |        |        |        | BEQ   | 70\$               |  | ;BR TO MAKE THEM NOT EQUAL        |
| 2556 | 007320 | 012767 | 002670 | 003464 | 45\$:  | MOV   | #TSTDAT, SOURCE    |  | ;ADDRESS FOR CHKGEN               |
| 2557 | 007326 | 004767 | 003532 |        |        | JSR   | PC, CHKGEN         |  | ;GO GENERATE CHECK BITS           |
| 2558 | 007332 | 004567 | 004046 |        |        | JSR   | R5, BITCOM         |  | ;BIT COMP ROUTINE                 |
| 2559 | 007336 | 002674 |        |        |        | .WORD | SBEMSK             |  | ;MASK                             |
| 2560 | 007340 | 002670 |        |        |        | .WORD | TSTDAT             |  | ;DATA WORD                        |
| 2561 | 007342 | 004567 | 004036 |        |        | JSR   | R5, BITCOM         |  | ;MUST FORCE SECOND ERROR          |
| 2562 | 007346 | 002700 |        |        |        | .WORD | DBEMSK             |  | ;MASK                             |
| 2563 | 007350 | 002670 |        |        |        | .WORD | TSTDAT             |  |                                   |
| 2564 | 007352 | 016702 | 003436 |        |        | MOV   | CHECK, R2          |  | ;GET CHECKBITS                    |
| 2565 | 007356 | 056702 | 173330 |        |        | BIS   | DIAGA, R2          |  | ;SET DIAGNOSTIC A BIT             |
| 2566 | 007362 | 013701 | 000320 |        | 50\$:  | MOV   | #MINMEM, R1        |  | ;TEST LOCATION                    |
| 2567 | 007366 | 016777 | 173316 | 173334 |        | MOV   | ECCDIS, #CSRADR    |  | ;DISABLE ECC                      |
| 2568 | 007374 | 016721 | 173270 |        |        | MOV   | TSTDAT, (R1)+      |  | ;WRITE FIRST 16 BITS              |
| 2569 | 007400 | 010277 | 173324 |        |        | MOV   | R2, #CSRADR        |  | ;LOAD CSR WITH IMAGE FROM R2      |
| 2570 | 007404 | 016711 | 173262 |        |        | MOV   | TSTDAT+2, (R1)     |  | ;WRITE SECOND 16 BITS + CHECKBITS |
| 2571 | 007410 | 005077 | 173314 |        |        | CLR   | #CSRADR            |  | ;CLEAR CSR                        |
| 2572 | 007414 | 013702 | 000320 |        |        | MOV   | #MINMEM, R2        |  | ;GET ADDRESS OF TEST LOC          |
| 2573 | 007420 | 010203 |        |        |        | MOV   | R2, R3             |  | ;R2 DESIGNATES FIRST BYTE         |
| 2574 | 007422 | 062703 | 000003 |        |        | ADD   | #3, R3             |  | ;R3 DESIGNATES LAST BYTE          |
| 2575 | 007426 | 112722 | 000360 |        | 55\$:  | MOVB  | #360, (R2)+        |  | ;TRY WRITING A BYTE               |
| 2576 | 007432 | 013701 | 000320 |        |        | MOV   | #MINMEM, R1        |  |                                   |
| 2577 | 007436 | 026711 | 173226 |        |        | CMP   | TSTDAT, (R1)       |  | ;CHECK FOR UNCHANGED DATA         |
| 2578 | 007442 | 001405 |        |        |        | BEQ   | 60\$               |  | ;BR IF OK                         |
| 2579 | 007444 | 016700 | 173220 |        |        | MOV   | TSTDAT, R0         |  | ;FOR ERROR MSG                    |
| 2580 | 007450 | 004767 | 004064 |        |        | JSR   | PC, ERROR          |  | ;*ERROR* REPORT ERROR MESSAGE     |
| (1)  | 007454 | 000056 |        |        |        | 56    |                    |  | ;*****ERROR NUMBER 56*****        |
| (1)  |        |        |        |        |        |       |                    |  |                                   |
| 2581 | 007456 | 062701 | 000002 |        | 60\$:  | ADD   | #2, R1             |  | ;UPPER WORD                       |
| 2582 | 007462 | 026711 | 173204 |        |        | CMP   | TSTDAT+2, (R1)     |  | ;READ SECOND WORD                 |
| 2583 | 007466 | 001405 |        |        |        | BEQ   | 65\$               |  | ;BR IF UNCHANGED                  |
| 2584 | 007470 | 016700 | 173176 |        |        | MOV   | TSTDAT+2, R0       |  |                                   |
| 2585 | 007474 | 004767 | 004040 |        |        | JSR   | PC, ERROR          |  | ;*ERROR* REPORT ERROR MESSAGE     |
| (1)  | 007500 | 000057 |        |        |        | 57    |                    |  | ;*****ERROR NUMBER 57*****        |
| (1)  |        |        |        |        |        |       |                    |  |                                   |
| 2586 | 007502 | 020203 |        |        | 65\$:  | CMP   | R2, R3             |  | ;TESTED LAST BYTE ?               |
| 2587 | 007504 | 001350 |        |        |        | BNE   | 55\$               |  | ;BR IF NO                         |
| 2588 | 007506 | 005737 | 000406 |        | 70\$:  | TST   | #SPASS             |  | ;FIRST PASS ?                     |
| 2589 | 007512 | 001424 |        |        |        | BEQ   | 100\$              |  | ;BR IF YES                        |
| 2590 | 007514 | 005767 | 173162 |        |        | TST   | DBEMSK+2           |  | ;CHECKING FOR LAST ERROR BIT      |
| 2591 | 007520 | 100404 |        |        |        | BMI   | 80\$               |  | ;BR IF DONE HERE                  |
| 2592 | 007522 | 004567 | 003626 |        |        | JSR   | R5, DASHL          |  | ;NOT DONE, SHIFT LEFT             |
| 2593 | 007526 | 002700 |        |        |        | .WORD | DBEMSK             |  | ;DOUBLE ERROR MASK                |
| 2594 | 007530 | 000644 |        |        |        | BR    | 35\$               |  |                                   |
| 2595 | 007532 | 005767 | 173140 |        | 80\$:  | TST   | SBEMSK+2           |  | ;LAST SBE MASK                    |
| 2596 | 007536 | 100404 |        |        |        | BMI   | 90\$               |  | ;BR IF DONE WITH THIS PASS        |
| 2597 | 007540 | 004567 | 003610 |        |        | JSR   | R5, DASHL          |  | ;DOUBLE WORD SHIFT ROUTINE        |
| 2598 | 007544 | 002674 |        |        |        | .WORD | SBEMSK             |  |                                   |
| 2599 | 007546 | 000630 |        |        |        | BR    | 30\$               |  |                                   |
| 2600 | 007550 | 105737 | 000302 |        | 90\$:  | TSTB  | #PASFLG            |  | ;TEST PASS FLAG                   |
| 2601 | 007554 | 001003 |        |        |        | BNE   | 100\$              |  | ;NON ZERO MEANS WE'RE DONE        |
| 2602 | 007556 | 105237 | 000302 |        |        | INCB  | #PASFLG            |  | ;NOT DONR                         |
| 2603 | 007562 | 000611 |        |        |        | BR    | 10\$               |  |                                   |
| 2604 | 007564 | 016777 | 173120 | 173136 | 100\$: | MOV   | ECCDIS, #CSRADR    |  | ;DISABLE ECC                      |

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

```

2605 007572 013701 000320      MOV      @#MINMEM,R1      ;TEST LOCATION
2606 007576 005021      CLR      (R1)+
2607 007600 005011      CLR      (R1)           ;TO ERASE ANY DBE'S FROM TESTING
2608 007602 005077 173122      CLR      @CSRADR        ;RESTORE CSR
2609 007606 004767 003526      JSR      PC,RSTOT4      ;RESTORE REGS R0 TO R4
2610 007612 000167 172770      END11:  JMP      TSTSCP
2611
2612      ;DOUBLE BIT ERROR WRITE CANCEL WITH
2613      ;WORD WRITE.
2614      ;CHECKS WRITE INHIBIT WITH WORD WRITES TO
2615      ;WORD WITH DOUBLE ERROR.
2616
2617 007616 122737 000012 000404  TST12:  CMPB     #12,@#STESTN    ;CHECK FOR PROPER TEST SEQUENCE
2618 007624 001403      BEQ      1$            ;BR IF OK
2619 007626 004767 004362      JSR      PC,SEQERR     ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 007632 000060      B0           ;*****ERROR NUMBER 60*****
2620 007634 004467 003466      1$:      JSR      R4,SAVOT4    ;SAVE REGS R0 TO R4
2621 007640 005067 173020      T12A:    CLR      DATBUF        ;BACKGROUND FOR DOUBLE ERRORS
2622 007644 005067 173016      CLR      DATBUF+2      ;2 WORDS WORTH
2623 007650 012767 000001 173016      MOV      #1,SBEMSK     ;SINGLE ERROR MASK
2624 007656 005067 173014      CLR      SBEMSK+2
2625 007662 012767 000001 173010  T12B:    MOV      #1,DBEMSK     ;DOUBLE ERROR MASK
2626 007670 005067 173006      CLR      DBEMSK+2
2627 007674 016767 172764 172766  35$:    MOV      DATBUF,TSTDAT ;DATA FOR TEST
2628 007702 016767 172760 172762      MOV      DATBUF+2,TSTDAT+2 ;BOTH WORDS
2629 007710 105737 000302      TSTB    @#PASFLG      ;COMP DATA ON SECOND PASS ONLY
2630 007714 001404      BEQ      40$          ;BR IF FIRST PASS
2631 007716 005167 172746      COM      TSTDAT        ;COMP FIRST WORD
2632 007722 005167 172744      COM      TSTDAT+2      ;NOW SECOND WORD
2633 007726 026767 172742 172744  40$:    CMP      SBEMSK,DBEMSK ;CHECK FOR IDENTICAL MASKS
2634 007734 001004      BNE      45$          ;BR IF DIFFERENT
2635 007736 026767 172734 172736      CMP      SBEMSK+2,DBEMSK+2 ;UPPER WORD TOO
2636 007744 001501      BEQ      70$          ;BR TO MAKE THEM NOT EQUAL
2637 007746 012767 002670 003036  45$:    MOV      @TSTDAT,SOURCE ;NEED ADDR OF DATA FOR CHKGEN
2638 007754 004767 003104      JSR      PC,CHKGEN     ;GO GENERATE CHECK BITS
2639 007760 004567 003420      JSR      R5,BITCOM     ;BIT COMP ROUTINE TO FORCE ERRORS
2640 007764 002674      .WORD   SBEMSK        ;FIRST ERROR
2641 007766 002670      .WORD   TSTDAT        ;DATA
2642 007770 004567 003410      JSR      R5,BITCOM     ;CALL IT AGAIN
2643 007774 002700      .WORD   DBEMSK        ;FOR SECOND ERROR
2644 007776 002670      .WORD   TSTDAT
2645 010000 016700 003010      MOV      CHECK,R0      ;GET CHECKBITS
2646 010004 056700 172702      BIS      DIAG,R0       ;SET DIAGNOSTIC BIT
2647 010010 013701 000320      50$:    MOV      @#MINMEM,R1   ;FIRST TEST ADDRESS
2648 010014 016777 172670 172706      MOV      ECCDIS,@CSRADR ;INHIBIT ECC
2649 010022 016721 172642      MOV      TSTDAT,(R1)+  ;WRITE FIRST 16 BITS
2650 010026 010077 172676      MOV      R0,@CSRADR   ;LOAD CSR FROM R0
2651 010032 016711 172634      MOV      TSTDAT+2,(R1) ;WRITE SECOND 16 BITS + CHECKBITS
2652 010036 105067 003010      CLRB    UPPFLG        ;SET FOR 2 LOOPS
2653 010042 162701 000002      SUB     #2,R1          ;POINT TO LOW WORD
2654 010046 005077 172656      55$:    CLR      @CSRADR      ;CLEAR CSR
2655 010052 012721 177400      MOV     #177400,(R1)+ ;TRY WRITING LOCATION
2656 010056 013701 000320      MOV     @#MINMEM,R1
2657 010062 026711 172602      CMP     TSTDAT,(R1)   ;CHECK FOR ORIGINAL DATA
2658 010066 001405      BEQ     60$          ;SHOULD BE UNCHANGED

```

DZMMLB.P11 05-AUG-77 12:57

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

```

2659 010070 016700 172574      MOV    TSTDAT,RO      ;FOR ERROR MSG
2660 010074 004767 003440      JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 010100 000061                61      ;*****ERROR NUMBER 61*****
(1)
2661 010102 062701 000002      60$:   ADD    #2,R1      ;UPPER WORD
2662 010106 026711 172560      CMP    TSTDAT+2,(R1) ;THIS SHOULD BE UNCHANGED ALSO
2663 010112 001405                BEQ    65$
2664 010114 016700 172552      MOV    TSTDAT+2,RO   ;
2665 010120 004767 003414      JSR    PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 010124 000062                62      ;*****ERROR NUMBER 62*****
(1)
2666 010126 105767 002720      65$:   TSTB   UPPFLG     ;WHICH LOOP ?
2667 010132 001003                BNE   66$           ;SECOND, BR OUT
2668 010134 105267 002712      INCB  UPPFLG     ;FIRST, KEEP GOING
2669 010140 000742                BR    55$
2670 010142 005737 000406      66$:   TST   @#SPASS   ;FIRST PASS ?
2671 010146 001426                BEQ   85$           ;BR IF YES
2672 010150 005767 172526      70$:   TST   DBEMSK+2 ;LAST BIT ?
2673 010154 100404                BMI  75$           ;MINUS = BIT 31
2674 010156 004567 003172      JSR   R5,DASHL    ;SHIFT ROUTINE
2675 010162 002700                .WORD DBEMSK      ;THIS 32 BIT WORD GETS SHIFTED
2676 010164 000643                BR    35$
2677 010166 005767 172504      75$:   TST   SBEMSK+2 ;LAST BIT IN THIS MASK ?
2678 010172 100405                BMI  80$           ;BR IF LAST BIT
2679 010174 004567 003154      JSR   R5,DASHL
2680 010200 002674                .WORD SBEMSK
2681 010202 000167 177454      JMP   T12B
2682 010206 105737 000302      80$:   TSTB   @#PASFLG  ;FIRST PASS ?
2683 010212 001004                BNE  85$           ;BR IF SECOND
2684 010214 105237 000302      INCB  @#PASFLG    ;INDICATE SECOND PASS COMING
2685 010220 000167 177414      JMP   T12A
2686 010224 016777 172460 172476 85$:   MOV   ECCDIS,@CSRADR ;
2687 010232 013701 000320      MOV   @#MINMEM,R1 ;RESTORE TEST ADDRESS
2688 010236 005021                CLR  (R1)+        ;CLEAR ANY DBE'S FROM TEST
2689 010240 005011                CLR  (R1)
2690 010242 005077 172462      CLR  @CSRADR     ;CLEAR CSR
2691 010246 004767 003066      JSR   PC,RSTOT4  ;RESTORE REGS R0 TO R4
2692 010252 000167 172330      END12: JMP  TSTSCP
2693                ;TEST DUAL ADDRESS TEST
2694                ;
2695                ;CHECKS FOR DUAL ADDRESSING BY WRITING
2696                ;AND READING THE ADDRESS IN THE LOCATION
2697                ;AND WRITING AND READING ITS COMPLEMENT
2698 010256 122737 000013 000404 TST13: CMPB  @13,@#STESTN
2699 010264 001403                BEQ  63$
2700 010266 004767 003722      JSR   PC,SEGERR   ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 010272 000063                63      ;*****ERROR NUMBER 63*****
(1)
2701
2702 010274 005003                65$:   CLR   R3          ;R3 INDICATES FIRST PASS OR
2703 010276 010100                1$:   MOV   R1,RO      ;COMPLEMENT PASS
2704 010300 005703                TST  R3           ;IF R3= ZERO, STORE ADDRESS
2705 010302 001401                BEQ  2$           ;IN THE LOCATION
2706 010304 005100                COM  RO          ;OTHERWISE STORE COMPLEMENT
2707 010306 010021                2$:   MOV   RO,(R1)+  ;OF ADDRESS
2708 010310 020105                CMP  R1,R5       ;UNTIL HIGHEST LOC IS REACHED

```

## B05

DZMPL MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-52  
 DZMPLB.P11 05-AUG-77 12:57

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

SEQ 0054

```

2709 010312 103771          BLO      1$
2710 010314 020041          3$:    CMP      RD,-(R1)      ;CHECK FOR CORRECT CONTENTS
2711 010316 001405          BEQ      4$              ;BRANCH IF OK
2712 010320 105237 000275    INCB     2$SADERR      ;PROBABLY AN ADDRESSING PROB.
2713 010324 004767 003210    JSR      PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 010330 000064          B4              ;*****ERROR NUMBER 64*****
(1)
2714
2715 010332 010100          4$:    MOV      R1,R0
2716 010334 162700 000002    SUB      #2,R0          ;CHECK THAT ADDRESS IS STORED AT
2717 010340 005703          TST      R3              ;LOCATION IF R3 IS 0
2718 010342 001401          BEQ      5$              ;OTHERWISE CHECK FOR
2719 010344 005100          COM      R0              ;ADDRESS COMPLEMENT.
2720 010346 020104          5$:    CMP      R1,R4
2721 010350 101361          BHI      3$
2722 010352 112737 000001 000302    MOVB     #1,2$PASFLG    ;SET PASFLG FOR ERROR REPORT
2723 010360 005103          COM      R3              ;COMPLEMENT R3
2724 010362 001345          BNE      1$              ;REPEAT TEST, COMPLEMENTING ADE
2725 010364 000167 172216    END13:  JMP      TSTSCP
2726
2727
2728          ;*(1)  THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
2729          ;*    OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
2730          ;*    OF BAKPAT AND READING IT
2731          ;*(2)  MEMORY IS WRITTEN USING A BYTE AT A TIME
2732          ;*(3)  STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
2733
2734 010370 122737 000014 000404    TST14:  CMPB     #14,2$TESTN    ;CHECK FOR PROPER TEST SEQUENCE
2735 010376 001403          BEQ      1$
2736 010400 004767 003610    JSR      PC,SEQERR      ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 010404 000065          B5              ;*****ERROR NUMBER 65*****
(1)
2737 010406 013700 000312          1$:    MOV      2$BAKPAT,R0    ;GET THE PATTERN INTO R0
2738 010412 110021          MOVB     R0,(R1)+        ;WRITE A BYTE
2739 010414 113721 000313    MOVB     2$BAKPAT+1,(R1)+ ;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
2740 010420 020105          CMP      R1,R5          ;COMPARE TEST LOC TO TOP + 2
2741 010422 103771          BLO      1$              ;BRANCH IF LOWER
2742
2743 010424 020041          2$:    CMP      RD,-(R1)      ;TEST THE MEMORY TO SEE IF IT CONTAINS
2744          ;THE WORD STORED IN BAKPAT
2745 010426 001416          BEQ      8$
2746 010430 062701 000002    ADD      #2,R1
2747 010434 123741 000313    CMPB     2$BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
2748 010440 001402          BEQ      4$
2749 010442 120041          CMPB     R0,-(R1)      ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
2750 010444 001002          BNE      6$
2751 010446 105237 000275          4$:    INCB     2$SADERR      ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
2752 010452 042701 000001          6$:    BIC      #1,R1          ;MAKE THE ADDRESS IN R1 EVEN
2753 010456 004767 003056    JSR      PC,ERROR      ;*ERROR* REPORT ERROR MESSAGE
(1) 010462 000066          B6              ;*****ERROR NUMBER 66*****
(1)
2754 010464 020104          8$:    CMP      R1,R4
2755 010466 101356          BHI      2$
2756 010470 000337 000312    SWAB     2$BAKPAT
2757 010474 001744          BEQ      1$
2758          ;KEEP ON TESTING THE MEMORY UNTIL
          ;R1 EQUALS THE LOWEST ADDRESS
          ;CHANGE THE DATA PATTERN
          ;IF THE DATA PATTERN DOES NOT HAVE LOW
          ;BYTE =0 THEN FALL THRU

```

```

2759 010476 000167 172104      END14: JMP    TSTSCP
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769 010502 122737 000015 000404 TST15: CMPB   #15,0#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2770 010510 001403          BEQ    1$
2771 010512 004767 003022      JSR    PC,ERROR ;#ERROR# REPORT ERROR MESSAGE
(1) 010516 000067          B7     ;*****ERROR NUMBER 67*****
(1)
2772 010520 010446          1$:    MOV    R4,-(SP) ;SAVE R4
2773 010522 016703 172164      MOV    DIAGA,R3 ;DIAGNOSTIC CHECK MODE
2774 010526 052703 004000      BIS    #4000,R3 ;CHECK BIT CT
2775 010532 013701 000320      MOV    #MINMEM,R1 ;FIRST TEST LOC
2776 010536 010146      MOV    R1,-(SP) ;SAVE IT FOR LATER
2777 010540 005004      CLR    R4
2778 010542 010377 172162      MOV    R3,0CSRADR ;SET UP CSR
2779 010546 010411          2$:    MOV    R4,(R1) ;WRITE CHECKBITS BY WRITING LOC
2780 010550 012761 100000 000002      MOV    #100000,2(R1) ;SET BIT 31 TO MATCH CHECKBITS
2781 010556 062701 000004      ADD    #4,R1 ;POINT TO NEXT WORD
2782 010562 020105      CMP    R1,R5 ;TOP + 2
2783 010564 103770          BLO   2$ ;BR IF NOT
2784 010566 162701 000704          3$:    SUB    #4,R1 ;ADJUST R1
2785 010572 005711          TST   (R1) ;READ
2786 010574 032777 003740 172126      BIT    #3740,0CSRADR ;SHOULD BE ALL ZEROS
2787 010602 001421          BEQ   4$ ;BR IF OK
2788 010604 012700 004000      MOV    #4000,R0 ;GOOD DATA
2789 010610 004767 005036      JSR    PC,BITCHK ;USE XOR TO COUNT BAD CHECKBITS
2790 010614 032777 004000 172110      BIT    #4000,0SWR ;ECC DIS ?
2791 010622 001004          BNE   33$ ;PRNIT SINGLE ERRORS
2792 010624 022767 000001 005102      CMP    #1,BITCNT ;MORE THAN 1 BAD BIT ?
2793 010632 001405          BEQ   4$ ;YES
2794 010634 012700 004000          33$:  MOV    #4000,R0 ;FOR ERROR MSG
2795 010640 004767 002674      JSR    PC,ERROR ;#ERROR# REPORT ERROR MESSAGE
(1) 010644 000070          70     ;*****ERROR NUMBER 70*****
(1)
2796 010646 052777 003740 172054 4$:    BIS    #3740,0CSRADR ;MAKE CHECKBITS ALL 1'S EXCEPT FOR CT
2797 010654 042777 004000 172046      BIC    #4000,0CSRADR ;MAKE CT = 0
2798 010662 010411          MOV    R4,(R1) ;WRITE THE CHECKBITS
2799 010664 010461 000002      MOV    R4,2(R1) ;BOTH WORDS
2800 010670 020116      CMP    R1,(SP) ;BOTTOM YET ?
2801 010672 001335          BNE   3$ ;BR IF NO
2802 010674 010377 172030          5$:    MOV    R3,0CSRADR ;RESTORE CSR
2803 010700 005711          TST   (R1) ;READ THE LOC AGAIN
2804 010702 017700 172022      MOV    0CSRADR,R0 ;GET CSR CONTENTS INTO R0
2805 010706 042700 170037      BIC    #170037,R0 ;ONLY WANT BITS 11-5
2806 010712 022700 003740      CMP    #3740,R0 ;READ FOR ALL 1'S
2807 010716 001421          BEQ   6$ ;BR IF OK
2808 010720 012700 003740      MOV    #3740,R0 ;FOR ERROR MSG
2809 010724 004767 004722      JSR    PC,BITCHK ;
2810 010730 032777 004000 171774      BIT    #4000,0SWR ;ECC DIS ?

```

|      |        |        |        |        |     |               |       |                               |
|------|--------|--------|--------|--------|-----|---------------|-------|-------------------------------|
| 2811 | 010736 | 001004 |        |        | BNE | 55\$          |       |                               |
| 2812 | 010740 | 022767 | 000001 | 004766 | CMP | #1,BITCNT     |       | :MORE THAN ONE ?              |
| 2813 | 010746 | 001405 |        |        | BEQ | 6\$           |       | :NO                           |
| 2814 | 010750 | 012700 | 003740 |        | MOV | #3740,R0      | 55\$: | :FOR MSG                      |
| 2815 | 010754 | 004767 | 002560 |        | JSR | PC,ERROR      |       | :*ERROR* REPORT ERROR MESSAGE |
| (1)  | 010760 | 000071 |        |        | 71  |               |       | :*****ERROR NUMBER 71*****    |
| (1)  |        |        |        |        |     |               |       |                               |
| 2816 | 010762 | 010377 | 171742 |        | MOV | R3,ACSRADR    | 6\$:  | :RESTORE CSR                  |
| 2817 | 010766 | 010411 |        |        | MOV | R4,(R1)       |       | :ACCESS THE LOC               |
| 2818 | 010770 | 012761 | 100000 | 000002 | MOV | #100000,2(R1) |       | :SECOND WORD                  |
| 2819 | 010776 | 062701 | 000004 |        | ADD | #4,R1         |       | :POINT TO NEXT ADDRESS        |
| 2820 | 011002 | 020105 |        |        | CMP | R1,R5         |       | :TOP + 2 YET?                 |
| 2821 | 011004 | 103733 |        |        | BLO | 5\$           |       | :BR IF NOT                    |
| 2822 | 011006 | 011601 |        |        | MOV | (SP),R1       |       | :RESTORE MINMEM               |
| 2823 | 011010 | 010377 | 171714 |        | MOV | R3,ACSRADR    | 7\$:  | :RESTORE CSR                  |
| 2824 | 011014 | 005711 |        |        | TST | (R1)          |       | :READ                         |
| 2825 | 011016 | 032777 | 003740 | 171704 | BIT | #3740,ACSRADR |       | :SHOULD BR 0                  |
| 2826 | 011024 | 001421 |        |        | BEQ | 8\$           |       |                               |
| 2827 | 011026 | 012700 | 004000 |        | MOV | #4000,R0      |       | :GOOD DATA                    |
| 2828 | 011032 | 004767 | 004614 |        | JSR | PC,BITCHK     |       | :CHECK # OF BAD BITS          |
| 2829 | 011036 | 032777 | 004000 | 171666 | BIT | #4000,ASWR    |       | :ECC DIS ?                    |
| 2830 | 011044 | 001004 |        |        | BNE | 77\$          |       | :YES                          |
| 2831 | 011046 | 022767 | 000001 | 004660 | CMP | #1,BITCNT     |       | :MORE THAN ONE                |
| 2832 | 011054 | 001405 |        |        | BEQ | 8\$           |       | :NO                           |
| 2833 | 011056 | 012700 | 004000 |        | MOV | #4000,R0      | 77\$: | :FOR PRINTOUT                 |
| 2834 | 011062 | 004767 | 002452 |        | JSR | PC,ERROR      |       | :*ERROR* REPORT ERROR MESSAGE |
| (1)  | 011066 | 000072 |        |        | 72  |               |       | :*****ERROR NUMBER 72*****    |
| (1)  |        |        |        |        |     |               |       |                               |
| 2835 | 011070 | 052777 | 003740 | 171632 | BIS | #3740,ACSRADR | 8\$:  | :COMP CHECKBITS               |
| 2836 | 011076 | 042777 | 004000 | 171624 | BIC | #4000,ACSRADR |       | :CLEAR CT                     |
| 2837 | 011104 | 010411 |        |        | MOV | R4,(R1)       |       | :WRITE                        |
| 2838 | 011106 | 010461 | 000002 |        | MOV | R4,2(R1)      |       |                               |
| 2839 | 011112 | 062701 | 000004 |        | ADD | #4,R1         |       | :NEXT WORD                    |
| 2840 | 011116 | 020105 |        |        | CMP | R1,R5         |       | :TOP + 2 ?                    |
| 2841 | 011120 | 103733 |        |        | BLO | 7\$           |       | :BR IF NO                     |
| 2842 | 011122 | 010377 | 171602 |        | MOV | R3,ACSRADR    |       | :RESTORE CSR                  |
| 2843 | 011126 | 162701 | 000004 |        | SUB | #4,R1         | 9\$:  | :ADJUST R1                    |
| 2844 | 011132 | 005711 |        |        | TST | (R1)          |       | :READ                         |
| 2845 | 011134 | 017700 | 171570 |        | MOV | ACSRADR,R0    |       | :GET CSR CONTENTS             |
| 2846 | 011140 | 042700 | 170037 |        | BIC | #170037,R0    |       | :ONLY WANT CHECKBITS          |
| 2847 | 011144 | 022700 | 003740 |        | CMP | #3740,R0      |       | :ALL ONES ?                   |
| 2848 | 011150 | 001421 |        |        | BEQ | 10\$          |       | :BR IF OK                     |
| 2849 | 011152 | 012700 | 003740 |        | MOV | #3740,R0      |       | :FOR ERROR MSG                |
| 2850 | 011156 | 004767 | 004470 |        | JSR | PC,BITCHK     |       | :COUNT BAD BITS               |
| 2851 | 011162 | 032777 | 004000 | 171542 | BIT | #4000,ASWR    |       | :ECC DIS                      |
| 2852 | 011170 | 001004 |        |        | BNE | 99\$          |       | :YES                          |
| 2853 | 011172 | 022767 | 000001 | 004534 | CMP | #1,BITCNT     |       | :MORE THAN ONE ?              |
| 2854 | 011200 | 001405 |        |        | BEQ | 10\$          |       | :NO                           |
| 2855 | 011202 | 012700 | 003740 |        | MOV | #3740,R0      | 99\$: | :FOR MSG.                     |
| 2856 | 011206 | 004767 | 002326 |        | JSR | PC,ERROR      |       | :*ERROR* REPORT ERROR MESSAGE |
| (1)  | 011212 | 000073 |        |        | 73  |               |       | :*****ERROR NUMBER 73*****    |
| (1)  |        |        |        |        |     |               |       |                               |
| 2857 | 011214 | 010377 | 171510 |        | MOV | R3,ACSRADR    | 10\$: | :RESTORE CSR                  |
| 2858 | 011220 | 010411 |        |        | MOV | R4,(R1)       |       | :WRITE NEW CHECKBITS          |
| 2859 | 011222 | 012761 | 100000 | 000002 | MOV | #100000,2(R1) |       | :BIT 31                       |
| 2860 | 011230 | 020116 |        |        | CMP | R1,(SP)       |       | :BOTTOM YET ?                 |



## E05

DZMPL MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-55  
 DZMPLB.P11 05-AUG-77 12:57

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

SEQ 0057

|      |        |        |        |        |        |      |                |  |  |
|------|--------|--------|--------|--------|--------|------|----------------|--|--|
| 2861 | 011232 | 001335 |        |        |        | BNE  | 9\$            |  | ;BR IF NOT                                       |
| 2862 | 011234 | 005711 |        |        | 11\$:  | TST  | (R1)           |  | ;RERAD   |
| 2863 | 011236 | 032777 | 003740 | 171464 |        | BIT  | #3740, @CSRADR |  | ;SHOULD BE ALL 0'S                               |
| 2864 | 011244 | 001421 |        |        |        | BEQ  | 12\$           |  | ;BR IF OK  |
| 2865 | 011246 | 012700 | 004000 |        |        | MOV  | #4000, R0      |  | ;GOOD DATA                                       |
| 2866 | 011252 | 004767 | 004374 |        |        | JSR  | PC, BITCHK     |  | ;COUNT BAD BITS                                  |
| 2867 | 011256 | 032777 | 004000 | 171446 |        | BIT  | #4000, @SWR    |  | ;ECC DIS ?                                       |
| 2868 | 011264 | 001004 |        |        |        | BNE  | 111\$          |  | ;YES   |
| 2869 | 011266 | 022767 | 000001 | 004440 |        | CMP  | #1, BITCNT     |  | ;MORE THAN 1                                     |
| 2870 | 011274 | 001405 |        |        |        | BEQ  | 12\$           |  | ;BR NO   |
| 2871 | 011276 | 012700 | 004000 |        | 111\$: | MOV  | #4000, R0      |  | ;FOR MSG   |
| 2872 | 011302 | 004767 | 002232 |        |        | JSR  | PC, ERROR      |  | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 011306 | 000074 |        |        |        | 74   |                |  | ;*****ERROR NUMBER 74*****                       |
| (1)  |        |        |        |        |        |      |                |  |  |
| 2873 | 011310 | 062701 | 000004 |        | 12\$:  | ADD  | #4, R1         |  | ;POINT TO NEXT WORD                              |
| 2874 | 011314 | 010377 | 171410 |        |        | MOV  | R3, @CSRADR    |  | ;RESTORE CSR                                     |
| 2875 | 011320 | 020105 |        |        |        | CMP  | R1, R5         |  | ;TOP + 2 YET?                                    |
| 2876 | 011322 | 103744 |        |        |        | BLO  | 11\$           |  | ;BR IF NOT                                       |
| 2877 | 011324 | 005726 |        |        |        | TST  | (SP)+          |  | ;FINALLY RESTORE STACK                           |
| 2878 | 011326 | 012604 |        |        |        | MOV  | (SP)+, R4      |  | ;RESTORE R4                                      |
| 2879 | 011330 | 000167 | 171252 |        | END15: | JMP  | TSTSCP         |  | ;AND EXIT  |
| 2880 |        |        |        |        |        |      |                |  |  |
| 2881 |        |        |        |        |        |      |                |  |  |
| 2882 |        |        |        |        |        |      |                |  |  |
| 2883 |        |        |        |        |        |      |                |  |  |
| 2884 |        |        |        |        |        |      |                |  |  |
| 2885 |        |        |        |        |        |      |                |  |  |
| 2886 |        |        |        |        |        |      |                |  |  |
| 2887 |        |        |        |        |        |      |                |  |  |
| 2888 |        |        |        |        |        |      |                |  |  |
| 2889 |        |        |        |        |        |      |                |  |  |
| 2890 |        |        |        |        |        |      |                |  |  |
| 2891 |        |        |        |        |        |      |                |  |  |
| 2892 |        |        |        |        |        |      |                |  |  |
| 2893 |        |        |        |        |        |      |                |  |  |
| 2894 | 011334 | 122737 | 000016 | 000404 | TST16: | CMPB | #16, @STESTN   |  | ;CHECK FOR PROPER TEST SEQUENCE                  |
| 2895 | 011342 | 001403 |        |        |        | BEQ  | 1\$            |  |  |
| 2896 | 011344 | 004767 | 002644 |        |        | JSR  | PC, SEQERR     |  | ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT |
| (1)  | 011350 | 000075 |        |        |        | 75   |                |  | ;*****ERROR NUMBER 75*****                       |
| (1)  |        |        |        |        |        |      |                |  |  |
| 2897 | 011352 | 004737 | 000120 |        | 1\$:   | JSR  | PC, @WRTMEM    |  | ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE |
| 2898 |        |        |        |        |        |      |                |  | ;WORD STORED IN BAKPAT                           |
| 2899 | 011356 | 020041 |        |        | 2\$:   | CMP  | R0, -(R1)      |  | ;READ THE CONTENTS OF LOCATION POINTED BY R1     |
| 2900 | 011360 | 001403 |        |        |        | BEQ  | 3\$            |  | ;TO SEE IF IT HAS THE SAME VALUE AS R0           |
| 2901 | 011362 | 004767 | 002152 |        |        | JSR  | PC, ERROR      |  | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 011366 | 000076 |        |        |        | 76   |                |  | ;*****ERROR NUMBER 76*****                       |
| (1)  |        |        |        |        |        |      |                |  |  |
| 2902 | 011370 | 000300 |        |        | 3\$:   | SWAB | R0             |  |  |
| 2903 | 011372 | 010011 |        |        |        | MOV  | R0, (R1)       |  | ;SWAP THE BYTES AT (R1)                          |
| 2904 | 011374 | 021100 |        |        |        | CMP  | (R1), R0       |  | ;READ (R1) FOR CORRECT VALUE                     |
| 2905 | 011376 | 001403 |        |        |        | BEQ  | 4\$            |  |  |
| 2906 | 011400 | 004767 | 002134 |        |        | JSR  | PC, ERROR      |  | ;*ERROR* REPORT ERROR MESSAGE                    |
| (1)  | 011404 | 000077 |        |        |        | 77   |                |  | ;*****ERROR NUMBER 77*****                       |
| (1)  |        |        |        |        |        |      |                |  |  |
| 2907 |        |        |        |        |        |      |                |  |  |
| 2908 | 011406 | 000300 |        |        | 4\$:   | SWAB | R0             |  | ;SWAP THE BYTES OF THE REGISTER                  |

```

;*(1) THIS TEST WRITES A BACK GROUND OF THE WORD STORED
;*(2) AT BAKPAT.
;*(3) READS EVERY LOCATION FOR CORRECT DATA, SWAPS BYTES
;*(4) AT THE LOCATION AND PROCEEDS IN MAX. TO MIN
;*(5) DIRECTION OF MEMORY LOCATIONS.
;*(1) READS EVERY LOCATION FOR SWAPPED BAKPAT PATTERN
;*(2) WRITES BAKPAT BACKGROUND IN THE LOCATION AND PROCEEDS
;*(3) IN MIN. TO MAX. DIRECTION
;*(4) REPEATS STEP 2 GOING IN MIN. TO MAX. DIRECTION
;*(5) REPEATS STEP 3 GOING IN MAX. TO MIN. DIRECTION

```

# F05

DZMLB.P11 MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-56  
 05-AUG-77 12:57

SEQ 0058

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

```

2909
2910 011410 001023          BNE      9$
2911
2912
2913
2914
2915
2916
2917
2918 011412 005703          5$:     TST      R3
2919
2920
2921
2922
2923 011414 001023          BNE      10$
2924 011416 062701 000002    6$:     ADD      #2,R1
2925 011422 020105          CMP      R1,R5
2926
2927 011424 103006          BHS      8$
2928 011426 020011          7$:     CMP      RO,(R1)
2929 011430 001757          BEQ      3$
2930
2931 011432 004767 002102    JSR      PC,ERROR
2932 (1) 011436 000100
2933 (1)
2934 011440 000753          BR       3$
2935 011442 105237 000302    8$:     INCB    #PASFLG
2936 011446 000300          SWAB    RO
2937 011450 001742          BEQ      2$
2938
2939 011452 005103          COM      R3
2940 011454 010401          MOV      R4,R1
2941 011456 000763          BR       7$
2942
2943 011460 005703          9$:     TST      R3
2944 011462 001353          BNE      5$
2945
2946
2947 011464 020104          10$:    CMP      R1,R4
2948
2949 011466 101333          BHI      2$
2950 011470 105237 000302    INCB    #PASFLG
2951 011474 000300          SWAB    RO
2952 011476 001753          BEQ      7$
2953
2954
2955 011500 000167 171102    END16:  JMP      TSTSCP
2956
2957
2958
2959
2960
2961
2962

; CONTAINING BACKGROUND PATTERN
; IF THE LOWER BYTE OF THE REGISTER
; IS NOT 0 THEN THE PROGRAM IS READING
; THE MEMORY TO CONTAIN A BACK GROUND OF
; BAKPAT AND WRITING THE SWAPPED WORD
; IN WHICH CASE GO TO 9$

; R3 WAS 0 WHEN THE PROGRAM ENTERED
; THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
; IF R3 EQUAL 0 THEN THE PROGRAM IS
; READING/WRITING MIN. TO MAX. OTHERWISE
; IT IS GOING IN MAX. TO MIN. DIRECTION
; IF R3 IS NOT CLEAR THEN GO TO 10$
; OTHERWISE ADD 2 TO THE CONTENTS OF R1
; COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
; BE TESTED
; IF R1>R5 THEN GO TO 8$ OTHERWISE
; READ (R1) FOR THE CORRECT DATA
; WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
; AND REPEAT UNTIL R1 > R5
; *ERROR* REPORT ERROR MESSAGE
; *****ERROR NUMBER 100*****

; IF THE LOWER BYTE OF RO IS ALL 0'S
; THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
; AND READING BAKPAT GOING FROM MAX. TO MIN. [PASFLG=4]
; OTHERWISE CLEAR RO
; PUT THE LOWEST TESTING ADDRESS IN R1
; AND BEGIN READING 0'S, WRITING 1'S AND
; READING 1'S IN MIN. TO MAX. DIRECTION [PASFLG=3]

; IF R3 IS NON 0, I.E. PASFLG=3
; THEN READ BAKPAT, WRITE
; SWAPPED BAKPAT AND READ SWAPPED BAKPAT
; IN MIN. TO MAX. DIRECTION
; OTHERWISE TEST IS PROCEEDING IN MAX. TO
; MIN. DIRECTION.
; KEEP ON LOOPING UNTIL R1=R4

; IF RO SWAPPED HAS LOWER BYTE=0
; THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
; AND READ BAKPAT GOING FROM MIN. TO MAX.

;*(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

```

## G05

DZMPL MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-57  
 DZMPLB.P11 05-AUG-77 12:57

SEQ 0059

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

```

2963          ;*(4) REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
2964          ;*(5) IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
2965          ;*   BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
2966          ;*   SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
2967          ;*(6) REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
2968          ;*   BAKPAT INSTEAD OF BAKPAT.
2969 011504 122737 000017 000404 TST17: CMPB  #17, #STESTN ;CHECK FOR PROPER TEST SEQUENCE
2970 011512 001403          BEQ    .+10
2971 011514 004767 002474          JSR    PC, SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 011520 000101          ;*****ERROR NUMBER 101*****
(1)
2972 011522 004737 000120          RPT17: JSR    PC, #WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
2973          ;WORD STORED AT LOCATION BAKPAT
2974 011526 005037 000302          CLR    #PASFLG
2975 011532 010403          1$: MOV   R4, R3 ;SET R3
2976 011534 010401          2$: MOV   R4, R1 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
2977 011536 020011          3$: CMP   R0, (R1) ;CHECK (R1) FOR CORRECT DATA
2978 011540 001403          BEQ    4$
2979 011542 004767 001772          JSR    PC, ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 011546 000102          ;*****ERROR NUMBER 102*****
(1)
2980 011550 062701 000002          4$: ADD   #2, R1 ;INCREMENT R1 BY 2
2981 011554 020105          CMP   R1, R5 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
2982 011556 103767          BLO   3$
2983 011560 132737 000001 000302 BITB  #1, #PASFLG ;CHECK TO SEE IF PASFLG=0 OR 2
2984 011566 001002          BNE   5$
2985 011570 105237 000302          INCB  #PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
2986
2987 011574 020305          5$: CMP   R3, R5 ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
2988 011576 103012          BHIS  7$
2989 011600 012702 037776          MOV   #37776, R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
2990 011604 000313          6$: SWAB (R3)
2991 011606 005302          DEC   R2
2992 011610 001375          BNE   6$
2993 011612 010337 000350          MOV   R3, #SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
2994 011616 062703 020000          ADD   #20000, R3 ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
2995          ;R3 TO POINT TO A LOCATION IN THE NEXT
2996          ;4K BANK OF MEMORY
2997 011622 000744          BR    2$
2998 011624 105237 000302          7$: INCB  #PASFLG ;MAKE PASFLG=2
2999 011630 000337 000312          SWAB  #BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
3000 011634 001732          BEQ   RPT17 ;THEN GO BACK TO THE LOCATION RPT17
3001 011636 000167 170744          END17: JMP   TSTSCP
3002
3003
3004
3005
3006
3007
3008
3009

```

```

3014 011642 005077 171062 RELOC: CLR      @CSRADR
3015 011646 012737 000377 000312 MOV      @377,@#BAKPAT
3016 011654 105737 000272 TSTB    @MMAVA
3017 011660 001170 BNE     CONTMM
3018
3019 011662 005767 171030 TST     LDFLG
3020 011666 001542 BEQ     CKDONE
3021 011670 032777 001000 171034 BIT     @1000,@SWR
3022 011676 001136 BNE     CKDONE
3023 011700 005767 171022 TST     INHREL
3024 011704 001133 BNE     CKDONE
3025 011706 105737 000170 TSTB    @REL
3026 011712 100473 BMI     RELOER
3027 011714 112737 000200 000170 MOVB    @200,@REL
3028
3029
3030
3031
3032 011722 012701 100500 MOV      @100000+BEGIN,R1 ;START OF RELOCATED VERSION
3033 011726 012702 116132 MOV      @100000+ENDPRG,R2 ;END
3034 011732 005711 3S: TST     (R1) ;READ A WORD
3035 011734 032777 100000 170766 BIT     @100000,@CSRADR ;CHECK FOR DBE
3036 011742 001005 BNE     5$ ;BR IF ERROR
3037 011744 062701 000004 ADD     @4,R1 ;NEXT WORD
3038 011750 020102 CMP     R1,R2 ;END YET ?
3039 011752 103767 BLO    3$ ;BR IF NOT
3040 011754 000405 BR     4$ ;BR IF ALL IS OK
3041 011756 105037 000170 5S: CLRB   @REL ;INDICATE NO RELOC
3042 011762 004767 002226 JSR     PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 011766 000103 103 ;*****ERROR NUMBER 103*****
(1)
3043 011770 032777 000040 170734 4S: BIT     @40,@SWR ;PRINTING WANTED
3044 011776 001005 BNE     1$ ;BR IF NO
3045 012000 004767 002434 JSR     PC,PNTMES ;TYPE "RELOC"
3046 012004 042522 047514 000103 .ASCIZ /RELOC/
3047 .EVEN
3048 012012 004767 003270 1S: JSR     PC,CLMM ;CLEAR MM REGS
3049 012016 012737 000500 000320 MOV     @BEGIN,@MINMEM ;NEW TEST ADDRESS
3050 012024 052767 000010 170656 BIS     @10,ECCDIS ;PROTECT 2ND 16K
3051 012032 052767 000010 170652 BIS     @10,DIAGA ;DITTO
3052 012040 012705 100000 MOV     @10000,R5 ;START OF 2ND 16K
3053 012044 010567 170650 MOV     R5,BASE ;BASE FOR RELOC
3054 012050 012704 000430 MOV     @BEGIN-50,R4 ;FIRST LOC TO BE RELOCATED
3055 012054 060405 ADD     R4,R5
3056 012056 012425 2S: MOV     (R4)+,(R5)+ ;MOV A WORD
3057 012060 020437 000342 CMP     R4,@$AVR4 ;LAST LOC YET ?
3058 012064 103774 BLO    2$ ;BR IF NO
3059 012066 012704 000430 MOV     @BEGIN-50,R4 ;FIRST TEST ADDRESS
3060 012072 016705 170622 MOV     BASE,R5
3061 012076 000137 100500 JMP     @BEGIN+100000 ;JUMP TO RELOCATED VERSION
3062
3063
3064
3065
3066 012102 105737 000170 RELOER: TSTB   @REL ;IS DIAGNOSTIC IN RELOCATED STATE?
3067 012106 100032 BPL     CKDONE ;BRANCH IF NO
    
```

;RELOCATE THE DIAGNOSTIC TO SECOND 16K

;\*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY

|      |        |        |        |        |             |                  |   |
|------|--------|--------|--------|--------|-------------|------------------|---|
| 3068 | 012110 | 005067 | 170604 |        | CLR         | BASE             |   |
| 3069 | 012114 | 042767 | 000010 | 170566 | BIC         | #10,ECCDIS       |   |
| 3070 | 012122 | 042767 | 000010 | 170562 | BIC         | #10,DIAGA        |   |
| 3071 |        |        |        |        |             |                  |   |
| 3072 | 012130 | 016737 | 170566 | 000320 | MOV         | SAVMIN, @#MINMEM | ; RESTORE TEST ADDRESS FOR ECC                      |
| 3073 | 012136 | 012704 | 000430 |        | MOV         | #BEGIN-50, R4    | ; PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE     |
| 3074 | 012142 | 010405 |        |        | MOV         | R4, R5           |   |
| 3075 | 012144 | 062705 | 100000 |        | ADD         | #100000, R5      |   |
| 3076 | 012150 | 012524 |        |        | MOV         | (R5)+, (R4)+     |   |
| 3077 | 012152 | 020437 | 000342 |        | CMP         | R4, @#SAVR4      |   |
| 3078 | 012156 | 103774 |        |        | BLO         | 2\$              |   |
| 3079 | 012160 | 105037 | 000170 |        | CLRB        | @#REL            |   |
| 3080 | 012164 | 012706 | 000500 |        | MOV         | #BEGIN, SP       | ; RESET STACK TO LOWER MEMORY                       |
| 3081 | 012170 | 010637 | 000346 |        | MOV         | SP, @#SAVR6      | ; "BEGIN" USES THIS TO RESET THE STACK.             |
| 3082 | 012174 | 000137 | 012200 |        | JMP         | @#LOWER          | ; TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE    |
| 3083 |        |        |        |        |             |                  |   |
| 3084 |        |        |        |        |             |                  |   |
| 3085 |        |        |        |        |             |                  |   |
| 3086 | 012200 | 105737 | 000402 |        | LOWER: TSTB | @#SFATAL         | ; FATAL ERROR                                       |
| 3087 | 012204 | 001405 |        |        | BEQ         | 1\$              | ; BR IF NOT   |
| 3088 | 012206 | 005737 | 000042 |        | TST         | @#42             | ; APT ?   |
| 3089 | 012212 | 001402 |        |        | BEQ         | 1\$              | ; NO KEEP TESTING                                   |
| 3090 | 012214 | 000167 | 002040 |        | JMP         | APTHLT           |   |
| 3091 | 012220 | 105737 | 000311 |        | 1\$: TSTB   | @#SAVKBB         | ; HERE DUE TO IC TYPED?                             |
| 3092 | 012224 | 001141 |        |        | BNE         | STPSTK           | ; BRANCH IF YES (TYPE ERROR STACK)                  |
| 3093 | 012226 | 004767 | 002504 |        | TSTM: JSR   | PC, MEMMNG       | ; SET THE REGISTERS IF THE MEMORY MANAGEMENT        |
| 3094 |        |        |        |        |             |                  | ; IS AVAILABLE                                      |
| 3095 | 012232 | 105737 | 000272 |        | TSTB        | @#MMAVA          | ; IS MEM. MANAG. AVAILABLE ?                        |
| 3096 | 012236 | 001474 |        |        | BEQ         | ENDPAS           | ; BRANCH IF NO                                      |
| 3097 | 012240 | 000406 |        |        | BR          | \$CNTMM          | ; BEGIN TESTING ABOVE 28K                           |
| 3098 | 012242 | 022737 | 007400 | 172352 | CONTMM: CMP | #7400, @#172352  | ; DON'T WANT TO WRAP AROUND                         |
| 3099 | 012250 | 001464 |        |        | BEQ         | ENDMAX           | ; GO TO END-OF-PASS                                 |
| 3100 | 012252 | 004767 | 002636 |        | JSR         | PC, UPMM         | ; GO TO UPDATE MEM. MANAG. REGISTERS                |
| 3101 | 012256 | 012703 | 000322 |        | SCNTMM: MOV | #LOWTWO, R3      | ; MAKE R3 POINT TO THE LOCATION LOWTWO              |
| 3102 | 012262 | 004767 | 002744 |        | JSR         | PC, GETSIZ       | ; LOAD BITS 6-10 OF R2 WITH THE BITS 13-17          |
| 3103 |        |        |        |        |             |                  | ; OF THE LOWEST ADDRESS UNDER TEST                  |
| 3104 | 012266 | 012704 | 040000 |        | MOV         | #40000, R4       | ; MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK  |
| 3105 |        |        |        |        |             |                  | ; POINTED BY PAGE ADDRESS REGISTER 2 (PAR2)         |
| 3106 | 012272 | 020237 | 172344 |        | CMP         | R2, @#172344     | ; IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF    |
| 3107 |        |        |        |        |             |                  | ; PAR2 ?  |
| 3108 | 012276 | 103405 |        |        | BLO         | 2\$              | ; IF SO THEN GO 2\$                                 |
| 3109 | 012300 | 050104 |        |        | BIS         | R1, R4           | ; SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12        |
| 3110 |        |        |        |        |             |                  | ; OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4        |
| 3111 | 012302 | 162702 | 000200 |        | SUB         | #200, R2         |   |
| 3112 | 012306 | 004767 | 002430 |        | JSR         | PC, MAREG        | ; SET MEM. MANAG. REGISTERS                         |
| 3113 | 012312 | 010437 | 000320 |        | MOV         | R4, @#MINMEM     | ; NEW MINMEM FOR ECC TESTS                          |
| 3114 | 012316 | 004767 | 002710 |        | JSR         | PC, GETSIZ       | ; PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED |
| 3115 |        |        |        |        |             |                  | ; IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12   |
| 3116 |        |        |        |        |             |                  | ; OF LOCATION HIGHADD IN R1                         |
| 3117 | 012322 | 004767 | 000020 |        | JSR         | PC, MAXADR       | ; GET THE ADDRESS OF MAX. MEM. UNDER TEST           |
| 3118 | 012326 | 010005 |        |        | MOV         | R0, R5           |   |
| 3119 | 012330 | 004767 | 002676 |        | JSR         | PC, GETSIZ       | ; PREPARE TO SET UP LOCATION MAXMEM                 |
| 3120 | 012334 | 004767 | 000006 |        | JSR         | PC, MAXADR       | ; GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY       |
| 3121 | 012340 | 010013 |        |        | MOV         | R0, (R3)         | ; AND STORE INTO "MAXMEM"                           |
| 3122 | 012342 | 000167 | 167664 |        | JMP         | CLRMEM           | ; GO TEST A 20K SLICE ABOVE 28K.                    |
| 3123 |        |        |        |        |             |                  |   |

```

3124
3125
3126
3127
3128
3129
3130
3131 012346 010046
3132 012350 012700 172356
3133
3134 012354 162716 020000
3135 012360 050116
3136 012362 020240
3137 012364 001411
3138 012366 020027 172340
3139 012372 101370
3140
3141 012374 062700 000004
3142 012400 021002
3143 012402 003006
3144 012404 012716 157776
3145 012410 012600
3146 012412 062700 000002
3147 012416 000207
3148
3149 012420 022626
3150
3151 012422 016737 170274 000320

```

```

;MAXADR - SUBROUTINE TO GET CURRENT 20K SLICE OF MEMORY ADDRESSES ABOVE 28K.
;REGISTERS:
;R0= ON ENTRY= #160000 AND ON EXIT=HIGHEST VIRTUAL ADDR. UNDER TEST
;R1= LOW ORDER 12 BITS OF VIRTUAL TEST ADDRESS
;R2= PAR BLOCK NO. CURRENTLY UNDER TEST.

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

4$: CMP (SP)+, (SP)+ ;FIXUP STACK
;AND FALL THRU TO ENDPAS.
ENDMAX: MOV SAVMIN, @#MINMEM ;BECAUSE WE WON'T SIZE AGAIN

```



L05

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-62  
DZMMLB.P11 05-AUG-77 12:57

TYPE ROUTINE FOR ERROR STACK

SEQ 0064

3212 012644 012703 000376  
3213 012650 000743

MOV #376,R3  
BR RETSTK

;RESET SCRATCH STACK FOR EACH BIT PRINTED.



```

3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229 012652 005002          SEOP: CLR      R2          ;SET R2= PARITY MODULE DISABLE CODE
3230 012654 105737 000311  TSTB   @#SAVKBB      ;CONTROL-C TYPED?
3231 012660 001050          BNE    CTLC         ;BRANCH IF YES-RESTORE LOADERS AND HALT-
3232 012662 005237 000406  INC    @#$PASS      ;INCREMENT PASS COUNT
3233 012666 005067 170032  CLR    ERRFLG       ;RESET ERROR HEADER FOR NEXT PASS
3234 012672 032777 000040 170032 BIT    #40,@$WR     ;"END PASS #XX" PRINTOUT WANTED?
3235 012700 001015          BNE    ACT11        ;BRANCH IF NO
3236 012702 004767 001524  TYPEOP: JSR   PC,TPCRLF ; TYPE CR, LF, AND "END PASS #"
3237 012706 047105 020104 040520 .ASCIZ /END PASS #/
3238 012714 051523 021440 000 .EVEN
3239 012722 013737 000406 000306 MOV    @#$PASS,@#DECDWD ;GET PASS COUNT
3240 012730 004767 001532  ACT11: JSR   PC,$TPDEC ;TYPE IT
3241 012734 013700 000042  MOV    @#42,R0      ;GET THE MONITOR ADDRESS
3242 012740 001405          BEQ    $DOAGN       ;IF NONE
3243 012742 004767 000012  JSR   PC,RLODER     ;RESTORE XXDP MONITOR
3244 012746 000005          RESET ;RETURN TO ACT11 MONITOR.
3245
3246
3247 ;* SERVICE XXDP/ACT11
3248 012750 000137 000156  JMP    @#$ENDAD     ;JUMP TO ACT SERVICE
3249
3250 012754 000137 000250  $DOAGN: JMP   @#RESTRT ;REPEAT TEST IF NOT UNDER ACT11/XXDP
3251
3252 012760 004767 002322  RLODER: JSR   PC,CLRMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
3253 012764 013704 000342  MOV    @#SAVR4,R4   ;RESTORE R4 WITH SAVR4
3254 012770 014445          4$:  MOV    -(R4),-(R5) ;RESTORE LOADERS
3255 012772 020437 000304  CMP    R4,@#ENDSTK
3256 012776 101374          BHI    4$
3257 013000 000207          RTS    PC           ;RETURN FROM RLODER CALL
3258
3259 ;CONTROL C HANDLER
3260
3261 013002 004767 177752  CTLC:  JSR   PC,RLODER ;RESTORE ABS LOADER
3262 013006 000167 001246  JMP    APTHLT       ;IF NOT APT HALT AT FATHLT
3263
3264
3265 ;*****
3266
3267 ;CHECK BIT GENERATOR ROUTINE
3268 ;CALL: JSR PC,CHKGEN
3269
3270 ;
3271 ; SOURCE = ADDRESS OF DATA
3272 ; CHECK = WORD CONTAINING GENERATED CHECKBITS
    
```

|      |        |        |        |         |       |             |  |
|------|--------|--------|--------|---------|-------|-------------|--|
| 3273 | 013012 | 000000 |        | SOURCE: | .WORD | 0           |  |
| 3274 | 013014 | 000000 |        | CHECK:  | .WORD | 0           |  |
| 3275 | 013016 | 125252 |        | CHKTAB: | .WORD | 125252      | : C1 BIT TABLE                         |
| 3276 | 013020 | 125252 |        |         | .WORD | 125252      |  |
| 3277 | 013022 | 146314 |        |         | .WORD | 146314      | : C2 BIT TABLE                         |
| 3278 | 013024 | 146314 |        |         | .WORD | 146314      |  |
| 3279 | 013026 | 170360 |        |         | .WORD | 170360      | : C4 BIT TABLE                         |
| 3280 | 013030 | 170360 |        |         | .WORD | 170360      |  |
| 3281 | 013032 | 177777 |        |         | .WORD | 177777      | : C8 BIT TABLE                         |
| 3282 | 013034 | 177400 |        |         | .WORD | 177400      |  |
| 3283 | 013036 | 000377 |        |         | .WORD | 377         | : C16 BIT TABLE                        |
| 3284 | 013040 | 177777 |        |         | .WORD | 177777      |  |
| 3285 | 013042 | 177400 |        |         | .WORD | 177400      | : C32 BIT TABLE                        |
| 3286 | 013044 | 177777 |        |         | .WORD | 177777      |  |
| 3287 | 013046 | 064551 |        |         | .WORD | 64551       | : CT BIT TABLE                         |
| 3288 | 013050 | 113151 |        |         | .WORD | 113151      |  |
| 3289 |        |        |        |         |       |             |  |
| 3290 | 013052 | 000    |        | UPPFLG: | .BYTE | 0           |  |
| 3291 |        |        |        |         |       |             |  |
| 3292 | 013053 | 000    |        | CHKCNT: | .BYTE | 0           | : C1 PARITY COUNTER                    |
| 3293 | 013054 | 000    |        |         | .BYTE | 0           | : C2 PARITY COUNTER                    |
| 3294 | 013055 | 000    |        |         | .BYTE | 0           | : C4 PARITY COUNTER                    |
| 3295 | 013056 | 000    |        |         | .BYTE | 0           | : C8 PARITY COUNTER                    |
| 3296 | 013057 | 000    |        |         | .BYTE | 0           | : C16 PARITY COUNTER                   |
| 3297 | 013060 | 000    |        |         | .BYTE | 0           | : C32 PARITY COUNTER                   |
| 3298 | 013061 | 000    |        |         | .BYTE | 0           | : CT PARITY COUNTER                    |
| 3299 | 013062 | 077    |        | PARBYT: | .BYTE | 77          | : ODD/EVEN PARITY FLAGS                |
| 3300 |        |        |        |         |       |             | : 0=EVEN                               |
| 3301 |        |        |        |         |       |             | : 1=ODD                                |
| 3302 |        |        |        |         |       |             | : CT C32 C16 C8 C4 C2 C1               |
| 3303 |        | 013064 |        | CHKGEN: | .EVEN |             |  |
| 3304 | 013064 | 004467 | 000236 |         | JSR   | R4, SAVOT4  | : SAVE R0 TO R4                        |
| 3305 | 013070 | 005000 |        |         | CLR   | R0          | : CLEARING OUT PARITY COUNTERS         |
| 3306 | 013072 | 012701 | 013053 |         | MOV   | #CHKCNT, R1 | : START 7 COUNTERS                     |
| 3307 | 013076 | 066701 | 167616 |         | ADD   | BASE, R1    |  |
| 3308 | 013102 | 022700 | 000007 | 10\$:   | CMP   | #7, R0      | : FINISHED?                            |
| 3309 | 013106 | 001403 |        |         | BEQ   | 20\$        | : BRANCH IF YES                        |
| 3310 | 013110 | 105021 |        |         | CLRB  | (R1)+       | : NO, DO THE NEXT BYTE                 |
| 3311 | 013112 | 005200 |        |         | INC   | R0          | : BUMP THE COUNT                       |
| 3312 | 013114 | 000772 |        |         | BR    | 10\$        | : LOOP                                 |
| 3313 | 013116 | 012701 | 000001 | 20\$:   | MOV   | #1, R1      | : SET THE TEST BIT                     |
| 3314 | 013122 | 012702 | 013016 | 30\$:   | MOV   | #CHKTAB, R2 | : START OF CHECK BIT TABLE             |
| 3315 | 013126 | 066702 | 167566 |         | ADD   | BASE, R2    |  |
| 3316 | 013132 | 012703 | 013053 |         | MOV   | #CHKCNT, R3 | : START ADDRESS OF PARITY COUNTERS     |
| 3317 | 013136 | 066703 | 167556 |         | ADD   | BASE, R3    |  |
| 3318 | 013142 | 012704 | 000006 |         | MOV   | #6, R4      | : FOR SEVEN PARITY COUNTERS            |
| 3319 | 013146 | 105067 | 177700 | 40\$:   | CLRB  | UPPFLG      | : INDICATE LOWER WORD                  |
| 3320 | 013152 | 016700 | 177634 |         | MOV   | SOURCE, R0  | : SET ADDRESS OF DATA IN R0            |
| 3321 | 013156 | 066700 | 167536 |         | ADD   | BASE, R0    |  |
| 3322 | 013162 | 030110 |        | 42\$:   | BIT   | R1, (R0)    | : CHECK LOWER 16 BITS FOR TEST BIT     |
| 3323 | 013164 | 001403 |        |         | BEQ   | 50\$        | : BRANCH IF NOT SET                    |
| 3324 | 013166 | 030112 |        |         | BIT   | R1, (R2)    | : LOOK FOR TEST BIT IN CHECK BIT TABLE |
| 3325 | 013170 | 001401 |        |         | BEQ   | 50\$        | : BRANCH IF NOT FOUND                  |
| 3326 | 013172 | 105213 |        |         | INCB  | (R3)        | : FOUND A MATCH, COUNT IT              |
| 3327 | 013174 | 062700 | 000002 | 50\$:   | ADD   | #2, R0      | : R0 NOW = SOURCE +2                   |
| 3328 | 013200 | 062702 | 000002 |         | ADD   | #2, R2      | : POINT TO SECOND WORD IN TABLE        |

|      |        |        |        |             |               |                                   |
|------|--------|--------|--------|-------------|---------------|-----------------------------------|
| 3329 | 013204 | 105767 | 177642 | TSTB        | UPPFLG        | :CHECK FOR UPPER/LOWER            |
| 3330 | 013210 | 001003 |        | BNE         | 60\$          | :BRANCH IF UPPER                  |
| 3331 | 013212 | 105267 | 177634 | INCB        | UPPFLG        | :CHANGE FLAG TO UPPER             |
| 3332 | 013216 | 000761 |        | BR          | 42\$          | :CHECK SECOND 16 BITS             |
| 3333 | 013220 | 005704 |        | 60\$: TST   | R4            | :LOOK FOR LAST PARITY COUNTER     |
| 3334 | 013222 | 001403 |        | BEQ         | 70\$          | :BRANCH IF DONE                   |
| 3335 | 013224 | 005203 |        | INC         | R3            | :ADVANCE PARITY COUNTER POINTER   |
| 3336 | 013226 | 005304 |        | DEC         | R4            | :ADVANCE CHECK BIT FLAG COUNTER   |
| 3337 | 013230 | 000746 |        | BR          | 40\$          | :GO CHECK NEXT CHECK BIT IN TABLE |
| 3338 | 013232 | 006301 |        | 70\$: ASL   | R1            | :SHIFT THE BIT OVER               |
| 3339 | 013234 | 103332 |        | BCC         | 30\$          | :ADJUST THE PARITY AND ASSEMBLE   |
| 3340 |        |        |        |             |               | :CHECK BIT WORD                   |
| 3341 | 013236 | 012700 | 000001 | MOV         | #1,R0         | :SET FIRST BIT IN R1              |
| 3342 | 013242 | 012701 | 013053 | MOV         | #CHKCNT,R1    | :ADDRESS OF C1 PARITY COUNTER     |
| 3343 | 013246 | 066701 | 167446 | ADD         | BASE,R1       |                                   |
| 3344 | 013252 | 130067 | 177604 | 80\$: BITB  | R0,PARBYT     | :CHECKING FOR ODD OR EVEN PARITY  |
| 3345 | 013256 | 001401 |        | BEQ         | 90\$          | :BRANCH IF EVEN PARITY            |
| 3346 | 013260 | 105211 |        | INCB        | (R1)          | :ODD, INC THE PARITY COUNTER      |
| 3347 | 013262 | 132711 | 000001 | 90\$: BITB  | #1,(R1)       | :TEST PARITY COUNTER FOR SET BIT  |
| 3348 | 013266 | 001401 |        | BEQ         | 100\$         |                                   |
| 3349 | 013270 | 050004 |        | BIS         | R0,R4         | :SET THE CHECK BIT IN CHECK       |
| 3350 | 013272 | 005201 |        | 100\$: INC  | R1            | :NEXT COUNTER                     |
| 3351 | 013274 | 006300 |        | ASL         | R0            | :SHIFT TEST BIT TO NEXT BIT       |
| 3352 | 013276 | 032700 | 000200 | BIT         | #200,R0       | :CT YET?                          |
| 3353 | 013302 | 001763 |        | BEQ         | 80\$          | :BRANCH IF NO                     |
| 3354 | 013304 | 000304 |        | SWAB        | R4            |                                   |
| 3355 | 013306 | 006004 |        | ROR         | R4            |                                   |
| 3356 | 013310 | 006004 |        | ROR         | R4            |                                   |
| 3357 | 013312 | 006004 |        | ROR         | R4            | :POSITION CHECK BITS IN BITS 11-5 |
| 3358 | 013314 | 010467 | 177474 | MOV         | R4,CHECK      |                                   |
| 3359 | 013320 | 004767 | 000014 | JSR         | PC,RSTOT4     | :RESTORE R0 TO R4                 |
| 3360 | 013324 | 000207 |        | RTS         | PC            |                                   |
| 3361 |        |        |        |             |               |                                   |
| 3362 |        |        |        |             |               |                                   |
| 3363 |        |        |        | :CALL       | JSR R4,SAVOT4 |                                   |
| 3364 | 013326 | 010346 |        | SAVOT4: MOV | R3,-(SP)      | :SAVE R3                          |
| 3365 | 013330 | 010246 |        | MOV         | R2,-(SP)      | :SAVE R2                          |
| 3366 | 013332 | 010146 |        | MOV         | R1,-(SP)      | :SAVE R1                          |
| 3367 | 013334 | 010046 |        | MOV         | R0,-(SP)      | :SAVE R0                          |
| 3368 | 013336 | 010407 |        | MOV         | R4,PC         | :R4 IS ALREADY SAVED              |
| 3369 |        |        |        |             |               |                                   |
| 3370 |        |        |        | :CALL       | JSR PC,RSTOT4 |                                   |
| 3371 | 013340 | 012604 |        | RSTOT4: MOV | (SP)+,R4      | :RETURN ADDRESS                   |
| 3372 | 013342 | 012600 |        | MOV         | (SP)+,R0      | :RESTORE R0                       |
| 3373 | 013344 | 012601 |        | MOV         | (SP)+,R1      | :R1                               |
| 3374 | 013346 | 012602 |        | MOV         | (SP)+,R2      | :R2                               |
| 3375 | 013350 | 012603 |        | MOV         | (SP)+,R3      | :R3                               |
| 3376 | 013352 | 000204 |        | RTS         | R4            | :RESTORE R4 AND RETURN            |
| 3377 |        |        |        |             |               |                                   |
| 3378 |        |        |        |             |               |                                   |
| 3379 |        |        |        |             |               |                                   |
| 3380 |        |        |        |             |               |                                   |
| 3381 |        |        |        |             |               |                                   |
| 3382 | 013354 | 010046 |        | DASHL: MOV  | R0,-(SP)      | :SAVE R0                          |
| 3383 | 013356 | 012500 |        | MOV         | (R5)+,R0      | :PICK UP ARGUMENT                 |
| 3384 | 013360 | 066700 | 167334 | ADD         | BASE,R0       | :                                 |

```

3385 013364 006360 000002          ASL      2(R0)          ;SHIFT HIGH 16 BITS
3386 013370 006310                ASL      (R0)          ;SHIFT LOW 16 BITS
3387 013372 103002                BCC      10$          ;BR IF LOW WORD BIT 15 = 0
3388 013374 005260 000002          INC      2(R0)        ;IT WAS = 1 INC HI WORD
3389 013400 012600                10$:    MOV      (SP)+,R0 ;RESTORE R0
3390 013402 000205                RTS
3391
3392 013404 012567 000112          BITCOM: MOV      (R5)+,MSKADR ;FIRST ARG IS ADDR OF MASK
3393 013410 012567 000110          MOV      (R5)+,DATADR ;SECOND IS ADDR OF DATA
3394 013414 010046                MOV      R0,-(SP)
3395 013416 010146                MOV      R1,-(SP)
3396 013420 010246                MOV      R2,-(SP)
3397 013422 066767 167272 000072  ADD      BASE,MSKADR ;SAVE THESE REGS
3398 013430 066767 167264 000066  ADD      BASE,DATADR
3399 013436 017700 000060          MOV      @MSKADR,R0 ;GET MASK
3400 013442 017701 000056          MOV      @DATADR,R1 ;GET DATA
3401 013446 004767 000054          JSR      PC,XOR      ;DO THE XOR
3402 013452 010277 000046          MOV      R2,@DATADR ;PUT RESULT BACK AS DATA
3403 013456 062767 000002 000036  ADD      #2,MSKADR ;NEXT WORD
3404 013464 062767 000002 000032  ADD      #2,DATADR ;DITTO
3405 013472 017700 000024          MOV      @MSKADR,R0
3406 013476 017701 000022          MOV      @DATADR,R1
3407 013502 004767 000020          JSR      PC,XOR      ;DO THE SAME FOR SECOND WORD
3408 013506 010277 000012          MOV      R2,@DATADR ;PUT RESULT BACK TOO
3409 013512 012602                MOV      (SP)+,R2
3410 013514 012601                MOV      (SP)+,R1
3411 013516 012600                MOV      (SP)+,R0
3412 013520 000205                RTS
3413
3414
3415 013522 000000                MSKADR: .WORD 0
3416 013524 000000                DATADR: .WORD 0
3417
3418
3419
3420
3421
3422
3423
3424
3425 013526 010102          XOR:    MOV      R1,R2 ;SAVE B OPERAND
3426 013530 040002          BIC      R0,R2 ;A NOT B
3427 013532 040100          BIC      R1,R0 ;B NOT A
3428 013534 050002          BIS      R0,R2 ;A XOR B
3429 013536 000207          RTS      PC ;EXIT AND RETURN
3430
3431
;*****

```

ERROR HANDLING ROUTINE

3436  
3437  
3438  
3439  
3440  
3441  
3442  
3443  
3444  
3445  
3446  
3447  
3448  
3449  
  
3450  
3451  
3452  
3453  
3454  
3455  
3456  
3457  
3458  
3459  
3460  
3461  
3462  
3463  
3464  
3465  
3466  
3467  
3468  
3469  
3470  
3471  
3472  
3473  
3474  
3475  
3476  
3477  
3478  
3479  
3480  
3481  
3482

\* ERROR HANDLING ROUTINE

-----

\* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS  
\* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER

```

013540 017637 000000 000402 ERROR: MOV 2(SP),2$SFATAL ;LOAD THE LOCATION SFATAL WITH THE ERROR NUMBER
013546 005767 167152 TST ERRFLG ;FIRST ERROR ?
013552 001050 BNE 1$ ;BR IF NOT FIRST ERROR
013554 032777 020000 167150 BIT 20000,2$WR ;ERROR PRINTOUTS DESIRED ?
013562 001044 BNE 1$ ;BR IN NO
013564 004767 000642 JSR PC,TPCRLF ;TYPE ERROR HEADER FOR FIRST ERROR
013570 040440 042104 020122 .ASCII / ADDR GOOD BAD PC ERR # PASFLG XOR /
013576 020040 020040 047507
013604 042117 020040 020040
013612 041040 042101 020040
013620 020040 020040 050040
013626 020103 020040 020040
013634 051105 020122 020043
013642 020040 050040 051501
013650 046106 020107 020040
013656 054040 051117 020040
013664 005015 000 .ASCIIZ <15><12>
013670 013670 .EVEN
013670 005267 167030 INC ERRFLG ;INDICATE NOT FIRST ERROR
013674 010346 1$: MOV R3,-(SP) ;SAVE R3
013676 010046 MOV R0,-(SP) ;AND R0 ON THE STACK

;SETUP BANK NO. IN FATAL FOR APT
013700 010103 MOV R1,R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
013702 004767 001420 JSR PC,GETBNK ;GET BANK NO. UNDER TEST INTO PBNK
013706 013703 000306 MOV 2$PBANK,R3 ;GET BANK UNDER TEST
013712 110337 000403 MOVB R3,2$SFATAL+1 ;STORE FAILING BANK NO. FOR APT

;
013716 010346 MOV R3,-(SP) ;TEMPORARILY STORE R3
013720 012703 000376 MOV 2$76,R3 ;MAKE R3 AS THE STACK POINTER
013724 013743 000302 MOV 2$PASFLG,-(R3) ;OUTPUT THE WORD STORED AT
013730 005043 2$: CLR -(R3)
013732 113713 000402 MOVB 2$SFATAL,(R3) ;PUT ERROR NO. ON ERROR STACK
013736 016643 000006 MOV 6(SP),-(R3) ;PLACE THE RETURN PC AT (R3)
013742 011143 MOV (R1),-(R3) ;PLACE BAD DATA
013744 010043 MOV R0,-(R3) ;AND GOOD DATA ON THE STACK
013746 005043 CLR -(R3)
013750 016313 000004 MOV 4(R3),(R3) ;TAKE THE
013754 040013 BIC R0,(R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
013756 046300 000004 BIC 4(R3),R0 ;TO FIND THE BITS THAT FAILED
013762 050013 BIS R0,(R3) ;AND PLACE IT ON THE STACK
013764 011367 000744 MOV (R3),BDCHIP ;TO PRINT OUT XOR OF GOOD VS BAD
013770 012700 002112 MOV 2$ENDPROG--24.,R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
013774 060700 ADD PC,R0 ;OF THE STARTING OF THE ERROR STACK
013776 062700 6$: ADD 2$18.,R0 ;FOR THE SPECIFIC 4K BANK
014002 005316 DEC (SP)

```

```

3483 014004 002374          BGE      6S
3484 014006 005726          TST      (SP)+      ;RESTORE THE STACK POINTER
3485
3486 014010 105037 000273      ERRYP:  CLR      2#TYPENB      ;DISABLE ANY TYPE OUT
3487 014014 105737 000274      1S:    TSTB     2#SPRERR      ;IF THIS IS PARITY PROBLEM
3488 014020 001007          BNE      3S          ;THEN GO TO 3S
3489 014022 105720          TSTB     (R0)+      ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
3490 014024 105737 000275      TSTB     2#ADERR      ;IF THIS IS ADDRESSING PROBLEM
3491 014030 001003          BNE      3S          ;THEN GO TO 3S
3492 014032 105720          TSTB     (R0)+      ;INCREMENT THE POINTER R0 BY 1
3493 014034 005713      2S:    TST      (R3)      ;IS BIT 15 OF (R3) SET?
3494 014036 100015          BPL      4S          ;IF NOT THEN GO TO 4S
3495 014040 122710 000377      3S:    CMPB     #377,(R0)      ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
3496 014044 001401          BEQ      5S          ;IF SO DON'T BUMP ERROR COUNT
3497 014046 105210          INCB     (R0)      ;INCREMENT THE ERROR COUNTER BY 1
3498 014050 122710 000001      5S:    CMPB     #1,(R0)      ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
3499 014054 001404          BEQ      7S          ;BRANCH IF NO
3500 014056 032777 000400 166646      BIT      #400,2SWR      ;STOP ERROR PRINTOUT AFTER 1 WANTED?
3501 014064 001002          BNE      4S          ;BRANCH IF YES (DON'T TYPE ERROR)
3502 014066 105237 000273      7S:    INCB     2#TYPENB      ;ENABLE THE TYPE OUT ROUTINE
3503 014072 105737 000275      4S:    TSTB     2#ADERR      ;ADDRESS ERROR?
3504 014076 001403          BEQ      6S          ;BRANCH IF NO
3505 014100 004767 000166      JSR      PC,TPADERR      ;PRINT "ADR ERR"
3506 014104 000403          BR       8S
3507 014106 105720      6S:    TSTB     (R0)+      ;POINT TO NEXT ENTRY IN ERROR STACK
3508 014110 006313          ASL     (R3)      ;IS THERE STILL AN ERROR BIT SET IN ERROR.
3509 014112 001350          BNE      2S          ;BR IF YES - KEEP FILLING ERROR STACK
3510 014114 112737 000006 000310 8S:    MOVB     #6,2#TYPCNT      ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
3511                                     ;THE STACK POINTED BY R3
3512 014122 004767 000774      JSR      PC,PUTADR      ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
3513                                     ;AT LOCATIONS (R3) AND (R3-2)
3514 014126 004767 000416      JSR      PC,TYPERR      ;TYPE ERROR STACK (7 WORDS)
3515
3516 014132 005037 000274      10S:   CLR      2#SPRERR      ;CLEAR ADDRESS ERROR FLAG
3517 014136 012600          MOV      (SP)+,R0      ;RESTORE R0
3518 014140 012603          MOV      (SP)+,R3      ;AND R3
3519 014142 105737 000420      FNDERR: TSTB     2#SENV      ;ARE WE RUNNING UNDER APT?
3520 014146 001404          BEQ      2S          ;IF NOT THEN TEST FOR HALT
3521 014150 012737 000001 000400      MOV      #1,2#MSGTY      ;OTHERWISE INFORM THE APT
3522 014156 000443          BR       FATHLT      ;GOTO FATHLT AND WAIT FOR APT.
3523
3524 014160 010246      2S:    MOV      R2,-(SP)      ;SAVE R2 TEMP
3525 014162 005777 166544      TST      2SWR          ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
3526                                     ;ON ERROR
3527 014166 100405          BMI      4S          ;IF SO THEN HALT ON ERROR
3528                                     ;CHECK FOR CONTROL-C KEY
3529
3530 014170 004767 001660      JSR      PC,CHECKC      ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
3531                                     ;AND HALT AT FATHLT.
3532 014174 105737 000042      7S:    TSTB     2#42          ;ARE WE RUNNING UNDER ACT?
3533 014200 001401          BEQ      6S          ;BRANCH IF NO
3534
3535 014202 000777      4S:    BR       4S          ;PROGRAM HAS HALTED ON ERROR, R1 IS POINTING
3536                                     ;TO A LOCATION WHICH SHOULD HAVE CONTAINED
3537                                     ;THE WORD STORED IN R0
3538 014204 012602      6S:    MOV      (SP)+,R2      ;RESTORE R2

```

```

3539 014206 062716 000002          ADD    #2,(SP)      ;RESTORE THE RETURN ADDRESS
3540 014212 000207          RTS     PC          ;RETURN FROM THE SUBROUTINE
3541
3542
3543
3544 014214          FATERR:
3545 014214 004767 000212          SEQERR: JSR    PC,TPCRLF ;TYPE "ERROR #"
3546 014220 051105 047522 020122      .ASCIZ  /ERROR #/
3547 014226 000043          .EVEN
3548
3549 014230 017637 000000 000402      MOV    2(SP),2$SFATAL ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
3550 014236 105237 000310          INCB   2$TYPCNT      ;TELL STPNUM TO TYPE 1 WORD
3551 014242 012703 000376          MOV    #376,R3      ;STPNUM USES R3 AS STACK
3552 014246 013743 000402      MOV    2$SFATAL,-(R3) ;PUT ERROR NO. ON STACK
3553 014252 005743          TST    -(R3)        ;STPNUM REQUIRES THIS
3554 014254 004767 000356          JSR    PC,FATYP     ;TYPE ERROR NO.
3555 014260 105737 000420      APTHLT: TSTB  2$ENV    ;RUNNING UNDER APT?
3556 014264 001326          BNE    FNDERR       ;BRANCH IF YES
3557 014266 000240          FATHLT: NOP
3558 014270 000776          BR     FATHLT       ;FATAL ERROR OR IC HALT.
3559                                     ;RESTART AT 250 IF DESIRED
3560 014272 105737 000273      TPADER: TSTB  2$TYPENB ;BR IF NO TYPE ENABLED
3561 014276 001406          BEQ    1$           ;PRINT MESSAGE ROUTINE
3562 014300 004767 000134          JSR    PC,PNTMES
3563 014304 042101 020122 051105      .ASCIZ  /ADR ERR/
3564 014312 000122          .EVEN
3565 014314 000207          1$:   RTS     PC
3566
3567
3568
3569
3570
3571
3572
3573
3574
3575 014316 010146          NOTYP: MOV    R1,-(SP)
3576 014320 016601 000002      4$:   MOV    2(SP),R1
3577 014324 105721          TSTB  (R1)+
3578 014326 001376          BNE    4$           ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
3579 014330 000412          BR     RETTYP       ;PREPARE TO RETURN
3580 014332 010146          $TYPE: MOV    R1,-(SP)
3581 014334 010046          MOV    R0,-(SP)
3582 014336 016601 000004      2$:   MOV    4(SP),R1
3583 014342 112100          MOVB  (R1)+,R0
3584 014344 001403          BEQ    4$           ;SAVE R1
3585 014346 004767 000022      JSR    PC,$TPCHR    ;AND R0 ON THE STACK
3586 014352 000773          BR     2$           ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
3587 014354 012600          4$:   MOV    (SP)+,R0   ;PLACE THE BYTE TO BE TYPED IN R0
3588 014356 005201          RETTYP: INC   R1      ;IF IT IS END OF MESSAGE THEN GO TO 4$
3589 014360 042701 000001      BIC   #1,R1        ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
3590 014364 010166 000002      MOV    R1,2(SP)
3591 014370 012601          MOV    (SP)+,R1
3592 014372 000416          BR     EXTYP        ;RESTORE R0
                                     ;CAUSE R1 TO
                                     ;POINT TO EVEN ADDRESS
                                     ;MODIFY THE RETURN ADDRESS
                                     ;RESTORE R1
                                     ;AND RETURN VIA RTS PC
    
```

```

;* TYPE OUT ROUTINE
;* -----
;*
;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
;*
    
```

|      |        |        |        |        |         |        |             |  |  |  |
|------|--------|--------|--------|--------|---------|--------|-------------|--|--|--|
| 3593 |        |        |        |        |         |        |             |  |  |  |
| 3594 | 014374 | 132737 | 000040 | 000421 | STPCHR: | BITB   | #40, @#SENV |  |  | ; HAVE TYPE OUTS BEEN DISABLED?          |
| 3595 | 014402 | 001005 |        |        |         | BNE    | 4S          |  |  | ; IF SO THEN RETURN FROM THE SUBROUTINE  |
| 3596 | 014404 | 105737 | 177564 |        | 2S:     | TSTB   | @#STPS      |  |  | ; WAIT HERE                              |
| 3597 | 014410 | 100375 |        |        |         | BPL    | 2S          |  |  | ; UNTIL THE PRINTER IS READY             |
| 3598 | 014412 | 110037 | 177566 |        |         | MOVB   | RO, @#STPB  |  |  | ; LOAD DATA TO BE TYPED INTO DATA REG.   |
| 3599 | 014416 | 000404 |        |        | 4S:     | BR     | EXTYP       |  |  | ; RETURN                                 |
| 3600 |        |        |        |        |         |        |             |  |  |  |
| 3601 | 014420 | 004767 | 177706 |        | PCRLF:  | JSR    | PC, STYPE   |  |  |  |
| 3602 | 014424 | 005015 | 000    |        |         | .ASCIZ | <15><12>    |  |  | ; CR/LF                                  |
| 3603 |        | 014430 |        |        |         | .EVEN  |             |  |  |  |
| 3604 | 014430 | 000207 |        |        | EXTYP:  | RTS    | PC          |  |  | ; RETURN                                 |
| 3605 |        |        |        |        |         |        |             |  |  |  |
| 3606 | 014432 | 004767 | 177762 |        | TPCRLF: | JSR    | PC, PCRLF   |  |  | ; TYPE CR/LF                             |
| 3607 | 014436 | 000735 |        |        |         | BR     | STYPE       |  |  | ; NOW GO TO TYPE THE REST OF THE MESSAGE |
| 3608 |        |        |        |        |         |        |             |  |  |  |
| 3609 |        |        |        |        |         |        |             |  |  |  |
| 3610 | 014440 | 032777 | 000020 | 166264 | PNTMES: | BIT    | #20, @#SWR  |  |  | ; PRINTOUTS ALLOWED?                     |
| 3611 | 014446 | 001323 |        |        |         | BNE    | NOTYP       |  |  | ; BRANCH IF NO                           |
| 3612 | 014450 | 123737 | 000042 | 000046 |         | CMPB   | @#42, @#46  |  |  | ; RUNNING UNDER ACT 11?                  |
| 3613 | 014456 | 001717 |        |        |         | BEQ    | NOTYP       |  |  | ; BRANCH IF YES -NOT PRINTOUT-           |
| 3614 | 014460 | 000764 |        |        |         | BR     | TPCRLF      |  |  | ; SEND CR/LF AND TYPE MESSAGE.           |



# H06

DZMML MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-71  
 DZMMLB.P11 05-AUG-77 12:57 ROUTINE TO TYPE OUT A DECIMAL NUMBER

SEQ 0073

3619  
3620  
3621  
3622  
3623  
3624  
3625  
3626  
3627  
3628  
3629  
3630  
3631  
3632  
3633  
3634  
3635  
3636  
3637  
3638  
3639  
3640  
3641  
3642  
3643  
3644  
3645  
3646

```

    * 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
    *
    014462 004767 177732      TYPDEC: JSR    PC,PCRLF      ;TYPE CR/LF
    014466 005046             STPDEC: CLR    -(SP)
    014470 013746 000306      MOV    2#DECWRD,-(SP) ;GET THE WORD THAT HAS TO BE CONVERTED TO A
    ;DECIMAL NUMBER
    014474 162716 000012      2$: SUB    #10.,(SP)
    014500 002403             BLT    4$                ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
    ;GO TO 4$
    014502 005266 000002      INC    2(SP)           ;OTHERWISE ADD 1 TO THE LOCATION STORING 10'S DIGIT
    014506 000772             BR     2$                ;AND RETURN TO 2$
    014510 062716 000012      4$: ADD    #10.,(SP)
    014514 052716 000060      BIS    #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
    014520 112667 000020      MOVB   (SP)+,6$-2      ;PLACE THE 1'S DIGIT TO BE TYPED
    014524 052716 000060      BIS    #60,(SP)        ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
    014530 112667 000007      MOVB   (SP)+,6$-3      ;PLACE THE 10'S DIGIT TO BE TYPED
    014534 004767 177572      JSR    PC,$TYPE       ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
    ;3 SPACES
    014540 020040 030040 000060 .ASCIZ  / 00/
    014546 000207             6$: RTS    PC          ;RETURN FROM THE SUBROUTINE
  
```



J06

DZMPL MACY11 30(1046) 05-AUG-77 12:59 PAGE 1-73  
DZMPLB.P11 05-AUG-77 12:57 OCTAL TYPE OUT ROUTINE

SEQ 0075

3707 014726 005743  
3708 014730 000736  
3709 014732 000207  
3710  
3711 014734 000000

TST -(R3)  
BR RPTOCT  
OCTXT: RTS PC  
BDCHIP: .WORD 0

;ADJUST POINTER  
;PRINT LAST 6 OCTAL DIGITS

;XOR OF GOOD VS BAD DATA

SUBROUTINE FOR MEMORY MANAGEMENT

```

3716
3717
3718
3719
3720
3721
3722
3723
3724
3725 014736 012702 001400 MEMMNG: MOV #1400,R2
3726 014742 105037 000272 MMREG: CLRB @#MMAVA ; CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
3727 ; THAT MEM. MANAG. IS AVAILABLE FOR TESTING
3728 014746 032777 010000 165756 BIT #10000,@SWR ; HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
3729 014754 001043 BNE RETMM ; IF NOT THEN RETURN FROM THE SUBROUTINE
3730 014756 012700 000004 MOV #4,R0 ; PREPARE TO SETUP TIME OUT VECTOR
3731 014762 012720 015066 MOV #NOMM,(R0)+ ; RETURN ADDRESS TO NOMM
3732 014766 012710 000340 MOV #340,(R0) ; AND WITH A PSW OF 340
3733 014772 005037 177572 CLR @#SRO ; TRY TO REACH MEM. MANAG. SRO
3734 014776 105237 000272 INCB @#MMAVA ; IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
3735 ; BYTE
3736 015002 012701 172340 MOV #172340,R1 ; R1 IS POINTING TO PAR0
3737 015006 005021 CLR (R1)+ ; PAR0 WILL POINT TO BANK 0
3738 015010 012721 000200 MOV #200,(R1)+ ; SET UP PAR 1
3739 015014 062702 000200 2S: ADD #200,R2
3740 015020 010221 MOV R2,(R1)+ ; SETUP PAR1-PAR6
3741 015022 020127 172356 CMP R1,#172356 ; ADDRESS OF PAR7
3742 015026 103772 BLO 2S ; PAR7 IS POINTING TO THE I/O PAGE
3743 015030 012711 007600 MOV #7600,(R1)
3744 015034 012701 172300 MOV #172300,R1
3745 015040 012721 077406 4S: MOV #77406,(R1)+ ; SETUP PDR0-PDR7
3746 015044 020127 172316 CMP R1,#172316
3747 015050 101773 BLOS 4S
3748 015052 005237 177572 INC @#SRO ; ENABLE MEM. MANAG.
3749 015056 005010 SRETMM: CLR (R0) ; RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
3750 015060 012740 000352 MOV #BUSER,-(R0)
3751 015064 000207 RETMM: RTS PC
3752
3753 015066 022626 NOMM: CMP (SP)+,(SP)+ ; RESTORE STACK POINTER
3754 015070 004767 177336 JSR PC,TPCRLF ; TYPE "NO MEMORY MANAGEMENT MESSAGE"
3755 015074 047516 046440 043516 .ASCIZ /NO MNG/
3756 015102 000 .EVEN
3757 015104 004767 177104 JSR PC,FATERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 015110 000104 104 ; *****ERROR NUMBER 104*****
(1)
3758 015112 000761 BR SRETMM ; RESTORE TIME OUT TRAP VECTOR
3759
3760 015114 013702 172354 UPMM: MOV @#172354,R2 ; PREPARE TO UPDATE MEMORY MANAG. REGISTERS
3761 015120 000710 BR MMREG
    
```

```

3766
3767
3768
3769
3770
3771
3772
3773
3774
3775
3776
3777 015122 005063 177776 PUTADR: CLR -2(R3)
3778 015126 010113 MOV R1,(R3) ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
3779 015130 105737 000272 TSTB @#MMAVA ;IS THE MEM. MANAG. AVAILABLE?
3780 015134 001425 BEQ 6$ ;IF NOT THEN RETURN FROM THE SUBROUTINE
3781 015136 010146 MOV R1,-(SP) ;SAVE R1
3782 015140 042701 017777 BIC #17777,R1 ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
3783 015144 040113 BIC R1,(R3) ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
3784 015146 052701 004000 BIS #4000,R1 ;PREPARE TO SHIFT R1 BY 12 PLACES
3785 015152 006001 2$: ROR R1
3786 015154 103376 BCC 2$ ;GET THE NUMBER OF PAR IN R1
3787 015156 062701 172340 ADD #172340,R1 ;GET THE ADDRESS OF PAR IN R1
3788 015162 011101 MOV (R1),R1 ;LOAD R1 WITH THE CONTENTS OF PAR
3789 015164 052701 010000 BIS #10000,R1
3790 015170 006101 4$: ROL R1
3791 015172 103376 BCC 4$ ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
3792 015174 006301 ASL R1 ;SO WE DON'T PICK UP C BIT
3793 015176 006143 ROL -(R3) ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
3794 015200 006101 ROL R1
3795 015202 006123 ROL (R3)+ ;PLACE BIT 16 OF THE ADDRESS
3796 015204 050113 BIS R1,(R3) ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
3797 015206 012601 MOV (SP)+,R1 ;RESTORE R1
3798 015210 000207 6$: RTS PC ;RETURN FROM THE SUBROUTINE

;* GET ADDRESS FROM THE APT MAILBOX
;* -----
;*
;* THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
;* PLACE IT IN THE LOCATION USED BY THE PROGRAM TO DEFINE THE
;* MEMORY BOUNDARIES.
;* PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
;* ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
;* THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
;* TO BE PLACED
;*
3815 015212 016143 000001 GETADR: MOV 1(R1),-(R3) ;PLACE THE LOW ORDER BITS OF THE ADDRESS
3816 015216 005043 CLR -(R3) ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
3817 ;HAVE TO BE PLACED
3818 015220 116113 177777 2$: MOVB -1(R1),(R3) ;PLACE BITS 16 & 17
3819 015224 000207 RTS PC ;RETURN FROM THE SUBROUTINE
    
```

```

3824
3825
3826
3827
3828
3829
3830
3831
3832
3833
3834 015226 105237 000311          $GTSIZ: INCB      2#SAVKBB      ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3835                                     ;0-4 OF R2
3836
3837 015232 012301          GETSIZ: MOV      (R3)+,R1
3838 015234 011302          MOV      (R3),R2      ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3839 015236 042702 017777      BIC      #17777,R2    ;CLEAR ADDRESS BITS 0-12
3840 015242 052702 000040      2$:     BIS      #40,R2
3841 015246 006001          4$:     ROR      R1
3842 015250 006002          ROR      R2      ;ROTATE R1 AND R2 7 TIMES
3843 015252 103375          BCC      4$
3844 015254 105737 000311      TSTB    2#SAVKBB
3845 015260 001405          BEQ      6$
3846 015262 105037 000311      CLRB    2#SAVKBB
3847 015266 052702 000100      BIS      #100,R2
3848 015272 000765          BR       4$
3849 015274 012301          6$:     MOV      (R3)+,R1      ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3850 015276 012700 160000      MOV      #160000,R0
3851 015302 040001          BIC      R0,R1      ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3852 015304 000207          RTS      PC          ;RETURN FROM THE SUBRORNE
3853
3854
3858
3859
3860
3861
3862
3863
3864
3865 015306 105737 000272          CLRMM: TSTB    2#MMAVA      ;WAS THE MEMORY MANAGEMENT ENABLED ?
3866 015312 001404          BEQ      1$      ;IF NOT THEN GO TO 1$
3867 015314 005037 177572          CLR      2#SRO      ;DISABLE THE MEMORY MANAGEMENT
3868 015320 105037 000272          CLRB    2#MMAVA      ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
3869 015324 000207          1$:     RTS      PC          ;RETURN FROM THE SUBROUTINE
3870
3871
3872
3873
3874
3875
3876
3877
3878
3879 015326 010046          ;* GET BANK NO. UNDER TEST
3880 015330 010346          ;CALLED BY ERRYP TO GET BANK NO. UNDER TEST INTO PBNK.
3881 015332 042703 017777      ;REGISTERS
3882 015336 052703 010000      ;RO=POINTER TO PAR UNDER TEST
3883                                     ;R3=VIRTUAL ADDRESS ON ENTRY
3884                                     ;RO+R3 ARE RESTORED ON EXIT.
3885
3886
3887
3888
3889
3890
3891
3892
3893
3894
3895
3896
3897
3898
3899
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909
3910
3911
3912
3913
3914
3915
3916
3917
3918
3919
3920
3921
3922
3923
3924
3925
3926
3927
3928
3929
3930
3931
3932
3933
3934
3935
3936
3937
3938
3939
3940
3941
3942
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952
3953
3954
3955
3956
3957
3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000

```

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

```

3883 015342 000241
3884 015344 006003
3885 015346 103376
3886 015350 105737 000272
3887 015354 001407
3888
3889
3890 015356 006303
3891 015360 062703 172340
3892 015364 011300
3893 015366 006300
3894 015370 000300
3895 015372 110003
3896 015374 010337 000306
3897 015400 012603
3898 015402 012600
3899 015404 000207
3900
3901
3902
3903
3904
3905
3906
3907
3908
3909 015406 010146
3910 015410 042701 003777
3911 015414 052701 002000
3912 015420 000241
3913 015422 006001
3914 015424 103376
3915 015426 010100
3916 015430 105737 000272
3917 015434 001416
3918
3919 015436 010146
3920 015440 006201
3921 015442 042701 000001
3922 015446 062701 172340
3923 015452 011100
3924 015454 006300
3925 015456 006300
3926 015460 006300
3927 015462 000300
3928 015464 042716 000034
3929 015470 052600
3930 015472 012601
3931 015474 000207
3932
3933
3934
3935
3936
3937
3938

1S:   CLC
      ROR    R3
      BCC    1S
      TSTB  @#MMAVA
      BEQ    2S

;SHIFT A BANK BIT
;UNTIL IN BITS <2:0> OF R3
;MEMORY MANAGEMENT UNDER TEST?
;NO EXIT

;GET PAR ADDRESS AND PHYSICAL BANK NO.
      ASL   R3
      ADD   #172340,R3
      MOV   (R3),RO
      ASL   RO
      SWAB  RO
      MOVB  RO,R3
2S:   MOV   R3,@#PBANK
      MOV   (SP)+,R3
      MOV   (SP)+,RO
      RTS   PC

;MAKE R3 PAR ADDRESS OFFSET.
;MAKE FULL PAR ADDRESS.
;GET PAR CONTENTS
;SHIFT BANK BITS TO BITS <7:0>
;SET R3 TO PHYSICAL BANK NO.
;STORE PHYSICAL BANK NO.
;RESTORE R3
;RESTORE RO
;RETURN TO CALLER

;GET THE 1K BANK UNDER TEST
;CALLED BY TEST 7
;REGISTERS
;R1 = VIRTUAL ADDRESS ON ENTRY
;RO = POINTER TO 1K UNDER TEST ON EXIT
;R1 WILL BE RESTORED ON EXIT

GET1K: MOV   R1,-(SP)
      BIC   #3777,R1
      BIS   #2000,R1
1S:   ROR   R1
      BCC   1S
      MOV   R1,RO
      TSTB  @#MMAVA
      BEQ   2S

;SAVE R1
;ONLY WANT 1K BIT COUNT
;SET UP SHIFT BIT
;CLEAR CARRY BIT
;ROTATE THE BANK
;UNTIL THEY'RE IN BITS 4:0
;IN CASE NO MEM MGNT
;MEM MGNT AVAILABLE ?
;BR IF NO

;GET PAR AND PHYSICAL BANK
      MOV   R1,-(SP)
      ASR   R1
      BIC   #1,R1
      ADD   #172340,R1
      MOV   (R1),RO
      ASL   RO
      ASL   RO
      ASL   RO
      SWAB  RO
      BIC   #34,(SP)
      BIS   (SP)+,RO
2S:   MOV   (SP)+,R1
      RTS   PC

;SAVE 1K BANK
;MAKE R1 A PAR ADDRESS
;PAR ADDRESSES ARE 4K BOUNDARIES
;MAKE IT A FULL PAR ADDR.
;GET THE PAR CONTENTS
;SHIFT RO
;ONLY WANT THE 2 LSB'S
;SET THE IN PAR
;RESTORE R1
;AND EXIT

;THIS ROUTINE CHECKS THE ADDRESS IN R1
;TO MAKE SURE IT IS ERROR FREE, INCLUDING
;THE CHECKBITS.
;R3 = 0 IF NO ERROR
;R3 = NONZERO IF ERROR IS FOUND
    
```

```

3939
3940 015476 016777 165206 165224 TSTADD: MOV      ECCDIS, @CSRADR ;DISABLE ECC
3941 015504 005011          CLR      (R1) ;CLEAR A LOC
3942 015506 005061 000002          CLR      2(R1) ;UPPER WORD TOO
3943 015512 005711          TST      (R1) ;READ ZEROS
3944 015514 001047          BNE     1$ ;BR IF ERROR
3945 015516 005761 000002          TST      2(R1) ;SHOULD BE ZEROS ALSO
3946 015522 001044          BNE     1$ ;
3947 015524 005111          COM      (R1) ;COMPLEMENT OF 0 = 177777
3948 015526 005161 000002          COM      2(R1) ;BOTH WORDS
3949 015532 022711 177777          CMP      @-1, (R1) ;READ ALL ONES
3950 015536 001036          BNE     1$ ;BR IF NO
3951 015540 022761 177777 000002          CMP      @-1, 2(R1) ;READ AGAIN
3952 015546 001032          BNE     1$ ;BR IF ERROR
3953 015550 016703 165154          MOV      CSRADR, R3 ;GET ADDRESS OF CSR
3954 015554 016713 165132          MOV      DIAGA, (R3) ;SET DIAGNOSTIC BIT
3955 015560 052713 000002          BIS      @2, (R3) ;AND ECCDIS BIT
3956 015564 012711 000000          MOV      @0, (R1) ;TO WRITE ALL ZERO CHECKBITS
3957 015570 005761 000002          TST      2(R1) ;READ AND RECORD CHECKBITS
3958 015574 032713 007740          BIT      @7740, (R3) ;NO CHECKBIT SHOULD BE SET
3959 015600 001015          BNE     1$ ;BR IF SET
3960 015602 052713 007740          BIS      @7740, (R3) ;MAKE CHECKBITS ALL ONES
3961 015606 012711 177777          MOV      @-1, (R1) ;WRITE THEM LIKE THIS
3962 015612 042713 007740          BIC      @7740, (R3) ;SO WE KNOW IF WE READ THEM BACK
3963 015616 005761 000002          TST      2(R1) ;READ THEM
3964 015622 042713 170037          BIC      @170037, (R3) ;ONLY WANT CHECKBITS
3965 015626 022713 007740          CMP      @7740, (R3) ;SHOULD BE ALL ONES
3966 015632 001403          BEQ     2$ ;BR IF OK
3967 015634 012703 000001          1$: MOV      @1, R3 ;INDICATE ERROR FOUND
3968 015640 000401          BR      3$ ;AND EXIT
3969 015642 005003          2$: CLR      R3 ;CLEARED = GOOD LOC
3970 015644 005077 165060          3$: CLR      @CSRADR ;CLEAR CSR BEFORE LEAVING
3971 015650 000207          RTS     PC ;AND EXIT
    
```

```

;THIS ROUTINE IS USED TO COUNT THE BAD CHECKBITS
;FROM TEST 15. IT USES THE ROUTINE 'XOR' AND THEN
;COUNTS THE BAD ONES
    
```

```

3972
3973
3974
3975
3976
3977
3978 015652 005067 000056          BITCHK: CLR      BITCNT ;WANT TO START CLEAN
3979 015656 010146          MOV      R1, -(SP) ;SAVE ADDRESS
3980 015660 017701 165044          MOV      @CSRADR, R1 ;GET CONTENTS OF CSR
3981 015664 042701 170037          BIC      @170037, R1 ;ONLY WANT CHECK BITS
3982 015670 010146          MOV      R1, -(SP) ;SAVE THIS TOO
3983 015672 004767 175630          JSR      PC, XOR ;R0 XOR R1 = R2
3984 015676 012701 000020          MOV      @20, R1 ;GETTING READY TO COUNT
3985 015702 006301          1$: ASL      R1 ;SHIFT TO NEXT POSITION
3986 015704 020127 010000          CMP      R1, @10000 ;FINISHED YET ?
3987 015710 001405          BEQ     2$ ;BR IF DONE
3988 015712 030102          BIT      R1, R2 ;SEE IF THIS BIT (R1) IS BAD
3989 015714 001772          BEQ     1$ ;BR IF NOT BAD
3990 015716 005267 000012          INC      BITCNT ;IT IS, COUNT IT
3991 015722 000767          BR      1$ ;NOT DONE YET, KEEP GOING
3992 015724 012600          2$: MOV      (SP)+, R0 ;FIRST GET DATA
3993 015726 012601          MOV      (SP)+, R1 ;THEN ADDRESS
3994 015730 010011          MOV      R0, (R1) ;PUT BAD DATA AWAY FOR PRINTOUT
    
```



## SUBROUTINE TO DISABLE MEMORY MANAGEMENT

```

3995 015732 000207          RTS    PC          ;THEN EXIT
3996
3997 015734 000000          BITCNT: .WORD 0          ;# OF BAD CHECKBITS FOUND ABOVE
3998
3999
4000          ;THIS ROUTINE PRINTS THE TITLE AND MESSAGE
4001
4002 015736 004767 176470          PRTITL: JSR    PC,TPCRLF          ;PRINT ROUTINE
4003 015742 055104 046515 026514          .ASCII /DZMML-A/
4004 015751 0040          .ASCII / USE TO TEST MF11S-K ECC MEMORY/
015756 047524 052040 051505
015764 020124 043115 030461
015772 026523 020113 041505
016000 020103 042515 047515
016006 054522
4005 016010 005015 054524 042520          .ASCIZ <15><12>/TYPE <CONTROL C> TO EXIT/
016016 036040 047503 052116
016024 047522 020114 037103
016032 052040 020117 054105
016040 052111 000
4006          .EVEN
4007 016044 012767 172136 164656          MOV    #172136,CSRADR          ;ORIGINAL CSR ADDRESS
4008 016052 000207          RTS    PC          ;RETURN TO START ROUTINE
4009
4010
4011
4012
4013
4014          ;CHECKC
4015          ; THIS ROUTINE CHECKS IF CONTROL-C WAS TYPED AT THE END OF EACH
4016          ; TEST OR IN THE ERROR TYPE ROUTINE.
4017          ; IF CONTROL-C TYPED THE PROGRAM IS RETURNED TO LOWER MEMORY IF IT WAS
4018          ; RELOCATED AND THE ERROR HISTORY IS PRINTED OUT.
4019          ; FINALLY IT HALTS AT FATHLT.
4020
4021 016054 105037 000311          CHECKC: CLRB   @#SAVKBB          ; INIT CONTROL-C FLAG.
4022 016060 105737 177560          TSTB   @#TKS          ; ANY CHAR. TYPED?
4023 016064 100021          BPL    EXITC          ; BR IF NO-EXIT VIA RTS PC-
4024 016066 113702 177562          MOVB   @#SKBB,R2          ; GET THE CHAR TYPED.
4025 016072 042702 000200          BIC    #200,R2          ; CLEAR THE PARITY BIT.
4026 016076 122702 000003          CMPB   #3,R2          ; IS IT CONTROL-C?
4027 016102 001012          BNE    EXITC          ; BRANCH IF NO -EXIT VIA RTS PC-
4028 016104 110237 000311          MOVB   R2,@#SAVKBB          ; ELSE STORE THE CHAR. FOR USE AS A FLAG.
4029 016110 004767 176316          JSR    PC,TPCRLF          ; PRINT "tC"
4030 016114 041536 000          .ASCIZ /tC/
4031          .EVEN
4032 016120 005067 164600          CLR    ERRFLG          ; CLEAR ERROR HEADER FLAG FOR NEXT START
4033 016124 000167 173752          JMP    RELOER          ; GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
4034 016130 000207          EXITC: RTS    PC
4035
4036 016132 000000          ENDPROG: 0          ; THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
4037          ; STACK FOR EACH 4K BANK 18. BYTES ARE SAVED.
4038          ; ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
4039          ; AFTER THE ERROR STACK.
4040          .END
000001

```

|                 |               |               |                |                |
|-----------------|---------------|---------------|----------------|----------------|
| ABASE = 000000  | BDCHIP 014734 | EXITC 016130  | RLODER 012760  | TST3 003544    |
| ACDW1 = 000000  | BEGIN 000500  | EXTYP 014430  | RPTOCT 014626  | TST4 003776    |
| ACDW2 = 000000  | BITCHK 015652 | FAILNM 012614 | RPT17 011522   | TST5 004152    |
| ACPUOP = 000000 | BITCNT 015734 | FATERR 014214 | RSTOT4 013340  | TST6 004622    |
| ACT11 012734    | BITCOM 013404 | FATHLT 014266 | SAVKBB 000311  | TST7 005542    |
| ADDMD = 000000  | BRTPSZ 001276 | FATYP 014636  | SAVLDR 002034  | TYPCNT 000310  |
| ADDW1 = 000000  | BUSER 000352  | FNDERR 014142 | SAVLOC 000350  | TYPDEC 014462  |
| ADDW10 = 000000 | CHECK 013014  | GETADR 015212 | SAVMAX 000340  | TYPENB 000273  |
| ADDW11 = 000000 | CHECKC 016054 | GETBNK 015326 | SAVMIN 002722  | TYPEOP 012702  |
| ADDW12 = 000000 | CHKCNT 013053 | GETSIZ 015232 | SAVR4 000342   | TYPERR 014550  |
| ADDW13 = 000000 | CHKGEN 013064 | GETIK 015406  | SAVR5 000344   | TYPMEM 001742  |
| ADDW14 = 000000 | CHKTAB 013016 | HIGHAD 000330 | SAVR6 000346   | TYP OCT 014602 |
| ADDW15 = 000000 | CKDONE 012174 | HIGHTW 000326 | SAVOT4 013326  | TYPSIZ 001650  |
| ADDW2 = 000000  | CLMEM 002232  | INHREL 002726 | SAV7 006532    | TYPSTK 012540  |
| ADDW3 = 000000  | CLMM 015306   | LDFLG 002716  | SBEMSK 002674  | T12A 007640    |
| ADDW4 = 000000  | CNTSCP 002624 | LOOP 002460   | SCOPE = 000240 | T12B 007662    |
| ADDW5 = 000000  | CONT 002422   | LOWADD 000324 | SEGERR 014214  | T5A 004174     |
| ADDW6 = 000000  | CONTHM 012242 | LOWBNK 000300 | SETSTK 001754  | T6A 004724     |
| ADDW7 = 000000  | CSRADR 002730 | LOWER 012200  | SETSWR 001126  | T6B 004736     |
| ADDW8 = 000000  | CTLC 013002   | LONTWO 000322 | SLFSIZ 001310  | T6FLG 005540   |
| ADDW9 = 000000  | DASHL 013354  | M = 000200    | SOURCE 013012  | UPMM 015114    |
| ADEVCT = 000000 | DATADR 013524 | MAXADR 012346 | SRO = 177572   | UPPFLG 013052  |
| ADEVN = 000000  | DATBUF 002664 | MAXMEM 000336 | START 000200   | WRTMEM 000120  |
| AREN = 000000   | DATSAV 002704 | MEMING 014736 | STRTDI 000276  | XOR 013526     |
| ARENH = 000000  | DBEMSK 002700 | MENTST 002224 | SWAPAT 000314  | SA = 000001    |
| AFATAL = 000000 | DECHRD 000306 | MINMEM 000320 | SWHALT 002644  | SADERR 000275  |
| AMADR1 = 000000 | DIAGA 002712  | MMAVA 000272  | SWR 002732     | SAPTHD 000272  |
| AMADR2 = 000000 | DISPLA 002734 | MREG 014742   | SWREG 000176   | SCNTMM 012256  |
| AMADR3 = 000000 | DSPREG 000174 | MSKADR 013522 | SW11 = 004000  | SCPUOP 000426  |
| AMADR4 = 000000 | ECCDIS 002710 | MSYES 002714  | TBL 002544     | SDEVCT 000410  |
| AMAMS1 = 000000 | ENDMAX 012422 | N = 000105    | TKS = 177560   | SDOAGN 012754  |
| AMAMS2 = 000000 | ENDPAS 012430 | NOMM 015066   | TPADR 014272   | SENDAD 000156  |
| AMAMS3 = 000000 | ENDPRO 016132 | NOTYP 014316  | TPCRLF 014432  | SENV 000420    |
| AMAMS4 = 000000 | ENDSTK 000304 | OCTTYP 014572 | TRYSR 001300   | SENVH 000421   |
| AMSGAO = 000000 | END0 003070   | OCTXT 014732  | TSTADD 015476  | SEOP 012652    |
| AMSGLG = 000000 | END1 003360   | ONEPAS 001030 | TSTDAT 002670  | SETABL 000420  |
| AMSGTY = 000000 | END10 007160  | PARBYT 013062 | TSTGO 002646   | SETEND 000450  |
| AMTYP1 = 000000 | END11 007612  | PASFLG 000302 | TSTMH 012226   | SFATAL 000402  |
| AMTYP2 = 000000 | END12 010252  | PBNK 000306   | TSTREL 002104  | SGTSIZ 015226  |
| AMTYP3 = 000000 | END13 010364  | PCRLF 014420  | TSTRP 001044   | SHD = 000002   |
| AMTYP4 = 000000 | END14 010476  | PNTMES 014440 | TSTSCP 002606  | SHIBTS 000272  |
| APASS = 000000  | END15 011330  | PRTITL 015736 | TSTSIZ 002110  | SHIMAX 000332  |
| APRIOR = 000000 | END16 011500  | PUTADR 015122 | TST0 002736    | SKBB = 177562  |
| APTHLT 014260   | END17 011636  | PWRDN 000044  | TST1 003072    | SMADR1 000432  |
| APTSIZ 001210   | END2 003540   | PWRUP 000136  | TST10 006536   | SMADR2 000436  |
| ASWREG = 000000 | END3 003772   | REL 000170    | TST11 007164   | SMADR3 000442  |
| ATESTN = 000000 | END4 004146   | RELBOT 000316 | TST12 007616   | SMADR4 000446  |
| AUNIT = 000000  | END5 004616   | RELOC 011642  | TST13 010256   | SMAIL 000400   |
| AUSWR = 000000  | END6 005534   | RELOER 012102 | TST14 010370   | SMAMS1 000430  |
| AVECT1 = 000000 | END7 006526   | RESTRT 000250 | TST15 010502   | SMAMS2 000434  |
| AVECT2 = 000000 | ERRFLG 002724 | RETMH 015064  | TST16 011334   | SMAMS3 000440  |
| BAKPAT 000312   | ERROR 013540  | RETSTK 012560 | TST17 011504   | SMAMS4 000444  |
| BASE 002720     | ERRTYP 014010 | RETTYP 014356 | TST2 003364    | SMAXM 000334   |

E07

DZMLB MACY11 30(1046) 05-AUG-77 12:59 PAGE 2-1  
DZMLB.P11 05-AUG-77 12:57 SYMBOL TABLE

SEQ 0083

|        |        |        |        |        |        |        |        |        |     |   |        |
|--------|--------|--------|--------|--------|--------|--------|--------|--------|-----|---|--------|
| SMBADR | 000274 | SMTYP4 | 000445 | SSWREG | 000422 | STPS   | =      | 177564 | SZ  | = | 000362 |
| SMSGAD | 000414 | SPASS  | 000406 | STESTN | 000404 | STPSTK |        | 012530 | .   | = | 016134 |
| SMSGLG | 000416 | SPASTM | 000300 | STN    | =      | 000000 | STSTM  | 000276 | .SX | = | 000272 |
| SMSGTY | 000400 | SPRERR | 000274 | STPB   | =      | 177566 | STYPE  | 014332 |     |   |        |
| SMTYP1 | 000431 | SRETMM | 015056 | STPCHR |        | 014374 | SUNIT  | 000412 |     |   |        |
| SMTYP2 | 000435 | SSVPC  | =      | 000044 | STPDEC | 014466 | SUNITM | 000302 |     |   |        |
| SMTYP3 | 000441 | SSWR   | =      | 000000 | STPNUM | 014640 | SUSWR  | 000424 |     |   |        |

. ABS. 016134 000

ERRORS DETECTED: 0

DZMLB, DZMLB/DOC=DZMLB  
RUN-TIME: 8 9 .3 SECONDS  
RUN-TIME RATIO: 151/17=8.4  
CORE USED: 12K (23 PAGES)

DOCUMENT PAGES: 83