



801

EOF1DZDL CBSEQ 00010000 770720 PDP10 411 SHOR1DZMMLASEQ 00010000 770720  
PDP10 411SEQ 0002 DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1  
DZMMLA.P11 23-JUN-77 10:17

.REPT 0

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZMML-A-D  
PRODUCT NAME: MF11S-K MEMORY DIAGNOSTIC  
DATE CREATED: MAY 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 OF 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
[1.0]	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-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 LOFLG MUST = 1 IF THE PROGRAM  
IS RESIDING IN THE MS11K MEMORY.

BIT05(000040) INHIBIT "END PASS #XX" AND "RELOC" PRINTOUTS

BIT04-BIT00 NUMBER OF TEST (0-21) 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 XXXXX" THE LAST PART OF  
TEST 7 CHECKS FOR PROPER 1K ADDRESSES IN  
THE CSR AFTER A DOUBLE ERROR. THIS MES-  
SAGE 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 --	REASON -----	RECOVERY -----
112	TRAP TO LOC. 4	EXAMINE R6. IT CONTAINS THE POINTER TO THE PC WHERE THE TRAP OCCURRED. (SEE NOTE) 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.

### 16.1] 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 \$MCGTY 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.
;               ;RO = GOOD DATA
;               ;RI = 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     ;[TSTO] 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
;STEST (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
;STESTN (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
;STESTN (404) SHOULD = 03
; THE DIAGNOSTIC HAS BEEN CORRUPTED.

;ERROR # 26 ;[TST3] BYTE ADDRESSING TEST
;COMPARE GOOD VS. BAD
;ADDRESSING PROBLEM

```

K01

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-9  
DZMMLA.P11 23-JUN-77 10:17

SEQ 0011

;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 ENABLED, 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] TEST 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
      ;$TESTN SHOULD = 13
      ;DIAGNOSTIC HAS BEEN CORRUPTED
;ERROR # 64  ;[TST13] DUAL ADDRESSING ERROR
      ;IF PASFLG = 0 THEN FAILING LOCATION
      ;AND FAILING DATA ARE DUAL ADDRESS
;ERROR # 65  ;[TST14] TEST SEQUENCE ERROR
      ;$TESTN 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
      ;$TESTN SHOULD = 15
      ;DIACNOSTIC HAS BEEN CORRUPTED
;ERROR # 70  ;[TST15] CHECK BIT WRITE ERROR
      ;CHECK BITS SHOULD BE 4000
      ;FROM WRITE OPERATION
;ERROR # 71  ;[TST15] CHECK BIT DATA ERROR
      ;MARCHING MIN TO MAX
      ;CHECK BITS SHOULD BE 3740
;ERROR # 72  ;[TST15] CHECKBIT DATA ERROR
      ;MARCHING MIN TO MAX
      ;CHECKBITS SHOULD BE 4000
;ERROR # 73  ;[TST15] CHECK BIT DATA ERROR
      ;MARCHING MAX TO MIN
      ;CHECKBITS SHOULD BE 3740
;ERROR # 74  ;[TST15] CHECKBIT DATA ERROR
      ;MARCHING MIN TO MAX
      ;CHECKBITS SHOULD BE 4000
;ERROR # 75  ;[TST16] TEST SEQUENCE ERROR
      ;$TESTN 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 ;[RFLOC] 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) MF11S-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:

1. THE PAR (PAGE ADDRESS REGISTER) CONTENTS ARE SHOWN IN A TEST THAT SELF SIZES. IF THE LIMITS OF TESTING ARE SET BY THE OPERATOR AND IF THE BANK IS ABOVE 28K PAR NO. 1 WILL BE SET TO THE BEGINNING PAGE. FOR EXAMPLE IF THE TESTING WAS TO BEGIN WITH BANK 8 PAR NO. 3 WOULD EQUAL 2000, PAR 4 WOULD EQUAL 2200 ETC.

## [B.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.

## [B.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).

## [B.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. [TSTMM] 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 & ACT11 & 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 (\$ENDAD) CONTROL IS RETURNED TO THE ACT11 MONITOR VIA LOCATION 42.

APT  
---

OPERATION IS SIMILAR TO STAND ALONE EXCEPT:

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

#### APT MANAGER INFORMATION

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

THE EXAMPLE ASSUMES YOU ARE LOGGED INTO THE APT MONITOR.

READY

RUN APPLU  
APT 11 PAPER TAPE PROGRAM LOAD UTILITY

THE FOLLOWING COMMANDS ARE VALID

ED	EDIT A PROGRAM
LI	LIST A PROGRAM

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

MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 2 <000>:  
MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 3 <000>:  
MAXIMUM ADDRESS<00000000>:  
MEMORY TYPE 4 <000>: 1  
MAXIMUM ADDRESS<00000000>: 17776  
WHICH ETABLE DO YOU WISH TO EDIT?  
COMMAND: OFF

```

.ENOR
.ABS
.NLIST MD,MC,CND

1159          .LIST ME,BIN,SEQ,LOC
1164          .TITLE DZMML
(1)          ;*COPYRIGHT (C) FEBRUARY 1977
(1)          ;*DIGITAL EQUIPMENT CORP.
(1)          ;*MAYNARD, MASS. 01754
(1)          ;*
(1)          ;*PROGRAM BY JIM RYAN
(1)          ;*
(1)          ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
(1)          ;*PACKAGE (MAINDEC-11-DZGAC-C3), JAN 19, 1977.
(1)          ;*
(1)          160000      $SWR=160000      ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT

1165
1166
1167
1168
1200          ;;TRAP CATCHER OF .+2 AND HALT FOR 0-776 LOCATIONS
1201
1209
1210
1217
1218          000240      SCOPE =NOP
1219
1220          000042      .=42
1221 000042 000000      .WORD 0      ;FOR ACT/XXDP
1222
1223          .SBTTL ACT11 HOOKS
(1)          ;*****
(2)          ;HOOKS REQUIRED BY ACT11
(1)          000044      $SVPC=.      ;SAVE PC
(1)          000046      .=46
(1) 000046 000156      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
(1)          000052      .=52
(1) 000052 040000      .WORD 40000      ;;2)SET LOC.52 TO 40000
(1)          000044      .=52      ;; RESTORE PC
1224
1225          000070      .=70
1226 000070 012737 000136 000024 PWRDN: MOV #PWRUP,0#24
1227 000076 000000      HALT
1228

```

02MML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-21  
02MMLA.P11 23-JUN-77 10:17 ACT11 HOOKS

1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1243  
1244  
1245  
1246  
1247  
1248  
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

000104  
000104 012737 000001 000400  
000112 000774  
000120  
000120 010401 000312  
000122 013700 000312  
000126 010021  
000130 020105  
000132 103775  
000134 000207  
000136 013706 000346  
000142 012700 013702  
000146 060600  
000150 004710  
000152 000120  
000154 000411  
000156 004710  
000160 000240  
000162 000240  
000164 000240  
000166 000430  
000174 000174  
000176 000000  
000176 000000  
000200 005077 002466

```

.=104
: GET HERE IF AN ILLEGAL TRAP TO LOC 4 OCCURRED
BUSER: MOV @1, @MSGTY ; TELL APT FATAL ERROR #000
        BR BUSER ; LOOP SO APT CAN GET INFO
; 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
; ; TYPE ROUTINE AND TYPE CR, LF & A "P".
        JSR PC, (R0)
        .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

.=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
```

```

1289 000204 013706 000346      MOV      @SAVR6,SP      ;SETUP STACK POINTER
1290 000210 012703 000412      MOV      @SUNIT,R3    ;CLEAR THE APT MAILBOX FROM $MAIL TO $DEVCT
1291 000214 005043          CLR      -(R3)         ;CLEAR A MAILBOX LOCATION
1292 000216 022703 000400      CMP      @SMAIL,R3    ;DONE?
1293 000222 001374          BNE     IS            ;BRANCH IF NO
1294 000224 105737 000042      TSTB    @42          ;ACT11 MODE?
1295 000230 001007          BNE     RESTR        ;BRANCH IF YES
1296 000232 105737 000352      TSTB    @REL         ;ARE WE RELOCATED?
1297 000236 100404          BMI     RESTR        ;BR IF YES- SINCE TPCRLF IS RELOCATED ALSO-
1298 000240 004767 015434      JSR     PC,PRITL     ;PRINT "DZMML-A" TITLE
1299 000244 000240          NOP                     ;SO WE FALL THRU TO RESTART
1300 000246 000240          NOP                     ;DITTO
1301
1302 000250 012703 000344      RESTR:  MOV      @SAVR5,R3 ;POINT R3 TO THE LOC SAVR5
1303 000254 012305          MOV      (R3)+,R5    ;RESTORE R5
1304 000256 012306          MOV      (R3)+,SP    ;AND RESTORE R6 JUST IN CASE IT IS A RESTART
1305 000260 010600          MOV      SP,R0       ;PLACE THE STARTING ADDRESS OF THE TEST IN R0
1306 000262 012746 000340      MOV      @340,-(SP)  ;SET HIGH PRIORITY FOR RTI
1307 000266 010046          MOV      R0,-(SP)
1308 000270 000002          RTI
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318

```

.SBTTL APT PARAMETER BLOCK

```

(1)
(2)
(1)
(2)
(1) 000272          .SX=.      ;SAVE CURRENT LOCATION
(1) 000024          .=24      ;SET POWER FAIL TO POINT TO START OF PROGRAM
(1) 000024          200      ;FOR APT START UP
(1) 000044          .=44      ;POINT TO APT INDIRECT ADDRESS PNTR.
(1) 000044          $APTHDR  ;POINT TO APT HEADER BLOCK
(1) 000272          .=.SX     ;RESET LOCATION COUNTER
(2)
(1)
(1)
(1)
(1) 000272          $APTHD:
(1) 000272 000000      $HIBTS: .WORD 0      ;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(1) 000274 000400      $MBAOR: .WORD $MAIL  ;ADDRESS OF APT MAILBOX (BITS 0-15)
(1) 000276 000031      $STMT:  .WORD 25     ;RUN TIM OF LONGEST TEST
(1) 000300 000156      $PASTM: .WORD 110.   ;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(1) 000302 000000      $UNITM: .WORD       ;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(1) 000304 000024          .WORD $ETEND-$MAIL/? ;LENGTH MAILBOX-ETABLE(WORDS)
1319
1320
1321 000272          MMAVA:  .=$APTHD
1322 000272          ;THIS BYTE IS USED TO DETERMINE IF MEMORY
1323          ;MANAGEMENT IS AVAILABLE OR NOT

```



```

1324
1325
1326 000273 000273      TYPENB:  .=MMAVA+1      ;THIS BYTE IS USED TO DETERMINE IF THE
1327                                     ;TYPE OUT OF ERROR HAS BEEN ENABLED OR NOT
1328
1329 000274 000274      $PRERR:  .=TYPENB+1     ;THIS BYTE DETERMINES IF THE PROGRAM HAS FOUND
1330                                     ;A PARITY ERROR
1331
1332
1333 000275 000275      $ADERR:  .=$PRERR+1    ;THIS BYTE IS USED TO DETERMINE IF THE
1334                                     ;PROGRAM HAS ENCOUNTERED ADDRESS ERROR
1335
1336
1337 000276 000276      STRTDI:  .=$ADERR+1
1338
1339 000300 000300      LOWBNK:  .=STRTDI+2
1340
1341 000302 000302      PASFLG:  .=LOWBNK+2   ;LOWER BYTE OF THIS WORD GIVES THE PASS NUMBER FOR
1342                                     ;THE SPECIFIC TEST WHEREAS THE UPPER BYTE
1343                                     ;HAS BEEN USED BY DIFFERENT TEST FOR DIFFERENT PURPOSES
1344
1345
1346 000304 000304      ENDSTK:  .=PASFLG+2
1347
1348 000306 000306      PBNK:    .=ENDSTK+2   ;HOLDS BANK UNDER TEST FOR "TST BNK XX" PRINTOUT.
1349
1350 000306 000306      DECWRD:
1351
1352 000310 000310      TYPCNT:  .=DECWRD+2   ;THIS BYTE DETERMINES THE NUMBER OF WORDS
1353                                     ;TO BE TYPED
1354 000311 000311      SAVKBB:  .BYTE  0     ;THIS LOCATION IS USED TO SAVE THE CHARACTER
1355                                     ;HIT BY THE OPERATOR
1356                                     ;ALSO IS USED AS TEMP IN ROUTINE $GTSIZ.
1357
1358 .EVEN
1359
1360 177560 177560      TKS=    177560
1361 177562 177562      $KBB=   177562
1362 177564 177564      $TPS=   177564
1363 177566 177566      $TPB=   177566
1364 177572 177572      SRQ=    177572
1365 000312 000377      BAKPAT:  .WORD  377   ;BACKGROUND PATTERN WRITTEN TO MEMORY.
1366
1367 000314 000000      SWAPAT:  .WORD
1368 000316 000430      RELBOT:  BEGIN-50    ;HOLDS LOWEST TEST ADDRESS WHEN RELOCATED.
1369 000320 100000      MINMEM:  .WORD  100000 ;FIRST ADDR FOR ECC IN CASE OF APT
1370
1371 ;;*****
1372 ;;LOCATIONS TO BE MODIFIED IF LIMITS SET BY OPERA.OR
1373 000322 000000      LOWTWO:  0           ;HOLDS BITS 17:16 OF LOW TEST ADDRESS
1374 000324 000000      LOWADD:  0           ;HOLDS BITS 15:0 OF LOW TEST ADDRESS
1375
1376 000326 000000      HIGHTWO:  0           ;HOLDS BITS 17:16 OF HIGH TEST ADDRESS
1377 000330 037776      HIGHADD:  37776      ;HOLDS BITS 15:0 OF HIGH TEST ADDRESS
1378 ;;*****
1379

```

M02

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-24  
 DZMMLA.P11 23-JUN-77 10:17 APT PARAMETER BLOCK

SEQ 0026

1380	000332	000000	\$HIMAX: 0	; HOLDS BITS 17:16 OF MAXIMUM AVAILABLE MEMORY
1381	000334	017776	\$HAXH: 17776	; HOLDS BITS 15:0 OF MAXIMUM AVAILABLE MEMORY
1382				
1383	000336	000000	MAXMEM: .WORD	; MAXIMUM CURRENT VIRTUAL MEMORY UNDER TEST
1384				
1385	000340	000000	SAVMAX: .WORD	
1386	000342	000000	SAVR4: .WORD	
1387	000344	000000	SAVR5: .WORD	
1388				
1389			;* SAVR6 POINTS TO WHERE THE PROGRAM STARTS EVEN WHEN RELOCATED.	
1390	000346	000500	SAVR6: .WORD BEGIN	; CONTAINS START ADDRESS WHEN RELOCATED ALSO.
1391	000350	000000	SAVLOC: 0	; TEST 17 STORES ERROR INFO HERE
1392	000352	000	REL: .BYTE 0	; THIS BYTE TELLS IF WE'RE RELOCATED
1399				
1400				
1401			;*364-400 IS USED AS A STACK AREA BY ERRCHK ROUTINE FOR ERROR HISTORY PRINTOUT	

1403 000400  
1404  
(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)  
(1) 000430 000  
(1) 000431 000  
(1)  
(1)  
(1)  
(1)  
(1) 000432 000000  
(1)  
(1) 000434 000  
(1) 000435 000  
(1) 000436 000000  
(1) 000440 000  
(1) 000441 000  
(1) 000442 000000  
(1) 000444 000  
(1) 000445 000  
(1) 000446 000000  
(1) 000450

.=400  
.SBTTL APT MAILBOX-ETABLE  
;\*\*\*\*\*  
.EVEN  
\$MAIL: APT MAILBOX  
\$MSGTY: .WORD AMSGTY : MESSAGE TYPE CODE  
\$FATAL: .WORD AFATAL : FATAL ERROR NUMBER  
\$TESTN: .WORD ATESTN : TEST NUMBER  
\$PASS: .WORD APASS : PASS COUNT  
\$DEVCT: .WORD ADEVCT : DEVICE COUNT  
\$UNIT: .WORD AUNIT : I/O UNIT NUMBER  
\$MSGAD: .WORD AMSGAD : MESSAGE ADDRESS  
\$MSGLG: .WORD AMSGLG : MESSAGE LENGTH  
\$ETABLE: APT ENVIRONMENT TABLE  
\$ENV: .BYTE AENV : ENVIRONMENT BYTE  
\$ENVM: .BYTE AENVM : ENVIRONMENT MODE BITS  
\$SWREG: .WORD ASWREG : APT SWITCH REGISTER  
\$USWR: .WORD AUSWR : USER SWITCHES  
\$CPUOP: .WORD ACPUOP : CPU TYPE, OPTIONS  
BITS 15-11=CPU TYPE  
11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05  
11/70=06, PDQ=07, Q=10  
BIT 10=REAL TIME CLOCK  
BIT 9=FLOATING POINT PROCESSOR  
BIT 8=MEMORY MANAGEMENT  
\$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 POS=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

1405  
1406  
1407  
1408  
1409 000500  
1410 000500 010706  
1411 000502 005746  
1412 000504 012704 016074  
1413 000510 010637 000346  
1414 000514 012737 000070 000024  
1415 000522 004767 014522

;\*\*\*\*\*  
.SBTTL BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.  
BEGIN: . =500  
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, CLRMM ;CLEAR MEM MGMT.

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

1416	000526	005037	000310		CLR	20TYP CNT	
1417	000532	105737	000352		TSTB	20REL	:RELOCATED ?
1418	000536	100002			BPL	4\$	:BR IF NOT
1419	000540	000167	001306		JMP	TSTSIZ	
1420	000544	005067	002110		CLR	LDFLG	
1421	000550	012737	000566	000004	4\$: MOV	87\$, 204	:PREPARE FOR NON-EXISTANT
1422	000556	012737	000014	177746	MOV	814, 20177746	:11/60 CACHE CONTROL REG
1423	000564	000402			BR	6\$	:BR IF ALL OK
1424	000566	062706	000004		7\$: AOD	84, SP	:RECOVER FROM TRAP
1425	000572	012701	100000		6\$: MOV	810000, R1	:2ND 16K
1426	000576	005021			CLR	(R1)+	:CLEAR FIRST LOC
1427	000600	005011			CLR	(R1)	:CLEAR SECOND LOC
1428	000602	012777	020004	002062	MOV	820004, 20CSRADR	:DIAG BIT
1429	000610	005741			TST	-(R1)	:READ THE LOC TO GET CHECKBITS IF ANY
1430	000612	032777	007740	002052	BIT	87740, 20CSRADR	:IF SET WE'RE IN MS11K
1431	000620	001007			BNE	5\$	:BR IF IN MEM UNDER TEST
1432	000622	042767	020000	002022	BIC	820000, ECCDIS	:JUST IN CASE WE ARE TESTING
1433	000630	042767	020000	002016	BIC	820000, DIAGA	:THE SECOND CSR (172134)
1434	000636	000533			BR	SETSWR	:BR IF NOT IN MEM UNDER TEST
1435	000640	005077	002026		5\$: CLR	20CSRADR	:CLEAR CSR
1436	000644	012767	000001	002006	MOV	81, LDFLG	:INDICATE PROG IN MEM UNDER TEST
1437	000652	052767	020000	001772	BIS	820000, ECCDIS	:DISABLE ECCINH IN LOW 16K
1438	000660	052767	020000	001766	BIS	820000, DIAGA	:SAME FOR DIAG CHECK
1439	000666	005001			CLR	R1	
1440	000670	005000			CLR	R0	:IT IS, CHECK FOR ERRORS
1441	000672	005067	001772		CLR	INHREL	
1442	000676	005077	001770		1\$: CLR	20CSRADR	:DON'T WANT TO TRAP
1443	000702	005711			TST	(R1)	:READ THE LOC
1444	000704	005777	100000	001760	BIT	810000, 20CSRADR	:CHECK FOR DOUBLE ERROR
1445	000712	001403			BEQ	8\$	
1446	000714	004767	013236		JSR	PC, FATERR	:*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1)	000720	000001					:*****ERROR NUMBER 1*****
(1)							
1447	000722	005767	001742		8\$: TST	INHREL	:ANY ERRORS FOUND YET ?
1448	000726	001012			BNE	2\$	:YES, DON'T LOOK ANY FURTHER
1449	000730	020127	000500		CMP	R1, 8BEGIN	
1450	000734	101007			BHI	2\$	
1451	000736	032777	000020	001726	BIT	820, 20CSRADR	:LOOKING FOR SBE'S IN 0-500
1452	000744	001403			BEQ	2\$	:BR IF NO ERRORS
1453	000746	012767	000001	001714	MOV	81, INHREL	:INHIBIT RELOCATION
1454	000754	022701	016074		2\$: CMP	8E40PROG, R1	:END OF PROGRAM YET?
1455	000760	003403			BLE	3\$	:BR IF YES
1456	000762	062701	000004		AOD	84, R1	:NO POINT TO NEXT LOCATION
1457	000766	000743			BR	1\$	:KEY READING
1458	000770	005767	001674		3\$: TST	INHREL	:RELOCATION ALLOWED ?
1459	000774	001415			BEQ	ONEPAS	:BR IF YES
1460	000776	004767	013372		JSR	PC, TPCRLF	:PRINT ROUTINE
1461	001002	047516	051040	046105	.ASCIZ	/NO RELOC-SBE IN	:0-500/
	001010	041517	051455	042502			
	001016	044440	020116	026460			
	001024	030065	000060				
1462					.EVEN		
1463	001030	005000			ONEPAS: CLR	R0	:INITIALIZE POINTER
1464	001032	005737	000404		TST	20\$TESTN	:IS THIS THE FIRST PASS
1465	001036	001402			BEQ	TSTRP	:BR IF YES (TEST TRAP CATCHERS)
1466	001040	000167	000062		JMP	SETSWR	:SET UP SWREG

```

1467 001044 012704 016074 TSTRP: MOV #ENDPROG,R4 ;ADDRESS OF END OF PROGRAM
1468 001050 012700 000377 MOV #377,R0 ;
1469 001054 005001 CLR R1 ;POINT TO ADDRESS 0
1470 001056 012124 1S: MOV (R1)+,(R4)+ ;SAVE 0000 TO BEGIN-
1471 001060 020127 000400 CMP R1,#$MAIL ;BEGINNING OF ETABLE
1472 001064 103774 BLO 1S ;BR IF NOT DONE
1473 001066 005741 3S: TST -(R1) ;ADJUST POINTER TO TRAP VECTORS
1474 001070 010011 4S: MOV R0,(R1) ;WRITE 1'S AND 0'S
1475 001072 020011 CMP R0,(R1) ;NOW COMPARE
1476 001074 001403 BEQ 5S ;BR IF OK
1477 001076 004767 013054 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 001102 000002 2 ;*****ERROR NUMBER 2*****
(1)
1478
1479 001104 000300 5S: SWAB R0 ;
1480 001106 001370 BNE 4S ;BR IF BOTH BYTES NOT TESTED
1481 001110 005701 TST R1 ;HAVE WE REACHED THE BOTTOM YET
1482 001112 001365 BNE 3S ;BRANCH IF NO
1483 001114 012701 000400 MOV #MAIL,R1 ;APT RESTORE TO $MAIL
1484 001120 014441 6S: MOV -(R4),-(R1) ;HERE!
1485 001122 005701 TST R1 ;FINISHED?
1486 001124 001375 BNE 6S ;BR IF NOT
1487 001126 005077 001540 SETSWR: CLR @CSRADR ;CLEAR CSR
1488 001132 012700 000006 MOV #6,R0 ;ADDRESS OF PSW FOR TIMEOUT TRAP
1489 001136 012710 000340 MOV #340,(R0) ;SET UP TIME OUT TRAP PSW
1490 001142 012740 001170 MOV #2S,-(R0) ;AND RETURN ADDRESS
1491 001146 105737 000420 TSTB @SENV ;RUNNING UNDER APT
1492 001152 001403 BEQ 1S ;IF NOT, SET UP ADDRESS FOR SWR
1493 001154 012737 000422 002674 MOV #SSWRREG,@SWR ;OTHERWISE, PREPARE TO USE APT SWR
1494 001162 005777 001506 1S: TST @SWR ;DOES SWITCH REG POINTED TO BY SWR EXIST
1495 001166 000410 BR APTSIZ ;YES, GO TO APTSIZ
1496 001170 062706 000004 2S: ADD #4,SP ;RESTORE STACK POINTER
1497 001174 012737 000176 002674 MOV #SWREG,@SWR ;PLACE ADDRESS OF SWITCH REG DESIGNED
1498 001202 012737 000174 002676 MOV #DSPREG,@DISPLAY ;FOR COMPUTERS NOT HAVING HARDWARE
1499 ;SWITCH REG AND RUNNING STAND ALONE.
1500 ;R4=# END PROG ON EXIT
1501 ;
1502 ;
1503 ;
1504 ;
1505 ;
1506 ;
1507 ;
1508 ;
1509 001210 012703 000336 APTSIZ: MOV #MAXMEM,R3 ;POINT R3 TO MAXMEM
1510 001214 013737 000326 000332 MOV @HIGHTWO,@SHIMAX ;IN CASE NO SELF SIZING DONE
1511 001222 013737 000330 000334 MOV @HIGHADD,@SMAXM ;
1512 001230 105737 000421 TSTB @SENVH ;DOES APT ALLOW SELF SIZING
1513 001234 100021 BPL TRYSA ;BR IF YES
1514 001236 012701 000451 MOV #SMTYP4+4,R1 ;POINT R1 TO BLOCK TYPE 4+4
1515 001242 162701 000004 1S: SUB #4,R1 ;POINT TO NEXT BLOCK TYPE
1516 001246 105711 TSTB (R1) ;IS IT NON-ZERO
1517 001250 001006 BNE 2S ;BR IF YES (MEMORY EXISTS)
1518 001252 020127 000431 CMP R1,#SMTYP1 ;ALL APT BLOCK TYPES BEEN CHECKED
1519 001256 101371 BHI 1S ;BR IF NO
1520 001260 004767 012672 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
    
```

DZMPL MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-28  
 DZMPLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEG 0030

```

(1) 001264 000003 3 ;*****ERROR NUMBER 3*****
(1)
1521
1522 001266 004767 013662 2$: JSR PC,GETADR ;SET MAX RPT ADDR INTO $MAXM+$SHIMAX
1523 001272 004767 013656 JSR PC,GETADR ;ALSO INTO HIGHADD+HIGHTWO
1524 001276 000545 BRTPSZ: BR TYPsiz ;TYPE SIZE OF MEM UNDER TEST
1525
1526 001300 032777 000100 001366 TRYSR: BIT #100,2$WR ;USER DEFINED BOUNDARIES
1527 001306 001141 BNE TYPsiz ;BR IF YES (DON'T SIZE MEM)
1528
1529 001310 005067 001342 SLFSIZ: CLR MSYES ;INITIALIZE MSYES
1530 001314 012703 000324 MOV #LOWADD,R3 ;GET READY FOR LOW TEST ADDRESS
1531 001320 005001 CLR R1 ;FIRST TEST LOC (0)
1532 001322 012710 001514 MOV #4$, (R0) ;SET UP RETURN FROM TIMEOUT TRAP
1533 001326 005077 001340 1$: CLR @CSRADR
1534 001332 011146 MOV (R1),-(SP) ;SAVE THE CONTENTS OF TEST LOC.
1535 001334 016146 000002 MOV 2(R1),-(SP)
1536 001340 005011 CLR (R1) ;TO SET CHECK BITS OF MS11K
1537 001342 005061 000002 CLR 2(R1)
1538 001346 016777 001302 001316 MOV DIAGA,@CSRADR ;DIAGNOSTIC BIT
1539 001354 005711 TST (R1) ;READ TO GET CHECK BITS
1540 001356 017702 001310 MOV @CSRADR,R2 ;GET CSR CONTENTS
1541 001362 005077 001304 CLR @CSRADR
1542 001366 012661 000002 MOV (SP)+,2(R1) ;RESTORE THE HI WORD
1543 001372 012611 MOV (SP)+,(R1) ;RESTORE THE LO WORD
1544 001374 042702 170037 BIC #170037,R2 ;ONLY CARE ABOUT 11-5
1545 001400 005702 TST R2
1546 001402 001440 BEQ 3$ ;BR IF NO MS11K
1547 001404 005767 001246 TST MSYES ;MS11K-FIRST SUCCESSFUL TEST
1548 001410 001032 BNE 22$ ;NO, BRANCH
1549 001412 020127 100000 CMP R1,#100000 ;IF THIS IS FIRST READABLE LOC
1550 001416 001015 BNE 11$ ;THEN FIRST ADDRESS IS 0000, ELSE BR
1551 001420 105737 000272 TSTB @MMAVA ;MEM MIGHT AVAIL?
1552 001424 001012 BNE 11$ ;YES, GO PUT ADDR AWAY
1553 001426 005767 001226 TST LDFLG ;RUNNING IN MEM UNDER TEST
1554 001432 001407 BEQ 11$ ;BR IF NOT
1555 001434 005037 000324 CLR @LOWADD
1556 001440 005037 000322 CLR @LOWTWO
1557 001444 005267 001206 INC MSYES
1558 001450 000404 BR 2$ ;INDICATE SOME MS11K FOUND
1559 001452 004767 013406 11$: JSR PC,PUTADR ;CONTINUE TO FIND HIGH LIMIT
1560 001456 005267 001174 INC MSYES ;PLACE ADDR AWAY (START ADDR)
1561 001462 010137 000320 2$: MOV R1,@MINMEM ;INDICATE SOME MS11 FOUND
1562 001466 010167 001172 MOV R1,$AVMIN ;FIRST TEST ADDRESS
1563 001472 062703 000012 ADD #12,R3 ;SAVE FOR RECOVERY FROM RELOC
1564 001476 062701 020000 22$: ADD #20000,R1 ;POINT TO $MAXMEM
1565 001502 000711 BR 1$ ;NEXT 4K BOUNDARY
1566 001504 005767 001146 3$: TST MSYES ;GO TEST IT
1567 001510 001772 BEQ 22$ ;FOUND ANY MS11K YET
1568 001512 000430 BR 7$ ;BR IF NO
1569 001514 062706 000004 4$: ADD #4,SP ;RESTORE STACK POINTER AFTER TRAP
1570 001520 004767 013154 JSR PC,MMING ;SERVICE MM IF AVAILABLE
1571 001524 105737 000272 TSTB @MMAVA ;SEE IF MM HAS TO BE TESTED
1572 001530 001421 BEQ 7$ ;BR IF NO MM
1573 001532 012710 001544 5$: MOV #6$, (R0) ;SET UP RETURN ADDRESS FROM TRAP
1574 001536 012701 040000 MOV #40000,R1 ;BEGIN CHECKING ABOVE 29K

```

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

1575	001542	000671			BR	1\$		
1576	001544	062706	000004		ADD	#4,SP		;RESTORE STACK POINTER
1577	001550	022701	160000		CMP	#160000,R1		;IF R1 DID NOT READ THE HIGHEST
1578								;LOCATION POINTED TO BY PAR6,
1579								;IT HAS REACHED MAX MS11K.
1580	001554	001007			BNE	7\$		;GO TO 7\$ IF DONE
1581	001556	022737	007400	172352	CMP	#7400,#172352		;CHECK PARS FOR MAX
1582	001564	001403			BEQ	7\$		;BR IF DONE
1583	001566	004767	013264		JSR	PC,UPMM		;NOT DONE UPDATE MEM MGMT.
1584	001572	000757			BR	5\$		
1585								
1586								
1587	001574	024341			7\$: CMP	-(R3),-(R1)		;CAUSE R3 TO POINT TO \$MAXM AND
1588								;R1 TO MAX AVAILABLE MEMORY
1589	001576	004767	013262		JSR	PC,PUTADR		;PLACE ADDR IN R1 AT LOCATIONS
1590								; \$MAXM AND \$HIMAX
1591	001602	162703	000004		SUB	#4,R3		;POINT TO HIGHADD
1592	001606	004767	013252		JSR	PC,PUTADR		;PLACE ADDR IN R1 AT HIGHADD & HIGHTWO
1593								
1594								
1595	001612	012710	000104		TYPsiz: MOV	#BUSER,(R0)		;SET UP VECTOR FOR FUTURE TRAP
1596	001616	005737	000406		TST	#SPASS		;ONLY CARE ABOUT FIRST PASS
1597	001622	001035			BNE	SETSTK		;BR IF NOT
1598	001624	004767	012544		JSR	PC,TPCRLF		;PRINT ROUTINE
1599	001630	051503	020122	042101	.ASCIZ	/CSR ADDR = /		
	001636	051104	036440	000040				
1600					.EVEN			
1601	001644	012703	002626		MOV	#DATBUF,R3		;NEED A LOCATION TO PRINT FROM
1602	001650	005023			CLR	(R3)+		;CLEAR HIGH ORDER BITS
1603	001652	016713	001014		MOV	CSRADR,(R3)		;ADDRESS TO BE TYPED
1604	001656	105237	000310		INCB	#TYPCNT		;ONE WORD TO BE TYPED
1605	001662	004767	012656		JSR	PC,TYPCT		;TYPE THE WORD
1606	001666	010403			MOV	R4,R3		;POINT R3 TO LOWEST AVAIL MEM
1607	001670	012701	000322		MOV	#LOWTWO,R1		
1608	001674	004767	012462		JSR	PC,PCRLF		;TYPE <CR><LF>
1609	001700	004767	012630		JSR	PC,OCTTYP		;TYPE LOW TEST ADDRESS
1610								; (LOWTWO + LOWADD)
1611	001704	004767	012364		TYPMEM: JSR	PC,\$TYPE		
1612	001710	000055			.ASCIZ	1-1		;TYPE "--"
1613					.EVEN			
1614	001712	004767	012616		JSR	PC,OCTTYP		;TYPE HIGHEST TEST ADDRESS
1615								; (HIGHTWO + HIGHADD)
1616	001716	012703	000326		SETSTK: MOV	#HIGHTWO,R3		;POINT R3 TO HIGH ORDER BITS OF HIGH MEM
1617	001722	004767	013242		JSR	PC,\$GTSIZ		;GET THE BITS 13-17 ON TOP ADDRESS
1618								;PLACED IN BITS 0-4 OF R2
1619	001726	010401			MOV	R4,R1		;ENDPROG
1620	001730	062704	000022		4\$: ADD	#18.,R4		;APPEND THE ERROR STACK FOR MEMORY
1621								;UNDER TEST TO END OF PROGRAM
1622	001734	005302			DEC	R2		;R2 = # OF 4K BANKS
1623	001736	002374			BGE	4\$		
1624	001740	010437	000304		MOV	R4,#ENDSTK		;SAVE ADDR OF END OF ERROR STACK
1625	001744	005021			6\$: CLR	(R1)+		;CLEAR THE ERROR STACK
1626	001746	020104			CMP	R1,R4		
1627	001750	101775			BLOS	6\$		
1628	001752	012737	157776	000336	MOV	#157776,#MAXMEM		;SET MAXMEM TO MAX VIRTUAL ADDRESS
1629	001760	005723			TST	(R3)+		;TESTING MEM MGMT?(READ LOC \$HIMAX)





DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-31  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

```

1682 002160 012703 000340 10$: MOV #SAVMAX,R3
1683 002164 011143 000000 11$: MOV (R3)-(R3) ;RESTORE CONTENTS OF MAXMEM
1684 002166 062713 000002 MEMTST: ADD #2,(R3) ;MAKE THE CONTENTS OF MAXMEM=
1685 ;MAX AVAIL MEM+2
1686 002172 005725 TST (R5)+ ;SET RS=MAXMEM +2
1687
1688 ;INIT BAKPAT AND SWAPAT
1689 ;AND CONTINUE TO CLEAR ACTIVE
1690 ;CHUNK OF MEMORY UNDER TEST
1691
1692
1693 002174 012702 000312 CLRMEM: MOV #BAKPAT,R2
1694 002200 012212 MOV (R2)+,(R2) ;COPY IT INTO SWAPAT
1695 002202 000312 SWAB (R2) ;SWAP THE BYTES
1696 002204 017702 000464 MOV #SWR,R2 ;GET BITS IN SW REG
1697 002210 042702 177740 BIC #177740,R2 ;ONLY WANT THE TEST NUMBER FOR LOOPING
1698 002214 022702 000020 CMP #20,R2 ;CAN'T BE HIGHER THAN 17
1699 002220 101003 BHI 1$ ;BR IF OK
1700 002222 004767 011730 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002226 000006 6 ;*****ERROR NUMBER 6*****
(1)
1701
1702 002230 013701 000320 1$: MOV #MINMEM,R1 ;GET FIRST TEST LOC IN SLICE
1703 002234 004767 013200 2$: JSR PC,TSTADD ;CHECK FOR ERROR FREE TEST ADDRESS
1704 002240 005703 TST R3 ;R3 = 0 MEANS ERROR FREE LOC FOUND
1705 002242 001446 BEQ 4$ ;BR AND SAVE IT
1706 002244 016777 000402 000420 3$: MOV ECCDIS,ACSRADR ;ECC DISABLE
1707 002252 062701 000004 ADD #4,R1 ;NEXT WORD
1708 002256 020105 CMP R1,R5 ;END OF SLICE YET ?
1709 002260 103765 BLO 2$ ;BR IF NOT
1710 002262 004767 012106 JSR PC,TPCRLF ;PRINT ROUTINE
1711 002266 047516 042440 051122 .ASCII /NO ERROR FREE LOC FOR ECC TESTS/
002274 051117 043040 042522
002302 020105 047514 020103
002310 047506 020122 041505
002316 020103 042524 052123
002324 123
1712 002325 015 041012 043505 .ASCIZ <15><12> BEGINNING AT TST13/
002332 047111 044516 043516
002340 040440 020124 051524
002346 030524 000063
1713
1714 002352 012702 000013 .EVEN
1715 002356 000402 MOV #13,R2 ;START AT TST13
1716 002360 010137 000320 BR CONT
1717 002364 016777 000262 000300 4$: MOV R1,#MINMEM ;SET UP TEST LOC
1718 002372 010500 000262 000300 CONT: MOV ECCDIS,ACSRADR ;INHIBIT ECC
1719 002374 005040 5$: CLR R5,R0 ;BEGIN CLEARING FROM TOP
1720 002376 020004 CMP R0,R4 ;TO BOTTOM
1721 002400 101375 BHI 5$
1722 002402 005077 000264 CLR ACSRADR ;RESTORE CSR
1723
1724 ;ENTER HERE FROM TSTSCP ROUTINE AT END OF EACH TEST
1725
1726 002406 005037 000302 CLR #PASFLG ;INIT SOFTEST PASS FLAG
1727 002412 110237 000404 MOVB R2,#STESTN ;SET UP STESTN WITH THE TEST

```

```

1728 002416 110277 000254          MOVB  R2, @DISPLAY ;NUMBER TO BE EXECUTED & DISPLAYED
1729
1730 002422 005077 000244          LOOP: CLR  @CSRADR ;
1731 002426 032777 004000 000240  BIT  @4000, @SWR ;ENABLE PRINTOUT OF SBE'S(INH ECC)
1732 002434 001403          BEQ  @15 ;BR IF NO
1733 002436 016777 000210 000226  IS:  MOV  @ECCDIS, @CSRADR ;YES, INHIBIT ECC
1734 002444 010401          MOV  R4, R1 ;LOWEST ADDRESS UNDER TEST
1735 002446 010246          MOV  R2, - (SP) ;SAVE R2
1736 002450 012703 000376          MOV  @376, R3 ;POINT R3 TO SCRATCH STACK
1737 002454 004767 012404          JSR  PC, @PUTADR ;GEN 18 BIT ADDRESS FROM R1
1738
1739 002460 005743          TST  -(R3) ;AND STORE IT IN (R3) AND (R3-2)
1740 002462 004767 012502          JSR  PC, @GTSIZ ;POINT R3 TO HIGH ORDER BITS
1741
1742 002466 010400          MOV  R4, R0 ;PLACE BITS 13-17 OF ADDRESS BITS
1743
1744 002470 010401          MOV  R4, R1 ;INTO 0-4 OF R2.
1745 002472 010403          MOV  R4, R3 ;PLACE ADDRESS OF LOWEST LOC
1746 002474 012602          MOV  (SP)+, R2 ;UNDER TEST IN R0
1747 002476 006302          ASL  R2 ;AND INTO R1
1748 002500 060702          ADD  PC, R2 ;AND INTO R3
1749 002502 066207 000004          ADD  @TBL - (R2), PC ;RESTORE R2
1750
1751 002506 000172          TBL: TST0-TBL ;DOUBLE IT
1752 002510 000326          TST1-TBL ;GO TO TEST # STORED IN BITS
1753 002512 000620          TST2-TBL ;3-0 OF SWITCH REG.
1754 002514 001000          TST3-TBL ;RELATIVE ADDRESS OF TEST 0
1755 002516 001232          TST4-TBL ;RELATIVE ADDRESS OF TEST 1
1756 002520 001406          TST5-TBL ;RELATIVE ADDRESS OF TEST 2
1757 002522 002056          TST6-TBL ;RELATIVE ADDRESS OF TEST 3
1758 002524 002776          TST7-TBL ;RELATIVE ADDRESS OF TEST 4
1759 002526 003772          TST10-TBL ;RELATIVE ADDRESS OF TEST 5
1760 002530 004420          TST11-TBL ;RELATIVE ADDRESS OF TEST 6
1761 002532 005052          TST12-TBL ;RELATIVE ADDRESS OF TEST 7
1762 002534 005512          TST13-TBL ;RELATIVE ADDRESS OF TEST 10
1763 002536 005624          TST14-TBL ;RELATIVE ADDRESS OF TEST 11
1764 002540 005736          TST15-TBL ;RELATIVE ADDRESS OF TEST 12
1765 002542 006570          TST16-TBL ;RELATIVE ADDRESS OF TEST 13
1766 002544 006740          TST17-TBL ;RELATIVE ADDRESS OF TEST 14
1767 002546 007076          RELOC-TBL ;RELATIVE ADDRESS OF TEST 15
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777 002550 005077 000116          TSTSCP: CLR  @CSRADR ;
1778 002554 105737 000420          TSTB  @SENV ;RUNNING UNDER APT?
1779 002560 001002          BNE  @CNTSCP ;IF SO GO TO CNTSCP
1780 002562 004767 013230          JSR  PC, @CHECKC ;TEST FOR CTRL-C AND IF TYPED
1781
1782 002566 113702 000404          CNTSCP: MOVB @TESTN, R2 ;GO TO ERROR HISTORY AND HALT
1783 002572 005237 000410          INC  @SEVCT ;PUT TEST NUMBER IN R2 LO BYTE
;TELL APT EVERYTHING'S OK
    
```

```

;SCOPE ROUTINE
;PROG COMES HERE AFTER EACH TEST AND
;IF CTRL C GO TO ERROR HISTORY TYPEOUT.
;IF SR=2000 (BIT 10) THEN HALT
;IF SR=40000 (BIT 14) THEN LOOP ON TEST
;DEFINED BY SR BITS <3:0>, ELSE GO ON
;TO NEXT TEST.
    
```

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-33  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

```

1784 002576 032777 002000 000070          BIT      #2000,2SWR      ;HALT AFTER EACH TEST?
1785 002604 001401          BEQ      TSTGO        ;BRANCH IF NOT HALTING
1786 002606 000000          SWHALT: HALT
1787 002610 032777 040000 000056 TSTGO:  BIT      #40000,2SWR ;LOOP ON TEST DESIRED?
1788 002616 001301          BNE     LOOP        ;YES, GO START SAME TEST
1789 002620 105202          INCB   R2          ;
1790 002622 000167 177536          JMP     CONT        ;GO CONTINUE EXECUTING NEXT TEST
1791
1792 002626 000000 000000          DATBUF: .WORD    0,0
1793 002632 000000 000000          TSTDAT: .WORD    0,0
1794 002636 000000 000000          SBEMSK: .WORD    0,0
1795 002642 000000 000000          DBEMSK: .WORD    0,0
1796 002646 000000 000000          DATSAV: .WORD    0,0
1797 002652 000002          ECCOIS: .WORD    2
1798 002654 000004          DIAGA:  .WORD    4
1799 002656 000000          MSYES:  .WORD    0
1800 002660 000000          LDFLG:  .WORD    0
1801 002662 000000          BASE:   .WORD    0 ;OFFSET FOR RELOC
1802 002664 100000          SAVMIN: .WORD   100000 ;TO SAVE MINMEM FROM RELOC
1803 002666 000000          ERRFLG: .WORD    0 ;FLAG TO INDICATE FIRST ERROR FOR HEADER
1804 002670 000000          INHREL: .WORD    0 ;FLAG TO INDICATE NO RELOCATION ALLOWED
1805 002672 172136          CSRADR: 172136 ;DEFAULT CSR ADDRESS
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815 002674 177570          SWR:    177570 ;CHANGES TO SWREG IF NO HARDWARE SWITCH REGISTER
1816 002676 177570          DISPLAY:177570 ;CHANGES TO DSPREG IF NO HARDWARE DISPLAY REG
1817
1818          ;*(1) THIS TEST ASSUMES THAT THE MEMORY IS IN A STATE
1819          ;* OF ALL 0'S AND R0 HAS THE ADDRESS OF THE LOWEST
1820          ;* LOCATION UNDER TEST
1821          ;*(2) IT CHECKS FOR PROPER BANK SELECTION BY WRITING
1822          ;* 1'S IN A LOCATION AND CHECKING FOR 0'S IN THE SAME
1823          ;* LOCATIONS OF OTHER 4K BANKS OF THE MEMORY
1824          ;* [I.E. LOCATIONS LIKE 7766 AND 27766 ETC.]
1825
1826 002700 122737 000000 000404 TSTO:  CMPB     #0,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
1827 002706 001403          BEQ     .+10
1828 002710 004767 011242          JSR     PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 002714 000007          7 ;*****ERROR NUMBER 7*****
(1)
1829 002716 012703 177777          MOV     #177777,R3
1830 002722 010401          1$:    MOV     R4,R1 ;R1 = ADDRESS OF LOWEST LOCATION OF MEMORY UNDER TEST
1831 002724 010310          MOV     R3,(R0) ;SET ALL THE BITS AT (R0)
1832 002726 020001          2$:    CMP     R0,R1 ;IS R0 POINTING TO THE SAME MEMORY LOCATION AS R1
1833 002730 001417          BEQ     4$ ;IN WHICH CASE CHECK FOR ALL 1'S AT (R1)
1834 002732 005711          TST    (R1) ;OTHERWISE CHECK (R1) FOR ALL 0'S
1835 002734 001430          BEQ     5$
1836 002736 020311          CMP     R3,(R1) ;IF R1 IS NOT EQUAL TO R0 AND (R1)
1837          ;DOES NOT CONTAIN ALL 0'S THEN
    
```

```

1838                                     ;CHECK TO SEE IF (R0) = (R1)
1839 002740 001004 BNE 3$
1840 002742 012767 000010 000042 MOV #10,12$ ;*****ERROR NUMBER 10*****
(1)
1841 002750 000403 BR 10$
1842 002752 3$:
(1) 002752 012767 000011 000032 MOV #11,12$ ;*****ERROR NUMBER 11*****
(1)
1843 002760 010046 10$: MOV R0,-(SP) ;SAVE R0 ON STACK
1844 002762 105237 000275 INCB 2#$ADERR ;AN ADDRESSING ERROR IS SUSPECTED
1845 002766 000407 BR 11$
1846 002770 020311 4$: CMP R3,(R1) ;CHECK (R1) FOR ALL 1'S
1847 002772 001411 BEQ 5$
1848 002774 012767 000012 000010 MOV #12,12$ ;*****ERROR NUMBER 12*****
(1)
1849 003002 010046 MOV R0,-(SP) ;SAVE R0 ON STACK
1850 003004 010300 MOV R3,R0
1851 003006 004767 010470 11$: JSR PC,ERROR ;GO TO THE ERROR SUBROUTINE
1852 003012 000000 12$: .WORD ;ERROR NUMBER TO BE REPORTED WILL BE PLACED HERE
1853 003014 012600 MOV (SP)+,R0 ;RESTORE R0
1854
1855 003016 013706 000346 5$: MOV 2$SAVR6,SP ;RESTORE THE STACK POINTER
1856 003022 062701 020000 ADD #20000,R1 ;CAUSE R1 TO POINT TO THE SAME CHIP
1857 ;LOCATION IN THE NEXT 4K BANK OF MEMORY
1858 ;BY ADDING 1 TO THE 14TH BIT OF ADDRESS IN R1
1859 003026 020105 CMP R1,R5 ;COMPARE R1 WITH THE HIGHEST MEMORY
1860 ;LOCATION WHICH IS STORED IN R5
1861 003030 103736 BLO 2$ ;IF R1 LESS THAN R5 THEN REPEAT THE TEST FROM 2$
1862
1863 003032 000646 ENDO: BR TSTSCP
1864 ;INITIAL DATA TEST
1865 ; THIS TEST CHECKS THE DI/DO LINES BY
1866 ; SHIFTING A 1 THROUGH THE WORD.
1867
1868 003034 122737 000001 000404 TST1: CMPB #1,2$STESTN ;CHECK FOR PROPER SEQUENCE
1869 003042 001403 BEQ +10 ;BR IF OK
1870 003044 004767 011106 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003050 000013 13 ;*****ERROR NUMBER 13*****
(1)
1871 003052 013701 000320 MOV 2$MINMEM,R1 ;GET TEST ADDRESS
1872 003056 012767 000001 177542 1$: MOV #1,DATBUF ;SET THE FIRST TEST BIT
1873 003064 005067 177540 CLR DATBUF+2 ;CLEAR 2ND WORD
1874 003070 016711 177532 2$: MOV DATBUF,(R1) ;WRITE TEST WORD 1
1875 003074 016761 177530 000002 MOV DATBUF+2,2(R1) ;AND TEST WORD 2
1876 003102 026711 177520 CMP DATBUF,(R1) ;NOW READ THEM
1877 003106 001405 BEQ 4$ ;BR IF FIRST 16 OK
1878 003110 016700 177512 MOV DATBUF,R0 ;GET GOOD DATA FOR ERROR MSG
1879 003114 004767 010362 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003120 000014 14 ;*****ERROR NUMBER 14*****
(1)
1880
1881 003122 026761 177502 000002 4$: CMP DATBUF+2,2(R1) ;NOW READ SECOND WORD
1882 003130 001405 BEQ 5$ ;BR IF OK
1883 003132 016700 177472 MOV DATBUF+2,R0 ;GOOD DATA FOR ERROR MSG
1884 003136 004767 010340 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003142 000015 15 ;*****ERROR NUMBER 15*****

```

# K03

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-35  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0037

```

(1)
1885
1886
1887 003144 005767 177460      5$:  IST  DATBUF+2      ; HAS LAST BIT BEEN TESTED ?
1888 003150 004404              BAI  6$           ; MINUS MEANS BIT 31
1889 003152 004567 010140      JSR  R5,DASHL     ; NO, SHIFT TEST BIT LEFT
1890 003156 002626              WORD DATBUF       ; IN DATBUF
1891 003160 000743              BR   2$           ; GO WRITE NEW TEST DATA
1892
1893
1894 003162 012767 177776 177436 6$:  MOV  #177776,DATBUF ; NOW GOING TO SHIFT A 0 IN DATA DIRECTION
1895 003170 012767 177777 177432  MOV  #-1,DATBUF+2 ; PUT A 0 IN BIT 0
1896 003176 012702 002626      MOV  #DATBUF,R2   ; AND 1'S IN ALL OTHERS
1897 003202 066702 177454      ADD  BASE,R2      ; LOC OF WRITE DATA
1898 003206 016711 177414      MOV  DATBUF,(R1) ; OFFSET IT IN CASE OF RELOC
1899 003212 016761 177412 000002 66$: MOV  DATBUF+2,2(R1) ; WRITE THE DATA
1900 003220 026711 177402      MOV  DATBUF,(R1) ; 2 WORDS WORTH
1901 003224 001405              BEQ  7$           ; NOW READ FIRST WORD
1902 003226 016700 177374      MOV  DATBUF,R0   ; BR IF OK
1903 003232 004767 010244      JSR  PC,ERROR    ; GOOD DATA
(1) 003236 000016              16           ; *ERROR* REPORT ERROR MESSAGE
(1)
1904
1905 003240 026761 177364 000002 7$:  CMP  DATBUF+2,2(R1) ; NOW, READ SECOND WORD
1906 003246 001405              BEQ  8$           ; BR IF OK
1907 003250 016700 177354      MOV  DATBUF+2,R0 ; GOOD DATA
1908 003254 004767 010222      JSR  PC,ERROR    ; *ERROR* REPORT ERROR MESSAGE
(1) 003260 000017              17           ; *****ERROR NUMBER 17*****
(1)
1909
1910 003262 022762 077777 000002 8$:  CMP  #77777,2(R2) ; TESTED BIT 31 YET?
1911 003270 001410              BEQ  10$          ; BR IF YES, WE'RE DONE
1912 003272 006162 000002      ROL  2(R2)        ; SHIFT DATBUF
1913 003276 006112              ROL  (R2)         ; AND DATBUF +2
1914 003300 103403              BCS  9$           ;
1915 003302 005362 000002      DEC  2(R2)        ;
1916 003306 000261              SEC              ; SET CARRY FOR NEXT R0!
1917 003310 000736              BR   9$           ; KEEP GOING
1918 003312 062701 020000      9$:  ADD  #20000,R1  ; NEXT 4K BANK
1919 003316 020105      10$: CMP  R1,R5        ; COMPARE AGAINST HIGHEST VIRTJAL MEM
1920 003320 103656              BLO  1$           ;
1921
1922 003322 000167 177222      END1: JMP  TSTSCP
1923
1924
1925      ; TEST2 THIS TEST CHECKS TO SEE THAT EACH ADDRESS
1926      ; BIT IN EACH 4K BANK CAN BE ASSERTED UNIQUELY.
1927      ; IT CHECKS FOR ADDRESS BITS THAT MAY BE STUCK
1928      ; HIGH, STUCK LOW OR STUCK TOGETHER.
1929
1930
1931 003326 122737 000002 000404 TST2: CMPB #2,2#STESTN ; CHECK FOR PROPER TEST SEQUENCE
1932 003334 001403              BEQ  1$           ;
1933 003336 004767 010614      JSR  PC,SEQERR   ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003342 000020              20           ; *****ERROR NUMBER 20*****
(1)

```

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-36  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0038

```

1934 003344 012701 040000 1$: MOV #40000,R1 ;
1935 003350 005000 CLR R0 ;
1936 003352 010146 MOV R1,-(SP) ;SAVE START ADDRESS
1937 003354 005021 11$: CLR (R1)+ ;CLEAR A LOCATION
1938 003356 020105 CMP R1,R5 ;END OF 20K SLICE YET ?
1939 003360 103775 BLO 11$ ;BR IF NO
1940 003362 011601 MOV (SP),R1 ;RESTORE R1 NOW
1941 003364 012702 000001 2$: MOV #1,R2 ;SET FIRST BIT
1942 003370 050201 BIS R2,R1 ;POINT R1 TO FIRST BYTE
1943 003372 105711 TSTB (R1) ;READ AND COMPARE FOR ZEROS
1944 003374 001403 BEQ 3$ ;BR IF OK
1945 003376 004767 010100 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003402 000021 21 ;*****ERROR NUMBER 21*****
(1)
1946 003404 105100 3$: COMB R0 ;FOR ERROR MSG
1947 003406 105111 COMB (R1) ;COMPLEMENT THE BYTE
1948 003410 105711 TSTB (R1) ;READ FOR NON ZEROS
1949 003412 001003 BNE 4$ ;BR IF OK
1950 003414 004767 010062 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003420 000022 22 ;*****ERROR NUMBER 22*****
(1)
1951 003422 040201 4$: BIC R2,R1 ;MASK OFF THE ASSERTED BIT
1952 003424 006302 ASL R2 ;SHIFT R2 FOR NEXT BIT
1953 003426 050201 BIS R2,R1 ;SET THE NEW BIT INTO R1
1954 003430 005711 TST (R1) ;READ THE NEW ADDRESS
1955 003432 001404 BEQ 5$ ;READ FOR ZEROS
1956 003434 005000 CLR R0 ;
1957 003436 004767 010040 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003442 000023 23 ;*****ERROR NUMBER 23*****
(1)
1958 003444 005100 5$: COM R0 ;
1959 003446 005111 COM (R1) ;COMPL THE WORD
1960 003450 005711 TST (R1) ;READ IT AGAIN
1961 003452 001003 BNE 6$ ;
1962 003454 004767 010022 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 003460 000024 24 ;*****ERROR NUMBER 24*****
(1)
1963 003462 032702 010000 6$: BIT #10000,R2 ;CHECK FOR MSB IN 4K BANK
1964 003466 001755 BEQ 4$ ;NOT LAST BIT, BRANCH
1965 003470 040201 BIC R2,R1 ;LAST BIT, CLR THE MASK
1966 003472 062701 020000 ADD #20000,R1 ;MOVE TO NEXT 4K
1967 003476 020105 CMP R1,R5 ;TOP + 2
1968 003500 103731 BLO 2$ ;
1969 003502 000167 177042 END2: JMP TSTSCP ;
1970 ;
1971 ;TEST 3 THIS TEST CHECKS FOR PROPER
1972 ; BYTE ADDRESSING.
1973 ;
1974 003506 122737 000003 000404 TST3: CMPB #3,#$TESTN ;CHECK FOR PROPER SEQUENCE NUMBER
1975 003514 001403 BEQ 1$ ;OK, BRANCH
1976 003516 004767 010434 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003522 000025 25 ;*****ERROR NUMBER 25*****
(1)
1977 ;
1978 003524 010446 1$: MOV R4,-(SP) ;SAVE R4
1979 003526 013702 000320 MOV #MINMEM,R2 ;MINMEM IS LOWEST ADDRESS

```

M03

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-37  
 DZMMLA.P11 23-JUN-77 10:17

SEQ 0039

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

1980	003532	010203			MOV	R2,R3	PUT IT IN R3 ALSO
1981	003534	062702	000004		ADD	#4,R2	POINT R2 TO LAST BYTE +1
1982	003540	012713	177777		MOV	#-1,(R3)	WRITE ALL ONES IN
1983	003544	012763	177777	000002	MOV	#-1,2(R3)	THE 4 TEST BYTES
1984	003552	105013		25:	CLRB	(R3)	CLEAR A BYTE
1985	003554	013701	000320		MOV	20MINMEM,R1	INITIALIZE R1 FOR EACH PASS
1986	003560	020201		35:	CMP	R2,R1	IF EQUAL, JUST READ LAST BYTE
1987	003562	001456			BEQ	55	BR IF EQUAL
1988	003564	020301			CMP	R3,R1	IS THIS THE BYTE OF ZEROS
1989	003566	001022			BNE	45	BR IF NOT
1990	003570	122711	000000		CMPB	#0,(R1)	IT IS, COMPARE FOR ZEROS
1991	003574	001415			BEQ	65	
1992	003576	012700	000377		MOV	#377,RO	SET UP LO BYTE
1993	003602	032701	000001		BIT	#1,R1	WHICH BYTE ARE WE TESTING ?
1994	003606	001001			BNE	95	LO BYTE RO IS OK
1995	003610	000300			SWAB	RO	HIGH BYTE, MAKE RO RIGHT
1996	003612	010146		95:	MOV	R1,-(SP)	SAVE R1
1997	003614	042701	000001		BIC	#1,R1	FOR PRINTOUT
1998	003620	004767	007656		JSR	PC,ERROR	*ERROR* REPORT ERROR MESSAGE
(1)	003624	000026			26		*****ERROR NUMBER 26*****
(1)							
1999	003626	012601			MOV	(SP)+,R1	RESTORE R1 FOR FURTHER TESTING
2000	003630	005201		65:	INC	R1	NEXT BYTE
2001	003632	000752			BR	35	RETURN
2002	003634	122711	177777	45:	CMPB	#-1,(R1)	ITS NOT THE BYTE OF 0'S, READ 1'S
2003	003640	001425			BEQ	75	
2004	003642	010304			MOV	R3,R4	GET ADDRESS
2005	003644	042704	177775		BIC	#177775,R4	TO FIND BYTE
2006	003650	030104			BIT	R1,R4	IS IT SET IN TEST ADDRESS ?
2007	003652	001003			BNE	125	NO, BRANCH CAUSE IT'S NOT IN SAME WORD
2008	003654	012700	177777		MOV	#-1,RO	DATA SHOULD BE ALL ONES
2009	003660	000406			BR	105	GO REPORT ERROR
2010	003662	012700	000377	125:	MOV	#377,RO	IT'S IN SAME WORD, BUT WHICH BYTE ?
2011	003666	032701	000001		BIT	#1,R1	CHECK FOR LO BYTE
2012	003672	001401			BEQ	105	NO, LEAVE RO ALONE
2013	003674	000300			SWAB	RO	SWAP BYTES
2014	003676	010146		105:	MOV	R1,-(SP)	SAVE R1
2015	003700	042701	000001		BIC	#1,R1	FOR PRINTOUT
2016	003704	004767	007572		JSR	PC,ERROR	*ERROR* REPORT ERROR MESSAGE
(1)	003710	000027			27		*****ERROR NUMBER 27*****
(1)							
2017	003712	012601			MOV	(SP)+,R1	RESTORE R1
2018	003714	005201		75:	INC	R1	MOVE TO NEXT BYTE
2019	003716	000720			BR	35	
2020	003720	112713	177777	55:	MOVB	#-1,(R3)	RESTORE 1'S TO BYTE JUST TESTED
2021	003724	005203			INC	R3	INC TO NEXT BYTE
2022	003726	020302			CMP	R3,R2	WAS THAT JUST THE LAST ONE?
2023	003730	001310			BNE	25	BR IF NO
2024	003732	012604			MOV	(SP)+,R4	RESTORE R4
2025	003734	000167	176610	END3:	JMP	TSTSCP	
2026							
2027							
2028							
2029							
2030							
2031							

;STATUS REGISTER RESPONSE  
 ;THIS TEST INSURES THAT THE MS11K  
 ;CSR RESPONDS AND THAT EACH BIT  
 ;CAN BE WRITTEN UNIQUELY.

# N03

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-38  
 DZMMLA.P11 23-JUN-77 10:17

SEQ 0040

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

```

2032
2033 003740 122737 000004 000404 TST4:  CMPB  #4, #STESTN ; TEST FOR PROPER TEST SEQUENCE
2034 003746 001403          BEQ  15          ;
2035 003750 004767 010202          JSR  PC, SEQERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 003754 000030          30          ; *****ERROR NUMBER 30*****
(1)
2036
2037 003756 012702 004004          15:  MOV  #25, R2
2038 003762 066702 176674          ADD  BASE, R2 ; OFFSET FOR RELOC
2039 003766 010237 000004          MOV  R2, #4
2040 003772 012701 172136          MOV  #172136, R1 ; FOR ERROR PRINTOUT
2041 003776 005777 176670          TST  #CSRADR ; TRY ACCESSING CSR
2042 004002 000403          BR   75 ; IF WE GOT HERE ALL IS WELL
2043
(1) 004004          25:  JSR  PC, FATERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004010 000031          31          ; *****ERROR NUMBER 31*****
(1)
2044
2045 004012 012700 000001          35:  MOV  #1, R0 ; SET FIRST TEST BIT
2046 004016 012737 000104 000004          MOV  #BUSER, #4 ; SET UP FOR FUTURE TRAPS
2047 004024 005767 176630          45:  TST  LDFLG ; RUNNING IN MEM UNDER TEST?
2048 004030 001004          BNE  55 ; YES, BRANCH TO 55
2049 004032 032700 050020          BIT  #50020, R0 ; NO, CHECK FOR UNUSED BITS
2050 004036 001017          BNE  75
2051 004040 000403          BR   85
2052 004042 032700 050026          55:  BIT  #50026, R0
2053 004046 001013          BNE  75
2054 004050 010077 176616          85:  MOV  R0, #CSRADR ; WRITE THE TEST BIT
2055 004054 017702 176612          MOV  #CSRADR, R2 ; GET CSR CONTENTS
2056 004060 042702 000020          BIC  #20, R2 ; SINGLE ERRORS WILL CONFUSE US
2057 004064 020002          CMP  R0, R2 ; READ AND COMPARE
2058 004066 001403          BEQ  75
2059 004070 004767 007406          JSR  PC, ERROR ; *ERROR* REPORT ERROR MESSAGE
(1) 004074 000032          32          ; *****ERROR NUMBER 32*****
(1)
2060
2061 004076 006300          75:  ASL  R0 ; SHIFT TO NEXT BIT
2062 004100 103351          BCC  45 ; CONTINUE TESTING THRU BIT 15
2063 004102 012737 000104 000004          MOV  #BUSER, #4 ; FOR ANY FUTURE TRAPS
2064 004110 000167 176434          END4: JMP  TSTSCP
2065
2066 ; SINGLE BIT ERROR TEST
2067 ; THIS TEST FIRST CHECKS FOR UNCORRECTED DATA
2068 ; WITH THE INH ECC BIT SET, CHECKS THAT THE SINGLE
2069 ; ERROR INDICATE OCCURRED AND FINALLY, WITH ECC
2070 ; ENABLED, CHECKS FOR CORRECTED DATA.
2071
2072 004114 122737 000005 000404 TST5:  CMPB  #5, #STESTN ; CHECK FOR PROPER TEST SEQUENCE
2073 004122 001403          BEQ  15
2074 004124 004767 010026          JSR  PC, SEQERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 004130 000033          33          ; *****ERROR NUMBER 33*****
(1)
2075 004132 005077 176534          15:  CLR  #CSRADR
2076 004136 012767 000001 176462 TSA:  MOV  #1, DATBUF ; INITIAL DATA
2077 004144 005067 176460          CLR  DATBUF+2 ; 32 BITS WORTH
2078 004150 012767 000001 176460 205:  MOV  #1, SBEMSK ; INITIAL ERROR MASK
  
```



2079	004156	005067	176456		CLR	SBEMSK+2	:32 BITS WORTH
2080	004162	016767	176440	176442	30\$:	MOV	DATBUF,TSTDAT
2081	004170	016767	176434	176436		MOV	DATBUF+2,TSTDAT+2
2082	004176	105737	000302			TSTB	@PASFLG
2083	004202	001404				BEQ	40\$
2084	004204	005167	176422			COM	TSTDAT
2085	004210	005167	176420			COM	TSTDAT+2
2086	004214	016702	176412		40\$:	MOV	TSTDAT,R2
2087	004220	016703	176410			MOV	TSTDAT+2,R3
2088	004224	012767	002632	006522		MOV	@TSTDAT,SOURCE
2089	004232	004767	006570			JSR	PC,CHKGEN
2090	004236	004567	007104			JSR	RS,BITCOM
2091	004242	002636				.WORD	SBEMSK
2092	004244	002632				.WORD	TSTDAT
2093	004246	013701	000320		50\$:	MOV	@MINMEM,R1
2094	004252	016777	176374	176412		MOV	ECCDIS,@CSRADR
2095	004260	016721	176346			MOV	TSTDAT,(R1)+
2096	004264	016700	006466			MOV	CHECK,R0
2097	004270	056700	176360			BIS	DIAGA,R0
2098	004274	010077	176372			MOV	R0,@CSRADR
2099	004300	016711	176330			MOV	TSTDAT+2,(R1)
2100							
2101							
2102							
2103	004304	016777	176342	176360		MOV	ECCDIS,@CSRADR
2104	004312	016700	176314			MOV	TSTDAT,R0
2105	004316	020041				CMP	R0,-(R1)
2106	004320	001403				BEQ	55\$
2107	004322	004767	007154			JSR	PC,ERROR
(1)	004326	000034					34
(1)							
2108	004330	016700	176300		55\$:	MOV	TSTDAT+2,R0
2109	004334	062701	000002			ADD	#2,R1
2110	004340	020011				CMP	R0,(R1)
2111	004342	001403				BEQ	56\$
2112	004344	004767	007132			JSR	PC,ERROR
(1)	004350	000035					35
(1)							
2113	004352	005077	176314		56\$:	CLR	@CSRADR
2114	004356	010200				MOV	R2,R0
2115	004360	020041				CMP	R0,-(R1)
2116	004362	001403				BEQ	57\$
2117	004364	004767	007112			JSR	PC,ERROR
(1)	004370	000036					36
(1)							
2118	004372	032777	000020	176272	57\$:	BIT	#20,@CSRADR
2119	004400	001011				BNE	58\$
2120	004402	010146				MOV	R1,-(SP)
2121	004404	016701	176262			MOV	CSRADR,R1
2122	004410	012700	000020			MOV	#20,R0
2123	004414	004767	007062			JSR	PC,ERROR
(1)	004420	000037					37
(1)							
2124	004422	012601				MOV	(SP)+,R1
2125	004424	005077	176242		58\$:	CLR	@CSRADR
2126	004430	010300				MOV	R3,R0

2127	004432	062701	000002		ADD	#2, R1	: POINT TO HIGH WORD
2128	004436	020011			CMP	R0, (R1)	: SEE IF ITS BEEN CORRECTED
2129	004440	001403			BEQ	59\$	: BR IF OK
2130	004442	004767	007034		JSR	PC, ERROR	: *ERROR* REPORT ERROR MESSAGE
(1)	004446	000040			40		: *****ERROR NUMBER 40*****
(1)							
2131	004450	032777	000020	176214	59\$: BIT	#20, @CSRADR	: SBE BIT SET ?
2132	004456	001010			BNE	60\$	: BR IF YES
2133	004460	010146			MOV	R1, -(SP)	: SAVE R1
2134	004462	016701	176204		MOV	CSRADR, R1	: GET ADDRESS OF CSR BEING TESTED
2135	004466	012700	000020		MOV	#20, R0	: FOR PRINTOUT
2136	004472	004767	007004		JSR	PC, ERROR	: *ERROR* REPORT ERROR MESSAGE
(1)	004476	000041			41		: *****ERROR NUMBER 41*****
(1)							
2137	004500	005737	000406		60\$: TST	@\$PASS	
2138	004504	001425			BEQ	ENDS	: Q V ONLY ONE ITERATION
2139	004506	005767	176126		TST	SBEMSK+2	: TEST FOR LAST MASK BIT
2140	004512	100404			BMI	65\$	: MINUS MEANS BIT 31
2141	004514	004567	006576		JSR	R5, DASHL	: SHIFT LEFT ROUTINE FOR 32 BITS
2142	004520	002636			.WORD	SBEMSK	: WORD TO BE SHIFTED
2143	004522	000617			BR	30\$	
2144	004524	005767	176100		65\$: TST	DATBUF+2	: LAST DATA BIT ?
2145	004530	100404			BMI	70\$	: WHICH IS BIT 31
2146	004532	004567	006560		JSR	R5, DASHL	: SHIFT IT
2147	004536	002626			.WORD	DATBUF	
2148	004540	000603			BR	20\$	
2149	004542	105737	000302		70\$: TSTB	@\$PASFLG	: FIRST OR SECOND PASS ?
2150	004546	001004			BNE	ENDS	: NON ZERO MEANS WE'RE DONE
2151	004550	105237	000302		INCB	@\$PASFLG	: NOT DONE, GO DO SECOND PASS
2152	004554	000167	177356		JMP	TSA	
2153	004560	000167	175764		ENDS: JMP	TSTSCP	
2154							: SINGLE BIT ERROR TEST TO INSURE THAT A WRITE
2155							: BYTE CLEARS SINGLE BIT ERRORS.
2156							
2157	004564	122737	000006	000404	TST6: CMPB	#6, @\$TESTN	: CHECK FOR PROPER TEST SEQUENCE
2158	004572	001403			BEQ	1\$	
2159	004574	004767	007356		JSR	PC, SEQERR	: *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1)	004600	000042			42		: *****ERROR NUMBER 42*****
(1)							
2160	004602	010701			1\$: MOV	PC, R1	: GET PC
2161	004604	012700	005476		MOV	#END6, R0	: END OF THIS TEST
2162	004610	066700	176046		ADD	BASE, R0	: FOR OFFSET
2163	004614	005711			2\$: TST	(R1)	: READ A LOC
2164	004616	032777	000020	176046	BIT	#20, @CSRADR	: LOOKING FOR SBE'S IN THIS TEST
2165	004624	001403			BEQ	3\$	: BR IF NO ERROR
2166	004626	005267	000650		INC	T6FLG	: INDICATE WITH FLAG
2167	004632	000404			BR	4\$	: AND BRANCH
2168	004634	062701	000004		3\$: ADD	#4, R1	: MOVE TO NEXT WORD
2169	004640	020100			CMP	R1, R0	: END OF TEST 6 YET ?
2170	004642	103764			BLO	2\$	: NO, BRANCH
2171	004644	004467	006420		4\$: JSR	R4, SAVOT4	: SAVE REGS R0 TO R4
2172	004650	005077	176016		CLR	@CSRADR	: CLEAR CSR
2173	004654	012767	000001	175744	MOV	#1, DATBUF	: INITIAL DATA
2174	004662	005067	175742		CLR	DATBUF+2	: 32 BITS WORTH
2175	004666	012767	000001	175742	T6A: MOV	#1, SBEMSK	: INITIAL ERROR MASK
2176	004674	005067	175740		CLR	SBEMSK+2	: 32 BITS WORTH

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

2177	004700	016767	175722	175724	T68:	MOV	DATBUF,TSTDAT	:SAVE ORIGINAL DATA
2178	004706	016767	175716	175720		MOV	DATBUF+2,TSTDAT+2	:BOTH WORDS
2179	004714	016767	175706	175724		MOV	DATBUF,DATSAV	:IN CASE PROG HAS SBE
2180	004722	016767	175702	175720		MOV	DATBUF+2,DATSAV+2	:DITTO
2181	004730	012767	002632	006016		MOV	#TSTDAT,SOURCE	:NEED ADDRESS FOR CHKGEN
2182	004736	004767	006064			JSR	PC,CHKGEN	:GENERATE CHECK BITS
2183	004742	004567	006400			JSR	RS,BITCOM	:BIT COM ROUTINE
2184	004746	002636				.WORD	SBEEMSK	:MASK FOR COMPLEMENTING
2185	004750	002632				.WORD	TSTDAT	:WORD TO BE COMPLEMENTED
2186	004752	013701	000320		30\$:	MOV	@MINMEM,R1	:FIRST TEST ADDRESS
2187	004756	010104				MOV	R1,R4	:PUT IT IN R4 ALSO
2188	004760	016777	175666	175704		MOV	ECCDIS,@CSRADR	:INHIBIT ECC
2189	004766	016724	175640			MOV	TSTDAT,(R4)+	:WRITE 16 BITS
2190	004772	016703	005760			MOV	CHECK,R3	:GET CHECKBITS
2191	004776	056703	175652			BIS	DIAGA,R3	:SET DIAGNOSTIC A BIT
2192	005002	010377	175664			MOV	R3,@CSRADR	:LOAD CSR WITH NEW IMAGE
2193	005006	016714	175622			MOV	TSTDAT+2,(R4)	:WRITE HIGH WORD+CHECKBITS
2194	005012	005077	175654			CLR	@CSRADR	:IT'S DANGEROUS IF WE DON'T
2195	005016	012702	002636			MOV	#SBEEMSK,R2	:ADDRESS OF ERROR MASK
2196	005022	066702	175634			ADD	BASE,R2	
2197	005026	162704	000002			SUB	#2,R4	:ADJUST ADDRESS
2198	005032	112714	177777		40\$:	MOVB	#-1,(R4)	:WRITE A BYTE OF 1'S
2199	005036	132712	177777			BITB	#-1,(R2)	
2200	005042	001476				BEQ	60\$	
2201	005044	005767	000432			TST	T6FLG	:SINGLE ERRORS IN TST6 AREA ?
2202	005050	001016				BNE	45\$	:BR IF YES
2203	005052	005077	175614			CLR	@CSRADR	:CLEAR CSR
2204	005056	105714				TSTB	(R4)	:READ
2205	005060	032777	000020	175604		BIT	#20,@CSRADR	:SINGLE ERROR INDICATE SET ?
2206	005066	001453				BEQ	50\$	
2207	005070	016701	175576			MOV	CSRADR,R1	:FOR PRINTOUT
2208	005074	005000				CLR	R0	
2209	005076	004767	006400			JSR	PC,ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	005102	000043				43		:*****ERROR NUMBER 43*****
(1)								
2210	005104	000444				BR	50\$	:CONTINUE
2211	005106	010446			45\$:	MOV	R4,-(SP)	:NEED A SCRATCH LOC
2212	005110	163716	000320			SUB	@MINMEM,(SP)	:TO FIND BYTE OFFSET
2213	005114	012703	002646			MOV	#DATSAV,R3	:GET ADDRESS
2214	005120	066703	175536			ADD	BASE,R3	:OFFSET FOR RELOC
2215	005124	062603				ADD	(SP)+R3	:ADD BYTE COUNT
2216	005126	112713	177777			MOVB	#-1,(R3)	:WRITE BYTE OF 1'S LIKE TEST LOCATION
2217	005132	016777	175516	175532		MOV	DIAGA,@CSRADR	:DIAG BIT
2218	005140	105714				TSTB	(R4)	:READ THE BYTE TO GET THE CHECKBITS
2219								
2220	005142	017703	175524			MOV	@CSRADR,R3	:GET CHECKBITS
2221	005146	042703	170037			BIC	#170037,R3	:CLEAR OTHER BITS
2222	005152	005077	175514			CLR	@CSRADR	
2223	005156	012767	002646	005570		MOV	#DATSAV,SOURCE	:FOR CHKGEN
2224	005164	004767	005636			JSR	PC,CHKGEN	:GENERATE CHECKBITS
2225	005170	020367	005562			CMP	R3,CHECK	:COMPARE ACTUAL VS. EXPECTED
2226	005174	001410				BEQ	50\$	:BR IF OK
2227	005176	010300				MOV	R3,R0	:GOOD DATA
2228	005200	012701	012756			MOV	#CHECK,R1	:ADDRESS OF CHECKBITS
2229	005204	066701	175452			ADD	BASE,R1	:PLUS OFFSET
2230	005210	004767	006266			JSR	PC,ERROR	:ERROR ROUTINE

2231	005214	000043			43		
2232	005216	122714	177777	50\$:	CMPB	#-1, (R4)	:CHECK DATA
2233	005222	001461			BEQ	70\$	:BR IF OK
2234	005224	010401			MOV	R4, R1	:GET R4
2235	005226	042701	000001		BIC	#1, R1	:TO MAKE EVEN ADDR FOR PRINTOUT
2236	005232	004767	006244		JSR	PC, ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	005236	000044			44		:*****ERROR NUMBER 44*****
(1)							
2237	005240	005767	000236	60\$:	TST	T6FLG	:SINGLE ERRORS IN TST6 AREA ?
2238	005244	001011			BNE	65\$	:BR IF YES
2239	005246	105714			TSTB	(R4)	:READ THE BYTE
2240	005250	032777	000020	175414	BIT	#20, @CSRAOR	:SBE ERROR BIT SET ?
2241	005256	001357			BNE	50\$	:SHOULD BE SET, BR IF OK
2242	005260	004767	006216		JSR	PC, ERROR	:*ERROR* REPORT ERROR MESSAGE
(1)	005264	000045			45		:*****ERROR NUMBER 45*****
(1)							
2243	005266	000437			BR	70\$	:CONTINUE
2244	005270	010446		65\$:	MOV	R4, -(SP)	:SAVE R4
2245	005272	163716	000320		SUB	@MINMEM, (SP)	:FOR BYTE OFFSET
2246	005276	012703	002646		MOV	@DATSAV, R3	:ADDRESS OF DATA
2247	005302	066703	175354		ADD	BASE R3	:FOR RELOC
2248	005306	062603			ADD	(SP)+, R3	:BYTE OFFSET
2249	005310	112713	177777		MOVB	#-1, (R3)	:WRITE A BYTE OF 1'S
2250	005314	016777	175334	175350	MOV	DIAGA, @CSRAOR	:DIAG BIT
2251	005322	105714			TSTB	(R4)	:READ THE BYTE
2252	005324	017746	175342		MOV	@CSRAOR, -(SP)	:GETCONTENTS
2253	005330	005077	175336		CLR	@CSRAOR	:CLEAR THE CSR
2254	005334	042716	170037		BIC	#170037, (SP)	:ONLY WANT CHECKBITS
2255	005340	012767	002646	005406	MOV	@DATSAV, SOURCE	:FOR CHKGEN
2256	005346	004767	005454		JSR	PC, CHKGEN	:GENERATE CHECKBITS ON DATA
2257	005352	022667	005400		CMP	(SP)+, CHECK	:COMPARE ACTUAL VS. EXPECTED
2258	005356	001403			BEQ	70\$	:BR IF OK
2259	005360	004767	006116		JSR	PC, ERROR	:ERROR ROUTINE
2260	005364	000045			45		
2261	005366	132712	177777	70\$:	BITB	#-1, (R2)	:CHECK FOR LAST BYTE
2262	005372	001012			BNE	80\$	:
2263	005374	005202			INC	R2	:
2264	005376	005204			INC	R4	:MOVE TO NEXT BYTE
2265	005400	013701	000320		MOV	@MINMEM, R1	:FIRST TEST ADDRESS
2266	005404	032704	000002		BIT	#2, R4	:TEST FOR LOWER WORD
2267	005410	001610			BEQ	40\$	:BR IF IT'S LOW 16 BITS
2268	005412	062701	000002		ADD	#2, R1	:ADJUST POINTER FOR ERROR REPT.
2269	005416	000605			BR	40\$	
2270	005420	005737	000406	80\$:	TST	@\$PASS	:FIRST PASS ?
2271	005424	001420			BEQ	100\$	:BR IF YES
2272	005426	005767	175206		TST	SBEMSK+2	:LAST ERROR BIT ?
2273	005432	100405			BMI	90\$	:MINUS MEANS BIT 31
2274	005434	004567	005656		JSR	R5, DASHL	:DOUBLE ARITHMETIC SHIFT LEFT
2275	005440	002636			.WORD	SBEMSK	:ON THE ERROR MASK
2276	005442	000167	177232		JMP	T6B	
2277	005446	005767	175156	90\$:	TST	DATBUF+2	:LAST DATA BIT ?
2278	005452	100405			BMI	100\$	:MINUS = BIT 31
2279	005454	004567	005636		JSR	R5, DASHL	
2280	005460	002626			.WORD	DATBUF	:SHIFT IT LEFT
2281	005462	000167	177200		JMP	T6A	
2282	005466	004767	005610	100\$:	JSR	PC, RSTOT4	:RESTORE REGS R0 TO R4

# F04

DZMLL MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-43  
 DZMLLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0045

```

2283 005472 005067 000004          CLR      T6FLG          ;FOR NEXT TIME
2284 005476 000167 175046          END6:   JMP      TSTSCP
2285 005502 000000          T6FLG: .WORD      0
2286
2287          ;DOUBLE BIT ERROR FORCE TO CHECK DOUBLE ERROR
2288          ;STATUS BIT AND TRAP THROUGH LOCATION 114.
2289
2290 005504 122737 000007 000404 TST7:   CMPB     #7, #STESTN
2291 005512 001403          BEQ      15$
2292 005514 004767 006436          JSR      PC, SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 005520 000046          ;*****ERROR NUMBER 46*****
(1)
2293 005522 005077 175144          15$:   CLR      @CSRADR    ;CLEAR CSR
2294 005526 013701 000320          MOV     @MINMEM, R1
2295 005532 004467 005532          JSR     R4, SAVOT4    ;SAVE REGS R0 TO R4
2296 005536 005067 175064          10$:   CLR      DATBUF      ;MAKE INITIAL DATA
2297 005542 005067 175062          CLR     DATBUF+2     ;ALL ZEROS
2298 005546 012767 000001 175062 20$:   MOV     #1, SBEMSK   ;INITIAL SINGLE ERROR MASK
2299 005554 005067 175060          CLR     SBEMSK+2     ;SECOND WORD
2300 005560 012767 000001 175054 30$:   MOV     #1, DBEMSK   ;INITIAL DOUBLE ERROR MASK
2301 005566 005067 175052          CLR     DBEMSK+2     ;32 BITS HERE ALSO
2302 005572 016767 175030 175032 35$:   MOV     DATBUF, TSTDAT
2303 005600 016767 175024 175026          MOV     DATBUF+2, TSTDAT+2
2304 005606 105737 000302          TSTB   @PASFLG      ;NO COMPLEMENTING FIRST PASS
2305 005612 001404          BEQ     40$
2306 005614 005167 175012          COM     TSTDAT       ;COMP FIRST WORD
2307 005620 005167 175010          COM     TSTDAT+2     ;SECOND WORD
2308 005624 005077 175042          40$:   CLR      @CSRADR
2309 005630 026767 175002 175004          CMP     SBEMSK, DBEMSK ;CAN'T HAVE THE SAME ERROR BIT SET
2310 005636 001004          BNE     45$          ;IN BOTH MASKS
2311 005640 026767 174774 174776          CMP     SBEMSK+2, DBEMSK+2 ;COULD BE EQUAL IN SECOND WORD
2312 005646 001502          BEQ     65$          ;GO MAKE THEM NOT EQUAL
2313 005650 012767 002632 005076 45$:   MOV     #TSTDAT, SOURCE ;SOURCE ADDRESS FOR CHKGEN
2314 005656 004767 005144          JSR     PC, CHKGEN   ;GO GENERATE CHECK BITS
2315 005662 004567 005460          JSR     R5, BITCOM   ;FORCE SINGLE ERROR
2316 005666 002636          .WORD  SBEMSK        ;AS PER SBEMSK
2317 005670 002632          .WORD  TSTDAT       ;ON DATA IN TSTDAT
2318 005672 004567 005450          JSR     R5, BITCOM   ;FORCING DOUBLE ERROR
2319 005676 002642          .WORD  DBEMSK       ;THIS MASK INDICATES SECOND BIT IN ERROR
2320 005700 002632          .WORD  TSTDAT       ;SAME DATA WORD
2321 005702 016704 005050          MOV     CHECK, R4    ;GET CHECKBITS
2322 005706 056704 174742          BIS     DIAGA, R4    ;SET DIAGNOSTIC A BIT
2323 005712 016777 174734 174752          MOV     ECCDIS, @CSRADR ;INHIBIT ECC
2324 005720 016721 174706          MOV     TSTDAT, (R1)+ ;WRITE 16 BITS
2325 005724 010477 174742          MOV     R4, @CSRADR ;LOAD CSR
2326 005730 016711 174700          MOV     TSTDAT+2, (R1) ;WRITE HIGH WORD
2327 005734 005077 174732          CLR     @CSRADR     ;CLEAR CSR AGAIN
2328 005740 162701 000002          SUB     #2, R1       ;ADJUST TEST ADDRESS
2329 005744 005711          TST     (R1)         ;READ THE LOCATION
2330 005746 032777 100000 174716          BIT     #100000, @CSRADR ;LOOK FOR DBE STATUS BIT
2331 005754 001011          BNE     50$          ;IT SHOULD BE SET
2332 005756 010146          MOV     R1, -(SP)    ;SAVE R1
2333 005760 016701 174706          MOV     CSRADR, R1   ;GET CSR ADDRESS
2334 005764 012700 100000          MOV     #100000, R0  ;SHOW BIT 15 SET
2335 005770 004767 005506          JSR     PC, ERROR    ;*ERROR* REPORT ERROR MESSAGE
(1) 005774 000047          ;*****ERROR NUMBER 47*****

```

```

(1)
2336 005776 012601          MOV      (SP)+,R1      ;RESTORE R1
2337 005000 012702 006042    50$:     MOV      #60$,R2      ;SET UP FOR BUS PBL
2338 006004 066702 174652    ADD      BASE,R2      ;
2339 006010 010237 000114    MOV      R2,#114      ;
2340 006014 052777 000001 174650  BIS      #1,CSRADR    ;SET DOUBLE ERROR INDICATE
2341
2342 006022 005711          TST      (R1)         ;READ THE TEST LOCATION AGAIN
2343 006024 005077 174642    CLR      CSRADR       ;CLEAR STATUS REG
2344 006030 005000          CLR      R0           ;CLEAR R0
2345 006032 004767 005444    JSR      PC,ERROR     ;*ERROR* REPORT ERROR MESSAGE
(1) 006036 000050          SO
(1)
2346 006040 000402          BR       61$          ;IF ERROR, DON'T ADJUST STACK
2347 006042 062706 000004    60$:     ADD      #4,SP       ;ADJUST STACK FROM TRAP
2348 006046 005737 000406    61$:     TST      2$PASS      ;FIRST PASS ?
2349 006052 001424          BEQ      80$          ;BR IF YES
2350 006054 005767 174564    65$:     TST      DBEMSK+2    ;CHECK MASK FOR LAST BIT
2351 006060 100404          BMI     70$          ;MINUS = BIT31
2352 006062 004567 005230    JSR      R5,DASHL     ;DOUBLE SHIFT
2353 006066 002642          .WORD   DBEMSK
2354 006070 000640          BR       35$
2355 006072 005767 174542    70$:     TST      SBEMSK+2    ;CHECK SINGLE ERROR MASK TOO
2356 006076 100404          BMI     75$          ;BR IF DONE
2357 006100 004567 005212    JSR      R5,DASHL     ;SHIFT
2358 006104 002636          .WORD   SBEMSK
2359 006106 000624          BR       30$
2360 006110 105737 000302    75$:     TSTB    2$PASFLG     ;FIRST PASS
2361 006114 001003          BNE     80$          ;NON ZERO MEANS WE'RE DONE
2362 006116 105237 000302    INCB    2$PASFLG     ;FIRST PASS, NOT DONE
2363 006122 000605          BR      10$          ;KEEP GOING
2364 006124 005737 000406    80$:     TST      2$PASS      ;CHECKING PASS AGAIN
2365 006130 001406          BEQ     82$          ;CHECK 1K ADDR FUNCTION ON
2366                                     ;FIRST PASS ONLY
2367 006132 016777 174514 174532  MOV      ECCDIS,CSRADR ;OTHERWISE DISABLE ECC
2368 006140 005021          CLR     (R1)+        ;CLEAR THE DBE'S
2369 006142 005011          CLR     (R1)
2370 006144 000545          BR      90$
2371 006146 005077 174520    82$:     CLR      CSRADR     ;AND EXIT
2372 006152 005002          CLR     R2           ;CLEAR CSR
2373 006154 005711          TST     (R1)         ;OUR COUNTER
2374 006156 017703 174510    MOV     CSRADR,R3    ;READ THE LOCATION
2375 006162 005077 174504    CLR     CSRADR       ;MOV CHECKBITS INTO R3
2376 006166 042703 170037    BIC     #170037,R3   ;CLEAR CSR AGAIN
2377 006172 006303          ASL     R3           ;ONLY WANT CHECKBITS
2378 006174 006303          ASL     R3           ;POSITION THEM
2379 006176 006303          ASL     R3           ;INTO LOW BYTE
2380 006200 000303          SWAB   R3           ;HERE
2381 006202 004767 007142    JSR     PC,GET1K     ;GET THE 1K BANK
2382 006206 020300          CMP     R3,R0        ;BETTER BE THE SAME
2383 006210 001413          BEQ     85$          ;BR IF OK
2384 006212 010146          MOV     R1,-(SP)     ;SAVE R1 FOR NOW
2385 006214 016777 174432 174450  MOV     ECCDIS,CSRADR ;DISABLE ECC
2386 006222 010311          MOV     R3,(R1)     ;PUT BAD DATA TO BE PRINTED INTO R1
2387 006224 005077 174442    CLR     CSRADR       ;RESTORE CSR
2388 006230 004767 005246    JSR     PC,ERROR     ;*ERROR* REPORT ERROR MESSAGE
    
```

# H04

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-45  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0047

```

(1) 006234 000051          51          ;*****ERROR NUMBER 51*****
(1)
2389 006236 012601          MOV      (SP)+,R1      ;RESTORE R1
2390 006240 016777 174406 174424 85$: MOV      ECCDIS,CSRADR ;DISABLE ERROR CORRECTING
2391 006246 005021          CLR      (R1)+        ;TO CLEAR ANY DOUBLE ERRORS
2392 006250 005011          CLR      (R1)         ;
2393 006252 005077 174414  CLR      CSRADR      ;RESTORE CSR
2394 006256 042701 003777  BIC      #3777,R1     ;STRIP AWAY TO LAST 1K
2395 006262 062701 004000  ADD      #4000,R1     ;MOVE UP TO NEXT 1K
2396 006266 020105          CMP      R1,R5        ;OVER TOP YET ?
2397 006270 103073          BHS     90$           ;BR IF YES
2398 006272 004767 007142 88$: JSR      PC,TSTADD    ;CHECK FOR ERROR FREE LOC
2399 006276 005703          TST     R3            ;ZERO MEANS ERROR FREE
2400 006300 001456          BEQ     86$           ;BR IF OK
2401 006302 005202          INC     R2            ;COUNT ONE WORD CHECKED
2402 006304 020227 001000  CMP      R2,#1000     ;TESTED WHOLE 1K SLICE YET ?
2403 006310 103447          BLO     87$           ;BR IF NOT
2404 006312 004767 006056  JSR      PC,TPCRLF    ;PRINT ROUTINE
2405 006316 051524 033524 047055 .ASCIZ  /TST7-NO ERROR FREE LOC IN 1K SLICE AT /
      006324 020117 051105 047522
      006332 020122 051106 042505
      006340 046040 041517 044440
      006346 020116 045461 051440
      006354 044514 042503 040440
      006362 020124 000
2406          006366          .EVEN
2407 006366 042701 003776  BIC      #3776,R1     ;WANT STARTING ADDRESS OF 1K
2408 006372 012703 006476  MOV      #SAV7+2,R3   ;SCRATCH LOC FOR ADDRESS
2409 006376 066703 174260  ADD      BASE,R3      ;FOR OFFSET
2410 006402 010146          MOV      R1,-(SP)     ;SAVE R1
2411 006404 004767 006454  JSR      PC,PUTADR    ;PUT ADDRESS IN SAV7 & SAV7+2
2412 006410 105277 171674  INCB     @TYPCNT      ;PRINT ONE WORD
2413 006414 005741          TST     -(R1)         ;ADJUST FOR PRINTOUT
2414 006416 004767 006122  JSR      PC,TYPOCT    ;PRINT IT HERE
2415 006422 005002          CLR     R2            ;RESTORE R2 FOR NEXT SLICE
2416 006424 012601          MOV     (SP)+,R1     ;RESTORE R1
2417 006426 000704          BR     85$           ;GO LOOK FOR GOOD LOC
2418 006430 062701 000004 87$: ADD      #4,R1       ;NEXT WORD
2419 006434 000716          BR     88$           ;GO CHECK IT
2420 006436 016721 174170 86$: MOV      TSTDAT,(R1)+ ;WRITE FIRST WORD
2421 006442 010477 174224  MOV      R4,CSRADR    ;CSR IMAGE WITH BAD CHECKBITS
2422 006446 016711 174162  MOV      TSTDAT+2,(R1);WRITE 2ND WORD + CHECKBITS
2423 006452 162701 000002  SUB      #2,R1        ;ADJUST R1
2424 006456 000633          BR     82$           ;CONTINUE TESTING
2425 006460 005077 174206 90$: CLR      CSRADR      ;CLEAR CSR TO NORMAL
2426 006464 004767 004612  JSR      PC,RSTOT4    ;RESTORE THE REGS 0 TO 4
2427 006470 000167 174054  END7:  JMP      TSTSCP
2428 006474 000000 000000  SAV7:  .WORD 0,0
2429
2430          ;THIS TEST CHECKS THE WRITE INHIBIT ON DOUBLE
2431          ;BIT ERRORS DURING A DATIP OPERATION BY USE
2432          ;OF AN 'INC' INSTRUCTION.
2433
2434 006500 122737 000010 000404 TST10: CMPB     #10,@$TESTN
2435 006506 001403          BEQ     1$
2436 006510 004767 005442  JSR      PC,SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
  
```

```

(1) 006514 000052 52 ;*****ERROR NUMBER 52*****
(1)
2437 006516 004467 004546 15: JSR R4,SAVOT4 ;SAVE REGS R0 TO R4
2438 006522 005067 174100 10$: CLR DATBUF ;INITIAL DATA
2439 006526 005067 174076 CLR DATBUF+2 ;2 WORDS WORTH
2440 006532 012767 000001 174076 20$: MOV #1,SBEMSK ;INITIAL ERROR MASK
2441 006540 005067 174074 CLR SBEMSK+2
2442 006544 012767 000001 174070 30$: MOV #1,DBEMSK ;DOUBLE ERROR MASK
2443 006552 005067 174066 CLR DBEMSK+2 ;2 WORDS
2444 006556 016767 174044 174046 35$: MOV DATBUF,TSTDAT ;PRESERVE ORIG DATA
2445 006564 016767 174040 174042 MOV DATBUF+2,TSTDAT+2
2446 006572 105737 000302 TSTB #PPFLG ;SECOND PASS YET ?
2447 006576 001404 BEQ 40$ ;BR IF NO
2448 006600 005167 174026 COM TSTDAT ;COMPL DATA ON SECOND PASS
2449 006604 005167 174024 COM TSTDAT+2 ;
2450 006610 005077 174056 40$: CLR #CSRADR ;
2451 006614 026767 174022 174014 CMP DBEMSK,SBEMSK ;CHECK FOR SAME MASKS
2452 006622 001004 BNE 45$ ;BR IF OK
2453 006624 026767 174014 174006 CMP DBEMSK+2,SBEMSK+2
2454 006632 001476 BEQ 70$ ;BR IF THEY'RE EQUAL
2455 006634 012767 002632 004112 45$: MOV #TSTDAT,SOURCE ;SET UP ADDRESS FOR CHKGEN
2456 006642 004767 004160 JSR PC,CHKGEN ;GENERATE CHECK BITS
2457 006646 004567 004474 JSR R5,BITCOM ;COMPL ROUTINE
2458 006652 002636 .WORD SBEMSK ;MASK
2459 006654 002632 .WORD TSTDAT ;DATA
2460 006656 004567 004464 JSR R5,BITCOM ;TO MAKE DOUBLE ERROR
2461 006662 002642 .WORD DBEMSK
2462 006664 002632 .WORD TSTDAT
2463 006666 016702 004064 MOV CHECK,R2 ;GET CHECKBITS
2464 006672 056702 173756 BIS DIAGA,R2 ;SET DIAGNOSTIC A BIT
2465 006676 013701 000320 50$: MOV #MINMEM,R1 ;TEST ADDRESS
2466 006702 016777 173744 173762 MOV ECCDIS,#CSRADR ;INHIBIT ECC
2467 006710 016721 173716 MOV TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
2468 006714 010277 173752 MOV R2,#CSRADR ;LOAD CSR WITH IMAGE FROM R2
2469 006720 016711 173710 MOV TSTDAT+2,(R1) ;SECOND 16 BITS+CHECKBITS
2470 006724 105067 004064 CLRB UPPFLG ;INDICATE LOWER WORD
2471 006730 013703 000320 MOV #MINMEM,R3 ;TEST ADDRESS
2472 006734 005077 173732 55$: CLR #CSRADR ;CLEAR IT
2473 006740 005223 INC (R3)+ ;INC INSTRUCTION
2474 006742 026741 173664 CMP TSTDAT,-(R1) ;CHECK FOR UNCHANGED DATA
2475 006746 001405 BEQ 60$ ;SHOULD BE UNCHANGED
2476 006750 016700 173656 MOV TSTDAT,R0 ;FOR PRINTOUT
2477 006754 004767 004522 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 006760 000053 53 ;*****ERROR NUMBER 53*****
(1)
2478 006762 062701 000002 60$: ADD #2,R1 ;POINT TO UPPER WORD
2479 006766 026711 173642 CMP TSTDAT+2,(R1) ;READ IT
2480 006772 001434 BEQ 80$ ;BR IF UNCHANGED
2481 006774 016700 173634 MOV TSTDAT+2,R0 ;
2482 007000 004767 004476 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 007004 000054 54 ;*****ERROR NUMBER 54*****
(1)
2483 007006 105767 004002 65$: TSTB UPPFLG ;LOWER WORD
2484 007012 001003 BNE 66$ ;BR IF NO
2485 007014 105267 003774 INCB UPPFLG
2486 007020 000745 BR 55$
    
```



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

```

2487 007022 005737 000406      66$:  TST      2#PASS      ;CHECK PASS #
2488 007022 001424 173610      67$:  BEQ      90$          ;BR IF FIRST
2489 007030 005767 173610      70$:  TST      DBEMSK+2    ;LAST BIT IN MASK ?
2490 007034 100404 004254      71$:  BMI      75$          ;BR IF BIT 31
2491 007036 004567 004254      72$:  JSR      R5,DASHL    ;SHIFT ROUTINE
2492 007042 002642 173566      75$:  .WORD    DBEMSK
2493 007044 000644 004236      75$:  BR       35$          ;LAST BIT IN SINGLE ERROR MASK ?
2494 007046 005767 004236      75$:  TST      SBEMSK+2    ;BR IF YES
2495 007052 100404 004236      76$:  BMI      80$          ;SHIFT
2496 007054 004567 004236      77$:  JSR      R5,DASHL
2497 007060 002636 000302      80$:  .WORD    SBEMSK
2498
2499 007062 000630 000302      80$:  BR       30$          ;WHICH PASS
2500 007064 105737 000302      81$:  TSTB     2#PASFLG    ;BR IF WE'RE DONE
2501 007070 001003 000302      82$:  BNE      90$          ;INDICATE SECOND PASS COMING
2502 007072 105237 000302      83$:  INCB     2#PASFLG
2503 007076 000611 173546 173564 90$:  BR       10$          ;GO DO IT!
2504 007100 016777 173546 173564 90$:  MOV      ECCDIS,2CSRADR ;INHIBIT ECC TO CLEAR DBE'S
2505 007106 005011 173554      91$:  CLR      (R1)
2506 007110 005041 004160      92$:  CLR      -(R1)
2507 007112 005077 004160      93$:  CLR      2CSRADR     ;RESTORE CSR TO NORMAL
2508 007116 004767 173422      94$:  JSR      PC,RSTOT4   ;RESTORE THE REGS
2509 007122 000167 173422      END10: JMP      TSTSCP
2510
2511      ;CHECK FOR WRITE INHIBIT DURING A WRITE BYTE.
2512      ;CHECKS FOR UNCORRECTED DATA.
2513
2514 007126 122737 000011 000404 TST11: CMPB     111,2#STESTN
2515 007134 001403 005014      1$:  BEQ      1$
2516 007136 004767 000055      1$:  JSR      PC,SEQERR   ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
2517 (1) 007142 000055 004120      1$:  55          ;*****ERROR NUMBER 55*****
2518 (1) 007144 004467 004120      1$:  JSR      R4,SAVOT4   ;SAVE REGS R0 TO R4
2519 007150 005067 173452      10$:  CLR      DATBUF      ;INITIAL DATA
2520 007154 005067 173450      11$:  CLR      DATBUF+2    ;32 BITS WORTH
2521 007160 012767 000001 173450 20$:  MOV      #1,SBEMSK   ;SINGLE ERROR MASK
2522 007166 005067 173446      21$:  CLR      SBEMSK+2
2523 007172 012767 000001 173442 30$:  MOV      #1,DBEMSK   ;DOUBLE ERROR MASK
2524 007200 005067 173440      31$:  CLR      DBEMSK+2
2525 007204 016767 173416 173420 35$:  MOV      DATBUF,TSTDAT ;PRESERVE ORIG DATA
2526 007212 016767 173412 173414 36$:  MOV      DATBUF+2,TSTDAT+2
2527 007220 105737 000302      37$:  TSTB     2#PASFLG    ;WHICH PASS ?
2528 007224 001404 173400      38$:  BEQ      40$          ;FIRST PASS, NO COMPLEMENTING
2529 007226 005167 173376      39$:  COM      TSTDAT
2530 007232 005167 173430      40$:  COM      TSTDAT+2    ;SECOND PASS, COMPLEMENT TSTDAT
2531 007236 005077 173370 173372 40$:  CLR      2CSRADR
2532 007242 026767 173362 173364 41$:  CMP      SBEMSK,DBEMSK ;CHECK FOR SAME MASKS
2533 007250 001004 173362 173364 42$:  BNE      45$          ;BR IF NOT EQUAL
2534 007252 026767 002632 003464 43$:  CMP      SBEMSK+2,DBEMSK+2 ;SECOND WORD ALSO
2535 007260 001473 002632 003464 44$:  BEQ      70$          ;BR TO MAKE THEM NOT EQUAL
2536 007262 012767 002632 003464 45$:  MOV      #TSTDAT,SOURCE ;ADDRESS FOR CHKGEN
2537 007270 004767 003532      46$:  JSR      PC,CHKGEN   ;GO GENERATE CHECK BITS
2538 007274 004557 004046      47$:  JSR      R5,BITCOM   ;BIT COMP ROUTINE
2539 007300 002636 004036      48$:  .WORD    SBEMSK      ;MASK
2540 007302 002632 004036      49$:  .WORD    TSTDAT     ;DATA WORD
2540 007304 004567 004036      50$:  JSR      R5,BITCOM   ;MUST FORCE SECOND ERROR
    
```

K04

2541	007310	002642		.WORD	DBEMSK	;MASK
2542	007312	002632		.WORD	TSTDAT	
2543	007314	016702	003436	MOV	CHECK,R2	;GET CHECKBITS
2544	007320	056702	173330	BIS	DIAGA,R2	;SET DIAGNOSTIC A BIT
2545	007324	013701	000320	MOV	2#MINMEM,R1	;TEST LOCATION
2546	007330	016777	173316	MOV	ECCDIS,2CSRADR	;DISABLE ECC
2547	007336	016721	173270	MOV	TSTDAT,(R1)+	;WRITE FIRST 16 BITS
2548	007342	010277	173324	MOV	R2,2CSRADR	;LOAD CSR WITH IMAGE FROM R2
2549	007346	016711	173262	MOV	TSTDAT+2,(R1)	;WRITE SECOND 16 BITS + CHECKBITS
2550	007352	005077	173314	CLR	2CSRADR	;CLEAR CSR
2551	007356	013702	000320	MOV	2#MINMEM,R2	;GET ADDRESS OF TEST LOC
2552	007362	010203		MOV	R2,R3	;R2 DESIGNATES FIRST BYTE
2553	007364	062703	000003	ADD	#3,R3	;R3 DESIGNATES LAST BYTE
2554	007370	112722	000360	MOV	#360,(R2)+	;TRY WRITING A BYTE
2555	007374	013701	000320	MOV	2#MINMEM,R1	
2556	007400	026711	173226	CMP	TSTDAT,(R1)	;CHECK FOR UNCHANGED DATA
2557	007404	001405		BEQ	60\$	;BR IF OK
2558	007406	016700	173220	MOV	TSTDAT,RO	;FOR ERROR MSG
2559	007412	004767	004064	JSR	PC,ERROR	;*ERROR* REPORT ERROR MESSAGE
(1)	007416	000056		56		;*****ERROR NUMBER 56*****
(1)						
2560	007420	062701	000002	ADD	#2,R1	;UPPER WORD
2561	007424	026711	173204	CMP	TSTDAT+2,(R1)	;READ SECOND WORD
2562	007430	001405		BEQ	65\$	;BR IF UNCHANGED
2563	007432	016700	173176	MOV	TSTDAT+2,RO	
2564	007436	004767	004040	JSR	PC,ERROR	;*ERROR* REPORT ERROR MESSAGE
(1)	007442	000057		57		;*****ERROR NUMBER 57*****
(1)						
2565	007444	020203		65\$: CMP	R2,R3	;TESTED LAST BYTE ?
2566	007446	001350		BNE	55\$	;BR IF NO
2567	007450	005737	000406	70\$: TST	2#SPASS	;FIRST PASS ?
2568	007454	001424		BEQ	100\$	;BR IF YES
2569	007456	005767	173162	TST	DBEMSK+2	;CHECKING FOR LAST ERROR BIT
2570	007462	100404		BMI	80\$	;BR IF DONE HERE
2571	007464	004567	003626	JSR	R5,DASHL	;NOT DONE, SHIFT LEFT
2572	007470	002642		.WORD	DBEMSK	;DOUBLE ERROR MASK
2573	007472	000644		BR	35\$	
2574	007474	005767	173140	80\$: TST	SBEMSK+2	;LAST SBE MASK
2575	007500	100404		BMI	90\$	;BR IF DONE WITH THIS PASS
2576	007502	004567	003610	JSR	R5,DASHL	;DOUBLE WORD SHIFT ROUTINE
2577	007506	002636		.WORD	SBEMSK	
2578	007510	000630		BR	30\$	
2579	007512	105737	000302	90\$: TSTB	2#PASFLG	;TEST PASS FLAG
2580	007516	001003		BNE	100\$	;NON ZERO MEANS WE'RE DONE
2581	007520	105237	000302	INCB	2#PAS LG	;NOT DONR
2582	007524	000611		BR	10\$	
2583	007526	016777	173120	100\$: MOV	ECCDIS,2CSRADR	;DISABLE ECC
2584	007534	013701	000320	MOV	2#MINMEM,R1	;TEST LOCATION
2585	007540	005021		CLR	(R1)+	
2586	007542	005011		CLR	(R1)	;TO ERASE ANY DBE'S FROM TESTING
2587	007544	005077	173122	CLR	2CSRADR	;RESTORE CSR
2588	007550	004767	003526	JSR	PC,RSTOT4	;RESTORE REGS R0 TO R4
2589	007554	000167	172770	END11: JMP	TSTSCP	
2590						
2591						;DOUBLE BIT ERROR WRITE CANCEL WITH
2592						;WORD WRITE.

```

2593          :CHECKS WRITE INHIBIT WITH WORD WRITES TO
2594          :WORD WITH DOUBLE ERROR.
2595
2596 007560 122737 000012 000404 TST12: CMPB  #12,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2597 007566 001403          BEQ  1$          ;BR IF OK
2598 007570 004767 004362          JSR  PC,SEQERR  ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 007574 000060          60          ;*****ERROR NUMBER 60*****
(1)
2599 007576 004467 003466          1$: JSR  R4,SAVOT4 ;SAVE REGS R0 TO R4
2600 007602 005067 173020          T12A: CLR  DATBUF    ;BACKGROUND FOR DOUBLE ERRORS
2601 007606 005067 173016          CLR  DATBUF+2  ;2 WORDS WORTH
2602 007612 012767 000001 173016          MOV  #1,SBEMSK ;SINGLE ERROR MASK
2603 007620 005067 173014          CLR  SBEMSK+2
2604 007624 012767 000001 173010          T12B: MOV  #1,DBEMSK ;DOUBLE ERROR MASK
2605 007632 005067 173006          CLR  DBEMSK+2
2606 007636 016767 172764 172766          35$: MOV  DATBUF,TSTDAT ;DATA FOR TEST
2607 007644 016767 172760 172762          MOV  DATBUF+2,TSTDAT+2 ;BOTH WORDS
2608 007652 105737 000302          TSTB 2#PASFLG ;COMP DATA ON SECOND PASS ONLY
2609 007656 001404          BEQ  40$          ;BR IF FIRST PASS
2610 007660 005167 172746          COM  TSTDAT    ;COMP FIRST WORD
2611 007664 005167 172744          COM  TSTDAT+2  ;NOW SECOND WORD
2612 007670 026767 172742 172744          40$: CMP  SBEMSK,DBEMSK ;CHECK FOR IDENTICAL MASKS
2613 007676 001004          BNE  45$          ;BR IF DIFFERENT
2614 007700 026767 172734 172736          CMP  SBEMSK+2,DBEMSK+2 ;UPPER WORD TOO
2615 007706 001501          BEQ  70$          ;BR TO MAKE THEM NOT EQUAL
2616 007710 012767 002632 003036          45$: MOV  #TSTDAT,SOURCE ;NEED ADDR OF DATA FOR CHKGEN
2617 007716 004767 003104          JSR  PC,CHKGEN ;GO GENERATE CHECK BITS
2618 007722 004567 003420          JSR  R5,BITCOM ;BIT COMP ROUTINE TO FORCE ERRORS
2619 007726 002636          .WORD SBEMSK ;FIRST ERROR
2620 007730 002632          .WORD TSTDAT ;DATA
2621 007732 004567 003410          JSR  R5,BITCOM ;CALL IT AGAIN
2622 007736 002642          .WORD DBEMSK ;FOR SECOND ERROR
2623 007740 002632          .WORD TSTDAT
2624 007742 016700 003010          MOV  CHECK,R0 ;GET CHECKBITS
2625 007746 056700 172702          BIS  DIAG,R0 ;SET DIAGNOSTIC BIT
2626 007752 013701 000320          50$: MOV  2#MINMEM,R1 ;FIRST TEST ADDRESS
2627 007756 016777 172670 172706          MOV  ECCDIS,2CSRADR ;INHIBIT ECC
2628 007764 016721 172642          MOV  TSTDAT,(R1)+ ;WRITE FIRST 16 BITS
2629 007770 010077 172676          MOV  R0,2CSRADR ;LOAD CSR FROM R0
2630 007774 016711 172634          MOV  TSTDAT+2,(R1) ;WRITE SECOND 16 BITS + CHECKBITS
2631 010000 105067 003010          CLRB UPPFLG ;SET FOR 2 LOOPS
2632 010004 162701 000002          SUB  #2,R1 ;POINT TO LOW WORD
2633 010010 005077 172656          55$: CLR  2CSRADR ;CLEAR CSR
2634 010014 012721 177400          MOV  #177400,(R1)+ ;TRY WRITING LOCATION
2635 010020 013701 000320          MOV  2#MINMEM,R1
2636 010024 026711 172602          CMP  TSTDAT,(R1) ;CHECK FOR ORIGINAL DATA
2637 010030 001405          BEQ  60$          ;SHOULD BE UNCHANGED
2638 010032 016700 172574          MOV  TSTDAT,R0 ;FOR ERROR MSG
2639 010036 004767 003440          JSR  PC,ERROR  ;*ERROR* REPORT ERROR MESSAGE
(1) 010042 000061          61          ;*****ERROR NUMBER 61*****
(1)
2640 010044 062701 000002          60$: ADD  #2,R1 ;UPPER WORD
2641 010050 026711 172560          CMP  TSTDAT+2,(R1) ;THIS SHOULD BE UNCHANGED ALSO
2642 010054 001405          BEQ  65$          ;
2643 010056 016700 172552          MOV  TSTDAT+2,R0 ;
2644 010062 004767 003414          JSR  PC,ERROR  ;*ERROR* REPORT ERROR MESSAGE

```

```

(1) 010066 000062          62          ;*****ERROR NUMBER 62*****
(1)
2645 010070 105767 002720    65$:  TSTB  UPPFLG  ;WHICH LOOP ?
2646 010074 001003          BNE  66$  ;SECOND, BR OUT
2647 010076 105267 002712    INCB  UPPFLG  ;FIRST, KEEP GOING
2648 010102 000742          BR    55$
2649 010104 005737 000406    66$:  TST   2#SPASS ;FIRST PASS ?
2650 010110 001426          BEQ   85$  ;BR IF YES
2651 010112 005767 172526    70$:  TST   DBEMSK+2 ;LAST BIT ?
2652 010116 100404          BMI   75$  ;MINUS = BIT 31
2653 010120 004567 003172    JSR   RS,DASHL ;SHIFT ROUTINE
2654 010124 002642          .WORD DBEMSK ;THIS 32 BIT WORD GETS SHIFTED
2655 010126 000643          BR    35$
2656 010130 005767 172504    75$:  TST   SBEMSK+2 ;LAST BIT IN THIS MASK ?
2657 010134 100405          BMI   80$  ;BR IF LAST BIT
2658 010136 004567 003154    JSR   RS,DASHL
2659 010142 002636          .WORD SBEMSK
2660 010144 000167 177454    JMP   T12B
2661 010150 105737 000302    80$:  TSTB  2#PASFLG ;FIRST PASS ?
2662 010154 001004          BNE   85$  ;BR IF SECOND
2663 010156 105237 000302    INCB  2#PASFLG ;INDICATE SECOND PASS COMING
2664 010162 000167 177414    JMP   T12A
2665 010166 016777 172460 172476 85$:  MOV   ECCDIS,2CSRADR
2666 010174 013701 000320    MOV   2#MINMEM,R1 ;RESTORE TEST ADDRESS
2667 010200 005021          CLR   (R1)+ ;CLEAR ANY DBE'S FROM TEST
2668 010202 005011          CLR   (R1)
2669 010204 005077 172462    CLR   2CSRADR ;CLEAR CSR
2670 010210 004767 003066    JSR   PC,RSTOT4 ;RESTORE REGS R0 TO R4
2671 010214 000167 172330    END12: JMP   TSTSCP
2672          ;TEST DUAL ADDRESS TEST
2673          ;
2674          ;CHECKS FOR DUAL ADDRESSING BY WRITING
2675          ;AND READING THE ADDRESS IN THE LOCATION
2676          ;AND WRITING AND READING ITS COMPLEMENT
2677 010220 122737 000013 000404 TST13: CMPE  13,2#STESTN
2678 010226 001403          BEQ   63$
2679 010230 004767 003722    JSR   PC,SEGERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 010234 000063          63          ;*****ERROR NUMBER 63*****
(1)
2680
2681 010236 005003          65$:  CLR   R3 ;R3 INDICATES FIRST PASS OR
2682 010240 010100          15$:  MOV   R1,R0 ;COMPLEMENT PASS
2683 010242 005703          TST   R3 ;IF R3= ZERO, STORE ADDRESS
2684 010244 001401          BEQ   25$ ;IN THE LOCATION
2685 010246 005100          COM   R0 ;OTHERWISE STORE COMPLEMENT
2686 010250 010021          25$:  MOV   R0,(R1)+ ;OF ADDRESS
2687 010252 020105          CMP   R1,R5 ;UNTIL HIGHEST LOC IS REACHED
2688 010254 103771          BLO   15$
2689 010256 020041          35$:  CMP   R0,-(R1) ;CHECK FOR CORRECT CONTENTS
2690 010260 001405          BEQ   45$ ;BRANCH IF OK
2691 010262 105237 000275    INCB  2#SADERR ;PROBABLY AN ADDRESSING PROB.
2692 010266 004767 003210    JSR   PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 010272 000064          64          ;*****ERROR NUMBER 64*****
(1)
2693
2694 010274 010100          45$:  MOV   R1,R0

```

```

2695 010276 162700 000002          SUB    #2,R0          ;CHECK THAT ADDRESS IS STORED AT
2696 010302 005703          TST    R3            ;LOCATION IF R3 IS 0
2697 010304 001401          BEQ    5$            ;OTHERWISE CHECK FOR
2698 010306 005100          COM    R0            ;ADDRESS COMPLEMENT.
2699 010310 020104          5$:    CMP    R1,R4
2700 010312 101361          BHI    3$            ;
2701 010314 112737 000001 000302      MOVB   #1,2#PASFLG   ;SET PASFLG FOR ERROR REPORT
2702 010322 005103          COM    R3            ;COMPLEMENT R3
2703 010324 001345          BNE    1$            ;REPEAT TEST, COMPLEMENTING ADE
2704 010326 000167 172216      END13: JMP    TSTSCP
2705
2706
2707          ;*(1) THIS TEST CHECKS THE MEMORY FOR THE CAPABILITY
2708          ;* OF HOLDING 1'S AND 0'S BY WRITING A BACKGROUND
2709          ;* OF BAKPAT AND READING IT
2710          ;*(2) MEMORY IS WRITTEN USING A BYTE AT A TIME
2711          ;*(3) STEPS 1 & 2 ARE REPEATED WITH A SWAPPED BACKGROUND PATTERN
2712
2713 010332 122737 000014 000404      TST14: CMPB   #14,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2714 010340 001403          BEQ    1$            ;
2715 010342 004767 003610          JSR    PC,SEQERR    ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 010346 000065          6$          ;*****ERROR NUMBER 65*****
(1)
2716 010350 013700 000312          1$:    MOV    2#BAKPAT,R0 ;GET THE PATTERN INTO R0
2717 010354 110021          MOVB   R0,(R1)+     ;WRITE A BYTE
2718 010356 113721 000313          MOVB   2#BAKPAT+1,(R1)+ ;WRITE THE MEMORY WITH THE WORD STORED IN BAKPAT
2719 010362 020105          CMP    R1,R5        ;COMPARE TEST LOC TO TOP + 2
2720 010364 103771          BLO    1$            ;BRANCH IF LOWER
2721
2722 010366 020041          2$:    CMP    R0,-(R1)  ;TEST THE MEMORY TO SEE IF IT CONTAINS
2723          ;* THE WORD STORED IN BAKPAT
2724 010370 001416          BEQ    8$            ;
2725 010372 062701 000002          ADD    #2,R1        ;
2726 010376 123741 000313          CMPB   2#BAKPAT+1,-(R1);CHECK FOR BYTE SELECTION PROBLEM
2727 010402 001402          BEQ    4$            ;
2728 010404 120041          CMPB   R0,-(R1)    ;AGAIN CHECK FOR BYTE SELECTION PROBLEM
2729 010406 001002          BNE    6$            ;
2730 010410 105237 000275          4$:    INCB   2#SADERR ;PREPARE TO INFORM THAT IT IS ADDRESSING ERROR
2731 010414 042701 000001          6$:    BIC    #1,R1    ;MAKE THE ADDRESS IN R1 EVEN
2732 010420 004767 003056          JSR    PC,ERROR    ;*ERROR* REPORT ERROR MESSAGE
(1) 010424 000066          6$          ;*****ERROR NUMBER 66*****
(1)
2733 010426 020104          8$:    CMP    R1,R4    ;KEEP ON TESTING THE MEMORY UNTIL
2734 010430 101356          BHI    2$            ;R1 EQUALS THE LOWEST ADDRESS
2735 010432 000337 000312          SWAB   2#BAKPAT    ;CHANGE THE DATA PATTERN
2736 010436 001744          BEQ    1$            ;IF THE DATA PATTERN DOES NOT HAVE LOW
2737          ;* BYTE =0 THEN FALL THRU
2738 010440 000167 172104      END14: JMP    TSTSCP
2739
2740
2741          ;*THIS TEST IS CONCERNED ONLY WITH THE INTEGRITY
2742          ;*OF THE 4K MOS RAMS THAT STORE THE CHECKBITS.
2743          ;*USING DIAGA (BIT 2) AND BIT 13 TO PROTECT THE
2744          ;*PROGRAM IF NECESSARY, THE CHECKBITS ARE WRITTEN
2745          ;*AND READ VIA THE CSR.
2746

```

```

2747
2748 010444 122737 000015 003404 TST15:  CMPB    #15,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2749 010452 001403                BEQ     1$
2750 010454 004767 003022                JSR    PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 010460 000067                67     ;*****ERROR NUMBER 67*****
(1)
2751 010462 010446                1$:    MOV    R4, -(SP) ;SAVE R4
2752 010464 016703 172164                MOV    DIAGA,R3 ;DIAGNOSTIC CHECK MODE
2753 010470 052703 004000                BIS    #4000,R3 ;CHECK BIT CT
2754 010474 013701 000320                MOV    #MINMEM,R1 ;FIRST TEST LOC
2755 010500 010146                MOV    R1, -(SP) ;SAVE IT FOR LATER
2756 010502 005004                CLR    R4
2757 010504 010377 172162                MOV    R3,2CSRADR ;SET UP CSR
2758 010510 010411                2$:    MOV    R4,(R1) ;WRITE CHECKBITS BY WRITING LOC
2759 010512 012761 100000 000002                MOV    #100000,2(R1) ;SET BIT 31 TO MATCH CHECKBITS
2760 010520 062701 000004                ADD    #4,R1 ;POINT TO NEXT WORD
2761 010524 020105                CMP    R1,R5 ;TOP + 2
2762 010526 103770                BLC    2$ ;BR IF NOT
2763 010530 162701 000004                3$:    SUB    #4,R1 ;ADJUST R1
2764 010534 005711                TST   (R1) ;READ
2765 010536 032777 003740 172126                BIT    #3740,2CSRADR ;SHOULD BE ALL ZEROS
2766 010544 001421                BEQ    4$ ;BR IF OK
2767 010546 012700 004000                MOV    #4000,R0 ;GOOD DATA
2768 010552 004767 005036                JSR    PC,BITCHK ;USE XOR TO COUNT BAD CHECKBITS
2769 010556 032777 004000 172110                BIT    #4000,2SWR ;ECC DIS ?
2770 010564 001004                BNE    33$ ;PRINT SINGLE ERRORS
2771 010566 022767 000001 005102                CMP    #1,BITCNT ;MORE THAN 1 BAD BIT ?
2772 010574 001405                BEQ    4$ ;YES
2773 010576 012700 004000                33$:   MOV    #4000,R0 ;FOR ERROR MSG
2774 010602 004767 002674                JSR    PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 010606 000070                70     ;*****ERROR NUMBER 70*****
(1)
2775 010610 052777 003740 172054                4$:    BIS    #3740,2CSRADR ;MAKE CHECKBITS ALL 1'S EXCEPT FOR CT
2776 010616 042777 004000 172046                BIC    #4000,2CSRADR ;MAKE CT = 0
2777 010624 010411                MOV    R4,(R1) ;WRITE THE CHECKBITS
2778 010626 010461 000002                MOV    R4,2(R1) ;BOTH WORDS
2779 010632 020116                CMP    R1,(SP) ;BOTTOM YET ?
2780 010634 001335                BNE    3$ ;BR IF NO
2781 010636 010377 172030                5$:    MOV    R3,2CSRADR ;RESTORE CSR
2782 010642 005711                TST   (R1) ;READ THE LOC AGAIN
2783 010644 017700 172022                MOV    2CSRADR,R0 ;GET CSR CONTENTS INTO R0
2784 010650 042700 170037                BIC    #170037,R0 ;ONLY WANT BITS 11-5
2785 010654 022700 003740                CMP    #3740,R0 ;READ FOR ALL 1'S
2786 010660 001421                BEQ    6$ ;BR IF OK
2787 010662 012700 003740                MOV    #3740,R0 ;FOR ERROR MSG
2788 010666 004767 004722                JSR    PC,BITCHK
2789 010672 032777 004000 171774                BIT    #4000,2SWR ;ECC DIS ?
2790 010700 001004                BNE    55$
2791 010702 022767 000001 004766                CMP    #1,BITCNT ;MORE THAN ONE ?
2792 010710 001405                BEQ    6$ ;NO
2793 010712 012700 003740                55$:   MOV    #3740,R0 ;FOR MSG
2794 010716 004767 002560                JSR    PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 010722 000071                71     ;*****ERROR NUMBER 71*****
(1)
2795 010724 010377 171742                6$:    MOV    R3,2CSRADR ;RESTORE CSR
2796 010730 010411                MOV    R4,(R1) ;ACCESS THE LOC
    
```

2797	010732	012761	100000	000002		MOV	#100000,2(R1)	: SECOND WORD
2798	010740	062701	000004			ADD	#4,R1	: POINT TO NEXT ADDRESS
2799	010744	020105				CMP	R1,R5	: TOP + 2 YET?
2800	010746	103733				BLO	5\$	: BR IF NOT
2801	010750	011601				MOV	(SP),R1	: RESTORE MINMEM
2802	010752	010377	171714		7\$:	MOV	R3,2CSRADR	: RESTORE CSR
2803	010756	005711				TST	(R1)	: READ
2804	010760	032777	003740	171704		BIT	#3740,2CSRADR	: SHOULD BR 0
2805	010766	001421				BEQ	8\$	
2806	010770	012700	004000			MOV	#4000,R0	: GOOD DATA
2807	010774	004767	004614			JSR	PC,BITCHK	: CHECK # OF BAD BITS
2808	011000	032777	004000	171666		BIT	#4000,2SWR	: ECC DIS ?
2809	011006	001004				BNE	77\$	: YES
2810	011010	022767	000001	004660		CMP	#1,BITCNT	: MORE THAN ONE
2811	011016	001405				BEQ	8\$	: NO
2812	011020	012700	004000		77\$:	MOV	#4000,R0	: FOR PRINTOUT
2813	011024	004767	002452			JSR	PC,ERROR	: *ERROR* REPORT ERROR MESSAGE
(1)	011030	000072				72		: *****ERROR NUMBER 72*****
(1)								
2814	011032	052777	003740	171632	8\$:	BIS	#3740,2CSRADR	: COMP CHECKBITS
2815	011040	042777	004000	171624		BIC	#4000,2CSRADR	: CLEAR CT
2816	011046	010411				MOV	R4,(R1)	: WRITE
2817	011050	010461	000002			MOV	R4,2(R1)	
2818	011054	062701	000004			ADD	#4,R1	: NEXT WORD
2819	011060	020105				CMP	R1,R5	: TOP + 2 ?
2820	011062	103733				BLO	7\$	: BR IF NO
2821	011064	010377	171602			MOV	R3,2CSRADR	: RESTORE CSR
2822	011070	162701	000004		9\$:	SUB	#4,R1	: ADJUST R1
2823	011074	005711				TST	(R1)	: READ
2824	011076	017700	171570			MOV	2CSRADR,R0	: GET CSR CONTENTS
2825	011102	042700	170037			BIC	#170037,R0	: ONLY WANT CHECKBITS
2826	011106	022700	003740			CMP	#3740,R0	: ALL ONES ?
2827	011112	001421				BEQ	10\$	: BR IF OK
2828	011114	012700	003740			MOV	#3740,R0	: FOR ERROR MSG
2829	011120	004767	004470			JSR	PC,BITCHK	: COUNT BAD BITS
2830	011124	032777	004000	171542		BIT	#4000,2SWR	: ECC DIS
2831	011132	001004				BNE	99\$	: YES
2832	011134	022767	000001	004534		CMP	#1,BITCNT	: MORE THAN ONE ?
2833	011142	001405				BEQ	10\$	: NO
2834	011144	012700	003740		99\$:	MOV	#3740,R0	: FOR MSG.
2835	011150	004767	002326			JSR	PC,ERROR	: *ERROR* REPORT ERROR MESSAGE
(1)	011154	000073				73		: *****ERROR NUMBER 73*****
(1)								
2836	011156	010377	171510		10\$:	MOV	R3,2CSRADR	: RESTORE CSR
2837	011162	010411				MOV	R4,(R1)	: WRITE NEW CHECKBITS
2838	011164	012761	100000	000002		MOV	#100000,2(R1)	: BIT 31
2839	011172	020116				CMP	R1,(SP)	: BOTTOM YET ?
2840	011174	001335				BNE	9\$	: BR IF NOT
2841	011176	005711			11\$:	TST	(R1)	: RERAD
2842	011200	032777	003740	171464		BIT	#3740,2CSRADR	: SHOULD BE ALL 0'S
2843	011206	001421				BEQ	12\$	: BR IF OK
2844	011210	012700	004000			MOV	#4000,R0	: GOOD DATA
2845	011214	004767	004374			JSR	PC,BITCHK	: COUNT BAD BITS
2846	011220	032777	004000	171446		BIT	#4000,2SWR	: ECC DIS ?
2847	011226	001004				BNE	111\$	: YES
2848	011230	022767	000001	004440		CMP	#1,BITCNT	: MORE THAN 1

```

2849 011236 001405
2850 011240 012700 004000
2851 011244 004767 002232
(1) 011250 000074
(1)
2852 011252 062701 000004
2853 011256 011377 171410
2854 011262 020105
2855 011264 103744
2856 011266 005726
2857 011270 012604
2858 011272 000167 171252
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873 011276 122737 000016 000404 TST16: CMPB #16,2#STESTN ;CHECK FOR PROPER TEST SEQUENCE
2874 011304 001403 BEQ 1$
2875 011306 004767 002644 JSR PC,SEQERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 011312 000075 ;*****ERROR NUMBER 75*****
(1)
2876 011314 004737 000120 1$: JSR PC,2#WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
2877 ;WORD STORED IN BAKPAT
2878 011320 020041 2$: CMP RO,-(R1) ;READ THE CONTENTS OF LOCATION POINTED BY R1
2879 011322 001403 BEQ 3$ ;TO SEE IF IT HAS THE SAME VALUE AS RO
2880 011324 004767 002152 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 011330 000076 ;*****ERROR NUMBER 76*****
(1)
2881 011332 000300 3$: SWAB RO
2882 011334 010011 MOV RO,(R1) ;SWAP THE BYTES AT (R1)
2883 011336 021100 CMP (R1),RO ;READ (R1) FOR CORRECT VALUE
2884 011340 001403 BEQ 4$
2885 011342 004767 002134 JSR PC,ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 011346 000077 ;*****ERROR NUMBER 77*****
(1)
2886
2887 011350 000300 4$: SWAB RO ;SWAP THE BYTES OF THE REGISTER
2888 ;CONTAINING BACKGROUND PATTERN
2889 011352 001023 BNE 9$ ;IF THE LOWER BYTE OF THE REGISTER
2890 ;IS NOT 0 THEN THE PROGRAM IS READING
2891 ;THE MEMORY TO CONTAIN A BACK GROUND OF
2892
2893 ;BAKPAT AND WRITING THE SWAPPED WORD
2894
2895 ;IN WHICH CASE GO TO 9$
2896

```

```

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

```



## E05

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-55  
 DZMMLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEQ 0057

```

2897 011354 005703      5$:   TST   R3      ;R3 WAS 0 WHEN THE PROGRAM ENTERED
2898                                     ;THIS TEST, AND IT IS NOT ALTERED UNTIL PASFLG=3
2899                                     ;IF R3 EQUAL 0 THEN THE PROGRAM IS
2900                                     ;READING/WRITING MIN. TO MAX. OTHERWISE
2901                                     ;IT IS GOING IN MAX. TO MIN. DIRECTION
2902 011356 001023      6$:   BNE   10$      ;IF R3 IS NOT CLEAR THEN GO TO 10$
2903 011360 062701 000002   ADD   #2,R1      ;OTHERWISE ADD 2 TO THE CONTENTS OF R1
2904 011364 020105      CMP   R1,R5      ;COMPARE R1 WITH THE MAX. MEMORY LOCATION TO
2905                                     ;BE TESTED
2906 011366 103006      7$:   BHIS  8$      ;IF R1>R5 THEN GO TO 8$ OTHERWISE
2907 011370 020011      CMP   R0,(R1)    ;READ (R1) FOR THE CORRECT DATA
2908 011372 001757      BEQ   3$      ;WRITE COMPLEMENT OF THE DATA FOUND AT (R1)
2909                                     ;AND REPEAT UNTIL R1 > R5
2910 011374 004767 002102   JSR   PC,ERROR   ;*ERROR* REPORT ERROR MESSAGE
2911 (1) 011400 000100      100                                     ;*****ERROR NUMBER 100*****
2912 (1)
2911 011402 000753      8$:   BR    3$      ;IF THE LOWER BYTE OF R0 IS ALL 0'S
2912 011404 105237 000302   INCB  #PASFLG    ;THEN BEGIN READING BAKPAT SWAPPED WRITING BAKPAT
2913 011410 000300      SWAB  R0         ;AND READING BAKPAT GOING FROM MAX. TO MIN. (PASFLG=4)
2914 011412 001742      BEQ   2$      ;OTHERWISE CLEAR R0
2915                                     ;PUT THE LOWEST TESTING ADDRESS IN R1
2916                                     ;AND BEGIN READING 0'S, WRITING 1'S AND
2917 011414 005103      COM   R3         ;READING 1'S IN MIN. TO MAX. DIRECTION (PASFLG=3)
2918 011416 010401      MOV   R4,R1
2919 011420 000763      BR    7$
2920
2921
2922 011422 005703      9$:   TST   R3      ;IF R3 IS NON 0, I.E. PASFLG=3
2923 011424 001353      BNE   5$      ;THEN READ BAKPAT, WRITE
2924                                     ;SWAPPED BAKPAT AND READ SWAPPED BAKPAT
2925                                     ;IN MIN. TO MAX. DIRECTION
2926 011426 020104      10$:  CMP   R1,R4   ;OTHERWISE TEST IS PROCEEDING IN MAX. TO
2927                                     ;MIN. DIRECTION.
2928 011430 101333      BHI   2$      ;KEEP ON LOOPING UNTIL R1=R4
2929 011432 105237 000302   INCB  #PASFLG
2930 011436 000300      SWAB  R0
2931 011440 001753      BEQ   7$      ;IF R0 SWAPPED HAS LOWER BYTE=0
2932                                     ;THEN READ BAKPAT SWAPPED, WRITE BAKPAT,
2933                                     ;AND READ BAKPAT GOING FROM MIN. TO MAX.
2934 011442 000167 171102   END16: JMP   TSTSCP
2935
2936
2937 ;*(1) THIS TEST WRITES THE MEMORY WITH A BACK GROUND OF BAKPAT
2938 ;*(2) WITH PASFLG=0 THE TEST READS THE MEMORY FOR BAKPAT
2939 ;* AND THEN INCREMENTS PASFLG
2940 ;*(3) IT THEN READS/SHAPS BYTES/WRITES A LOCATION X FOR
2941 ;* OVER 2 MSEC AND THEN READS THE MEMORY FOR BAKPAT
2942 ;*(4) REPEATS STEP 3 WITH X=X+4K UNTIL END OF MEMORY IS ENCOUNTERED
2943 ;*(5) IT THEN INCREMENTS PASFLG AND WRITES THE MEMORY TO
2944 ;* BAKPAT AND WITH PASFLG=2 IT READS MEMORY FOR ALL
2945 ;* SWAPPED BAKPAT AFTER WHICH PASFLG IS INCREMENTED TO 3
2946 ;*(6) REPEATS STEPS 3 AND 4 READING THE MEMORY FOR SWAPPED
2947 ;* BAKPAT INSTEAD OF BAKPAT.
2948 011446 122737 000017 000404 TST17: CMPB  #17,#$TESTN ;CHECK FOR PROPER TEST SEQUENCE
2949 011454 001403      BEQ   +10
2950 011456 004767 002474      JSR   PC,SEGERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT

```

DZMLL MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-56  
 DZMLLA.P11 23-JUN-77 10:17 BEGIN OF AREA TESTED (+20) WHEN PROGRAM RELOCATES.

SEG 0058

```

(1) 011462 000101          101          ;*****ERROR NUMBER 101*****
(1)
2951 011464 004737 000120  RPT17: JSR    PC, @WRTMEM ;GO TO WRITE THE MEMORY WITH A BACKGROUND OF THE
2952                                     ;WORD STORED AT LOCATION BAKPAT
2953 011470 005037 000302          CLR    @PASFLG
2954 011474 010403          1$: MOV   R4, R3 ;SET R3
2955 011476 010401          2$: MOV   R4, R1 ;AND R1 TO THE STARTING ADDRESS OF MEMORY UNDER TEST
2956 011500 020011          3$: CMP   R0, (R1) ;CHECK (R1) FOR CORRECT DATA
2957 011502 001403          BEQ   4$
2958 011504 004767 001772  JSR    PC, ERROR ;*ERROR* REPORT ERROR MESSAGE
(1) 011510 000102          102          ;*****ERROR NUMBER 102*****
(1)
2959 011512 062701 000002  4$: ADD   #2, R1 ;INCREMENT R1 BY 2
2960 011516 020105          CMP   R1, R5 ;SEE IF R1 HAS REACHED THE MAX. OF MEMORY
2961 011520 103767          BLO   3$
2962 011522 132737 000001 000302 BITB  #1, @PASFLG ;CHECK TO SEE IF PASFLG=0 OR 2
2963 011530 001002          BNE   5$
2964 011532 105237 000302  INCB  @PASFLG ;IN WHICH CASE INCREMENT PASFLG COUNTER BY 1
2965
2966 011536 020305          5$: CMP   R3, R5 ;SEE IF R3 HAS REACHED THE MAX. OF THE MEMORY
2967 011540 103012          BHIS  7$
2968 011542 012702 037776  MOV   #37776, R2 ;WRITE INTO 1 LOC FOR >2MS (ABOUT 100MS)
2969 011546 000313          6$: SWAB (R3)
2970 011550 005302          DEC   R2
2971 011552 001375          BNE   6$
2972 011554 010337 000350  MOV   R3, @SAVLOC ;SAVE LOCATION WRITTEN FOR 2MS FOR ERROR REPORT.
2973 011560 062703 020000  ADD   #20000, R3 ;BY ADDING 1 TO THE 14TH ADDRESS BIT CAUSE
2974                                     ;R3 TO POINT TO A LOCATION IN THE NEXT
2975                                     ;4K BANK OF MEMORY
2976 011564 000744          BR    2$
2977 011566 105237 000302  7$: INCB @PASFLG ;MAKE PASFLG=2
2978 011572 000337 000312  SWAB @BAKPAT ;IF BAKPAT IS NOT BEING SWAPPED FOR THE 2ND
2979 011576 001732          BEQ   RPT17 ;THEN GO BACK TO THE LOCATION RPT17
2980 011600 000167 170744  END17: JMP  TSTSCP
2981
2982
2983
2984
2985
2986
2987
2988

```

```

2993 011604 005077 171062 RELOC: CLR @CSRADR ;
2994 011610 012737 000377 000312 MOV #377,@#BAKPAT ;
2995 011616 105737 000272 TSTB @#MMAVA ; IS THE MEMORY MANAGEMENT BEING TESTED ?
2996 011622 001170 BNE CONTMM ; IF SO THEN GO TO CONTMM AND CONTINUE TESTING
2997 ; MEMORY MANAGEMENT
2998 011624 005767 171030 TST LDPLG ; ONLY RELOCATE IF IN MEM UNDER TEST
2999 011630 001542 BEQ CKDONE ; DON'T RELOC, BRANCH
3000 011632 032777 001000 171034 BIT #1000,@SWR ; RELOCATION WANTED?
3001 011640 001136 BNE CKDONE ; BRANCH IF NO
3002 011642 005767 171022 TST INHREL ; RELOCATION ALLOWED ?
3003 011646 001133 BNE CKDONE ; NO, SINGLE ERROR IN LOC 0-500, BRANCH
3004 011650 105737 000352 TSTB @#REL ; IF THE PROGRAM HAS ALREADY BEEN RELOCATED THEN ALSO
3005 011654 100473 BMI RELOER ; PLACE THE PROGRAM BACK IN LOWER CORE
3006 011656 112737 000200 000352 MOVB #200,@#REL ; OTHERWISE PREPARE TO RELOCATE
3007
3008 ;RELOCATE THE DIAGNOSTIC TO SECOND 16K
3009
3010
3011 011664 012701 100500 MOV #100000+BEGIN,R1 ; START OF RELOCATED VERSION
3012 011670 012702 116074 MOV #100000+ENDPROG,R2 ; END
3013 011674 005711 3$: TST (R1) ; READ A WORD
3014 011676 032777 100000 170766 BIT #100000,@CSRADR ; CHECK FOR DBE
3015 011704 001005 BNE 5$ ; BR IF ERROR
3016 011706 062701 000004 ADD #4,R1 ; NEXT WORD
3017 011712 020102 CMP R1,R2 ; END YET ?
3018 011714 103767 BLO 3$ ; BR IF NOT
3019 011716 000405 BR 4$ ; BR IF ALL IS OK
3020 011720 105037 000352 5$: CLRB @#REL ; INDICATE NO RELOC
3021 011724 004767 002226 JSR PC,FATERR ; *ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 011730 000103 103 ; *****ERROR NUMBER 103*****
(1)
3022 011732 032777 000040 170734 4$: BIT #40,@SWR ; PRINTING WANTED
3023 011740 001005 BNE 1$ ; BR IF NO
3024 011742 004767 002434 JSR PC,PNTMES ; TYPE "RELOC"
3025 011746 042522 047514 000103 .ASCIZ /RELOC/
3026 .EVEN
3027 011754 004767 003270 1$: JSR PC,CLMM ; CLEAR MM REGS
3028 011760 012737 000500 000J20 MOV #BEGIN,@#MINMEM ; NEW TEST ADDRESS
3029 011766 052767 000010 170656 BIS #10,ECCDIS ; PROTECT 2ND 16K
3030 011774 052767 000010 170652 BIS #10,DIAGA ; DITTO
3031 012002 012705 100000 MOV #100000,R5 ; START OF 2ND 16K
3032 012006 010567 170650 MOV R5,BASE ; BASE FOR RELOC
3033 012012 012704 000430 MOV #BEGIN-50,R4 ; FIRST LOC TO BE RELOCATED
3034 012016 060405 ADD R4,R5 ;
3035 012020 012425 2$: MOV (R4)+(R5)+ ; MOV A WORD
3036 012022 020437 000342 CMP R4,@#SAVR4 ; LAST LOC YET ?
3037 012026 103774 BLO 2$ ; BR IF NO
3038 012030 012704 000430 MOV #BEGIN-50,R4 ; FIRST TEST ADDRESS
3039 012034 016705 170622 MOV BASE,R5 ;
3040 012040 000137 100500 JMP @#BEGIN+100000 ; JUMP TO RELOCATED VERSION
3041
3042 ;*RELOCATE THE DIAGNOSTIC BACK TO LOWER MEMORY
3043
3044
3045 012044 105737 000352 RELOER: TSTB @#REL ; IS DIAGNOSTIC IN RELOCATED STATE?
3046 012050 100032 BPL CKDONE ; BRANCH IF NO

```

DZML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-58  
 DZMLA.P11 23-JUN-77 10:17 RELOCATE

3047	012052	005067	170604		CLR	BASE	
3048	012056	042767	000010	170566	BIC	#10,ECCDIS	
3049	012064	042767	000010	170562	BIC	#10,DIAGA	
3050							
3051	012072	016737	170566	000320	MOV	SAVMIN,#MINMEM	;RESTORE TEST ADDRESS FOR ECC
3052	012100	012704	000430		MOV	#BEGIN-50,R4	;PREPARE TO RELOCATE THE PROGRAM TO LOWER CORE
3053	012104	010405			MOV	R4,R5	
3054	012106	062705	100000		ADD	#100000,R5	
3055	012112	012524			MOV	(R5)+,(R4)+	
3056	012114	020437	000342	2\$:	CMP	R4,#SAVR4	
3057	012120	103774			BLO	2\$	
3058	012122	105037	000352		CLRB	#REL	
3059	012126	012706	000500		MOV	#BEGIN,SP	;RESET STACK TO LOWER MEMORY
3060	012132	010637	000346		MOV	SP,#SAVR6	; "BEGIN" USES THIS TO RESET THE STACK.
3061	012136	000137	012142	CKDONE:	JMP	#LOWER	;TRANSFER THE PROGRAM CONTROL TO THE LOWER CORE
3062							
3063							
3064							
3065	012142	105737	000402	LOWER:	TSTB	#SFATAL	;FATAL ERROR
3066	012146	001405			BEQ	1\$	;BR IF NOT
3067	012150	005737	000042		TST	#42	;APT ?
3068	012154	001402			BEQ	1\$	;NO KEEP TESTING
3069	012156	000167	002040		JMP	APTHLT	
3070	012162	105737	000311	1\$:	TSTB	#SAVKBB	;HERE DUE TO ^C TYPED?
3071	012166	001141			BNE	STPSTK	;BRANCH IF YES (TYPE ERROR STACK)
3072	012170	004767	002504	TSTMM:	JSR	PC,MEMMNG	;SET THE REGISTERS IF THE MEMORY MANAGEMENT
3073							;IS AVAILABLE
3074	012174	105737	000272		TSTB	#MMAVA	;IS MEM. MANAG. AVAILABLE ?
3075	012200	001474			BEQ	ENDPAS	;BRANCH IF NO
3076	012202	000406			BR	SCNTMM	;BEGIN TESTING ABOVE 28K
3077	012204	022737	007400	172352	CONTMM:	CMP	#7400,#172352
3078	012212	001464			BEQ	ENDMAX	;DON'T WANT TO WRAP AROUND
3079	012214	004767	002636		JSR	PC,UPMM	;GO TO END-OF-PASS
3080	012220	012703	000322	SCNTMM:	MOV	#LOWTWO,R3	;GO TO UPDATE MEM. MANAG. REGISTERS
3081	012224	004767	002744		JSR	PC,GETSIZ	;MAKE R3 POINT TO THE LOCATION LOWTWO
3082							;LOAD BITS 6-10 OF R2 WITH THE BITS 13-17
3083	012230	012704	040000		MOV	#40000,R4	;OF THE LOWEST ADDRESS UNDER TEST
3084							;MAKE R4 POINT TO THE LOWEST LOCATION IN THE BANK
3085	012234	020237	172344		CMP	R2,#172344	;POINTED BY PAGE ADDRESS REGISTER 2 (PAR2)
3086							;IS THE CONTENT OF R2 LOWER THAN THE CONTENT OF
3087	012240	103405			BLO	2\$	;PAR2 ?
3088	012242	050104			BIS	R1,R4	;IF SO THEN GO 2\$
3089							;SUBROUTINE GETSIZ LOADED R1 WITH BITS 0-12
3090	012244	162702	000200		SUB	#200,R2	;OF LOWADD WHICH HAVE NOW BEEN LOADED IN R4
3091	012250	004767	002430		JSR	PC,MAREG	
3092	012254	010437	000320	2\$:	MOV	R4,#MINMEM	;SET MEM. MANAG. REGISTERS
3093	012260	004767	002710		JSR	PC,GETSIZ	;NEW MINMEM FOR ECC TESTS
3094							;PLACE BITS 13-17 OF HIGHEST LOCATION TO BE TESTED
3095							;IN BITS 6-10 OF R2, #160000 IN R0 AND BITS 0-12
3096	012264	004767	000020		JSR	PC,MAXADR	;OF LOCATION HIGHADD IN R1
3097	012270	010005			MOV	R0,R5	;GET THE ADDRESS OF MAX. MEM. UNDER TEST
3098	012272	004767	002676		JSR	PC,GETSIZ	
3099	012276	004767	000006		JSR	PC,MAXADR	;PREPARE TO SET UP LOCATION MAXMEM
3100	012302	010013			MOV	R0,(R3)	;GET THE MAXIMUM ADDRESS OF AVAILABLE MEMORY
3101	012304	000167	167664		JMP	CLAMEM	;AND STORE INTO "MAXMEM"
3102							;GO TEST A 20K SLICE ABOVE 28K.

```

3103
3104
3105
3106
3107
3108
3109
3110 012310 010046
3111 012312 012700 172356
3112
3113 012316 162716 020000
3114 012322 050116
3115 012324 020340
3116 012326 001411
3117 012330 020027 172340
3118 012334 101370
3119
3120 012336 062700 000004
3121 012342 021002
3122 012344 003006
3123 012346 012716 157776
3124 012352 012600
3125 012354 062700 000002
3126 012360 000207
3127
3128 012362 022626
3129
3130 012364 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 PAR?
    BHI 2$ ;NO KEEP TRYING
; **END LOOP TO FIND PAR ADDRESS UNDER TEST
    ADC #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

```



K05

DZML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-61  
DZMLA.P11 23-JUN-77 10:17 TYPE ROUTINE FOR ERROR STACK

SEQ 0063

3191 012606 012703 000376  
3192 012612 000743

MOV #376,R3  
BR RETSTK

;RESET SCRATCH STACK FOR EACH BIT PRINTED.

```

3197
3198
3199
3200
3201
3202
3203
3204
3205
3206
3207
3208 012614 005002 SEOP: CLR R2 ;SET R2= PARITY MODULE DISABLE CODE
3209 012616 105737 000311 YSTB 2#SAVKBB ;CONTROL-C TYPED?
3210 012622 001050 BNE CTLC ;BRANCH IF YES-RESTORE LOADERS AND HALT-
3211 012624 005237 000406 INC 2#SPASS ;INCREMENT PASS COUNT
3212 012630 005067 170032 CLR ERRFLG ;RESET ERROR HEADER FOR NEXT PASS
3213 012634 032777 000040 170032 BIT 4#JSWR ;"END PASS #XX" PRINTOUT WANTED?
3214 012642 001015 BNE ACT11 ;BRANCH IF NO
3215 012644 004767 001524 TYPEOP: JSR PC,TPCALF ; TYPE CR, LF, AND "END PASS #"
3216 012650 047105 020104 040520 .ASCIZ /END PASS #/
3217 012656 051523 021440 000 .EVEN
3218 012664 013737 000406 000306 MOV 2#SPASS,2#DECWRD ;GET PASS COUNT
3219 012672 004767 001532 JSR PC,STPDEC ;TYPE IT
3220 012676 013700 000042 ACT11: MOV 2#42,R0 ;GET THE MONITOR ADDRESS
3221 012702 001405 BEQ $DOAGN ;IF NONE
3222 012704 004767 000012 JSR PC,RLODER ;RESTORE XXDP MONITOR
3223 012710 000005 RESET ;RETURN TO ACT11 MONITOR.
3224
3225
3226 ;* SERVICE XXDP/ACT11
3227 012712 000137 000156 JMP 2#$ENDAD ;JUMP TO ACT SERVICE
3228
3229 012716 000137 000250 $DOAGN: JMP 2#RESTR ;REPEAT TEST IF NOT UNDER ACT11/XXDP
3230
3231 012722 004767 002322 RLODER: JSR PC,CLMM ;STOP MEMORY MANAGEMENT SO CAN RESTORE LOADERS
3232 012726 013704 000342 MOV 2#SAVR4,R4 ;RESTORE R4 WITH SAVR4
3233 012732 014445 4$: MOV -(R4) -(R5) ;RESTORE LOADERS
3234 012734 020437 000304 CMP R4,2#ENDSTK
3235 012740 101374 BHI 4$
3236 012742 000207 RTS PC ;RETURN FROM RLODER CALL
3237
3238 ;CONTROL C HANDLER
3239
3240 012744 004767 177752 CTLC: JSR PC,RLODER ;RESTORE ABS LOADER
3241 012750 000167 001246 JMP APTHLT ;IF NOT APT HALT AT FATHLT
3242
3243
3244 ;*****
3245
3246 ;CHECK BIT GENERATOR ROUTINE
3247 ;CALL: JSR PC,CHKGEN
3248
3249 ; SOURCE = ADDRESS OF DATA
3250 ; CHECK = WORD CONTAINING GENERATED CHECKBITS
3251

```



3252	012754	000000	
3253	012756	000000	
3254	012760	125252	
3255	012762	125252	
3256	012764	146314	
3257	012766	146314	
3258	012770	170360	
3259	012772	170360	
3260	012774	177777	
3261	012776	177400	
3262	013000	000377	
3263	013002	177777	
3264	013004	177400	
3265	013006	177777	
3266	013010	064551	
3267	013012	113151	
3268			
3269	013014	000	
3270			
3271	013015	000	
3272	013016	000	
3273	013017	000	
3274	013020	000	
3275	013021	000	
3276	013022	000	
3277	013023	000	
3278	013024	077	
3279			
3280			
3281			
3282	013026		
3283	013026	004467	000236
3284	013032	005000	
3285	013034	012701	013015
3286	013040	066701	167616
3287	013044	022700	000007
3288	013050	001403	
3289	013052	105021	
3290	013054	005200	
3291	013056	000772	
3292	013060	012701	000001
3293	013064	012702	012760
3294	013070	066702	167566
3295	013074	012703	013015
3296	013100	066703	167556
3297	013104	012704	000006
3298	013110	105067	177700
3299	013114	016700	177634
3300	013120	066700	167536
3301	013124	030110	
3302	013126	001403	
3303	013130	030112	
3304	013132	001401	
3305	013134	105213	
3306	013136	062700	000002
3307	013142	062702	000002

```

SOURCE: .WORD 0
CHECK: .WORD 0
CHKTAB: .WORD 125252 ;C1 BIT TABLE
        .WORD 125252 ;
        .WORD 146314 ;C2 BIT TABLE
        .WORD 146314 ;
        .WORD 170360 ;C4 BIT TABLE
        .WORD 170360 ;
        .WORD 177777 ;C8 BIT TABLE
        .WORD 177400 ;
        .WORD 377 ;C16 BIT TABLE
        .WORD 177777 ;
        .WORD 177400 ;C32 BIT TABLE
        .WORD 177777 ;
        .WORD 64551 ;CT BIT TABLE
        .WORD 113151 ;
        ;

UPPFLG: .BYTE 0

CHKCNT: .BYTE 0 ;C1 PARITY COUNTER
        .BYTE 0 ;C2 PARITY COUNTER
        .BYTE 0 ;C4 PARITY COUNTER
        .BYTE 0 ;C8 PARITY COUNTER
        .BYTE 0 ;C16 PARITY COUNTER
        .BYTE 0 ;C32 PARITY COUNTER
        .BYTE 0 ;CT PARITY COUNTER
PARBYT: .BYTE 77 ;ODD/EVEN PARITY FLAGS
        ;0=EVEN
        ;1=ODD
        ;CT C32 C16 C8 C4 C2 C1

CHKGEN: JSR R4, SAV0T4 ;SAVE R0 TO R4
        CLR R0 ;CLEARING OUT PARITY COUNTERS
        MOV #CHKCNT, R1 ;START 7 COUNTERS
        ADD BASE, R1
10$: CMP #7, R0 ;FINISHED?
     BEQ 20$ ;BRANCH IF YES
     CLRB (R1)+ ;NO, DO THE NEXT BYTE
     INC R0 ;BUMP THE COUNT
     BR 10$ ;LOOP
20$: MOV #1, R1 ;SET THE TEST BIT
30$: MOV #CHKTAB, R2 ;START OF CHECK BIT TABLE
     ADD BASE, R2
     MOV #CHKCNT, R3 ;START ADDRESS OF PARITY COUNTERS
     ADD BASE, R3
     MOV #6, R4 ;FOR SEVEN PARITY COUNTERS
40$: CLRB UPPFLG ;INDICATE LOWER WORD
     MOV SOURCE, R0 ;SET ADDRESS OF DATA IN R0
     ADD BASE, R0
42$: BIT R1, (R0) ;CHECK LOWER 16 BITS FOR TEST BIT
     BEQ 50$ ;BRANCH IF NOT SET
     BIT R1, (R2) ;LOOK FOR TEST BIT IN CHECK BIT TABLE
     BEQ 50$ ;BRANCH IF NOT FOUND
     INCB (R3) ;FOUND A MATCH, COUNT IT
50$: ADD #2, R0 ;R0 NOW = SOURCE +2
     ADD #2, R2 ;POINT TO SECOND WORD IN TABLE

```

N05

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-64  
 DZMMLA.P11 23-JUN-77 10:17 END OF PASS

SEQ 0066

3308	013146	105767	177642	TSTB	UPPFLG	: CHECK FOR UPPER/LOWER
3309	013152	001003		BNE	60\$	: BRANCH IF UPPER
3310	013154	105267	177634	INCB	UPPFLG	: CHANGE FLAG TO UPPER
3311	013160	000761		BR	42\$	: CHECK SECOND 16 BITS
3312	013162	005704		60\$: TST	R4	: LOOK FOR LAST PARITY COUNTER
3313	013164	001403		BEQ	70\$	: BRANCH IF DONE
3314	013166	005203		INC	R3	: ADVANCE PARITY COUNTER POINTER
3315	013170	005304		DEC	R4	: ADVANCE CHECK BIT FLAG COUNTER
3316	013172	000746		BR	40\$	: GO CHECK NEXT CHECK BIT IN TABLE
3317	013174	006301		70\$: ASL	R1	: SHIFT THE BIT OVER
3318	013176	103332		BCC	30\$	: ADJUST THE PARITY AND ASSEMBLE
3319						: CHECK BIT WORD
3320	013200	012700	000001	MOV	#1,R0	: SET FIRST BIT IN R1
3321	013204	012701	013015	MOV	#CHKCNT,R1	: ADDRESS OF C1 PARITY COUNTER
3322	013210	066701	167446	ADD	BASE,R1	
3323	013214	130067	177604	80\$: BITB	R0,PARBYT	: CHECKING FOR ODD OR EVEN PARITY
3324	013220	001401		BEQ	90\$	: BRANCH IF EVEN PARITY
3325	013222	105211		INCB	(R1)	: ODD, INC THE PARITY COUNTER
3326	013224	132711	000001	90\$: BITB	#1,(R1)	: TEST PARITY COUNTER FOR SET BIT
3327	013230	001401		BEQ	100\$	
3328	013232	050004		BIS	R0,R4	: SET THE CHECK BIT IN CHECK
3329	013234	005201		100\$: INC	R1	: NEXT COUNTER
3330	013236	006300		ASL	R0	: SHIFT TEST BIT TO NEXT BIT
3331	013240	032700	000200	BIT	#200,R0	: CT YET?
3332	013244	001763		BEQ	80\$	: BRANCH IF NO
3333	013246	000304		SWAB	R4	
3334	013250	006004		ROR	R4	
3335	013252	006004		ROR	R4	
3336	013254	006004		ROR	R4	: POSITION CHECK BITS IN BITS 11-5
3337	013256	010467	177474	MOV	R4,CHECK	
3338	013262	004767	000014	JSR	PC,RSTOT4	: RESTORE R0 TO R4
3339	013266	000207		RTS	PC	
3340						
3341						
3342				: CALL	JSR R4,SAVOT4	
3343	013270	010346		SAVOT4: MOV	R3,-(SP)	: SAVE R3
3344	013272	010246		MOV	R2,-(SP)	: SAVE R2
3345	013274	010146		MOV	R1,-(SP)	: SAVE R1
3346	013276	010046		MOV	R0,-(SP)	: SAVE R0
3347	013300	010407		MOV	R4,PC	: R4 IS ALREADY SAVED
3348						
3349				: CALL	JSR PC,RSTOT4	
3350	013302	012604		RSTOT4: MOV	(SP)+,R4	: RETURN ADDRESS
3351	013304	012600		MOV	(SP)+,R0	: RESTORE R0
3352	013306	012601		MOV	(SP)+,R1	: R1
3353	013310	012602		MOV	(SP)+,R2	: R2
3354	013312	012603		MOV	(SP)+,R3	: R3
3355	013314	000204		RTS	R4	: RESTORE R4 AND RETURN
3356						
3357						
3358						
3359						
3360						
3361	013316	010046		DASHL: MOV	R0,-(SP)	: SAVE R0
3362	013320	012500		MOV	(R5)+,R0	: PICK UP ARGUMENT
3363	013322	066700	167334	ADD	BASE,R0	

```

3364 013326 006360 000002      ASL      2(R0)      ;SHIFT HIGH 16 BITS
3365 013332 006310              ASL      (R0)      ;SHIFT LOW 16 BITS
3366 013334 103002              BCC      10$      ;BR IF LOW WORD BIT 15 = 0
3367 013336 005260 000002      INC      2(R0)      ;IT WAS = 1 INC HI WORD
3368 013342 012600              MOV      (SP)+,R0  ;RESTORE R0
3369 013344 000205              RTS      R5
3370
3371 013346 012567 000112      BITCOM: MOV      (R5)+,MSKADR ;FIRST ARG IS ADDR OF MASK
3372 013352 012567 000110      MOV      (R5)+,DATADR ;SECOND IS ADDR OF DATA
3373 013356 010046              MOV      R0,-(SP)
3374 013360 010146              MOV      R1,-(SP)
3375 013362 010246              MOV      R2,-(SP)      ;SAVE THESE REGS
3376 013364 066767 167272 000072      ADD      BASE,MSKADR
3377 013372 066767 167264 000066      ADD      BASE,DATADR
3378 013400 017700 000060      MOV      @MSKADR,R0 ;GET MASK
3379 013404 017701 000056      MOV      @DATADR,R1 ;GET DATA
3380 013410 004767 000054      JSR      PC,XOR      ;DO THE XOR
3381 013414 010277 000046      MOV      R2,@DATADR ;PUT RESULT BACK AS DATA
3382 013420 062767 000002 000036      ADD      @2,MSKADR ;NEXT WORD
3383 013426 062767 000002 000032      ADD      @2,DATADR ;DITTO
3384 013434 017700 000024      MOV      @MSKADR,R0
3385 013440 017701 000022      MOV      @DATADR,R1
3386 013444 004767 000020      JSR      PC,XOR      ;DO THE SAME FOR SECOND WORD
3387 013450 010277 000012      MOV      R2,@DATADR ;PUT RESULT BACK TOO
3388 013454 012602              MOV      (SP)+,R2
3389 013456 012601              MOV      (SP)+,R1
3390 013460 012600              MOV      (SP)+,R0 ;RESTORE THE REGS R0,R2
3391 013462 000205              RTS      R5
3392
3393
3394 013464 000000      MSKADR: .WORD 0
3395 013466 000000      DATADR: .WORD 0
3396
3397
3398
3399
3400
3401
3402
3403
3404 013470 010102      XOR:     MOV      R1,R2 ;SAVE B OPERAND
3405 013472 040002      BIC      R0,R2      ;A NOT B
3406 013474 040100      BIC      R1,R0      ;B NOT A
3407 013476 050002      BIS      R0,R2      ;A XOR B
3408 013500 000207      RTS      PC ;EXIT AND RETURN
3409
3410
;*****

```

3415  
3416  
3417  
3418  
3419  
3420  
3421  
3422  
3423  
3424  
3425  
3426  
3427  
3428

3429  
3430  
3431  
3432  
3433  
3434  
3435  
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

013502 017637 000000 000402  
013510 005767 167152  
013514 001050  
013516 032777 020000 167150  
013524 001044  
013526 004767 000642  
013532 040440 042104 020122  
013540 020040 020040 047507  
013546 042117 020040 020040  
013554 041040 042101 020040  
013562 020040 020040 050040  
013570 020103 020040 020040  
013576 051105 020122 020043  
013604 020040 050040 051501  
013612 046106 020107 020040  
013620 054040 051117 020040  
013626 005015 000  
013632 013632  
013632 005267 167030  
013636 010346  
013640 010046  
  
013642 010103  
013644 004767 001420  
013650 013703 000306  
013654 110337 000403  
  
013660 010346  
013662 012703 000376  
013666 013743 000302  
013672 005043  
013674 113713 000402  
013700 016643 000006  
013704 011143  
013706 010043  
013710 005043  
013712 016313 000004  
013716 040013  
013720 046300 000004  
013724 050013  
013726 011367 000744  
013732 012700 002112  
013736 060700  
013740 062700 000022  
013744 005316

```

* ERROR HANDLING ROUTINE
* -----
*
* PROGRAM COMES HERE EACH TIME AN ERROR IS ENCOUNTERED THIS
* ROUTINE TYPES OUT THE ERROR MESSAGE IN THE FORMAT GIVEN EARLIER
*
ERROR: MOV 2(SP), 2#SFATAL ;LOAD THE LOCATION SFATAL WITH THE ERROR NUMBER
TST ERRFLG ;FIRST ERROR ?
BNE IS ;BR IF NOT FIRST ERROR
BIT 20000, 2SWR ;ERROR PRINTOUTS DESIRED ?
BNE IS ;BR IN NO
JSR PC TPCRLF ;TYPE ERROR HEADER FOR FIRST ERROR
.ASCII / ADDR GOOD BAD PC ERR # PASFLG XOR /

.ASCIIZ <15><12>
.EVEN
INC ERRFLG ;INDICATE NOT FIRST ERROR
IS: MOV R3, -(SP) ;SAVE R3
MOV R0, -(SP) ;AND R0 ON THE STACK

;SE*JP BANK NO. IN FATAL FOR APT

MOV R1, R3 ;GET VIRTUAL ADDRESS UNDER TEST FOR GETBNK
JSR PC, GETBNK ;GET BANK NO. UNDER TEST INTO PBNK
MOV 2#PBNK, R3 ;GET BANK UNDER TEST
MOVVB R3, 2#SFATAL+1 ;STORE FAILING BANK NO. FOR APT

;
MOV R3, -(SP) ;TEMPORARILY STORE R3
MOV 376, R3 ;MAKE R3 AS THE STACK POINTER
MOV 2#PASFLG, -(R3) ;OUTPUT THE WORD STORED AT
25: CLR -(R3)
MOVVB 2#SFATAL, (R3) ;PUT ERROR NO. ON ERROR STACK
MOV 6(SP), -(R3) ;PLACE THE RETURN PC AT (R3)
MOV (R1), -(R3) ;PLACE BAD DATA
MOV R0, -(R3) ;AND GOOD DATA ON THE STACK
CLR -(R3)
MOV 4(R3), (R3) ;TAKE THE
BIC R0, (R3) ;EXCLUSIVE OR OF GOOD AND BAD DATA
BIC 4(R3), R0 ;TO FIND THE BITS THAT FAILED
BIS R0, (R3) ;AND PLACE IT ON THE STACK
MOV (R3), BOCHIP ;TO PRINT OUT XOR OF GOOD VS BAD
MOV 2#ENDPROG--24., R0 ;THIS CODE BRINGS THE RELATIVE ADDRESS
ADD PC, R0 ;OF THE STARTING OF THE ERROR STACK
65: ADD 18., R0 ;FOR THE SPECIFIC 4K BANK
DEC (SP)

```

DZMLL MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-67  
 DZMLLA.P11 23-JUN-77 10:17 ERROR HANDLING ROUTINE

SEQ 0069

```

3462 013746 002374          BGE      6$
3463 013750 005726          TST      (SP)+      ;RESTORE THE STACK POINTER
3464
3465 013752 105037 000273  ERRYP:  CLRB    2$TYPENB ;DISABLE ANY TYPE OUT
3466 013756 105737 000274  1$:    TSTB    2$SPRERR ;IF THIS IS PARITY PROBLEM
3467 013762 001007          BNE      3$        ;THEN GO TO 3$
3468 013764 105720          TSTB    (R0)+      ;OTHERWISE INCREMENT THE ERROR STACK POINTER BY 1
3469 013766 105737 000275  TSTB    2$ADERR    ;IF THIS IS ADDRESSING PROBLEM
3470 013772 001003          BNE      3$        ;THEN GO TO 3$
3471 013774 105720          TSTB    (R0)+      ;INCREMENT THE POINTER R0 BY 1
3472 013776 005713          TST      (R3)      ;IS BIT 15 OF (R3) SET?
3473 014000 100015          BPL      4$        ;IF NOT THEN GO TO 4$
3474 014002 122710 000377  3$:    CMPB    #377,(R0) ;OTHERWISE SEE IF THIS ERROR HAS OCCURED 377 TIMES
3475 014006 001401          BEQ     5$        ;IF SO DON'T BUMP ERROR COUNT
3476 014010 105210          INCB    (R0)      ;INCREMENT THE ERROR COUNTER BY 1
3477 014012 122710 000001  5$:    CMPB    #1,(R0)  ;MORE THAN 1 ERROR OCCURRED ON THIS BIT?
3478 014016 001404          BEQ     7$        ;BRANCH IF NO
3479 014020 032777 000400 166646  BIT      #400,2$SWR ;STOP ERROR PRINTOUT AFTER 1 WANTED?
3480 014026 001002          BNE     4$        ;BRANCH IF YES (DON'T TYPE ERROR)
3481 014030 105237 000273  7$:    INCB    2$TYPENB ;ENABLE THE TYPE OUT ROUTINE
3482 014034 105737 000275  4$:    TSTB    2$ADERR    ;ADDRESS ERROR?
3483 014040 001403          BEQ     6$        ;BRANCH IF NO
3484 014042 004767 000166  JSR     PC,TPADERR ;PRINT "ADR ERR"
3485 014046 000403          BR      8$
3486 014050 105720          TSTB    (R0)+      ;POINT TO NEXT ENTRY IN ERROR STACK
3487 014052 006313          ASL     (R3)      ;IS THERE STILL AN ERROR BIT SET IN ERROR.
3488 014054 001350          BNE     2$        ;BR IF YES - KEEP FILLING ERROR STACK
3489 014056 112737 000006 000310 8$:    MOVB    #6,2$TYPCNT ;TELL TYPOCT TO TYPE 6 WORDS OF ERROR STACK.
3490
3491 014064 004767 000774          JSR     PC,PUTADR  ;THE STACK POINTED BY R3
3492
3493 014070 004767 000416          JSR     PC,TYPERR  ;GO TO THE SUBROUTINE TO PLACE THE ADDRESS IN R1
3494
3495 014074 005037 000274  10$:   CLR      2$SPRERR ;CLEAR ADDRESS ERROR FLAG
3496 014100 012600          MOV     (SP)+,R0  ;RESTORE R0
3497 014102 012603          MOV     (SP)+,R3  ;AND R3
3498 014104 105737 000420  FNDERR: TSTB    2$SENV    ;ARE WE RUNNING UNDER APT?
3499 014110 001404          BEQ     2$        ;IF NOT THEN TEST FOR HALT
3500 014112 012737 000001 000400  MOV     #1,2$MSGTY ;OTHERWISE INFORM THE APT
3501 014120 000443          BR     FATHLT    ;GOTO FATHLT AND WAIT FOR APT.
3502
3503 014122 010246          MOV     R2,-(SP)  ;SAVE R2 TEMP
3504 014124 005777 166544          TST     2$SWR    ;DOES THE OPERATOR REQUIRE THE PROGRAM TO HALT
3505
3506 014130 100405          BMI     4$        ;ON ERROR
3507
3508          ;CHECK FOR CONTROL-C KEY
3509 014132 004767 001660          JSR     PC,CHECKC ;IF CONTROL-C TYPED THEN PRINT ERROR HISTORY
3510
3511 014136 105737 000042  7$:    TSTB    2$42    ;AND HALT AT FATHLT.
3512 014142 001401          BEQ     6$        ;ARE WE RUNNING UNDER ACT?
3513
3514 014144 000777          BR     4$        ;BRANCH IF NO
3515
3516
3517 014146 012602          MOV     (SP)+,R2 ;PROGRAM WAS HALTED ON ERROR, R1 IS POINTING
;TO A LOCATION WHICH SHOULD HAVE CONTAINED
;THE WORD STORED IN R0
;RESTORE R2

```

```

3518 014150 062716 000002          ADD    #2,(SP)      ;RESTORE THE RETURN ADDRESS
3519 014154 000207          RTS     PC          ;RETURN FROM THE SUBROUTINE
3520
3521
3522
3523 014156          FATERR:
3524 014156 004767 000212          SEQERR: JSR    PC,TPCRLF      ;TYPE "ERROR #"
3525 014162 051105 047522 020122          .ASCIZ  /ERROR #/
3526          .EVEN
3527
3528 014172 017637 000000 000402          MOV    2(SP),2#$FATAL      ;LOAD THE LOCATION $FATAL WITH THE ERROR NUMBER
3529 014200 105237 000310          INCB   2#TYPENT           ;TELL $TPNUM TO TYPE 1 WORD
3530 014204 012703 000376          MOV    #376,R3           ;$TPNUM USES R3 AS STACK
3531 014210 013743 000402          MOV    2#$FATAL,-(R3)     ;PUT ERROR NO. ON STACK
3532 014214 005743          TST    -(R3)             ;$TPNUM REQUIRES THIS
3533 014216 004767 000356          JSR    PC,FATYP          ;TYPE ERROR NO.
3534 014222 105737 000420          APTHLT: TSTB  2#$ENV        ;RUNNING UNDER APT?
3535 014226 001326          BNE    FNDERR           ;BRANCH IF YES
3536 014230 000240          FATHLT: NOP              ;FATAL ERROR OR IC HALT.
3537 014232 000776          BR     FATHLT           ;RESTART AT 250 IF DESIRED
3538
3539 014234 105737 000273          TPADER: TSTB  2#TYPEENB    ;
3540 014240 001406          BEQ    1$                ;BR IF NO TYPE ENABLED
3541 014242 004767 000134          JSR    PC,PNTMES        ;PRINT MESSAGE ROUTINE
3542 014246 042101 020122 051105          .ASCIZ  /ADR ERR/
3543          .EVEN
3544 014256 000207          1$:   RTS     PC
3545
3546
3547
3548          ;* TYPE OUT ROUTINE
3549          ;* -----
3550          ;*
3551          ;* THIS ROUTINE IS USED BY THE PROGRAM TO TYPE OUT ANY CHARACTER
3552          ;*
3553
3554 014260 010146          NOTYP: MOV    R1,-(SP)
3555 014262 016601 000002          MOV    2(SP),R1
3556 014266 105721          4$:   TSTB  (R1)+          ;IF THIS TYPE OUT HAS BEEN SUPRESSED THEN
3557 014270 001376          BNE    4$                ;PREPARE TO RETURN
3558 014272 000412          BR     RETTYP
3559 014274 010146          $TYPE: MOV    R1,-(SP)      ;SAVE R1
3560 014276 010046          MOV    R0,-(SP)         ;AND R0 ON THE STACK
3561 014300 016601 000004          MOV    4(SP),R1         ;PLACE THE ADDRESS OF MESSAGE TO BE TYPED IN R1
3562 014304 112100          2$:   MOVB  (R1)+,R0        ;PLACE THE BYTE TO BE TYPED IN R0
3563 014306 001403          BEQ    4$                ;IF IT IS END OF MESSAGE THEN GO TO 4$
3564 014310 004767 000022          JSR    PC,$TPCHR        ;OTHERWISE GO TO TYPE THE CONTENTS OF R0
3565 014314 000773          BR     2$
3566 014316 012600          4$:   MOV    (SP)+,R0      ;RESTORE R0
3567 014320 005201          RETTYP: INC   R1          ;CAUSE R1 TO
3568 014322 042701 000001          BIC   #1,R1             ;POINT TO EVEN ADDRESS
3569 014326 010166 000002          MOV    R1,2(SP)         ;MODIFY THE RETURN ADDRESS
3570 014332 012601          MOV    (SP)+,R1        ;RESTORE R1
3571 014334 000416          BP     EXTYP            ;AND RETURN VIA RTS PC

```

DZMML MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-69  
 DZMMLA.P11 23-JUN-77 10:17 ERROR HANDLING ROUTINE

SEG 0071

```

3572
3573 014336 132737 000040 000421 STPCHR: BITB #40,2#SENVN ;HAVE TYPE OUTS BEEN DISABLED?
3574 014344 001005 BNE 4$ ;IF SO THEN RETURN FROM THE SUBROUTINE
3575 014346 105737 177564 2$: TSTB 2#STPS ;WAIT HERE
3576 014352 100375 BPL 2$ ;UNTIL THE PRINTER IS READY
3577 014354 110037 177566 MOVB R0,2#STPB ;LOAD DATA TO BE TYPED INTO DATA REG.
3578 014360 000404 4$: BR EXTYP ;RETURN
3579
3580 014362 004767 177706 PCRLF: JSR PC,STYPE
3581 014366 005015 000 .ASCIZ <15><12> ;CR/LF
3582 014372 014372 .EVEN
3583 014372 000207 EXTYP: RTS PC ;RETURN
3584
3585 014374 004767 177762 TPCRLF: JSR PC,PCRLF ;TYPE CR/LF
3586 014400 000735 BR STYPE ;NOW GO TO TYPE THE REST OF THE MESSAGE
3587
3588
3589 014402 032777 000020 166264 PNTMES: BIT #20,2$WR ;PRINTOUTS ALLOWED?
3590 014410 001323 BNE NOTYP ;BRANCH IF NO
3591 014412 123737 000042 000046 CMPB 2#42,2#46 ;RUNNING UNDER ACT 11?
3592 014420 001717 BEQ NOTYP ;BRANCH IF YES -NOT PRINTOUT-
3593 014422 000764 BR TPCRLF ;SEND CR/LF AND TYPE MESSAGE.

```

```

3598
3599
3600
3601
3602
3603
3604
3605
3606 014424 004767 177732      TYPDEC: JSR      PC,PCRLF      ;TYPE CR/LF
3607
3608 014430 005046      STPDEC: CLR      -(SP)
3609 014432 013746 000306      MOV      @#DECWRD,-(SP) ;GET THE WORD THAT HAS TO BE CONVERTED TO A
3610                                     ;DECIMAL NUMBER
3611 014436 162716 000012      2$:      SUB      #10.,(SP)
3612 014442 002403      BLT      4$
3613                                     ;IF THE NUMBER IN (SP) WAS LESS THAN 10. THEN
3614 014444 005266 000002      INC      2(SP)
3615 014450 000772      BR       2$
3616 014452 062716 000012      4$:      ADD      #10.,(SP)
3617 014456 052716 000060      BIS      @60,(SP)
3618 014462 112667 000020      MOVB    (SP)+,6$-2
3619 014466 052716 000060      BIS      @60,(SP)
3620 014472 112667 000007      MOVB    (SP)+,6$-3
3621 014476 004767 177572      JSR      PC,$TYPE
3622                                     ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
3623 014502 020040 030040 000060      .ASCIZ  / 00/
3624                                     ;PLACE THE 1'S DIGIT TO BE TYPED
3625 014510 000207      6$:      RTS      PC
                                     ;MAKE THE CONTENTS OF (SP) A DECIMAL NUMBER
                                     ;PLACE THE 10'S DIGIT TO BE TYPED
                                     ;GO TO TYPE THE NUMBER IN DECIMAL FOLLOWED BY
                                     ;3 SPACES
                                     ;RETURN FROM THE SUBROUTINE
    
```

```

;* 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
;*
    
```



```

3630
3631
3632
3633
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645 014512 032777 020000 166154 TYPERR: BIT #20000, @SWR ;ERROR PRINTOUT WANTED?
3646 014520 001065 BNE OCTXT ;BRANCH IF NO
3647 014522 004767 177634 JSR PC, PCRLF ;TYPE CR/LF
3648 014526 004767 000012 JSR PC, TYPOCT ;TYPE OCTAL NO.
3649 014532 000460 BR OCTXT ;RETURN VIA RTS PC
3650 014534 012123 OCTTYP: MOV (R1)+, (R3)+ ;PLACE THE HIGH ORDER BITS AT LOCATION POINTED
3651 ;BY R3
3652 014536 012113 MOV (R1)+, (R3) ;AND NOW PLACE THE LOW ORDER BITS
3653 014540 105237 000310 INCB @#TYPCNT ;ENABLE THE TYPE OUT OF ONE OCTAL WORD
3654 014544 052743 000004 TYPOCT: BIS #4, -(R3)
3655 014550 106113 2$: ROLB (R3)
3656 014552 103376 BCC 2$
3657 014554 005000 CLR RO
3658 014556 106113 ROLB (R3) ;GET BITS 17 & 16 INTO RO
3659 014560 006100 ROL RO
3660 014562 106113 ROLB (R3)
3661 014564 006100 ROL RO
3662 014566 000405 BR $TYPNUM
3663 014570 004767 177500 RPTOCT: JSR PC, $TYPE ; TYPE 3 SPACES
3664 014574 020040 .ASCIZ / /
3665 .EVEN
3666 014600 005000 FATYP: CLR RO
3667 014602 012723 000006 $TYPNUM: MOV #6, (R3)+ ;ENABLE THE TYPE OUT OF 6 OCTAL DIGITS
3668 014606 000241 4$: CLC
3669 014610 006113 ROL (R3)
3670 014612 006100 ROL RO ;PLACE THE CARRY FROM (R3) IN RO
3671 014614 052700 000060 BIS #60, RO ;OR THE CONTENTS OF RO WITH AN ASCII 0
3672 014620 004767 177512 JSR PC, $TPCHR ; TYPE THE OCTAL NUMBER STORED IN RO
3673 014624 005000 CLR RO
3674 014626 006113 ROL (R3)
3675 014630 006100 ROL RO ;PLACE THE CARRY FROM (R3) IN RO
3676 014632 006113 ROL (R3)
3677 014634 006100 ROL RO ;PLACE THE CARRY FROM (R3) IN RO
3678 014636 105363 177776 DECB -2(R3) ;IF WE HAVEN'T TYPED THE 6 OCTAL DIGITS
3679 014642 001361 BNE 4$ ;THEN REPEAT FROM 4$
3680 014644 105337 000310 DECB @#TYPCNT ;IF ALL THE WORDS REQUIRED HAVE NOT BEEN
3681 014650 100411 BMI OCTXT ;BR IF = -1
3682 014652 001346 BNE RPTOCT ;TYPED THEN REPEAT FROM RPTOCT
3683 014654 032777 000200 166012 BIT #200, @SWR ;PRINT OUT XOR
3684 014662 001404 BEQ OCTXT ;BR IF NOT WANTED
3685 014664 016743 000006 MOV BDCHIP, -(R3) ;PUT IN LAST USED STACK LOC TO BE PRINTED
    
```

DZMLL MACY11 30(1046) 23-JUN-77 10:20 PAGE 1-72  
DZMLA.P11 23-JUN-77 10:17 OCTAL TYPE OUT ROUTINE

3686 014670 005743  
3687 014672 000736  
3688 014674 000207  
3689  
3690 014676 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

```

3695
3696
3697
3698
3699
3700
3701
3702
3703
3704 014700 012702 001400 MEMMNG: MOV #1400,R2
3705 014704 105037 000272 MMREG: CLRB @MMAVA ;CLEAR THE BYTE THAT IS SUPPOSED TO INDICATE
; THAT MEM. MANAG. IS AVAILABLE FOR TESTING
3706
3707 014710 032777 010000 165756 BIT #10000,@SWR ;HAS THE OPERATOR ASKED TO CHECK MEMORY MANAG. ?
3708 014716 001043 BNE RETMM ;IF NOT THEN RETURN FROM THE SUBROUTINE
3709 014720 012700 000004 MOV #4,R0 ;PREPARE TO SETUP TIME OUT VECTOR
3710 014724 012720 015030 MOV @NOMM,(R0)+ ;RETURN ADDRESS TO NOMM
3711 014730 012710 000340 MOV #340,(R0) ;AND WITH A PSW OF 340
3712 014734 005037 177572 CLR @SRO ;TRY TO REACH MEM. MANAG. SRO
3713 014740 105237 000272 INCB @MMAVA ;IF IT IS AVAILABLE THEN SET MEM. MANAG. AVAILABLE
3714 ;BYTE
3715 014744 012701 172340 MOV #172340,R1 ;R1 IS POINTING TO PAR0
3716 014750 005021 CLR (R1)+ ;PAR0 WILL POINT TO BANK 0
3717 014752 012721 000200 MOV #200,(R1)+ ;SET UP PAR 1
3718 014756 062702 000200 2$: ADD #200,R2
3719 014762 010221 MOV R2,(R1)+ ;SETUP PAR1-PAR6
3720 014764 020127 172356 CMP R1,#172356 ;ADDRESS OF PAR7
3721 014770 103772 BLO 2$
3722 014772 012711 007600 MOV #7600,(R1) ;PAR7 IS POINTING TO THE I/O PAGE
3723 014776 012701 172300 MOV #172300,R1
3724 015002 012721 077406 4$: MOV #77406,(R1)+ ;SETUP PDR0-PDR7
3725 015006 020127 172316 CMP R1,#172316
3726 015012 101773 BLOS 4$
3727 015014 005237 177572 INC @SRO ;ENABLE MEM. MANAG.
3728 015020 005010 $RETMM: CLR (R0) ;RESTORE TIME OUT TRAP VECTOR FOR ANY FUTURE TRAP
3729 015022 012740 000104 MOV @BUSER,-(R0)
3730 015026 000207 RETMM: RTS PC
3731
3732 015030 022626 NOMM: CMP (SP)+,(SP)+ ;RESTORE STACK POINTER
3733 015032 004767 177336 JSR PC,TPCRLF ;TYPE "NO MEMORY MANAGEMENT MESSAGE"
3734 015036 047516 046440 043516 .ASCIZ /NO MNG/
3735 015044 000 .EVEN
3736 015046 004767 177104 JSR PC,FATERR ;*ERROR* REPORT ERROR MESSAGE AND HALT AT FATHLT
(1) 015052 000104 104 ;*****ERROR NUMBER 104*****
(1)
3737 015054 000761 BR $RETMM ; RESTORE TIME OUT TRAP VECTOR
3738
3739 015056 013702 172354 UPMM: MOV @#172354,R2 ;PREPARE TO UPDATE MEMORY MANAG. REGISTERS
3740 015062 000710 BR MMREG
    
```

```

3745
3746
3747
3748
3749
3750
3751
3752
3753
3754
3755
3756 015064 005063 177776 PUTADR: CLR -2(R3)
3757 015070 010113 MOV R1,(R3) ;PLACE THE ADDRESS STORED IN R1 IN LOCATION (R3)
3758 015072 105737 000272 TSTB 2#MMAVA ;IS THE MEM. MANAG. AVAILABLE?
3759 015076 001425 BEQ 6$ ;IF NOT THEN RETURN FROM THE SUBROUTINE
3760 015100 010146 MOV R1,-(SP) ;SAVE R1
3761 015102 042701 017777 BIC #17777,R1 ;CLEAR BITS 0-12 OF THE ADDRESS IN R1
3762 015106 040113 BIC R1,(R3) ;LEAVE BITS 0-12 OF THE ADDRESS IN (R3)
3763 015110 052701 004000 BIS #4000,R1 ;PREPARE TO SHIFT R1 BY 12 PLACES
3764 015114 006001 2$: ROR R1
3765 015116 103376 BCC 2$ ;GET THE NUMBER OF PAR IN R1
3766 015120 062701 172340 ADD #172340,R1 ;GET THE ADDRESS OF PAR IN R1
3767 015124 011101 MOV (R1),R1 ;LOAD R1 WITH THE CONTENTS OF PAR
3768 015126 052701 010000 BIS #10000,R1
3769 015132 006101 4$: ROL R1
3770 015134 103376 BCC 4$ ;PLACE THE ADDRESS BITS 13-17 IN BITS 11-15 OF R1
3771 015136 006301 ASL R1 ;SO WE DON'T PICK UP C BIT
3772 015140 006143 ROL -(R3) ;PLACE BIT 17 IN LOCATION POINTED BY R3-2
3773 015142 006101 ROL R1
3774 015144 006123 ROL (R3)+ ;PLACE BIT 16 OF THE ADDRESS
3775 015146 050113 BIS R1,(R3) ;PLACE BITS 13-15 OF THE ADDRESS IN LOCATION (R3)
3776 015150 012601 MOV (SP)+,R1 ;RESTORE R1
3777 015152 000207 6$: RTS PC ;RETURN FROM THE SUBROUTINE

;* GET ADDRESS FROM THE APT MAILBOX
;* -----
;*
;* THIS SUBROUTINE IS USED TO GET ADDRESS FROM APT MAILBOX AND
;* PLACE IT IN THE LOCATION USED BY THE PROGRAM TO DEFINE THE
;* MEMORY BOUNDRIES.
;* PROGRAM CONTROL SHOULD COME TO THIS SUBROUTINE WITH R1 POINT-
;* ING TO THE MEMORY TYPE IN THE APT MAILBOX AND R3 POINTING TO
;* THE LOCATION+2 WHERE THE LOW ORDER BITS OF THE ADDRESS HAVE
;* TO BE PLACED
;*
3793
3794 015154 016143 000001 GETADR: MOV 1(R1),-(R3) ;PLACE THE LOW ORDER BITS OF THE ADDRESS
3795 015160 005043 CLR -(R3) ;CLEAR THE LOCATION WHERE THE HIGH ORDER BITS
3796 ;HAVE TO BE PLACED
3797 015162 116113 177777 MOVB -1(R1),(R3) ;PLACE BITS 16 & 17
3798 015166 000207 2$: RTS PC ;RETURN FROM THE SUBROUTINE
    
```

```

3803
3804
3805
3806
3807
3808
3809
3810
3811
3812
3813 015170 105237 000311      $GTSIZ: INCB      2#SAVKBB      ;PREPARE TO PLACE ADDRESS BITS 13-17 IN BITS
3814                                     ;0-4 OF R2
3815
3816 015174 012301      GETSIZ: MOV      (R3)+,R1
3817 015176 011302      MOV      (R3),R2      ;LOAD R2 WITH THE LOW ORDER BITS OF THE ADDRESS
3818 015200 042702 017777      BIC      #17777,R2      ;CLEAR ADDRESS BITS 0-12
3819 015204 052702 000040      2$:     BIS      #40,R2
3820 015210 006001      4$:     ROR      R1
3821 015212 006002      ROR      R2      ;ROTATE R1 AND R2 7 TIMES
3822 015214 103375      BCC      4$
3823 015216 105737 000311      TSTB    2#SAVKBB
3824 015222 001405      BEQ      6$
3825 015224 105037 000311      CLRB    2#SAVKBB
3826 015230 052702 000100      BIS      #100,R2
3827 015234 000765      BR      4$
3828 015236 012301      6$:     MOV      (R3)+,R1      ;PLACE THE LOW ORDER ADDRESS BITS IN R1
3829 015240 012700 160000      MOV      #160000,R0
3830 015244 040001      BIC      R0,R1      ;LEAVE BITS 0-12 OF THE ADDRESS IN R1
3831 015246 000207      RTS     PC      ;RETURN FROM THE SUBROUNE
3832
3833
3837
3838
3839
3840
3841
3842
3843
3844 015250 105737 000272      CLRMM: TSTB    2#MMAVA      ;WAS THE MEMORY MANAGEMENT ENABLED ?
3845 015254 001404      BEQ      1$      ;IF NOT THEN GO TO 1$
3846 015256 005037 177572      CLR      2#SRO      ;DISABLE THE MEMORY MANAGEMENT
3847 015262 105037 000272      CLRB    2#MMAVA      ;AND DO NOT ATTEMPT TO TEST MEM. MANAG.
3848 015266 000207      1$:     RTS     PC      ;RETURN FROM THE SUBROUTINE
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858 015270 010046      ;* GET BANK NO. UNDER TEST
3859 015272 010346      ;CALLED BY ERRYP TO GET BANK NO. UNDER TEST INTO PBNK.
3860 015274 042703 017777      ;REGISTERS
3861 015300 052703 010000      ;R0=POINTER TO PAR UNDER TEST
3861                                     ;R3=VIRTUAL ADDRESS ON ENTRY
3861                                     ;R0+R3 ARE RESTORED ON EXIT.
3861 GETBNK: MOV      R0,-(SP)      ;SAVE R0
3861                                     MOV      R3,-(SP)      ;SAVE R3
3861                                     BIC      #17777,R3      ;SAVE ONLY VIRTUAL BANK BITS
3861                                     BIS      #10000,R3      ;SETUP R3 SHIFT BIT
    
```

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

```

3862 015304 000241
3863 015306 006003
3864 015310 103376
3865 015312 105737 000272
3866 015316 001407
3867
3868
3869 015320 006303
3870 015322 062703 172340
3871 015326 011300
3872 015330 006300
3873 015332 000300
3874 015334 110003
3875 015336 010337 000306
3876 015342 012603
3877 015344 012600
3878 015346 000207
3879
3880
3881
3882
3883
3884
3885
3886
3887
3888 015350 010146
3889 015352 042701 003777
3890 015356 052701 002000
3891 015362 000241
3892 015364 006001
3893 015366 103376
3894 015370 010100
3895 015372 105737 000272
3896 015376 001416
3897
3898 015400 010146
3899 015402 006201
3900 015404 042701 000001
3901 015410 062701 172340
3902 015414 011100
3903 015416 006300
3904 015420 006300
3905 015422 006300
3906 015424 000300
3907 015426 042716 000034
3908 015432 052600
3909 015434 012601
3910 015436 000207
3911
3912
3913
3914
3915
3916
3917

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

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

;GET THE 1K BANK UNDER TEST
;CALLED BY TEST 7
;REGISTERS
; R1 = VIRTUAL ADDRESS ON ENTRY
; R0 = POINTER TO 1K UNDER TEST ON EXIT
;R1 WILL BE RESTORED ON EXIT

GET1K: MOV R1,-(SP) ;SAVE R1
BIC #3777,R1 ;ONLY WANT 1K BIT COUNT
BIS #2000,R1 ;SET UP SHIFT BIT
CLC ;CLEAR CARRY BIT
1$: ROR R1 ;ROTATE THE BANK
BCC 1$ ;UNTIL THEY'RE IN BITS 4:0
MOV R1,R0 ;IN CASE NO MEM MGNT
TSTB 2$MMAVA ;MEM MGNT AVAILABLE ?
BEQ 2$ ;BR IF NO

;GET PAR AND PHYSICAL BANK
MOV R1,-(SP) ;SAVE 1K BANK
ASR R1 ;MAKE R1 A PAR ADDRESS
BIC #1,R1 ;PAR ADDRESSES ARE 4K BOUNDARIES
ADD #172340,R1 ;MAKE IT A FULL PAR ADDR.
MOV (R1),R0 ;GET THE PAR CONTENTS
ASL R0 ;SHIFT R0
ASL R0
ASL R0
SWAB R0
BIC #34,(SP) ;ONLY WANT THE 2 LSB'S
BIS (SP)+,R0 ;SET THE IN PAR
2$: MOV (SP)+,R1 ;RESTORE R1
RTS PC ;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
    
```

```

3918
3919 015440 016777 165206 165224 TSTADD: MOV      ECCDIS, @CSRADR ;DISABLE ECC
3920 015446 005011          CLR      (R1)      ;CLEAR A LOC
3921 015450 005061 000002          CLR      2(R1)     ;UPPER WORD TOO
3922 015454 005711          TST      (R1)      ;READ ZEROS
3923 015456 001047          BNE     1$        ;BR IF ERROR
3924 015460 005761 000002          TST      2(R1)     ;SHOULD BE ZEROS ALSO
3925 015464 001044          BNE     1$        ;
3926 015466 005111          COM     (R1)      ;COMPLEMENT OF 0 = 177777
3927 015470 005161 000002          COM     2(R1)     ;BOTH WORDS
3928 015474 022711 177777          CMP     #-1,(R1)  ;READ ALL ONES
3929 015500 001036          BNE     1$        ;BR IF NO
3930 015502 022761 177777 000002          CMP     #-1,2(R1) ;READ AGAIN
3931 015510 001036          BNE     1$        ;BR IF ERROR
3932 015512 016703 165154          MOV     CSRADR,R3 ;GET ADDRESS OF CSR
3933 015516 016713 165132          MOV     DIAGA,(R3) ;SET DIAGNOSTIC BIT
3934 015522 052713 000002          BIS     #2,(R3)   ;AND ECCDIS BIT
3935 015526 012711 000000          MOV     #0,(R1)  ;TO WRITE ALL ZERO CHECKBITS
3936 015532 005761 000002          TST     2(R1)     ;READ AND RECORD CHECKBITS
3937 015536 032713 007740          BIT     #7740,(R3) ;NO CHECKBIT SHOULD BE SET
3938 015542 001015          BNE     1$        ;BR IF SET
3939 015544 052713 007740          BIS     #7740,(R3) ;MAKE CHECKBITS ALL ONES
3940 015550 012711 177777          MOV     #-1,(R1) ;WRITE THEM LIKE THIS
3941 015554 042713 007740          BIC     #7740,(R3) ;SO WE KNOW IF WE READ THEM BACK
3942 015560 005761 000002          TST     2(R1)     ;READ THEM
3943 015564 042713 170037          BIC     #170037,(R3) ;ONLY WANT CHECKBITS
3944 015570 022713 007740          CMP     #7740,(R3) ;SHOULD BE ALL ONES
3945 015574 001403          BEQ     2$        ;BR IF OK
3946 015576 012703 000001          1$: MOV     #1,R3    ;INDICATE ERROR FOUND
3947 015602 000401          BR      3$        ;AND EXIT
3948 015604 005003          2$: CLR     R3     ;CLEARED = GOOD LOC
3949 015606 005077 165060          3$: CLR     @CSRADR ;CLEAR CSR BEFORE LEAVING
3950 015612 000207          RTS     PC        ;AND EXIT
3951
3952
3953 ;THIS ROUTINE IS USED TO COUNT THE BAD CHECKBITS
3954 ;FROM TEST 15. IT USES THE ROUTINE 'XOR' AND THEN
3955 ;COUNTS THE BAD ONES
3956
3957 015614 005067 000056 BITCHK: CLR     BITCNT ;WANT TO START CLEAN
3958 015620 010146          MOV     R1,-(SP)  ;SAVE ADDRESS
3959 015622 017701 165044          MOV     @CSRADR,R1 ;GET CONTENTS OF CSR
3960 015626 042701 170037          BIC     #170037,R1 ;ONLY WANT CHECK BITS
3961 015632 010146          MOV     R1,-(SP)  ;SAVE THIS TOO
3962 015634 004767 175630          JSR     PC,XOR    ;R0 XOR R1 = R2
3963 015640 012701 000020          MOV     #20,R1   ;GETTING READY TO COUNT
3964 015644 006301          1$: ASL     R1     ;SHIFT TO NEXT POSITION
3965 015646 020127 010000          CMP     R1,#10000 ;FINISHED YET ?
3966 015652 001405          BEQ     2$        ;BR IF DONE
3967 015654 030102          BIT     R1,R2    ;SEE IF THIS BIT (R1) IS BAD
3968 015656 001772          BEQ     1$        ;BR IF NOT BAD
3969 015660 005267 000012          INC     BITCNT   ;IT IS, COUNT IT
3970 015664 000767          BR      1$        ;NOT DONE YET, KEEP GOING
3971 015666 012600          2$: MOV     (SP)+,R0 ;FIRST GET DATA
3972 015670 012601          MOV     (SP)+,R1 ;THEN ADDRESS
3973 015672 010011          MOV     R0,(R1)  ;PUT BAD DATA AWAY FOR PRINTOUT
    
```

SUBROUTINE TO DISABLE MEMORY MANAGEMENT

```

3974 015674 000207          RTS      PC          ;THEN EXIT
3975
3976 015676 000000          BITCNT: .WORD  0          ;# OF BAD CHECKBITS FOUND ABOVE
3977
3978
3979
3980
3981 015700 004767 176470          PRITL: JSR      PC,TPCRLF      ;PRINT ROUTINE
3982 015704 055104 046515 026514          .ASCII  /DZMML-A/
3983 015712 101
3983 015713 040 051525 020105          .ASCII  / USE TO TEST MF115-K ECC MEMORY/
3983 015720 047524 052040 051505
3983 015726 020124 043115 030461
3983 015734 026523 020113 041505
3983 015742 020103 042515 047515
3983 015750 054522
3984 015752 005015 054524 042520          .ASCIZ  <15><12>/TYPE <CONTROL C> TO EXIT/
3984 015760 036040 047503 052116
3984 015766 047522 020114 037103
3984 015774 052040 020117 054105
3984 016002 052111 000
3985
3986 016006 012767 172136 164656          .EVEN
3986 MOV      #172136,CSRADR      ;ORIGINAL CSR ADDRESS
3987 016014 000207          RTS      PC          ;RETURN TO START ROUTINE
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000 016016 105037 000311          CHECKC: CLRB   @#SAVKBB      ;INIT CONTROL-C FLAG.
4001 016022 105737 177560          TSTB   @#TKS              ;ANY CHAR. TYPED?
4002 016026 100021          BPL    EXITC              ;BR IF NO-EXIT VIA RTS PC-
4003 016030 113702 177562          MOVB   @#SKBB,R2         ;GET THE CHAR TYPED.
4004 016034 042702 000200          BIC    #200,R2           ;CLEAR THE PARITY BIT.
4005 016040 122702 000003          CMPB   #3,R2             ;IS IT CONTROL-C?
4006 016044 001012          BNE    EXITC              ;BRANCH IF NO -EXIT VIA RTS PC-
4007 016046 110237 000311          MOVB   R2,@#SAVKBB      ;ELSE STORE THE CHAR. FOR USE AS A FLAG.
4008 016052 004767 176316          JSR    PC,TPCRLF        ;PRINT "IC"
4009 016056 041536 000          .ASCII  /IC/
4010
4011 016062 005067 164600          .EVEN
4011 CLR     ERRFLG            ;CLEAR ERROR HEADER FLAG FOR NEXT START
4012 016066 000167 173752          JMP    RELOER            ;GO RETURN PROGRAM TO LOWER CORE IF RELOCATED.
4013 016072 000207          EXITC: RTS      PC
4014
4015 016074 000000          ENDPROG: 0
4016
4017
4018
4019          .END
          ;THIS BEGINS THE STORAGE FOR THE ERROR HISTORY
          ;STACK.FOR EACH 4K BANK 18. BYTES ARE SAVED.
          ;ALSO THE ABSOLUTE LOADER AND XXDP CODE IS SAVED
          ;AFTER THE ERROR STACK.
    
```



ABASE =	000000	BOCHIP	014676	EXITC	016072	RLODER	012722	TST3	003506
ACDM1 =	000000	BEGIN	000500	EXTYP	014372	RPTOCT	014570	TST4	003740
ACDM2 =	000000	BITCHK	015614	FAILNM	012556	RPT17	011464	TST5	004114
ACPUOP =	000000	BITCNT	015676	FATERR	014156	RSTOT4	013302	TST6	004564
ACT11	012676	BITCOM	013346	FATHLT	014230	SAVKBB	000311	TST7	005504
ADDW0 =	000000	BRTPSZ	001276	FATYP	014600	SAVLDR	001776	TYPCNT	000310
ADDW1 =	000000	BUSER	000104	FNDERR	014104	SAVLOC	000350	TYPDEC	014424
ADDW10 =	000000	CHECK	012756	GETADR	015154	SAVMAX	000340	TYPENB	000273
ADDW11 =	000000	CHECKC	016016	GETBNK	015270	SAVMIN	002664	TYPECP	012644
ADDW12 =	000000	CHKCNT	013015	GETSIZ	015174	SAVR4	000342	TYPERR	014512
ADDW13 =	000000	CHKGEN	013026	GETIK	015350	SAVRS	000344	TYPHEM	001704
ADDW14 =	000000	CHKTAB	012760	HIGHAD	000330	SAVR6	000346	TYP OCT	014544
ADDW15 =	000000	CKDONE	012136	HIGHTH	000326	SAVOT4	013270	TYPSIZ	001612
ADDW2 =	000000	CLRMEM	002174	INHREL	002670	SAV7	006474	TYPSTK	012502
ADDW3 =	000000	CLRMM	015250	LDFLG	002660	SBEMSK	002636	T12A	007602
ADDW4 =	000000	CNTSCP	002566	LOOP	002422	SCOPE =	000240	T12B	007624
ADDW5 =	000000	CONT	002364	LOWADD	000324	SEERR	014156	TSP	004136
ADDW6 =	000000	CONTHM	012204	LOWBNK	000300	SETSTK	001716	T6A	004666
ADDW7 =	000000	CSRADR	002672	LOWER	012142	SETSWR	001126	T6B	004700
ADDW8 =	000000	CTLC	012744	LOWTWO	000322	SLFSIZ	001310	T6FLG	005502
ADDW9 =	000000	DASHL	013316	M =	000200	SOURCE	012754	UPMM	015056
ADDEVCT =	000000	DATADR	013466	MAXADR	012310	SRO =	177572	UPPFLG	013014
ADDEVH =	000000	DATBUF	002626	MAXMEM	000336	START	000200	WRTMEM	000120
ADENV =	000000	DATSAV	002646	MEMANG	014700	STRTDI	000276	XOR	013470
ADENVH =	000000	DBEMSK	002642	MEMTST	002166	SWPAT	000314	SA =	000001
AFATAL =	000000	DECARD	000306	MINMEM	000320	SWALT	002606	SADERR	000275
AMADR1 =	000000	DIAGA	002654	MMAVA	000272	SWR	002674	SAPTHD	000272
AMADR2 =	000000	DISPLA	002676	MMREG	014704	SWREG	000176	SCNTHM	012220
AMADR3 =	000000	DSPREG	000174	MSKADR	013464	SW11 =	004000	SCPUOP	000426
AMADR4 =	000000	ECCDIS	002652	MSYES	002656	TBL	002506	SDEVCT	000410
AMAMS1 =	000000	ENDMAX	012364	N =	000105	TKS =	177560	SDOAGN	012716
AMAMS2 =	000000	ENDPAS	012372	NOMM	015030	TPADR	014234	SENDAD	000156
AMAMS3 =	000000	ENDPRO	016074	NOTYP	014260	TPCRLF	014374	SENV	000420
AMAMS4 =	000000	ENDSTK	000304	OCTTYP	014534	TRYSR	001300	SENVH	000421
AMSGAD =	000000	END0	003032	OCTXT	014674	TSTADD	015440	SEOP	012614
AMSGLG =	000000	END1	003322	ONEPAS	001030	TSTDAT	002632	SETABL	000420
AMSGTY =	000000	END10	007122	PARBYT	013024	TSTGO	002610	SETEND	000450
AMTYP1 =	000000	END11	007554	PASFLG	000302	TSTMM	012170	SFATAL	000402
AMTYP2 =	000000	END12	010214	PBNK	000306	TSTREL	002046	SGTSIZ	015170
AMTYP3 =	000000	END13	010326	PCRLF	014362	TSTRP	001044	SHD =	000002
AMTYP4 =	000000	END14	010440	PNTMES	014402	TSTSCP	002550	SHIBTS	000272
APASS =	000000	END15	011272	PRITL	015700	TSTSZ	002052	SHIMAX	000332
APRIOR =	000000	END16	011442	PUTADR	015064	TST0	002700	SKBB =	177562
APHLT	014222	END17	011600	PWRDN	000070	TST1	003034	SMAOR1	000432
APTSIZ	001210	END2	003502	PWRUP	000136	TST10	006500	SMAOR2	000436
ASWREG =	000000	END3	003734	REL	000352	TST11	007126	SMAOR3	000442
ATESTN =	000000	END4	004110	RELBOT	000316	TST12	007560	SMAOR4	000446
AUNIT =	000000	END5	004560	RELOC	011604	TST13	010220	SMAIL	000400
AUSWR =	000000	END6	005476	RELOER	012044	TST14	010332	SMAMS1	000430
AVECT1 =	000000	END7	006470	RESTR	000250	TST15	010444	SMAMS2	000434
AVECT2 =	000000	ERRFLG	002666	RETM	015026	TST16	011276	SMAMS3	000440
BAKPAT	000312	ERROR	013502	RETSTK	012522	TST17	011446	SMAMS4	000444
BASE	002662	ERRTYP	013752	RETTYP	014320	TST2	003326	SMAXM	000334

\$MADR	000274	\$MTYP4	000445	\$SWREG	000422	\$TPS	=	177564	\$Z	=	000353	
\$MSGAD	000414	\$PASS	000406	\$TESTN	000404	\$TPSTK		012472	.	=	016076	
\$MSGLG	000416	\$PASTM	000300	\$TN	=	000000	\$TSM		000276	.\$X	=	000272
\$MSGTY	000400	\$PRERR	000274	\$TPB	=	177566	\$TYPE		014274			
\$MTYP1	000431	\$RETM	015020	\$TPCHR		014336	\$UNIT		000412			
\$MTYP2	000435	\$SVPC	=	000044	\$TPDEC		\$JNITM		000302			
\$MTYP3	000441	\$SWR	=	000000	\$TPNUM		\$USWR		000424			

. ABS. 016076 000

ERRORS DETECTED: 0

DZMPLA, DZMPLA/DOC=DZMPLA  
RUN-TIME: 8 9 .3 SECONDS  
RUN-TIME RATIO: 367/17=20.4  
CORE USED: 12K (23 PAGES)

DOCUMENT PAGES: 82