

801

EOF1DZVSDDBSEQ
PDP10 411 SEQ 0001

00010000 770225

PDP10 411
IDENTIFICATION

HDR1DZQMCCSEQ

00010000

770225

=====

PRODUCT CODE: MAINDEC-11-DZQMC-C-D
PRODUCT NAME: 0-124K MEMORY EXERCISER, 16K VERSION.
MAINTAINER: DIAGNOSTIC ENGINEERING, CC301

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

THE SOFTWARE IN THIS DOCUMENT IS FURNISHED UNDER A LICENSE AND MAY ONLY BE USED OR COPIED IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE.

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

COPYRIGHT (C) 1975,1977 BY DIGITAL EQUIPMENT CORPORATION

REVISION HISTORY
=====

REVISION A:	MAY 1975
REVISION B:	OCTOBER 1975
REVISION C	JANUARY 1977

TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION.
- 1.1 PROGRAM PURPOSE (ABSTRACT)
- 1.2 SYSTEM REQUIREMENTS
- 1.3 RELATED DOCUMENTS AND STANDARDS
- 1.4 DIAGNOSTIC HIERARCHY PREREQUISITES
- 1.5 ASSUMPTIONS

- 2.0 OPERATING INSTRUCTIONS
- 2.1 LOADING AND STARTING PROCEDURE
- 2.2 SPECIAL ENVIRONMENTS
- 2.3 PROGRAM OPTIONS
- 2.4 EXECUTION TIMES

- 3.0 ERROR INFORMATION
- 3.1 ERROR REPORTING
- 3.2 ERROR HALTS

- 4.0 PERFORMANCE AND PROGRESS REPORTS

- 5.0 DEVICE INFORMATION TABLES
- 5.1 CORE PARITY REGISTER
- 5.2 MOS PARITY REGISTER
- 5.3 MSII-K CSR

- 6.0 SUB-TEST SUMMARIES
- 6.1 SECTION 1: ADDRESS TESTS
- 6.2 SECTION 2: WORST CASE NOISE TESTS
- 6.3 SECTION 3: INSTRUCTION EXLCUTION TESTS
- 6.4 SPECIAL TOGGLE IN TESTS

- 7.0 PROGRAM FUNCTIONAL FLOW CHARTS

- 8.0 PROGRAM LISTING

1.0 GENERAL PROGRAM INFORMATION.

1.1 PROGRAM PURPOSE (ABSTRACT)

THIS PROGRAM HAS THE ABILITY TO TEST MEMORY FROM ADDRESS 000000 TO ADDRESS 757777. IT DOES SO USING:

- A. UNIQUE ADDRESSING TECHNIQUES
- B. WORSE CASE NOISE PATTERNS, AND
- C. INSTRUCTION EXECUTION THRUOUT MEMORY.

THERE IS ALSO A SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESS RANGES WHICH DO NOT TIMEOUT, AS WELL AS TWO(2) TOGGLE IN ADDRESS TESTS PROVIDED IN SECTION 6.1 OF THIS DOCUMENT.

1.2 SYSTEM REQUIREMENTS

A. HARDWARE REQUIREMENTS

PDP11 FAMILY PROCESSOR WITH A MINIMUM OF 16K OF MEMORY.
OPTIONAL:
M7259 PARITY MEMORY CONTROL MODULE.
KT11 MEMORY MANAGEMENT.

B. SOFTWARE REQUIREMENTS

THE SMALLEST UNIT OF MEMORY THIS PROGRAM WILL RECOGNIZE IS 4K. IF ANY ADDRESS IN A 4K BANK CAUSES A TIME OUT TRAP, THAT ENTIRE BANK OF MEMORY IS IGNORED BY THE PROGRAM.

THE PROGRAM IS DESIGNED TO EXERCISE THE VECTOR PORTION OF MEMORY (LOCATIONS 0-776) IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY. TO MAKE THIS POSSIBLE, WITHOUT REQUIRING MEMORY MANAGEMENT, NO SOFTWARE TRAPS ARE USED IN THE PROGRAM. THIS MEANS THAT IF MEMORY MANAGEMENT IS NOT AVAILABLE OR IS DISABLED (SW12=1), IF THE PROGRAM IS RELOCATED OUT OF BANK 0, IF LOCATION 0-776 ARE SELECTED FOR TEST, AND IF AN UNEXPECTED HARDWARE TRAP OCCURS, THE RESULTS WILL BE UNPREDICTABLE.

THE PROGRAM HAS THE PROPER INTERFACE CODE TO ALLOW RUNNING UNDER THE AUTOMATED MANUFACTURING TEST LINE SYSTEM - ACT11.

1.3 RELATED DOCUMENTS AND STANDARDS

- A. PROGRAMMING PRACTICES - DOCUMENT NO. 175-003-009-01
- B. PDP-11 MAINDEC SYSMAC PACKAGE - MAINDEC-11-DZQAC-C2-D
- C. PDP11/40 PROCESSOR HANDBOOK
- D. MF11-U/UP CORE MEMORY SYSTEM MAINTENANCE MANUAL
DOCUMENT NO. DEC-11-HMFMA-B-D
- E. APPLICABLE CIRCUIT SCHEMATICS:

G235 - 16K X-Y DRIVE
G114 - 16K SENSE/INHIBIT
M8293 - 16K UNIBUS TIMING
M7259 - PARITY CONTROL

1.4 DIAGNOSTIC HIERARCHY PREREQUISITES

BEFORE RUNNING THIS PROGRAM, A CPU DIAGNOSTIC SHOULD BE RUN TO VERIFY THE FUNCTIONALITY OF THE PROCESSOR AND PDP-11 INSTRUCTION SET.

IF MEMORY MANAGEMENT IS TO BE USED, THEN THE KT11 DIAGNOSTIC SHOULD ALSO BE RUN BEFORE THIS PROGRAM.

PDP-11/05 - MAINDEC-11-DZQKC
PDP-11/20 - MAINDEC-11-DZQKC
PDP-11/40 - MAINDEC-11-DBQEA
 OR MAINDEC-11-DCQKC
PDP-11/45 - MAINDEC-11-DCQKC
KT11-C - MAINDEC-11-DCKTA THRU DCKTF
KT11-D - MAINDEC-11-DBKTA THRU DBKTF

1.5 ASSUMPTIONS

THIS PROGRAM ASSUMES THE CORRECT OPERATION OF THE CPU AND, IF USED, THE MEMORY MANAGEMENT OPTION.

2.0 OPERATING INSTRUCTIONS

2.1 LOADING AND STARTING PROCEDURES

2.1.1 LOAD THE PROGRAM USING ANY STANDARD ABSOLUTE LOADER.

2.1.2 STARTING ADDRESS 200:
NORMAL PROGRAM EXECUTION.

2.1.3 STARTING ADDRESS 204:
ALLOWS THE OPERATOR TO INPUT, VIA TELETYPE CONVERSATION, FIRST AND LAST ADDRESSES TO BE EXERCISED, AND A DATA PATTERN TO BE USED IN TESTS 6 AND 7.

2.1.4 STARTING ADDRESS 210:
RESTART PROGRAM USING PREVIOUSLY SELECTED PARAMETERS.

2.1.5 STARTING ADDRESS 214:
RESTORE LOADERS AND HALT. THIS ROUTINE IS CAPABLE OF RELOCATING THE PROGRAM BACK TO BANKS 0 AND 1 IF THE PROGRAM WAS HALTED WHILE RUNNING THE TOP TWO BANKS OF MEMORY. THERE ARE SPECIAL PROCEDURES REQUIRED FOR THIS SITUATION.

- A. IF MEMORY ADDRESSES 0-1000 HAVE NOT BEEN EXERCISED, EITHER THROUGH PARAMETER SELECTION (SA=204) OR BY RUNNING WITH SW05=1, THEN:

LOAD ADDRESS 214.
PRESS START.

- B. IF RUNNING WITHOUT MEMORY MANAGEMENT, THEN:

LOAD ADDRESS <214+RELOCATION FACTOR>
(RELOCATION FACTOR IS TYPED WHEN THE PROGRAM IS RELOCATED),
PRESS START.

- C. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS NOT BEEN INITIALIZED (VIA RESET INSTRUCTION, START SWITCH, ETC.), THEN:

LOAD ADDRESS 777707 (PC)
DEPOSIT 214
PRESS CONTINUE

- D. IF RUNNING WITH MEMORY MANAGEMENT AND THE UNIBUS HAS BEEN INITIALIZED:

LOAD ADDRESS 772340 (KIPAR0)
DEPOSIT <<(RELOCATION FACTOR)/100>
(EXAMPLE: RELOCATION FACTOR=540000, THEN
DEPOSIT 005400)
LOAD ADDRESS 777572 (SRO)
DEPOSIT 000001
LOAD ADDRESS 777707 (PC)
DEPOSIT 214
PRESS CONTINUE

2.1.6 STARTING ADDRESS 220:

BYTE ADDRESS MEMORY MAP TYPEOUT ROUTINE. THIS ROUTINE PERFORMS DATI, DATIP, DATO, AND DATOB ON ALL POSSIBLE ADDRESSES, AND TYPES THE RANGES OF ADDRESSES WHICH DO NOT CAUSE A TIMEOUT TRAP.

2.2 SPECIAL ENVIRONMENTS

IF THE PROGRAM IS RUN IN QUICK VERIFY MODE UNDER ACT11 THE PROGRAM IS DONE AFTER THE FIRST PASS. ALSO, THE PROGRAM DOES NOT RELOCATE TO TEST THE LOWER 8K OF MEMORY.

2.3 PROGRAM OPTIONS

SW15 = 1 OR UP.... HALT ON ERROR

SW14 = 1 OR UP.... LOOP ON TEST

SW13 = 1 OR UP.... INHIBIT ERROR TYPEOUT
 SW12 = 1 OR UP.... INHIBIT MEMORY MANAGEMENT (INITIAL
 START ONLY)
 SW11 = 1 OR UP.... INHIBIT SUBTEST ITERATION
 SW10 = 1 OR UP.... RING BELL ON ERROR
 SW9 = 1 OR UP.... LOOP ON ERROR
 SW8 = 1 OR UP.... LOOP ON TEST IN SWR<4:0>
 SW7 = 1 OR UP.... INHIBIT PROGRAM RELOCATION
 SW6 = 1 OR UP.... INHIBIT PARITY ERROR DETECTION

NOTE: WITH PARITY ERROR DETECTION ENABLED, A MEMORY FAILURE
 WHILE RUNNING THE WORSE CASE NOISE TESTS (NON-PARITY)
 CAN CAUSE A PARITY ERROR. THE ERROR PRINTOUT ON A
 PARITY ERROR DOES NOT TYPE THE GOOD DATA. THUS A BIT
 DROP OR PICKUP WILL NOT BE TYPED AS SUCH. IT IS BEST
 TO RUN THE PROGRAM FOR 1 PASS WITH PARITY DISABLED,
 THEN, RESTART THE PROGRAM WITH PARITY ENABLED.

SW5 = 1 OR UP.... INHIBIT EXERCISING VECTOR AREA
 (LOCATIONS 0-1000).

2.4 EXECUTION TIMES

EXECUTION TIME IS DEPENDENT ON TYPE OF MEMORY, AND AMOUNT OF
 MEMORY. WORSE CASE RUN TIMES WITH 900NS MEMORYS ARE:

- A. FOR NON-PARITY MEMORY
 FIRST PASS: 45 SECONDS + 3 SECONDS PER 4K
 FULL PASS: 3 MINUTES PER 4K
 ITERATION INHIBITED: SAME AS FIRST PASS
- B. FOR PARITY MEMORY
 FIRST PASS: 45 SECONDS +10 SECONDS PER 4K
 FULL PASS: 4 MINUTES PER 4K
 ITERATION INHIBITED: SAME AS FIRST PASS

3.C ERROR INFORMATION

3.1 ERROR REPORTING

THERE ARE A TOTAL OF 31(8) TYPES OF ERROR REPORTS GENERATED BY THE PROGRAM. SOME OF THE KEY COLUMN HEADING MNEMONICS ARE DESCRIBED BELOW FOR CLARITY:

PC = PROGRAM COUNTER OF ERROR DETECTION CODE.
(V/PC=P/PC)

V/PC = VIRTUAL PROGRAM COUNTER. THIS IS WHERE THE ERROR DETECTION CODE CAN BE FOUND IN THE PROGRAM LISTING.

P/PC = PHYSICAL PROGRAM COUNTER. THIS IS WHERE THE ERROR DETECTION CODE IS ACTUALLY LOCATED IN MEMORY.

TRP/PC = PHYSICAL PROGRAM COUNTER OF THE CODE WHICH CAUSED A TRAP.

MA = MEMORY ADDRESS

REG = PARITY REGISTER ADDRESS.

PS = PROCESSOR STATUS WORD.

IUT = INSTRUCTION UNDER TEST.

S/B = WHAT CONTENTS SHOULD BE.

WAS = WHAT CONTENTS WAS.

3.2 ERROR HALTS

WITH THE 'HALT ON ERROR' SWITCH (SW15) NOT SET THERE ARE SEVERAL PROGRAMMED 'HALTS' IN THE PROGRAM:

A. IN THE ERROR TRAP SERVICE ROUTINE FOR UNEXPECTED TRAPS TO VECTOR 4. THIS ONE WILL OCCUR IF A 2ND TRAP TO 4 OCCURS BEFORE THE ERROR REPORT FOR THE FIRST HAS HAD A CHANCE TO BE PRINTED OUT.

B. IN THE RELOCATION ROUTINE IF THE PROGRAM IS BEING RELOCATED BACK TO THE FIRST 8K OF MEMORY AND THE PROGRAM CODE WAS NOT ABLE TO BE TRANSFERRED PROPERLY.

C. IN THE CASE OF ERROR REPORTING AND THERE IS NO TERMINAL TO ALLOW THE INFORMATION TRANSFER.

D. IN THE POWER FAIL ROUTINE IF THE POWER UP SEQUENCE WAS STARTED BEFORE THE POWER DOWN SEQUENCE HAD A CHANCE TO COMPLETE ITSELF.

E. IN THE MEMORY MAPPING ROUTINE OR ANY OF THE ADDRESS CONTROL ROUTINES. FAILURES TO FIND A MEANINGFUL MAP.

4.0 PERFORMANCE AND PROGRESS REPORTS
NOT APPLICABLE

5.0 DEVICE INFORMATION TABLES

THE FOLLOWING IS A PICTURE VIEW OF A PARITY CONTROL STATUS REGISTERS, WHICH WILL SHOW BIT ASSIGNMENTS AND DEFINITIONS, TO PROVIDE A HANDY REFERENCE:

5.1 CORE PARITY REGISTER

```

-----
I I I I I I I I I I I I I I I I I I
|PE| | | | | | | | | | | | | | | | |
I I I I I I I I I I I I I I I I I I
-----

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BITS 11-5	ERROR ADDRESS	HIGH ORDER ADDRESS BITS OF ADDRESS OF PARITY ERROR (BITS 17-11 OF ADDRESS)
BIT02	WRITE WRONG PARITY	NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET
BIT00	ACTION ENABLE	NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

5.2 MOS PARITY REGISTER

```

-----
I I I I I I I I I I I I I I I I I I
|PE| | | | | | | | | | | | | | | |
I I I I I I I I I I I I I I I I I I
-----

```

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BIT02	WRITE WRONG PARITY	NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET

TEST 3 WRITES THE COMPLEMENT OF EACH WORD ADDRESS INTO THAT LOCATION AND CHECKS IT.

TEST 4 WRITES THE 4K BANK NUMBER INTO EACH BYTE OF THAT BANK AND CHECKS IT.

TEST 5 WRITES THE COMPLEMENT OF THE BANK NUMBER INTO EACH BYTE OF THAT BANK AND CHECKS IT.

6.2 SECTION 2: WORST CASE NOISE TESTS.

THESE ARE INTENDED TO APPLY MAXIMUM STRESS TO THE VARIOUS TYPES OF PDP-11 CORE MEMORIES.

TEST 6 AND TEST 7 ARE SUPPLIED TO ALLOW THE OPERATOR TO SELECT A SINGLE WORD DATA PATTERN (SA=204) AND SCOPE ON EITHER THE WRITING (DATO) IN TEST 6 OR THE READING (DATI) IN TEST 7 OF THAT DATA.

TEST 10 WRITES AND THEN CHECKS A SERIES OF SINGLE WORD PATTERNS WHICH ARE DESIGNED TO STRESS PARITY MEMORY.

TEST 11 WRITES ALL MEMORY WITH 1'S IN EVERY BIT AND THEN "RIPPLES" A "0" THROUGH IT.

TEST 12 WRITES ALL MEMORY WITH 0'S IN EVERY BIT AND THEN "RIPPLES" A "1" THROUGH IT.

TEST 13 WRITE A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 1 XOR ADDRESS BIT 8 COMPLEMENTS. THIS CREATES A "CHECKERBOARD" PATTERN IN CERTAIN MEMORY ARCHITECTURES.

TEST 14, 15, 16, AND 17 WRITE A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 3 XOR BIT 9 COMPLEMENTS.

TEST 20 WRITES A PATTERN WHICH COMPLEMENTS WHEN ADDRESS BIT 8 XOR BIT 13 COMPLEMENTS.

TEST 21 WRITES WRONG PARITY IN EACH BYTE OF MEMORY AND CHECKS THAT THE PARITY DETECTION LOGIC WORKS. THIS TEST IS SKIPPED FOR NON-PARITY MEMORY.

TEST 22 WRITE "RANDOM" PROGRAM CODE THROUGH MEMORY AND CHECKS IT.

6.3 SECTION 3: INSTRUCTION EXECUTION TESTS.

THIS GROUP OF TESTS PLACE INSTRUCTIONS IN THE MEMORY UNDER TEST, THEN EXECUTES THE INSTRUCTIONS, AND FINALLY, CHECKS THAT THEY EXECUTED CORRECTLY.

TEST 23 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATO ON THE MEMORY UNDER TEST.

TEST 24 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATOB ON THE LOW BYTE OF MEMORY UNDER TEST.

TEST 25 EXECUTES AN INSTRUCTION WHICH DOES A DATI AND A DATOB ON THE HIGH BYTE.

TEST 26 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATO.

TEST 27 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE LOW BYTE.

TEST 30 EXECUTES AN INSTRUCTION WHICH DOES A DATIP AND A DATOB ON THE HIGH BYTE.

TEST 31 EXECUTES A SERIES OF INSTRUCTIONS DESIGNED TO STRESS OR "HEAT" THE MEMORY.

6.4 SPECIAL TOGGLE IN TESTS

6.4.1 TOGGLE-IN-PROGRAM #1

THE FOLLOWING IS A TOGGLE IN MEMORY ADDRESS TEST. THIS TEST IS USEFUL WHEN ADDRESS SELECTION FAILURE IS SUSPECTED INVOLVING THE FIRST 8K OF MEMORY. THIS PROGRAM WRITES THE VALUE OF EACH ADDRESS INTO ITSELF STARTING WITH THE LOWER LIMIT AND CONTINUING TO THE UPPER LIMIT. AFTER ALL ADDRESSES HAVE BEEN WRITTEN EACH ADDRESS IS CHECKED FOR THE CORRECT CONTENTS STARTING WITH THE UPPER LIMIT AND CONTINUING TO THE LOWER LIMIT.

LOCATION	CONTENTS	MNEMONIC	COMMENT
10	012700	MOV #50,R0	;GET FIRST ADDRESS
* 12	000050		;TO TEST ;(EXAMPLE START ADDRESS)
14	010001	MOV R0,R1	;SAVE IN R1
16	020037	1\$: CMP R0,#SWR	;CHECK UPPER LIMIT
20	177570		; (IN SWITCH REGISTER)
22	001403	BEQ 2\$;BRANCH IF AT UPPER LIMIT
24	010010	MOV R0,(R0)	;LOAD VALUE INTO ADDRESS
26	005720	TST (R0)+	;STEP TO NEXT ADDRESS
30	000772	BR 1\$;LOOP UNTIL DONE
32	010004	2\$: MOV R0,R4	;SAVE UPPER LIMIT
34	020001	3\$: CMP R0,R1	;CHECK IF AT LOWER LIMIT
* 36	001767	BEQ 1\$;BRANCH IF DONE
40	024000	CMP -(R0),R0	;CHECK DATA WRITTEN
42	001774	BEQ 3\$;BRANCH IF OK
44	000000	HALT	;ERROR
46	000772	BR 3\$;LOOP BACK

AFTER TOGGING THE PROGRAM LA=10**SET UPPER LIMIT**, START

NOTES: THE UPPER LIMIT ADDRESS OBTAINED FROM THE SWITCH

REGISTER MAY BE CHANGED DURING PROGRAM OPERATION. HOWEVER OCCASIONALLY THE PROGRAM MAY HALT BECAUSE OF 'SWITCH BOUNCE'. (THE BEST PROCEDURE WHEN CHANGING LIMITS IS TO STOP THE PROGRAM MAKE THE CHANGE AND CONTINUE.) THE LOWER LIMIT ADDRESS (12) MAY BE PATCHED TO ANY DESIRED ADDRESS.

6.4.2 TOGGLE-IN-PROGRAM #2

THE FOLLOWING IS ALSO A TOGGLE IN PROGRAM TO BE USED WITH TOGGLE-IN-PROGRAM #1 FOR MORE COMPLETE ADDRESS TESTING. THIS PROGRAM WRITES THE COMPLEMENT VALUE OF EACH ADDRESS INTO ITSELF STARTING WITH THE UPPER LIMIT AND CONTINUING TO THE LOWER LIMIT. AFTER ALL ADDRESSES HAVE BEEN WRITTEN EACH ADDRESS IS CHECKED FOR THE CORRECT CONTENTS STARTING WITH THE LOWER LIMIT ADDRESS AND CONTINUING TO THE UPPER LIMIT. TOGGLE IN THE FOLLOWING PATCHES TO THE PROGRAM ABOVE.

THESE ARE THE PATCHES TO TOGGLE-IN-PROGRAM #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
12	100		;CHANGE LOWER LIMIT
36	001404	BEQ 4\$;BRANCH TO PROGRAM #2

THESE ARE THE ADDITIONS TO TOGGLE-IN-PROGRAM #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
50	010402	4\$: MOV R4,R2	;GET UPPER LIMIT
52	005142	5\$: COM -(R2)	;COMPLEMENT ADDRESS
54	020201	CMP R2,R1	;CHECK IF AT LOWER LIMIT
56	001375	BNE 5\$;LOOP UNTIL DONE
60	020204	6\$: CMP R2,R4	;CHECK IF AT UPPER LIMIT
62	001755	BEQ 1\$;GO TO PROGRAM 1 IF DONE
64	010203	MOV R2,R3	;GET VALUE OF ADDRESS
66	005103	COM R3	;COMPLEMENT VALUE
70	020322	CMP R3,(R2)+	;CHECK ADDRESS
72	001772	BEQ 6\$;BRANCH IF OK
74	000000	HALT	;ERROR
76	000770	BR 6\$;GO CHECK NEXT ADDRESS

7.0 PROGRAM FUNCTIONAL FLOW CHARTS
ATTACHED

8.0 PROGRAM LISTING
ATTACHED

FLOW CHART

MAINDEC-11-DZQMC-C D-124K MEMORY EXERCISER. (16K VERSION)

COPYRIGHT 1976
DIGITAL EQUIPMENT CORPORATION

TABLE OF CONTENTS

PAGE 01	DEFINITIONS, TRAP CATCHER, STARTING ADDRESSES.
PAGE 02	RESTART AND RESTORE ROUTINES
PAGE 04	POWER FAIL ROUTINES
PAGE 05	COMMON TAGS
PAGE 06	SETUP
PAGE 08	MAP MEMORY
PAGE 09	MEMORY BYTE MAP ROUTINE
PAGE 12	MAP PARITY REGISTERS
PAGE 13	MAP PARITY MEMORY
PAGE 14	TEST PARITY REGISTERS
PAGE 15	USER PARAMETER SELECTION SECTION
PAGE 16	START1: START OF PASS
PAGE 17	SECTION 1: ADDRESS TESTS. TEST 1
PAGE 18	TEST 2
PAGE 19	TEST 3
PAGE 20	TEST 4
PAGE 21	TEST 5
PAGE 22	SECTION 2: WORSE CASE NOISE TESTS. TEST 6
PAGE 23	TEST 7
PAGE 24	TEST 10
PAGE 25	TEST 11
PAGE 26	TEST 12
PAGE 27	TEST 13: 1 XOR 8
PAGE 29	TEST 14: 3 XOR 9

TABLE OF CONTENTS

PAGE 33 TEST 16: 3 XOR 9 (FOR PARITY)
 PAGE 35 TEST 17: 3 XOR 9 (FOR PARITY)
 PAGE 37 TEST 20: 8 XOR 13
 PAGE 39 TEST 21: PARITY BYTE TEST
 PAGE 43 TEST 22
 PAGE 44 TEST 23: EXICUTE DATI, DATO
 PAGE 45 TEST 24: EXICUTE DATI, DATOB (LO BYTE)
 PAGE 46 TEST 25: EXICUTE DATI, DATOB (HI BYTE)
 PAGE 47 TEST 26: EXICUTE DATIP, DATO
 PAGE 48 TEST 27: EXICUTE DATIP, DATOB (LO BYTE)
 PAGE 49 TEST 30: EXICUTE DATIP, DATOB (HI BYTE)
 PAGE 50 TEST 31: "BRANCH GOBBLE"
 PAGE 51 DONE
 PAGE 52 END OF PASS
 PAGE 53 MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES
 PAGE 55 SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS
 PAGE 56 RELOCATION SUBROUTINES
 PAGE 58 PARITY ROUTINES
 PAGE 60 SPECIAL PRINTOUT ROUTINES

E02

MAINDEC-11-DZQMC-C 0-124K MEMORY EXERCISER. (16K VERSION)
DEFINITIONS, TRAP CATCHER, STARTING ADDRESSES.

DECFL0 VER 00.12 08-OCT-76 13:38 PAGE 01

SEQ 0017

```
*****  
* SWITCH SETTINGS AND *  
* BASIC DEFINITIONS *  
* *  
*****
```

```
*****  
* TRAP CATCHER AND *  
* STARTING ADDRESSES *  
* *  
.=0
```



```

-----
/ RESTART FLAG \ YES *****
( RS=0 )? -----> *START1(16)*
\ \ *****
-----
I NO
V RESLDR(57)
*****
** RESTORE LOADERS **
** *****
I
V
*****
**HALT **

```

.=560


```

*****
**$PWRDN **
*****
  I
  V
*****
* $ILLUP -> VECTOR *
* SAVE REGISTERS *
* $PWRDN -> VECTOR *
*****
  I
  V
*****
**HALT **
*****

```

```

*****
**$PWRUP **
*****
  I
  V
*****
* WAIT LOOP FOR TTY *
* RESTORE REGISTERS *
* $PWRDN -> VECTOR *
*****
  I
  V
***** SPRINT(61)
/ TYPE POWER FAIL /
/ MESSAGE /
*****
  I
  V
*****
**RETURN **

```

```

*****
**$ILLUP **
*****
  I
  V
*****
**HALT **
*****

```

.=1100

```
*****  
* STANDARD 'SYSMAC' *  
* COMMON TAGS *  
* *  
*****
```

```
*****  
*COMMON TAGS FOR THIS *  
* PROGRAM *  
* *  
*****
```

```
*****  
* RELATIVE ADDRESSING *  
* TABLE. ERROR DATA *  
* POINTER *  
*****
```

```
*****  
* MEMORY PARITY WORSE *  
* CASE PATTERNS TABLE *  
* *  
*****
```

```
*****  
* MEMORY PARITY *  
*REGISTER ADDRESS AND *  
* MAP TABLE *  
*****
```

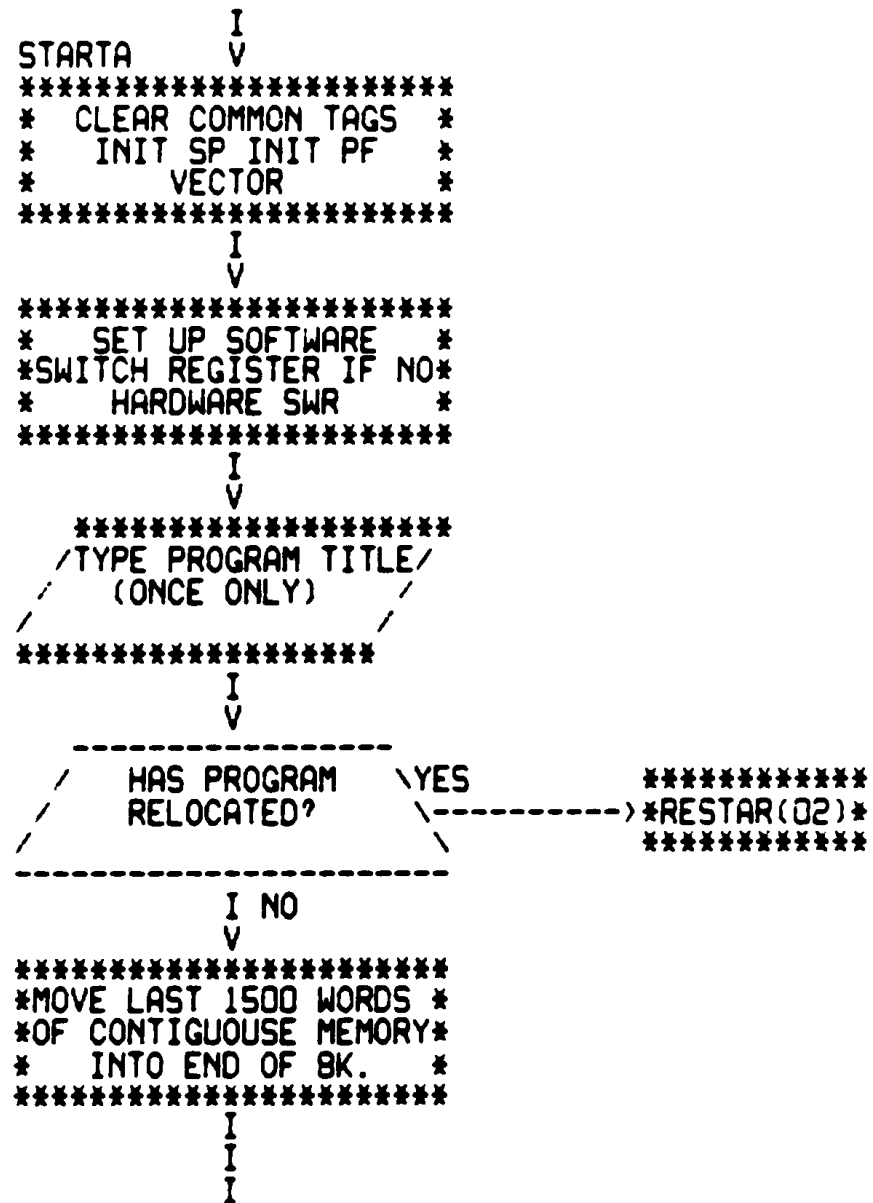
```
*****  
*ERROR MESSAGE POINTER*  
* TABLE *  
* *  
*****
```

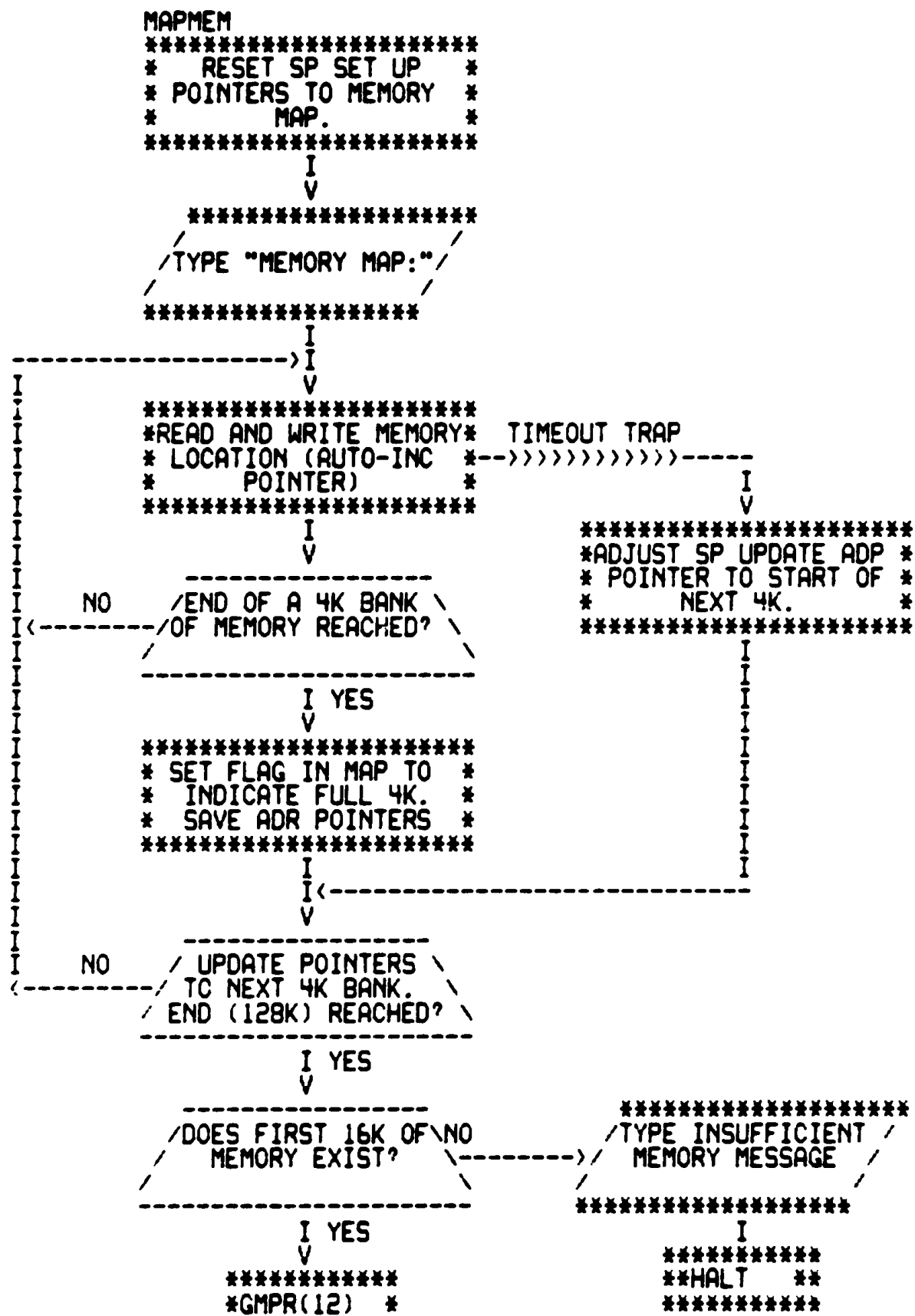
SA=204

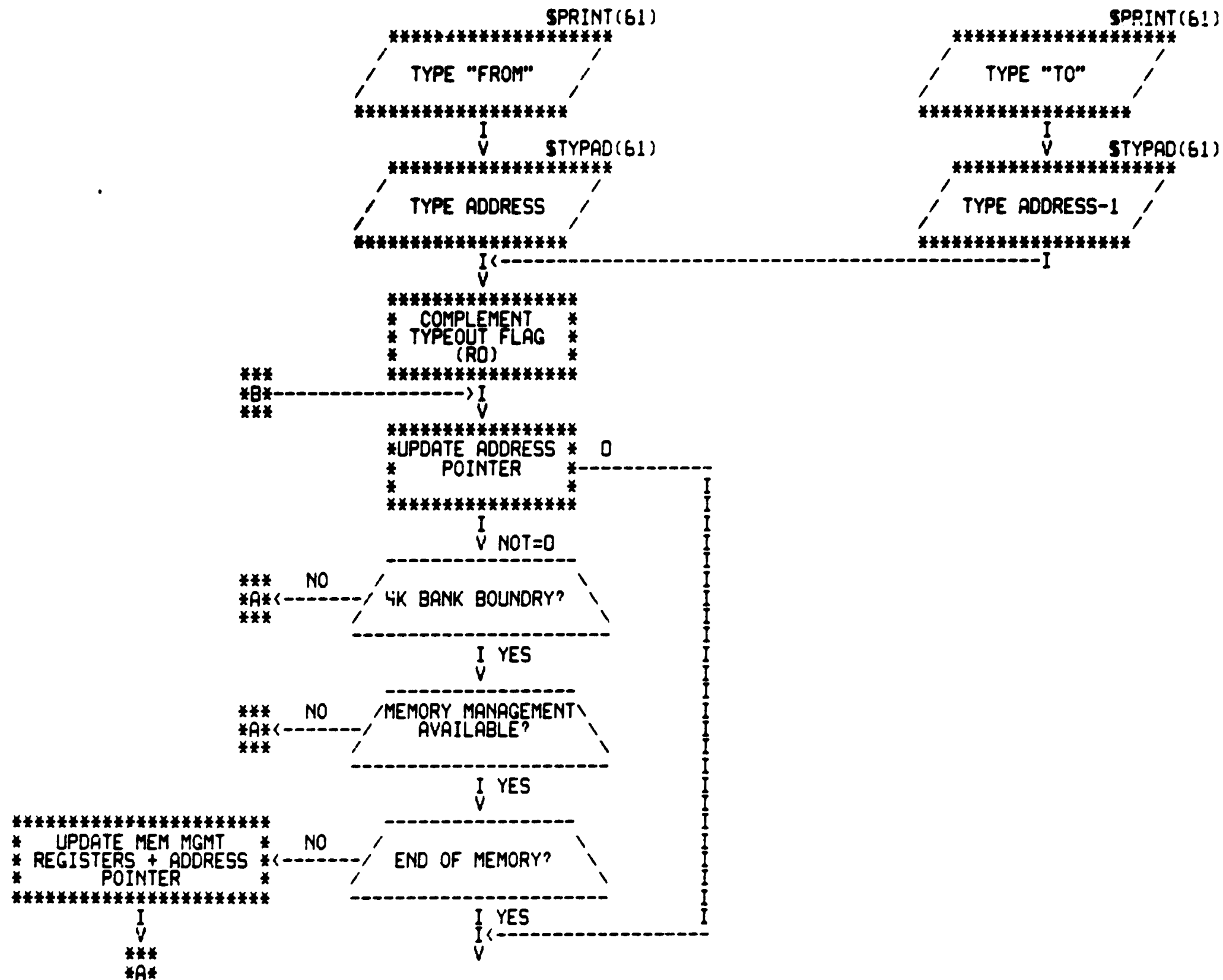
```
*****  
**SELECT **  
*****  
I  
*****  
* SET FLAG FOR *  
* SELECTING *  
* PARAMETERS *  
*****
```

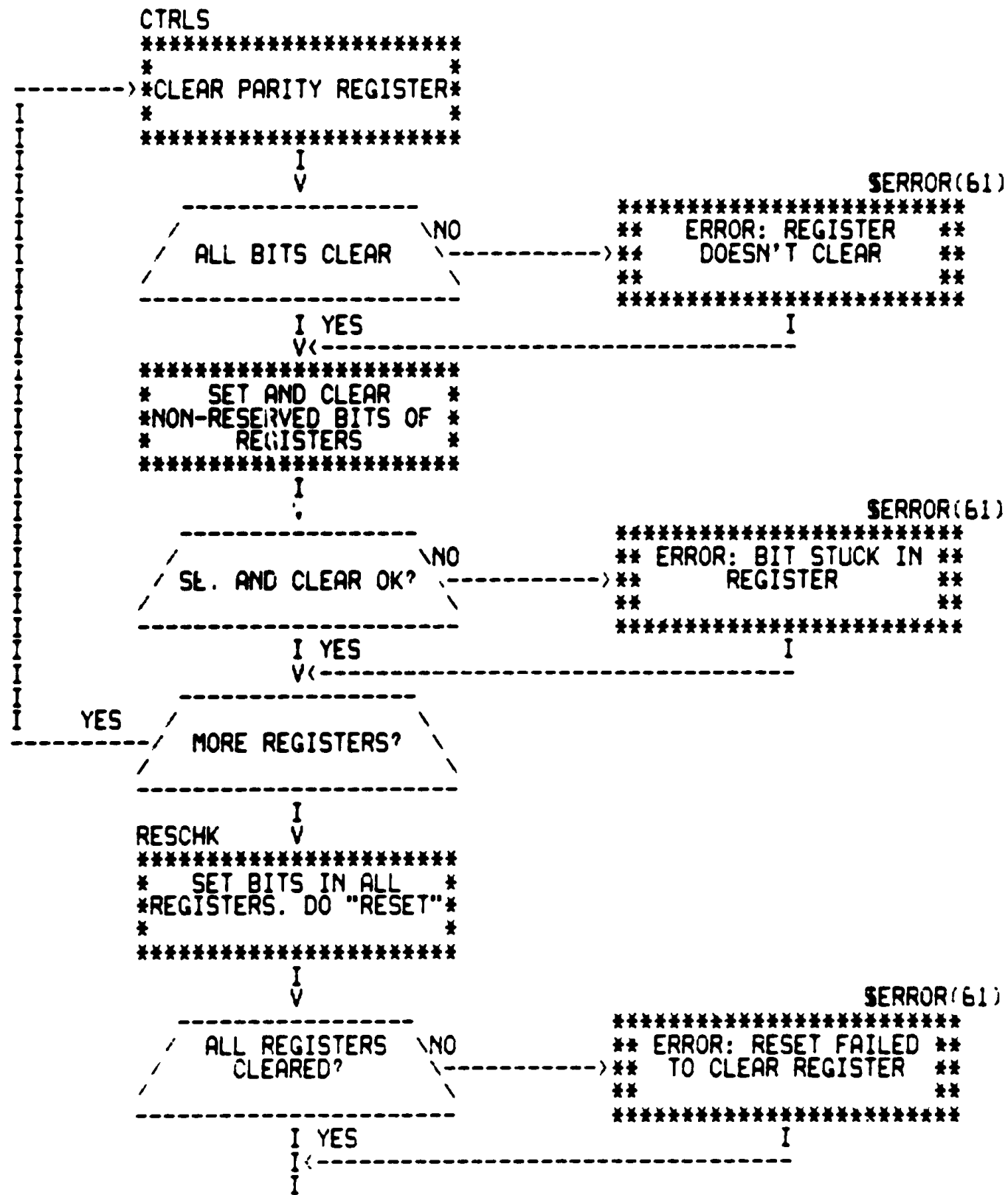
SA=200

```
*****  
**START **  
*****  
I  
*****  
*CLEAR FLAG FOR *  
* SELECTING *  
* PARAMETERS *  
*****
```










```

MANUL2
*****
* MAKE NECESSARY *
*ADJUSTMENTS TO FIRST *
* AND LAST ADDRESSES *
*****
  I
  I
  V
*****
**START1 **
*****
  I
  I
  I
  V
START1
*****
*INITIALIZE EVERYTING *
* FOR A NEW PASS *
* *
*****
  I
  I
  I
  I
  I

```

```

TST1          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
*****
*   WRITE PHYSICAL   *
* ADDRESS VALUE IN EACH*
*   WORD LOCATION    *
*****
          I
          V          MMUP(54)
*****
MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
          **                   **
          *****
          IDONE
          V          INITDN(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
          /-----\
          / DOES EACH \ NO
          / LOCATION HAVE \
          / ADDRESS VALUE? \
          \-----/
          I YES
          I<-----I
          V          MMDOWN(54)
*****
MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
          **                   **
          *****
          IDONE
          I
          I
          I

```

```

*****
**ERROR: ADDRESS VALUE **
** NOT IN LOCATION **
*****

```

```

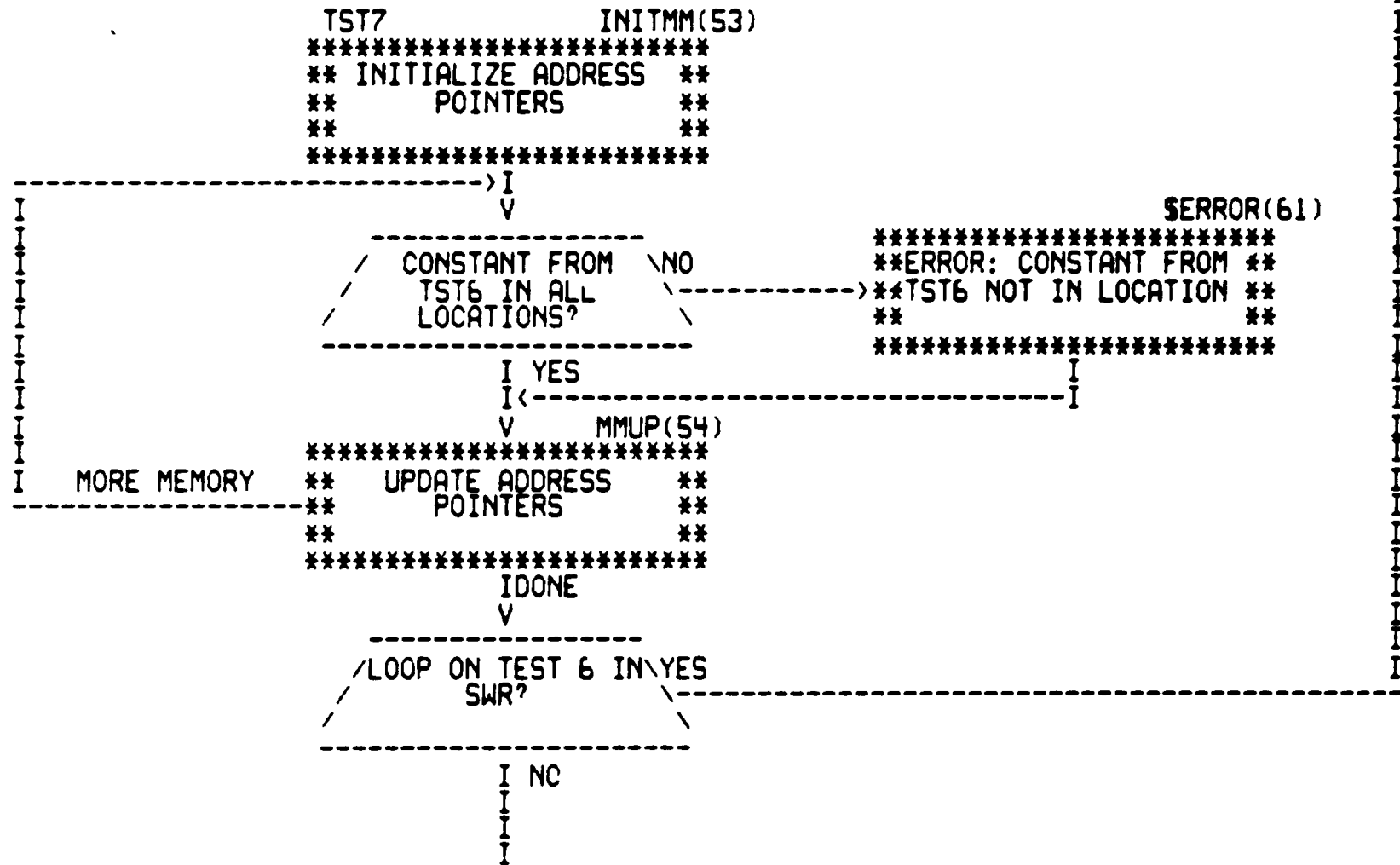
TST2          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
V
*****
*   WRITE PHYSICAL   *
* ADDRESS VALUE IN EACH*
*   BYTE LOCATION   *
*****
I
V          MMUP(54)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
IDONE
V          INITDN(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
V
*****
/ DOES EACH BYTE \ NO
/ LOCATION HAVE  \
/ ADDRESS VALUE? \
*****
I YES
I<-----I
V          MMDOWN(54)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
IDONE
I
I
I

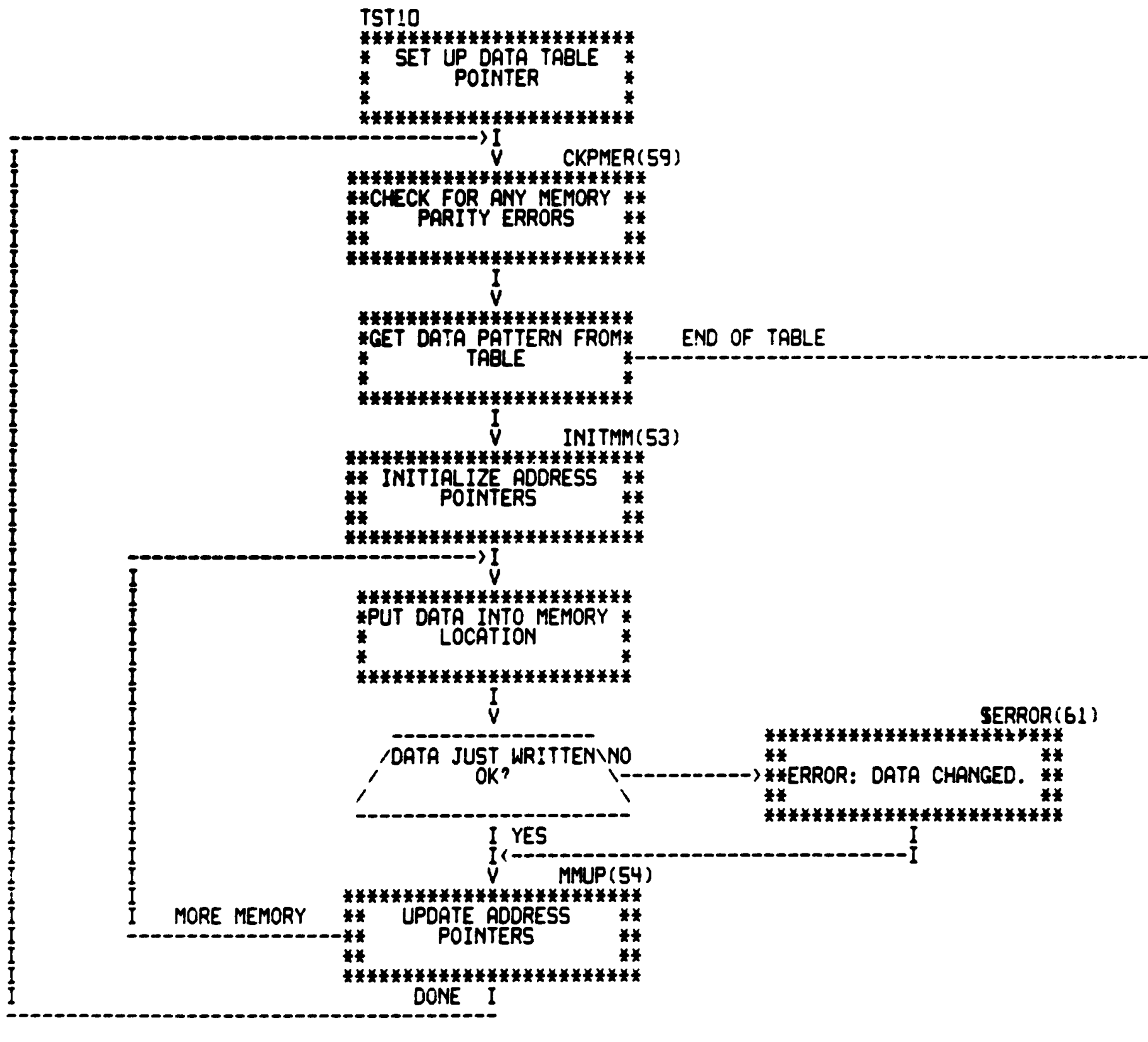
```

```

*****
**ERROR: ADDRESS VALUE **
**NOT IN BYTE LOCATION **
**                   **
*****

```



```
-----  
TST11      I      SETCON(55)  
           V  
*****  
**PUT ALL ONE'S IN ALL **  
**      MEMORY      **  
**                  **  
*****  
           I  
           V      INITMM(53)  
*****  
** INITIALIZE ADDRESS **  
**   POINTERS   **  
**              **  
*****  
-----> I  
           V      ROTATE(55)  
*****  
** CLEAR C-BIT AND **  
**ROTATE IT THROUGH TWO**  
**   BYTES   **  
*****  
           I  
           V  
-----  
/ C-BIT CLEAR AND \ NO  
-1 IN MEMORY \  
LOCATION? \  
-----  
I YES \ I  
I <----- I  
           V      MMUP(54)  
*****  
** UPDATE ADDRESS **  
**   POINTERS   **  
**              **  
*****  
I MORE MEMORY  
-----  
IDONE  
I  
I
```

```
SERROR(61)  
*****  
** ERROR: ROTATING 0 **  
**      FAILED.      **  
**                  **  
*****
```



```

TST13          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTER          **
**                   **
*****
----->I
          V
*****
**WRITE 128 WORD BLOCK **
** WITH 0,-1,0,-1... **
** THEN NEXT BLOCK COM. **
*****
          I
          V          MMUP(54)
MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS          **
          **                   **
          *****
          IDONE
          V          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS          **
**                   **
*****
----->I
          V
          /-----\
          / 128 WORD BLOCKS \ NO
          / WRITTEN WITH    \
          / 0,-1,0,-1...?   \
          \-----/
          I YES
          I <----- I
          V          MMUP(54)
MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS          **
          **                   **
          *****
          IDONE
          I
          I
          I
          I
          I

```

```

*****
** ERROR: 1 XOR 8 **
**   PATTERN FAILED **
**                   **
*****

```



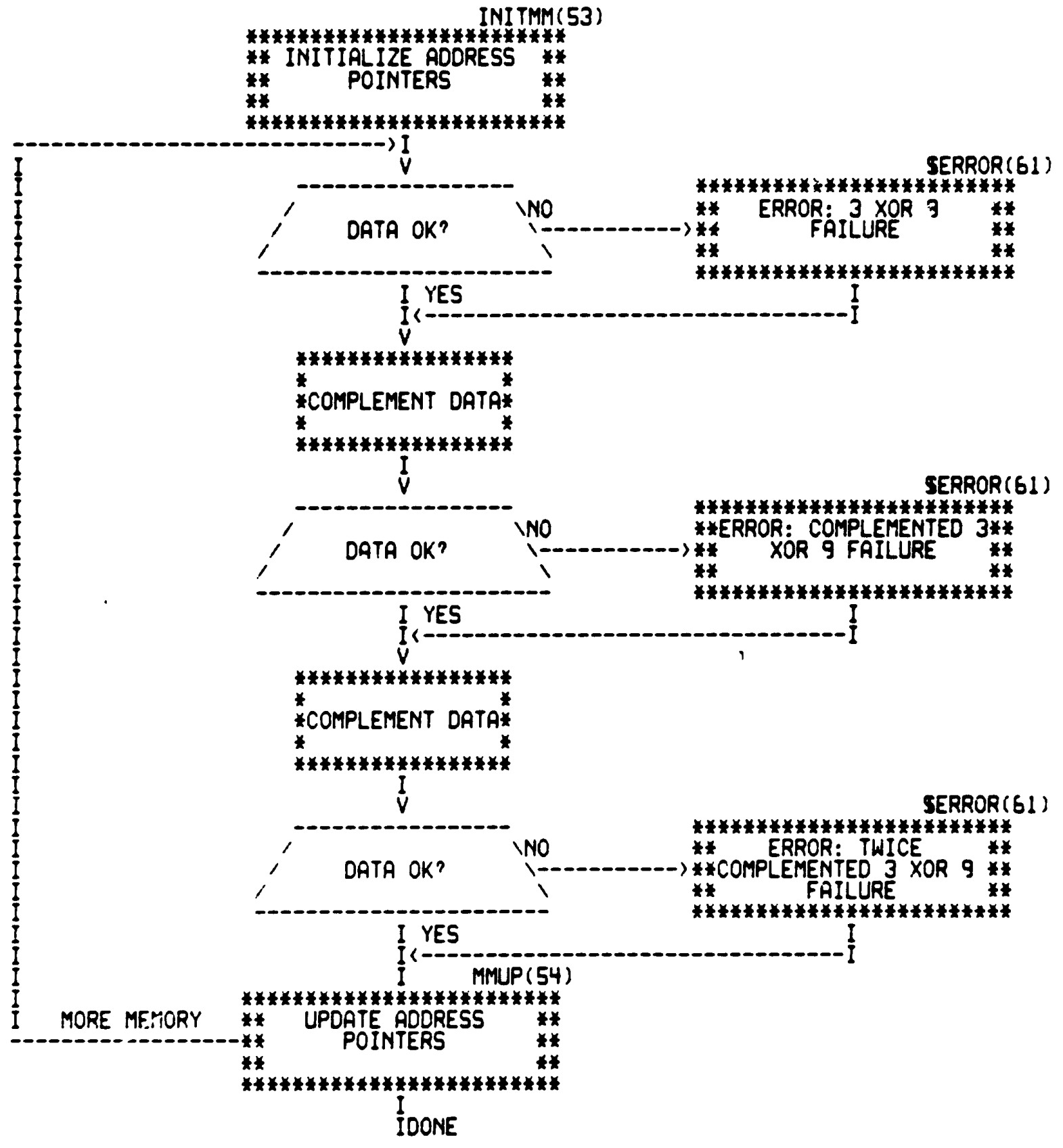
```

TST15          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V      W3X9(55)
*****
**   WRITE 256. WORD  **
**   BLOCKS WITH     **
** -1,-1,-1,-1,0,0,0 **
*****
          I
          V      MMUP(54)
*****
I MORE MEMORY **   UPDATE ADDRESS   **
----- **   POINTERS             **
**                   **
*****
          IDONE
          V      INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
          /256. WORD BLOCKS \NO
          / WRITTEN WITH  \----->
          / -1,-1,-1,-1,0,0,0 \
          /-----
          I YES
          I<-----I
          V      MMUP(54)
*****
I MORE MEMORY **   UPDATE ADDRESS   **
----- **   POINTERS             **
**                   **
*****
          IDONE
          I
          I
          I
          I

```

** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
**

\$ERROR(51)



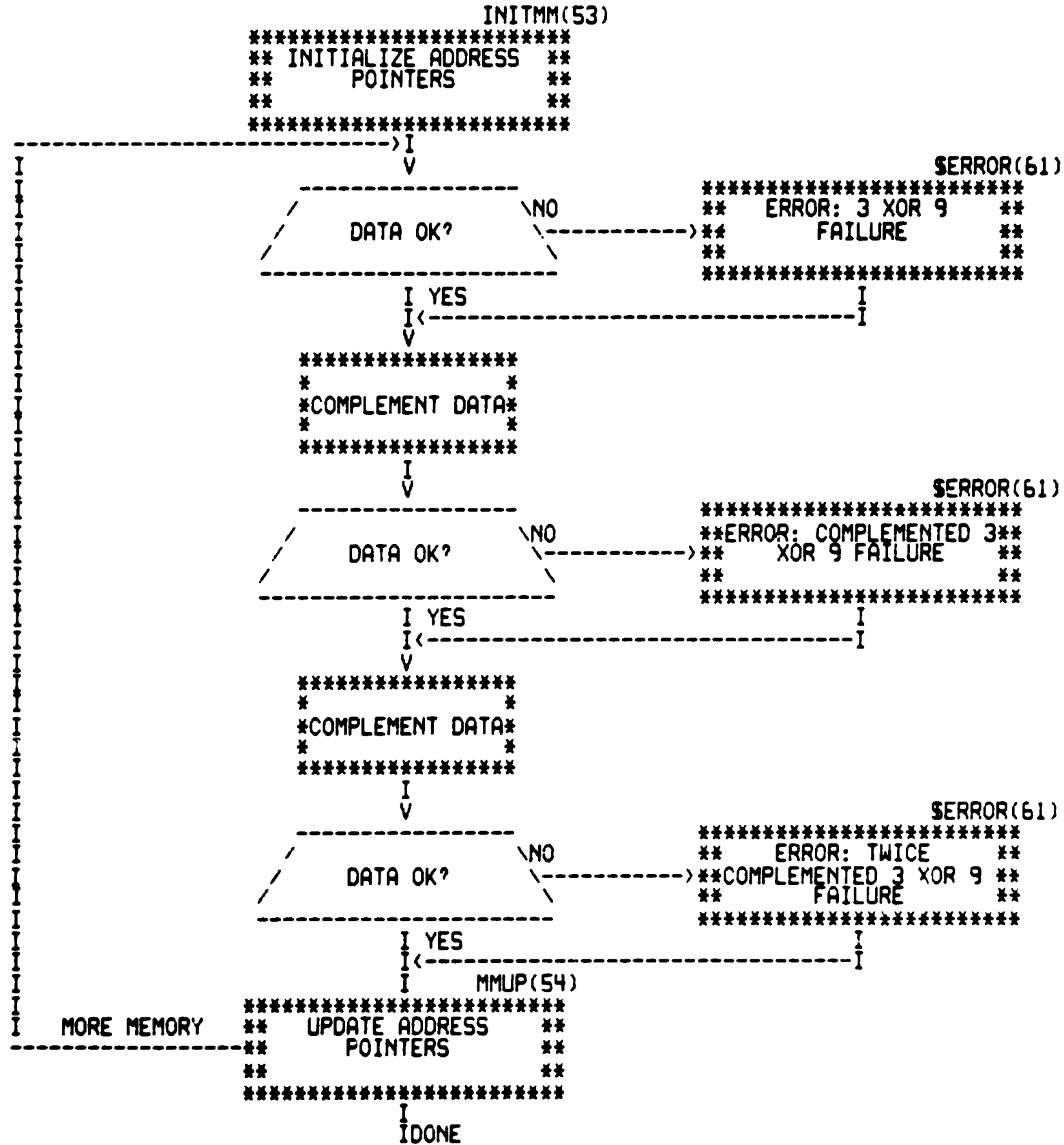
```

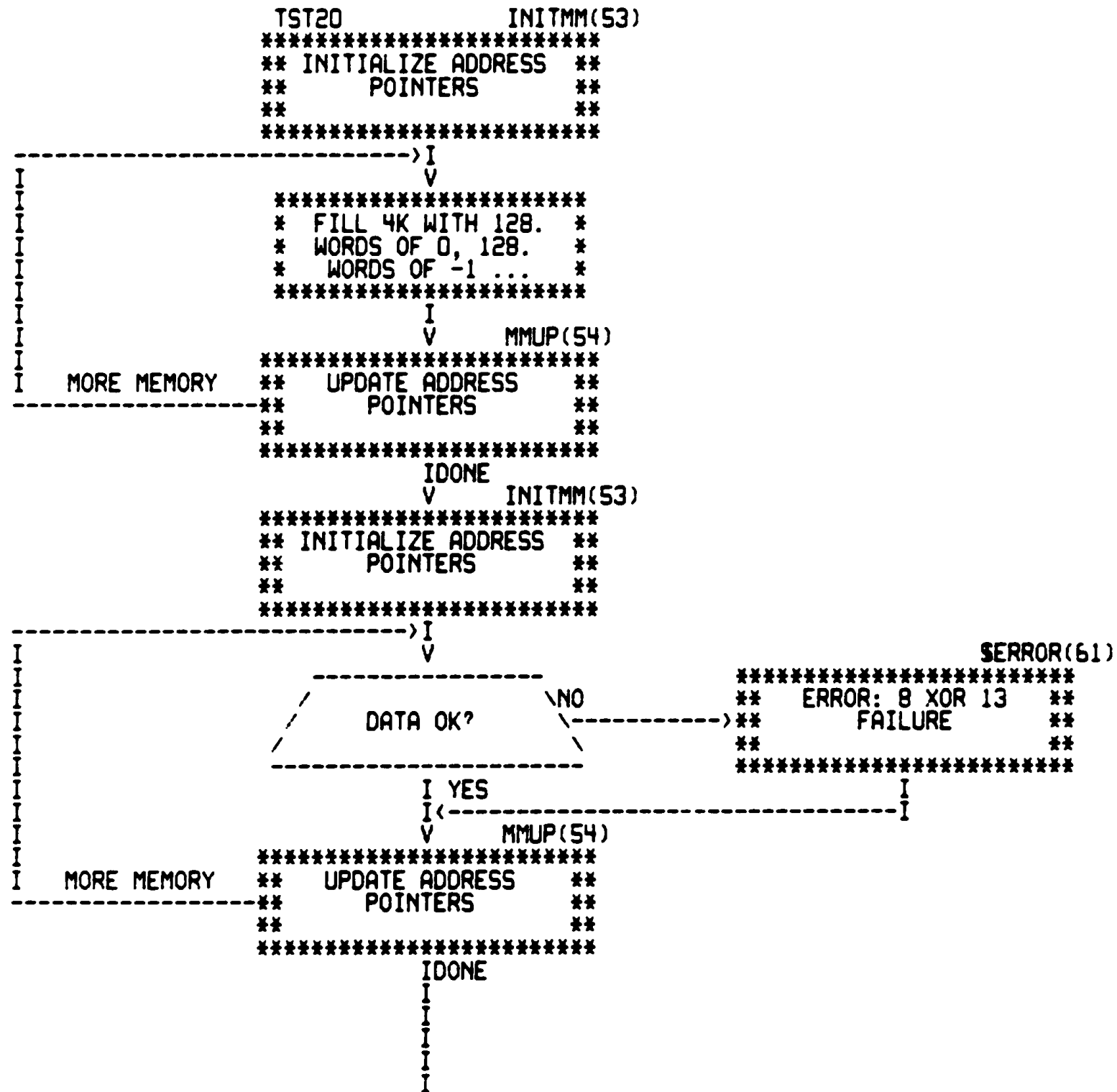
TST17          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
V          W3X9(55)
*****
**   WRITE 256. WORD  **
** BLOCKS WITH -1 AND **
**          401       **
*****
I
V          MMUP(54)
*****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
*****
IDONE
V          INITMM(53)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
V
/256. WORD BLOCKS \NO
/WRITTEN WITH -1 AND\----->
/          401?     \
-----
I YES
I<-----I
V          MMUP(54)
*****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
*****
IDONE
I
I
I
I

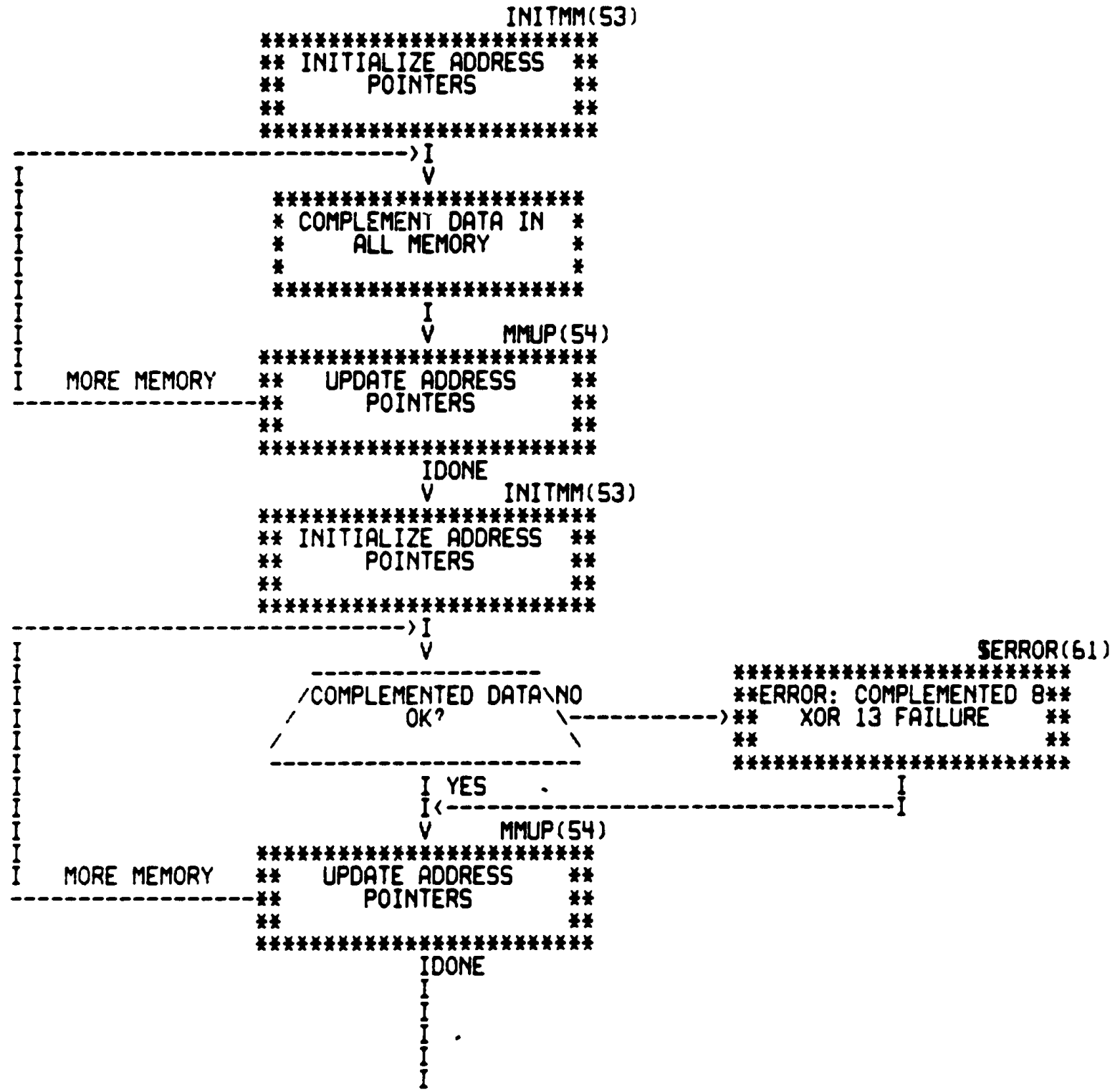
```

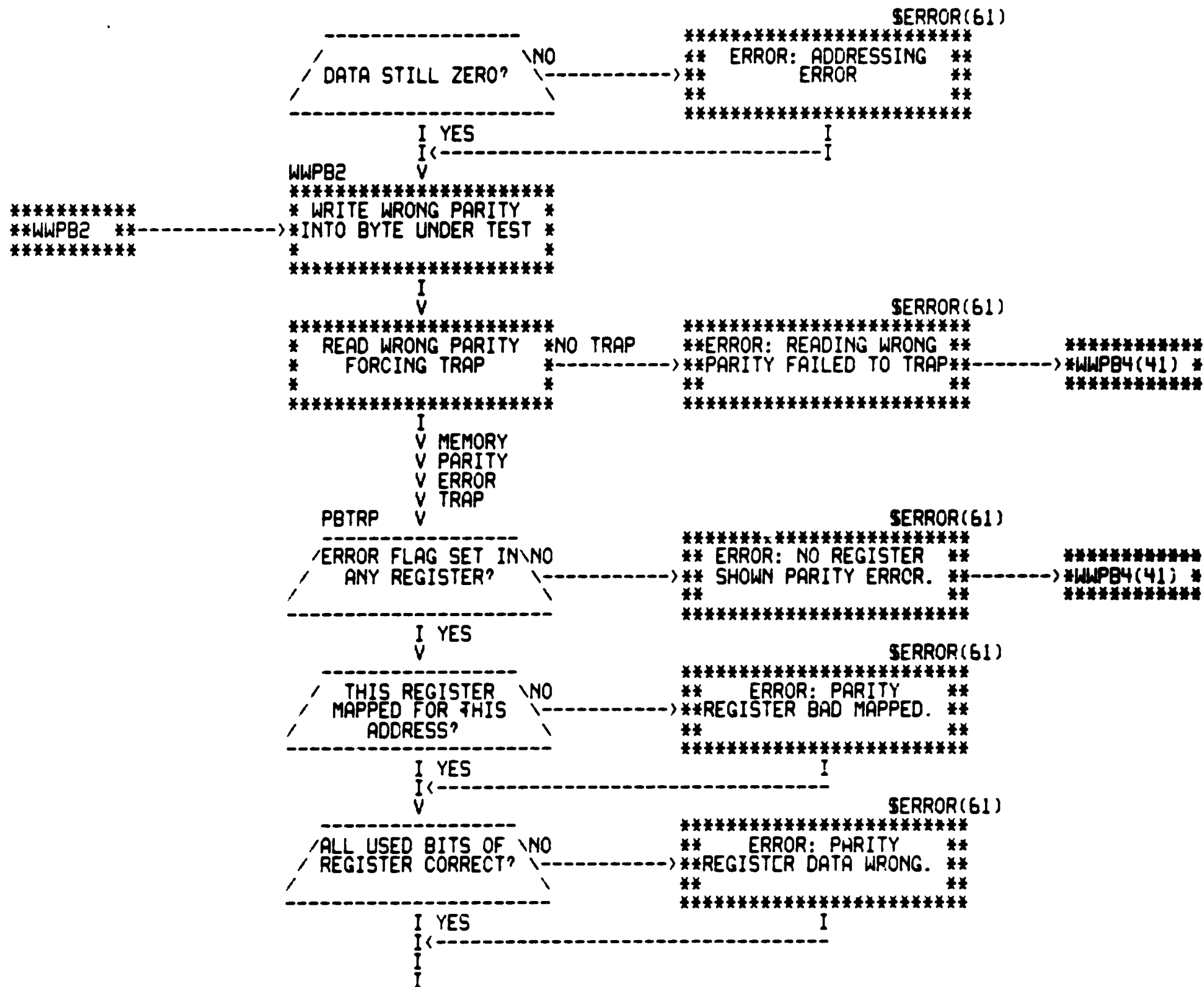
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

\$ERROR(61)

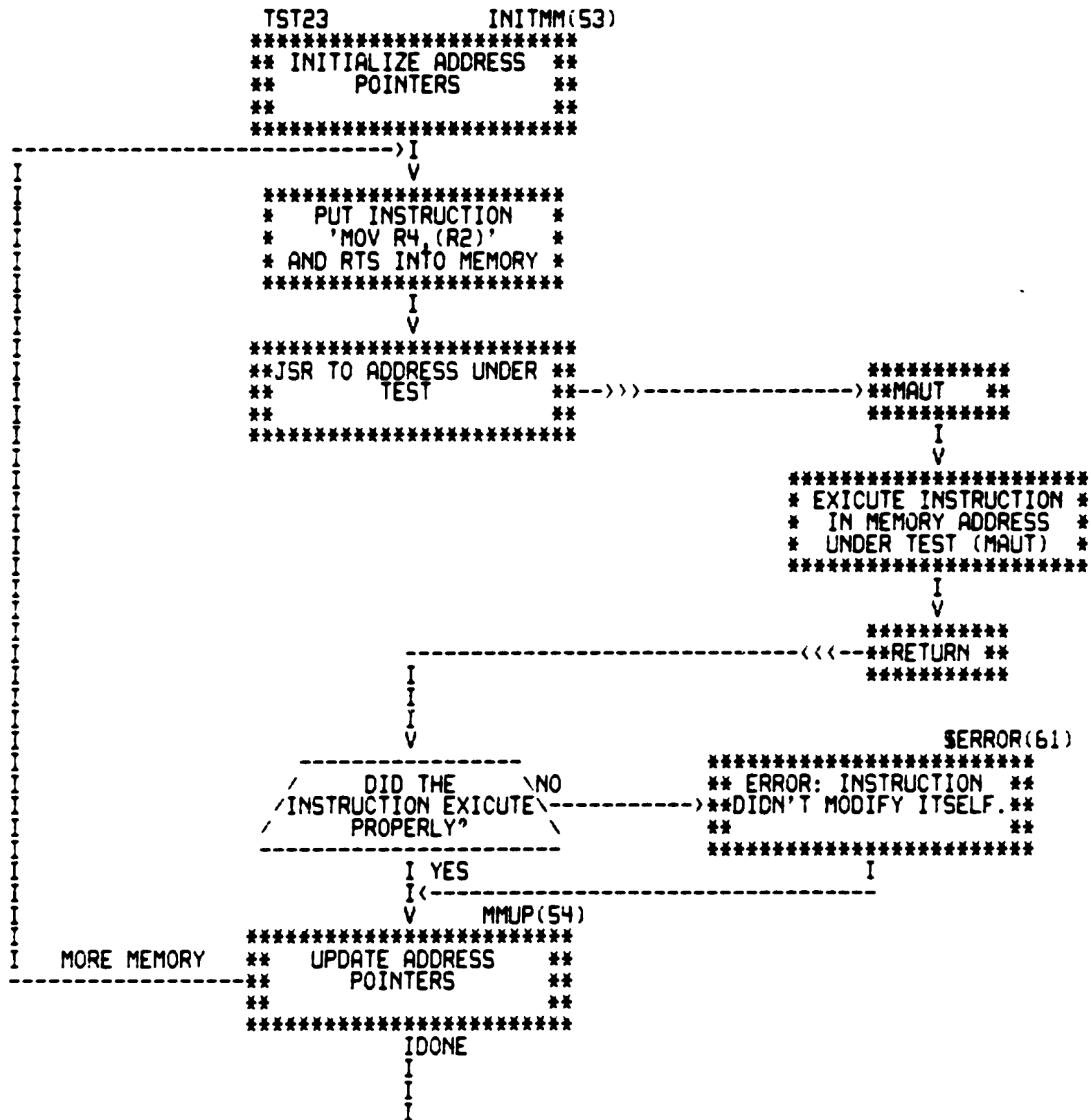


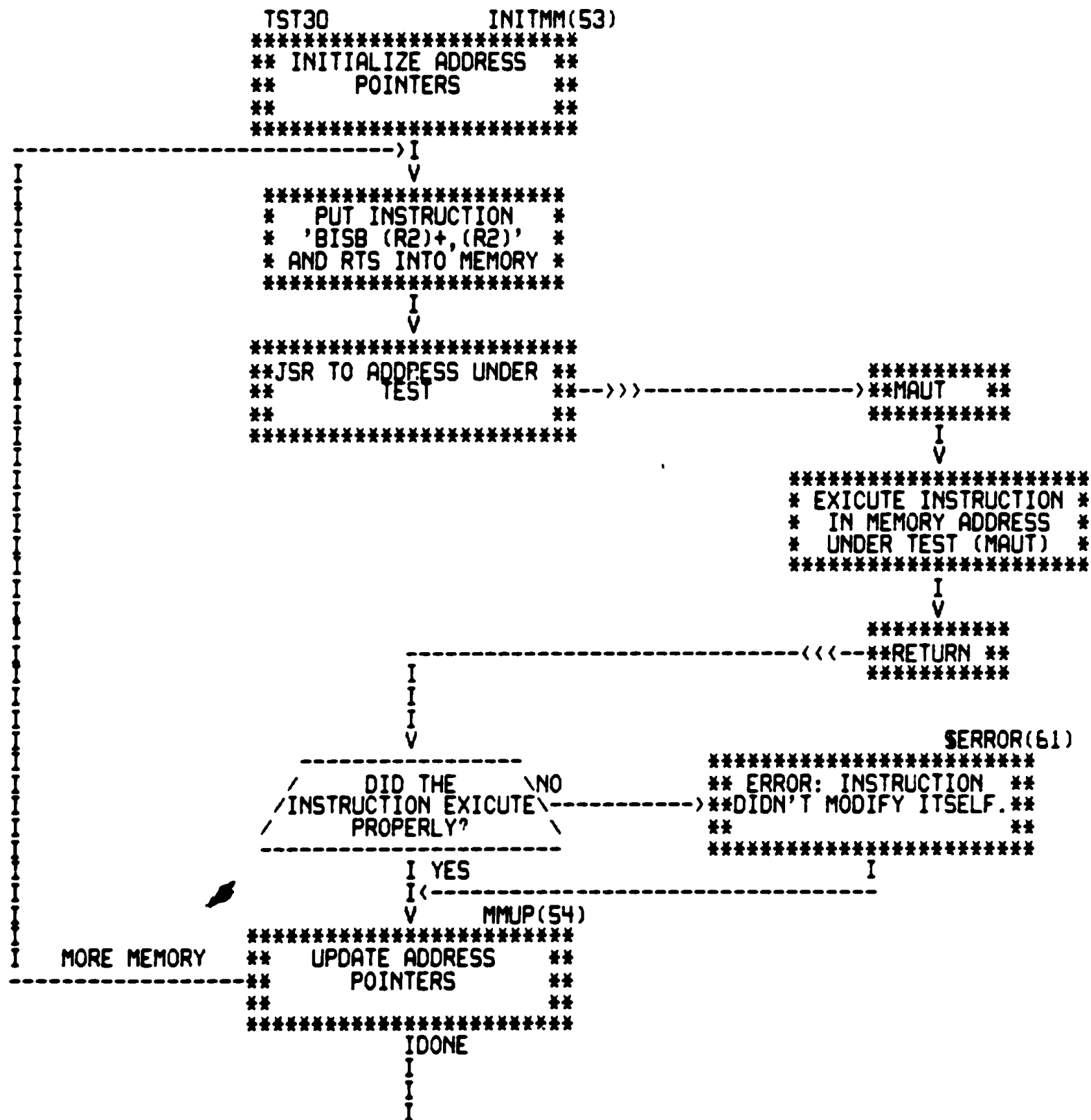


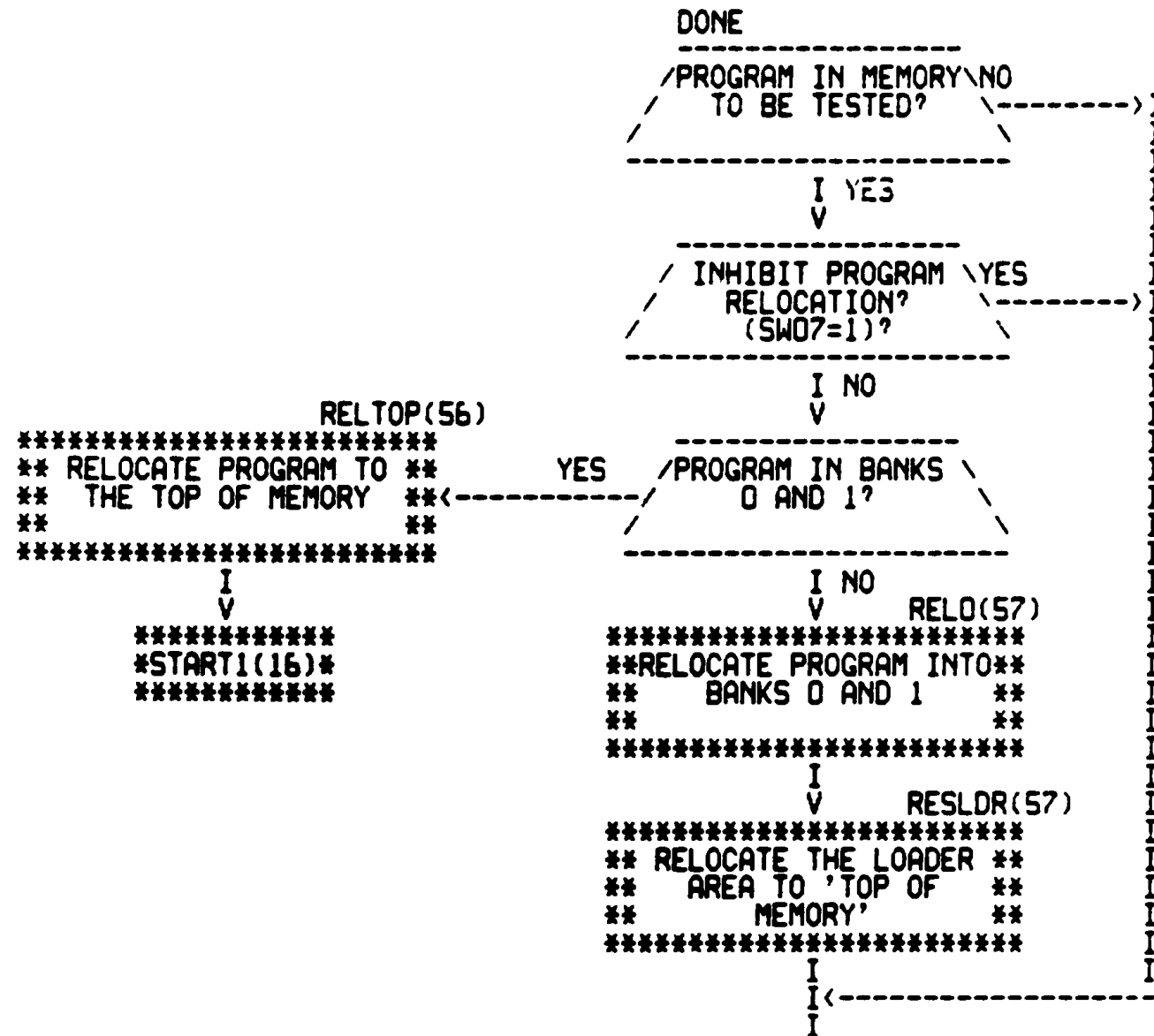


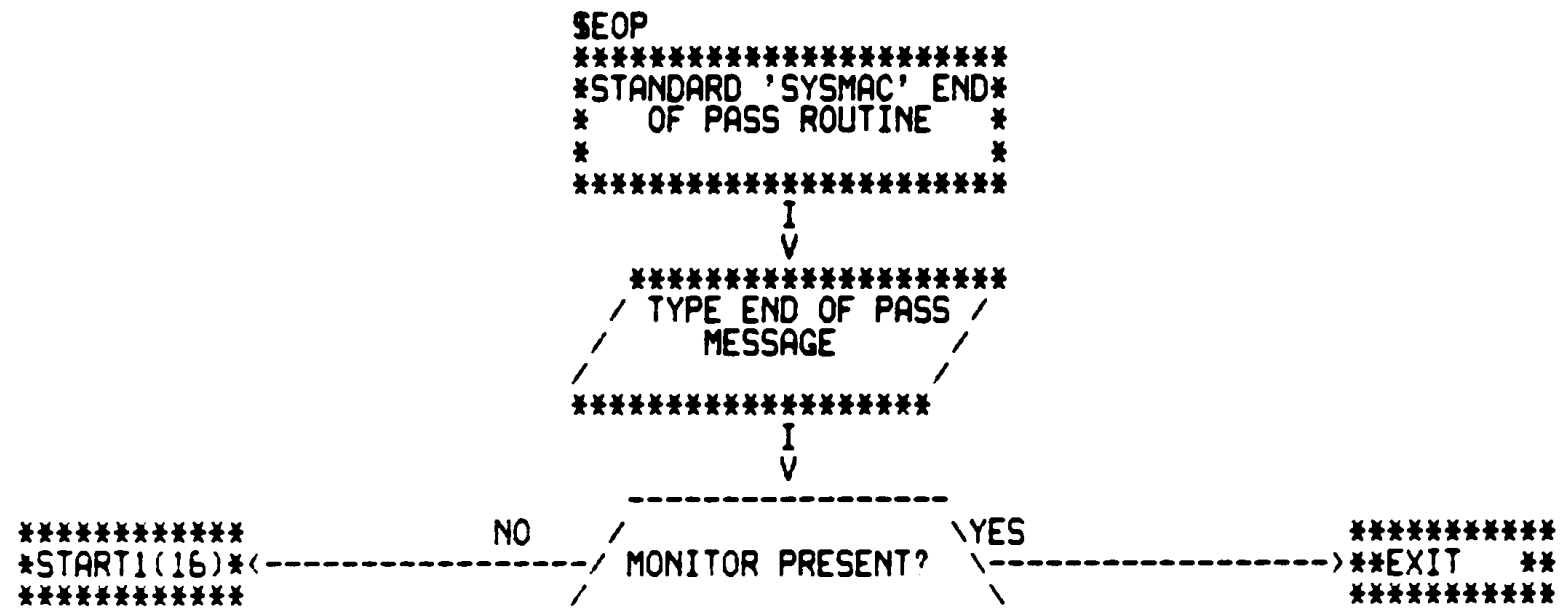



```
MMUP(54)
*****
***** MORE MEMORY ** UPDATE ADDRESS **
*WWPBT(39) *<-----** POINTERS **
*****
*****
IDONE
V MAMF(58)
*****
** RESET ALL PARITY **
** REGISTERS **
** **
*****
I
I
I
```








```

*****
**PHYADR **
*****
  I
PHYADR V
*****
* GET VIRTUAL *
* ADDRESS (FROM *
* R2) *
*****
  I
  V
-----
MEMORY MANAGEMENT\NO
AVAILABLE
-----
  I YES
  V
*****
*ADD INDEX FACTOR FROM*
* KIPAR2 TO GET *
* PHYSICAL ADR *
*****
  I<-----
  V
*****
**RETURN **
*****

```

```

*****
**BANKNO **
*****
  I
BANKNO V
*****
* CALCULATE BANK # *
* USING TEST MAP BANK *
* POINTER *
*****
  I
  V
*****
**RETURN **
*****

```

```

*****
**SETCON **
*****
  I
SETCON V INITMM(53)
*****
** INITIALIZE ADDRESS **
** POINTERS **
*****
----->I
  I
  V
*****
* PUT THE CONTENTS OF *
* RO INTO MEMORY *
*
*****
  I
  V MMUP(54)
*****
** UPDATE ADDRESS **
** POINTERS **
*****
I MORE
I-----
I MEMORY
  I
  V IDONE
  V
*****
**RETURN **
*****

```

```

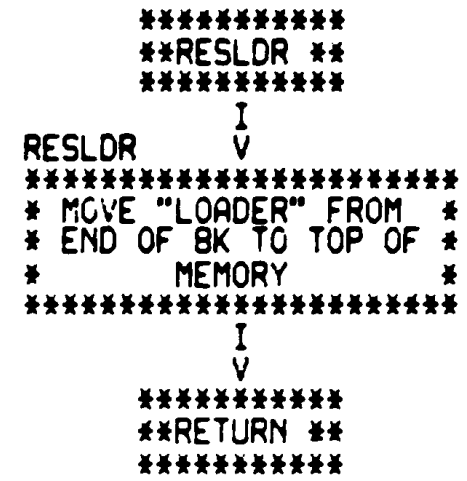
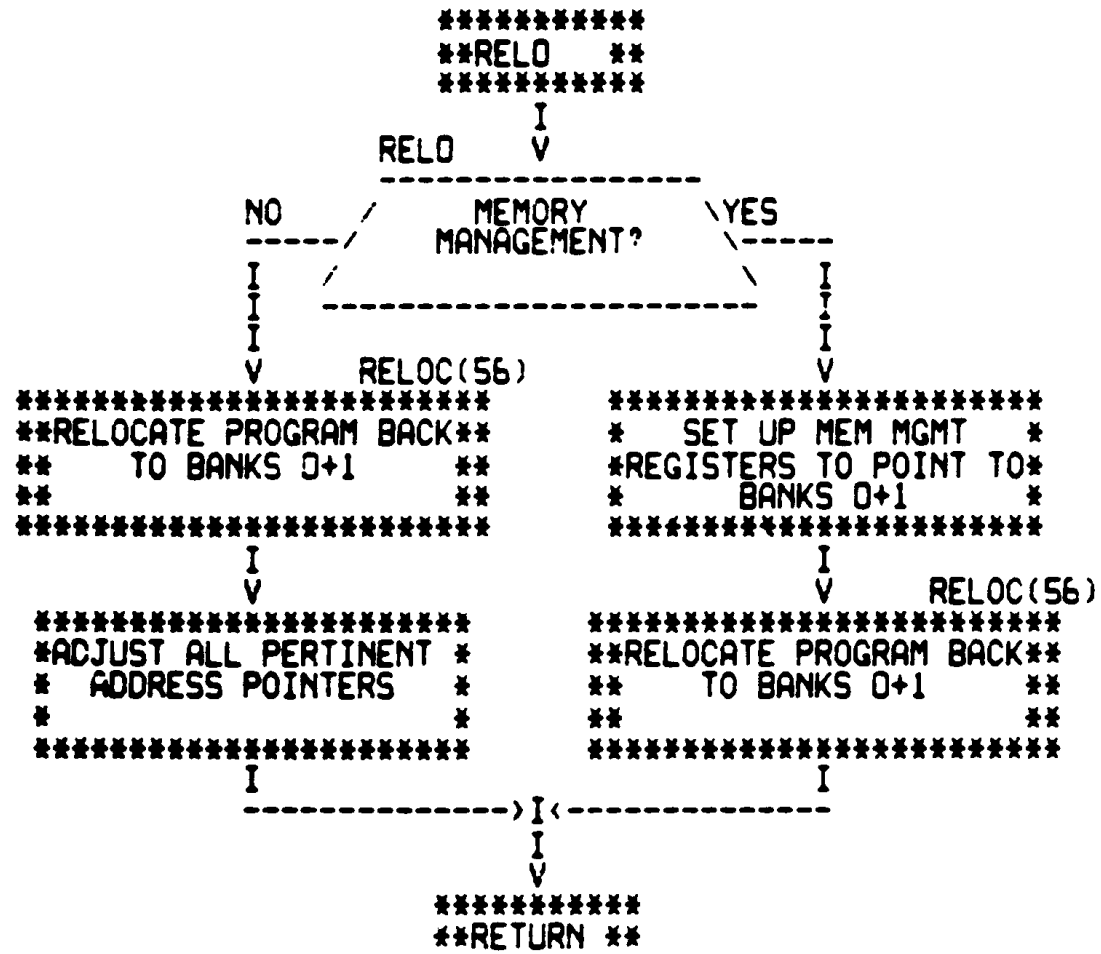
*****
**ROTATE **
*****
  I
ROTATE V
*****
*ROTATE C-BIT THROUGH *
* 16 BIT WORD. *
*
*****
  I
  V
*****
**RETURN **
*****

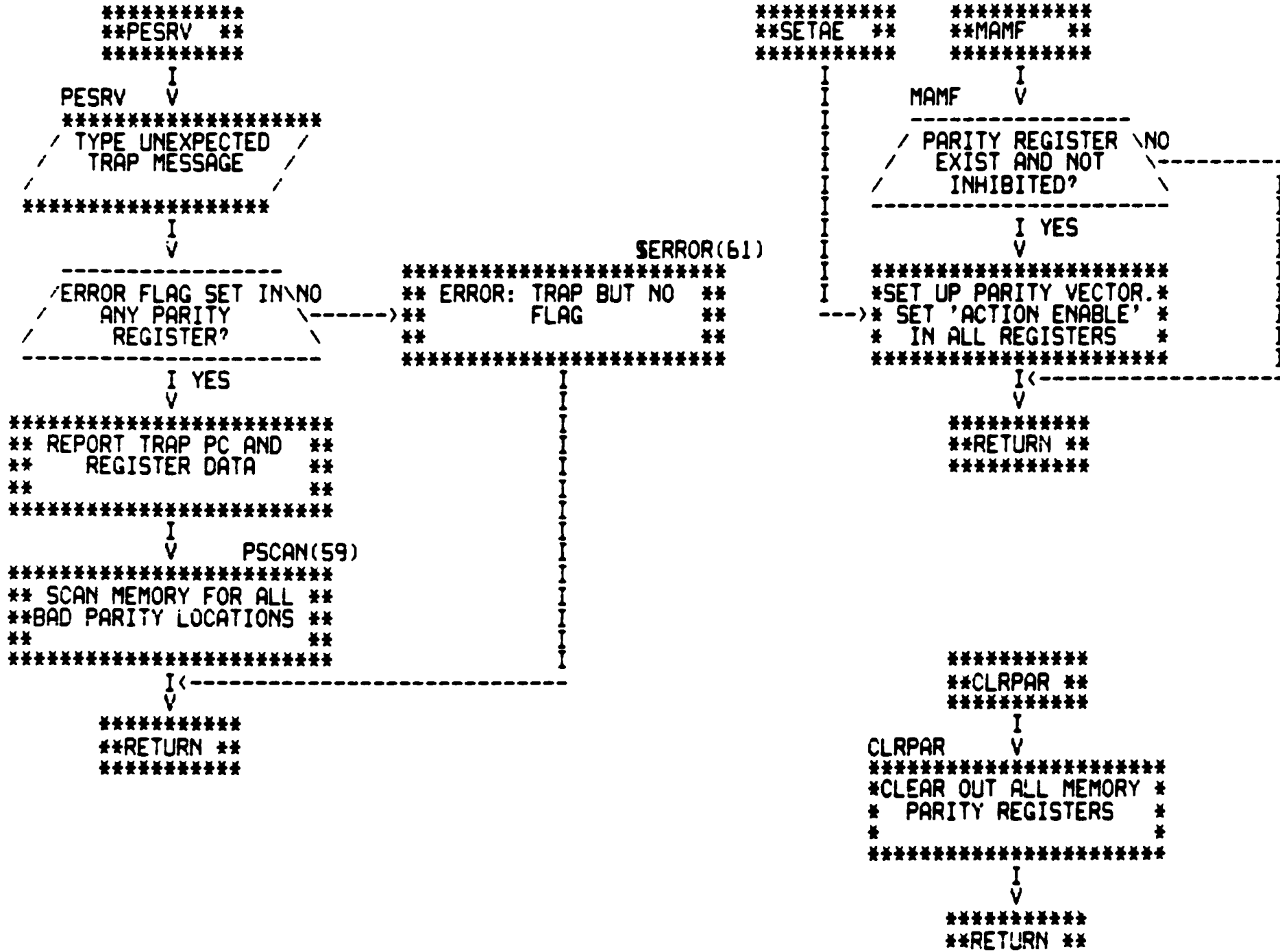
```

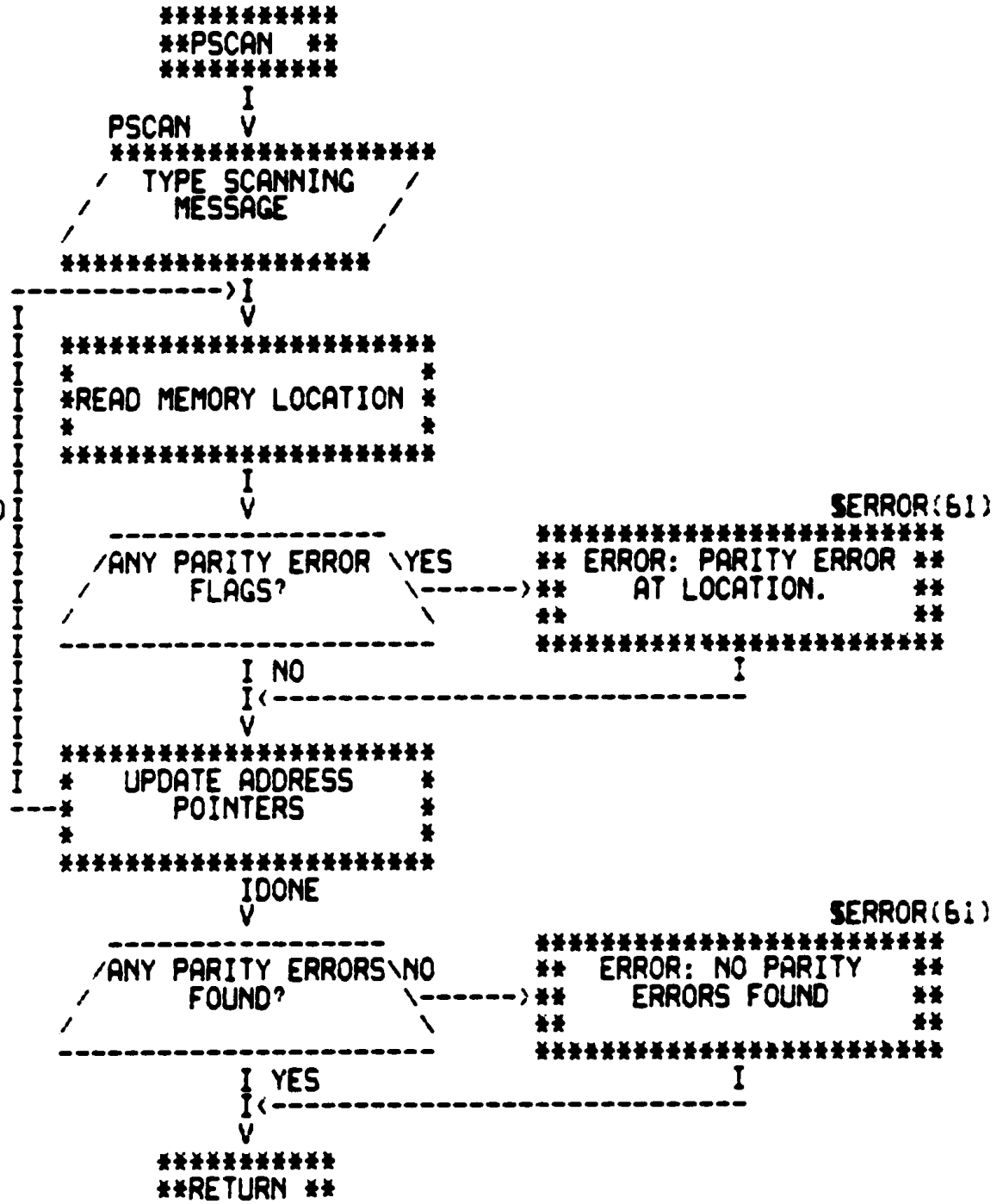
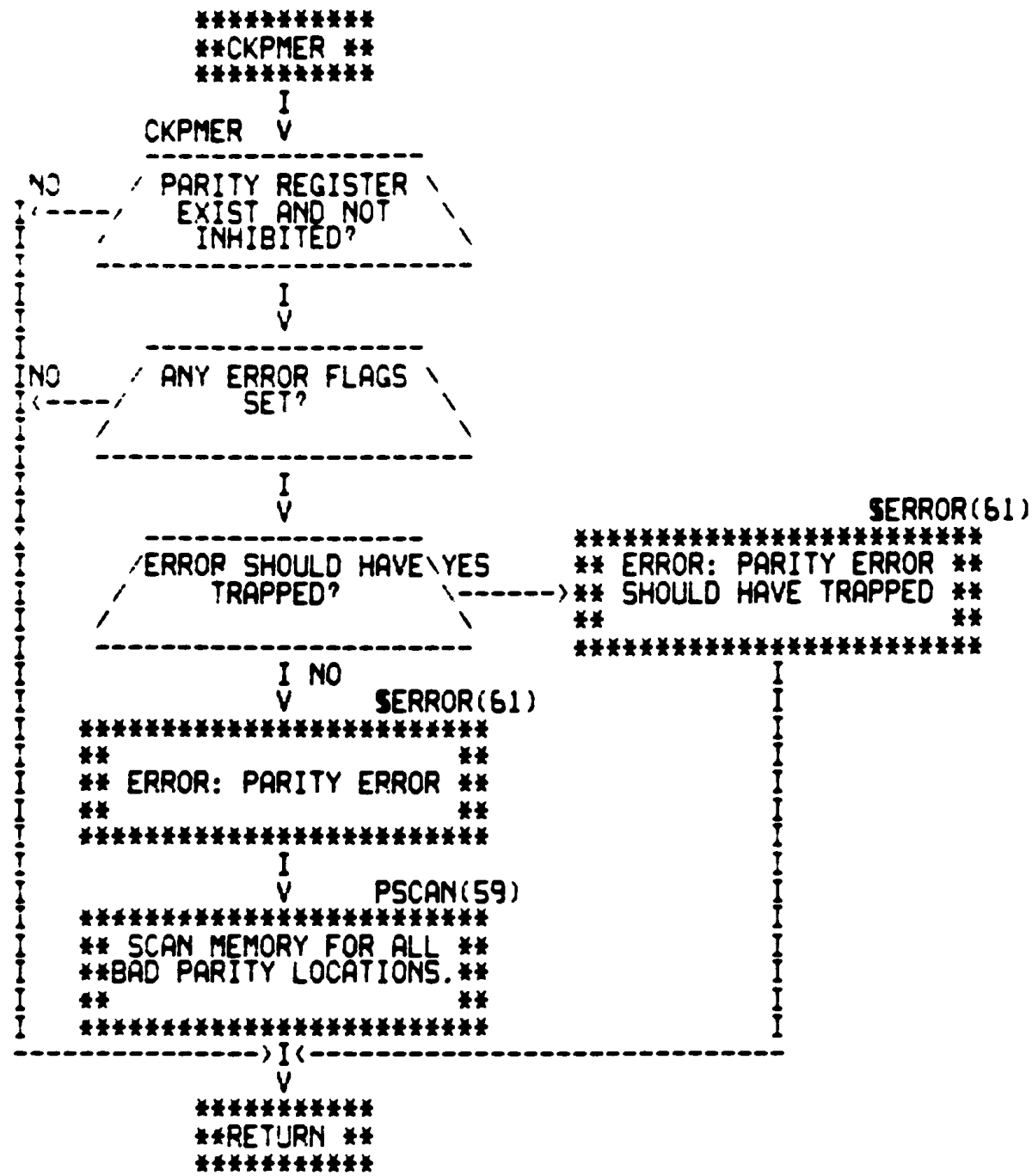
```

*****
**W3X9 **
*****
  I
W3X9 V
*****
*WRITE 256 WORD WITH 4*
* OF A PATTERN THEN 4 *
* OF ANOTHER *
*****
  I
  V
*****
**RETURN **
*****

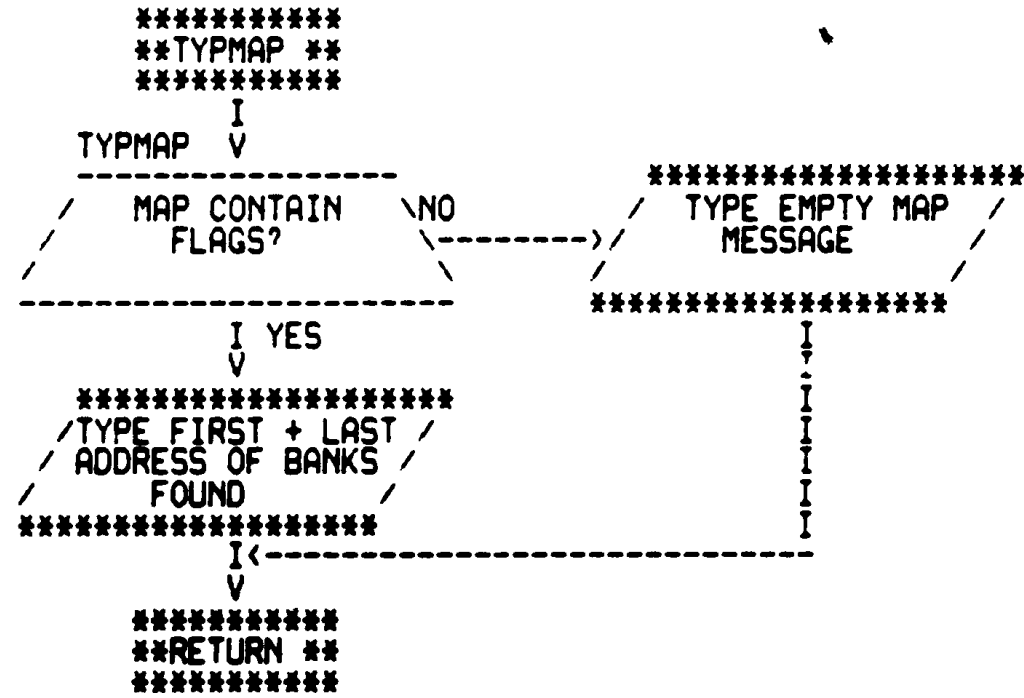
```





```
*****  
**SPRNT **--->I  
*****  
  
*****  
**SPRNTQ **--->I  
*****  
  
*****  
**SPRNT R **--->I  
*****  
  
*****  
**SPRNTD **--->I  
*****  
  
*****  
**SPRNT1 **--->I  
*****  
  
*****  
**SPRNT3 **--->I  
*****  
  
*****  
**SPRNT2 **--->I  
*****  
  
*****  
* ROUTINES TO SET UP *  
* DATA FOR ERROR *  
* TYPEOUTS. *  
*****  
I  
V  
*****  
**RETURN **
```



```

$SCOPE
*****
***** * CONTROLS LOOPING, * *****
**$SCOPE **-->* INTERATIONS, ETC. **-->**RETURN **
***** * BETWEEN SUBTESTS * *****
*****

```

```

$ERROR
*****
***** * COUNTS ERRORS, LOOPS. * *****
**$ERROR **-->* PASS DATA TO $ERRTYP **-->**RETURN **
***** * * *****
*****

```

```

$ERRTYP
*****
***** * TYPEOUT ERROR * *****
**$ERRTYP **-->* MESSAGE, HEADER, AND **-->**RETURN **
***** * DATA * *****
*****

```

```

$RDCHR
*****
***** * INPUTS CHARACTER FROM * *****
**$RDCHR **-->* TTY **-->**RETURN **
***** * * *****
*****

```

```

$ROLIN
*****
***** * INPUTS STRING OF * *****
**$ROLIN **-->* CHARACTERS FROM TTY **-->**RETURN **
***** * * *****
*****

```

```

$RDOCT
*****
***** * CONVERTS ASCII OCTAL * *****
**$RDOCT **-->* NUMBER TO MACHINE **-->**RETURN **
***** * NARY * *****
*****

```

```

$PRINT
*****
***** * RELOCATES MESSAGE * *****
**$PRINT **-->* ADDRESS FOR $TYPE **-->**RETURN **
***** * * *****
*****

```

```

$TYPE
*****
***** * TYPES OUT A MESSAGE * *****
**$TYPE **-->* ON TTY. **-->**RETURN **
***** * * *****
*****

```

```

$STYDPS
*****
***** * * *****
**$STYDPS **-->* TYPE A DECIMAL NUMBER **-->**RETURN **
***** * * *****
*****

```

```

$STYPOC
*****
***** * * *****
**$STYPOC **-->* TYPE AN OCTAL NUMBER **-->**RETURN **
***** * * *****
*****

```

```

$ERRTRP
*****
***** * UNEXPECTED TIMEOUT * *****
**$ERRTRP **-->* TRAP (TO 4) ROUTINE **-->**HALT **
***** * * *****
*****

```

```

$STYPAD
*****
***** * * *****
**$STYPAD **-->* TYPE AN 18-BIT * *****
***** * ADDRESS (OCTAL) **-->**RETURN **
***** * * *****
*****

```

```

*****
* *
* ASCII MESSAGES *
* *
*****

```

```

*****
* ERROR DATA FORMAT *
* TABLE *
* *
*****

```

```

*****
** .END **

```


\$PWRUP	04#			
\$RDCHR	61	61#		
\$RDOCT	15	15	61	61#
\$ROLIN	61	61#		
\$SCOPE	61	61#		
\$TYPAD	10	10	61	61#
\$TYPDS	61	61#		
\$TYPE	61	61#		
\$TYPOC	61	61#		

14	OPERATIONAL SWITCH SETTINGS
29	BASIC DEFINITIONS
141	MEMORY MANAGEMENT DEFINITIONS
190	TRAP CATCHER
199	STARTING ADDRESS(ES)
211	ACT11 HOOKS
293	POWER DOWN AND UP ROUTINES
335	COMMON TAGS
383	APT MAILBOX-ETABLE
449	APT PARAMETER BLOCK
471	APT STATISTICS TABLE
505	MEMORY PARITY PATTERNS TABLE
624	MEMORY PARITY REGISTER ADDRESS TABLE
704	ERROR POINTER TABLE
845	START: SETUP AND MAP MEMORY
856	INITIALIZE THE COMMON TAGS
890	TYPE PROGRAM NAME
1088	MAP PARITY REGISTERS
1122	MAP PARITY MEMORY
1262	TEST PARITY REGISTERS
1349	USER PARAMETER SELECTION SECTION
1475	SECTION 1: MEMORY ADDRESS TESTS
1476	T1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1523	T2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1561	T3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
1602	T4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1638	T5 WRITE 1'S COMPLEMENT OF BANK #.
1675	SECTION 2: WORST CASE NOISE TESTS
1680	T6 WRITE A CONSTANT INTO MEMORY.
1701	T7 READ MEMORY AND COMPARE TO CONSTANT.
1735	T10 WORSE CASE NOISE (PARITY) WORD TESTING
1760	T11 ROTATE A "0" BIT THROUGH A FIELD OF ONES.
1784	T12 ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
1807	T13 1 XOR 8 TEST PATTERN
1955	T14 3 XOR 9 TEST PATTERN.
2058	T15 COMPLEMENT 3 XOR 9 TEST PATTERN
2162	T16 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2348	T17 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2533	T20 8 XOR 13 TEST PATTERN
2637	T21 WORSE CASE NOISE PARITY BYTE TESTING
2794	T22 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
2829	SECTION 3: INSTRUCTION EXECUTION TESTS.
2830	T23 EXECUTE DATI, DATO THRU MEMORY.
2880	T24 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
2930	T25 EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
2981	T26 EXECUTE DATI, DATIP, DATO THRU MEMORY.
3031	T27 EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
3081	T30 EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
3132	T31 "BRANCH GOBBLE" TEST
3232	DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.
3257	END OF PASS ROUTINE
3299	SUBROUTINE AND TRAP ROUTINE SECTION.
3300	MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
3555	SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
3661	RELOCATION SUBROUTINES.
3858	PARITY MEMORY TRAP SERVICE AND SUBROUTINES.

4051	SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
4152	SCOPE HANDLER ROUTINE
4308	ERROR HANDLER ROUTINE
4365	ERROR MESSAGE TIMEOUT ROUTINE
4453	TTY INPUT ROUTINE
4568	READ AN OCTAL NUMBER FROM THE TTY
4641	TYPE ROUTINE
4720	APT COMMUNICATIONS ROUTINE
4815	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4883	BINARY TO OCTAL (ASCII) AND TYPE
4975	PHYSICAL ADDRESS TYPE ROUTINE
5031	STANDARD PROGRAM MESSAGES
5189	ERROR REPORTING MESSAGES AND TABLES.

.TITLE MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
:*COPYRIGHT (C) 1975,1976
:*DIGITAL EQUIPMENT CORP.
:*MAYNARD, MASS. 01754
:*
:*PROGRAM BY BRUCE BURGESS/KEN CHAPMAN
:*
:*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
:*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
:*

13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

.SBTTL OPERATIONAL SWITCH SETTINGS

SWITCH	USE
15	HALT ON ERROR
14	LOOP ON TEST
13	INHIBIT ERROR TYPEOUTS
12	INHIBIT KTI1 (AT START TIME ONLY)
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<4:0>
7	INHIBIT PROGRAM RELOCATION
6	INHIBIT PARITY ERROR DETECTION
5	INHIBIT EXERCISING VECTOR AREA.

.SBTTL BASIC DEFINITIONS

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***

001100

STACK= 1100
.EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

.*MISCELLANEOUS DEFINITIONS

000011	HT= 11	;;CODE FOR HORIZONTAL TAB
000012	LF= 12	;;CODE FOR LINE FEED
000015	CR= 15	;;CODE FOR CARRIAGE RETURN
000200	CRLF= 200	;;CODE FOR CARRIAGE RETURN-LINE FEED
177776	PS= 177776	;;PROCESSOR STATUS WORD
177774	.EQUIV PS,PSW	
177774	STKLMT= 177774	;;STACK LIMIT REGISTER
177772	PIRQ= 177772	;;PROGRAM INTERRUPT REQUEST REGISTER
177570	DSWR= 177570	;;HARDWARE SWITCH REGISTER
177570	DDISP= 177570	;;HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS

000000	R0= %0	;;GENERAL REGISTER
000001	R1= %1	;;GENERAL REGISTER
000002	R2= %2	;;GENERAL REGISTER
000003	R3= %3	;;GENERAL REGISTER
000004	R4= %4	;;GENERAL REGISTER
000005	R5= %5	;;GENERAL REGISTER
000006	R6= %6	;;GENERAL REGISTER
000007	R7= %7	;;GENERAL REGISTER
000006	SP= %6	;;STACK POINTER

```
57          000007          PC=      %7          ;;PROGRAM COUNTER
58
59          ;*PRIORITY LEVEL DEFINITIONS
60          000000          PR0=      0          ;;PRIORITY LEVEL 0
61          000040          PR1=      40         ;;PRIORITY LEVEL 1
62          000100          PR2=     100         ;;PRIORITY LEVEL 2
63          000140          PR3=     140         ;;PRIORITY LEVEL 3
64          000200          PR4=     200         ;;PRIORITY LEVEL 4
65          000240          PR5=     240         ;;PRIORITY LEVEL 5
66          000300          PR6=     300         ;;PRIORITY LEVEL 6
67          000340          PR7=     340         ;;PRIORITY LEVEL 7
68
69          ;*"SWITCH REGISTER" SWITCH DEFINITIONS
70          100000          SW15=    100000
71          040000          SW14=    40000
72          020000          SW13=    20000
73          010000          SW12=    10000
74          004000          SW11=    4000
75          002000          SW10=    2000
76          001000          SW09=    1000
77          000400          SW08=    400
78          000200          SW07=    200
79          000100          SW06=    100
80          000040          SW05=    40
81          000020          SW04=    20
82          000010          SW03=    10
83          000004          SW02=    4
84          000002          SW01=    2
85          000001          SW00=    1
86          .EQUIV SW09,SW9
87          .EQUIV SW08,SW8
88          .EQUIV SW07,SW7
89          .EQUIV SW06,SW6
90          .EQUIV SW05,SW5
91          .EQUIV SW04,SW4
92          .EQUIV SW03,SW3
93          .EQUIV SW02,SW2
94          .EQUIV SW01,SW1
95          .EQUIV SW00,SW0
96
97          ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
98          100000          BIT15=  100000
99          040000          BIT14=  40000
100         020000          BIT13=  20000
101         010000          BIT12=  10000
102         004000          BIT11=  4000
103         002000          BIT10=  2000
104         001000          BIT09=  1000
105         000400          BIT08=  400
106         000200          BIT07=  200
107         000100          BIT06=  100
108         000040          BIT05=  40
109         000020          BIT04=  20
110         000010          BIT03=  10
111         000004          BIT02=  4
112         000002          BIT01=  2
```

```

113          000001          BIT00= 1
114          .EQUIV BIT09,BIT9
115          .EQUIV BIT08,BIT8
116          .EQUIV BIT07,BIT7
117          .EQUIV BIT06,BIT6
118          .EQUIV BIT05,BIT5
119          .EQUIV BIT04,BIT4
120          .EQUIV BIT03,BIT3
121          .EQUIV BIT02,BIT2
122          .EQUIV BIT01,BIT1
123          .EQUIV BIT00,BIT0
124
125          ;*BASIC "CPU" TRAP VECTOR ADDRESSES
126          000004          ERRVEC= 4          ;; TIME OUT AND OTHER ERRORS
127          000010          RESVEC= 10         ;; RESERVED AND ILLEGAL INSTRUCTIONS
128          000014          TBITVEC=14        ;; "T" BIT
129          000014          TRIVEC= 14        ;; TRACE TRAP
130          000014          BPTVEC= 14        ;; BREAKPOINT TRAP (BPT)
131          000020          IOTVEC= 20        ;; INPUT/OUTPUT TRAP (IOT) **SCOPE**
132          000024          PWRVEC= 24        ;; POWER FAIL
133          000030          EMTVEC= 30        ;; EMULATOR TRAP (EMT) **ERROR**
134          000034          TRAPVEC=34        ;; "TRAP" TRAP
135          000060          TKVEC= 60         ;; TTY KEYBOARD VECTOR
136          000064          TPVEC= 64         ;; TTY PRINTER VECTOR
137          000240          PIRQVEC=240       ;; PROGRAM INTERRUPT REQUEST VECTOR
138
139
140          .SBTTL MEMORY MANAGEMENT DEFINITIONS
141
142          ;*KT11 VECTOR ADDRESS
143
144          000250          MMVEC= 250
145
146          ;*KT11 STATUS REGISTER ADDRESSES
147
148          177572          SR0= 177572
149          177574          SR1= 177574
150          177576          SR2= 177576
151          172516          SR3= 172516
152
153          ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
154
155          172300          KIPDR0= 172300
156          172302          KIPDR1= 172302
157          172304          KIPDR2= 172304
158          172306          KIPDR3= 172306
159          172310          KIPDR4= 172310
160          172312          KIPDR5= 172312
161          172314          KIPDR6= 172314
162          172316          KIPDR7= 172316
163
164          ;*KERNEL "I" PAGE ADDRESS REGISTERS
165
166          172340          KIPAR0= 172340
167          172342          KIPAR1= 172342
168          172344          KIPAR2= 172344
  
```


MEMORY MANAGEMENT DEFINITIONS

```

169          172346      KIPAR3= 172346
170          172350      KIPAR4= 172350
171          172352      KIPAR5= 172352
172          172354      KIPAR6= 172354
173          172356      KIPAR7= 172356
174
175          000000      UP = 0          ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
176          000006      RW = 6          ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
177
178          ;* PARITY MEMORY DEFINITIONS.
179          000004      WWP=4          ;WRITE WRONG PARITY
180          000001      AE=1          ;PARITY ACTION ENABLE
181          000114      PARVEC=114      ;PARITY TRAP VECTOR
182
183          ;* MISCELLANEOUS ASSIGNMENTS
184          017777      MASK4K= 17777      ;MASK FOR 4K ADDRESS BANK BOUNDRY.
185
186          ;* CACHE REGISTER DEFINITIONS.
187          177746      IMPCHE= 177746
188
189          .SBTTL TRAP CATCHER
190
191          000000      .=0
192          ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
193          ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
194          ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
195          000174      .=174
196          000174      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
197          000176      000000      SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
198          .SBTTL STARTING ADDRESS(ES)
199          000200      000137      002666      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
200          000204      000167      002464      JMP SELECT      ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
201          ;SELECT VARIOUS PARAMETERS.
202          000210      000167      000064      JMP RESTAR      ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
203          000214      000167      000064      JMP RESTOR      ;RESTORE LOADERS TO END OF MEMORY AND HALT.
204          000220      000167      003406      JMP TIMEOUT      ;TYPE OUT MEMORY MAP, BYTE BY BYTE.
205
206          000004      000004      .=ERRVEC
207          000004      025200      .WORD ERRTRP
208          000006      000000      .WORD 0
209
210          .SBTTL ACT11 HOOKS
211
212          ;*****
213          ;HOOKS FEQUIRED BY ACT11
214          000010      $$VPC=.          ;SAVE PC
215          000046      .=46
216          000046      014760      $ENDAD      ;;1)SET LOC.46 TO ADDRESS OF SENDAD IN .SECP
217          000052      .=52
218          000052      040000      .WORD BIT14      ;;2)SET LOC.52 TO BIT14
219          000010      .=$$VPC          ;; RESTORE PC

```

```

220          000300          .-300
221          :*****
222          :* THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
223          :* THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
224          :* THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
225          :* MANAGEMENT IS DISABLED BY "CONSOLE START".
226          :*****
227 000300 005005 RESTAR: CLR R5 ;CLEAR FLAG TO INDICATE RESTART.
228 000302 000401          BR REST1 ;GO RESTORE PROGRAM BEFORE RESTARTING.
229 000304 010705 RESTOR: MOV PC R5 ;PUT DATA INTO FLAG FOR RESTORE.
230 000306 012706 001100 REST1: MOV #STACK, SP ;SET UP THE STACK POINTER.
231 000312 005767 001206          *ST MEMMAP ;CHECK IF THE MEMORY HAS BEEN MAPPED.
232 000316 001002          BNE REST2 ;BR IF MEMORY MAPPED.
233 000320 000167 002356          JMP STARTA ;GO START
234 000324 005767 000256 REST2: *ST MMAPA ;CHECK IF MEM MGMT AVAILABLE.
235 000330 001470          BEQ 10$ ;BR IF NO MEM MGMT.
236 000332 032737 000001 177572 BIT #BIT0, @#SRO ;CHECK IF MEM MGMT ACTIVE.
237 000340 001034          BNE 2$ ;BR IF MEM MGMT ALREADY SET UP.
238 000342 012700 172300          MOV #KIPDR0, R0 ;POINT TO FIRST MEM MGMT DDATA REG.
239 000346 012701 000010          MOV #8, R1 ;SET UP COUNTER.
240 000352 012720 077406 1$: MOV #077406, (R0)+ ;MAP FIRST 28K 1-FOR-1.
241 000356 005301          DEC R1 ;COUNT REGESTERS.
242 000360 001374          BNE 1$ ;BR IF MORE REG.
243 000362 012700 172340          MOV #KIPAR0, R0 ;POINT TO FIRST MEM MGMT ADDRESS REG.
244 000366 005020          CLR (R0)+ ;PAR0 MAPPED INTO BANK0.
245 000370 012720 000200          MOV #200, (R0)+ ;PAR1 MAPPED INTO BANK1.
246 000374 012720 000400          MOV #400, (R0)+ ;PAR2 MAPPED INTO BANK2.
247 000400 012720 000600          MOV #600, (R0)+ ;PAR3 MAPPED INTO BANK3.
248 000404 012720 001000          MOV #1000, (R0)+ ;PAR4 MAPPED INTO BANK4.
249 000410 012720 001200          MOV #1200, (R0)+ ;PAR5 MAPPED INTO BANK5.
250 000414 012720 001400          MOV #1400, (R0)+ ;PAR6 MAPPED INTO BANK6.
251 000420 012720 007600          MOV #7600, (R0)+ ;PAR7 MAPPED INTO BANK37.
252 000424 012737 000001 177572 MOV #BIT0, @#SRO ;ENABLE MEM MGMT.
253 000432 005000 2$: CLR R0 ;INIT TEMP PAR REG.
254 000434 016701 000142          MOV PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
255 000440 016702 000140          MOV PRGMAP+2, R2 ;...HI 64K.
256 000444 006202 3$: ASR R2 ;SHIFT THE MAP POINTER...HI
257 000446 006001          ROR R1 ;...LO.
258 000450 103404          BCS 4$ ;BR WHEN FIRST BANK FOUND.
259 000452 062700 000200          ADD #200, R0 ;UPDATE TMP PAR TO NEXT BANK.
260 000456 100372          BPL 3$ ;BR IF MORE.
261 000460 000000          HALT ;FATAL ERROR!!! MAP EMPTY?
262 000462 010037 172340 4$: MOV R0, @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
263 000466 000137 000472          JMP @#5$ ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
264 000472 062700 000200 5$: ADD #200, R0 ;KEEP UPDATING TEMP PAR REG.
265 000476 006202          ASR R2 ;SHIFT POINTER...HI
266 000500 006001          ROR R1 ;...LO
267 000502 103373          BCC 5$ ;BR IF TOP BANK NOT YET FOUND.
268 000504 010037 172342          MOV R0, @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
269 000510 000410          BR 20$ ;BR TO RELOCATE SECTION.
270 000512 016700 000062 10$: MOV RELOCF, R0 ;GET RELOCATION FACTOR.
271 000516 062700 001100          ADD #STACK, R0 ;SET UP STACK POINTER.
272 000522 010006          MOV R0, SP ;SET STACK TO RELOCATE PROGRAM.
273 000524 062700 177432          ADD #20$-STACK, R0 ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
274 000530 000110          JMP (R0) ;GO TO "20$" (RELOCATED).
275 000532 022767 000003 000042 20$: CMP #3, PRGMAP ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.

```

K07

276	000540	001402	
277	000542	004767	017030
278	000546	005705	
279	000550	001006	
280	000552	005067	000412
281	000556	105067	000320
282	000562	000167	005220
283	000566	004767	017212
284	000572	000000	
285	000574	000167	002102
286			
287			
288			
289	000600	000000	
290	000602	000000	000000
291	000606	000000	

```

21$: BEQ 21$ ;BR IF IN BANKS 0 AND ..
      JSR PC, RELO ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
      TST R5 ;CHECK RESTART/RESTORE FLAG.
      BNE 22$ ;BR IF RESTORE.
      CLR $TIMES ;CLEAN UP BEFORE STARTING.
      CLRB $STNM
22$: JMP START1 ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
      JSR PC, RESLDR ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
      HALT ;HALT AFTER RESTORING THE LOADERS.
      JMP STARTA ;CONTINUE WILL RESTART THE PROGRAM.

```

;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
;* BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
;* CIRCUMSTANCES.

```

RELOCF: .WORD 0 ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
PRGMAP: .WORD 0,0 ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
MMAVA: .WORD 0 ;MEMORY MANAGEMENT AVAILABLE FLAG.

```

```

292          .SBTTL POWER DOWN AND UP ROUTINES
293
294          ;:*****
295          :POWER DOWN ROUTINE
296 000610 012737 000756 000024 $PWRDN: MOV    $SILLUP,@#PWRVEC ;:SET FOR FAST UP
297 000616 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;:PRIO:7
298 000624 010046      MOV    R0,-(SP) ;:PUSH R0 ON STACK
299 000626 010146      MOV    R1,-(SP) ;:PUSH R1 ON STACK
300 000630 010246      MOV    R2,-(SP) ;:PUSH R2 ON STACK
301 000632 010346      MOV    R3,-(SP) ;:PUSH R3 ON STACK
302 000634 010446      MOV    R4,-(SP) ;:PUSH R4 ON STACK
303 000636 010546      MOV    R5,-(SP) ;:PUSH R5 ON STACK
304 000640 017746 000274      MOV    @SWR,-(SP) ;:PUSH @SWR ON STACK
305 000644 010667 000112      MOV    SP,$SAVR6 ;:SAVE SP
306 000650 012737 000662 000024      MOV    $PWRUP,@#PWRVEC ;:SET UP VECTOR
307 000656 000000      HALT
308 000660 000776      BR      .-2 ;:HANG UP
309
310          ;:*****
311          :POWER UP ROUTINE
312 000662 012737 000756 000024 $PWRUP: MOV    $SILLUP,@#PWRVEC ;:SET FOR FAST DOWN
313 000670 016706 000066      MOV    $SAVR6,SP ;:GET SP
314 000674 005067 000062      CLR    $SAVR6 ;:WAIT LOOP FOR THE TTY
315 000700 005267 000056      1$: INC    $SAVR6 ;:WAIT FOR THE INC
316 000704 001375      BNE    1$ ;:OF WORD
317 000706 012677 000226      MOV    (SP)+,@SWR ;:POP STACK INTO @SWR
318 000712 012605      MOV    (SP)+,R5 ;:POP STACK INTO R5
319 000714 012604      MOV    (SP)+,R4 ;:POP STACK INTO R4
320 000716 012603      MOV    (SP)+,R3 ;:POP STACK INTO R3
321 000720 012602      MOV    (SP)+,R2 ;:POP STACK INTO R2
322 000722 012601      MOV    (SP)+,R1 ;:POP STACK INTO R1
323 000724 012600      MOV    (SP)+,R0 ;:POP STACK INTO R0
324 000726 012737 000610 000024      MOV    $PWRDN,@#PWRVEC ;:SET UP THE POWER DOWN VECTOR
325 000734 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;:PRIO:7
326 000742 004567 022630      JSR    R5,$PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
327 000746 025725      $PWRMG: .WORD PWRMSG ;:POWER FAIL MESSAGE POINTER
328 000750 012716      MOV    (PC)+,(SP) ;:RESTART AT RESTART
329 000752 000300      $PWRAD: .WORD RESTART ;:RESTART ADDRESS
330 000754 000002      RTI
331 000756 000000      $SILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
332 000760 000776      BR      .-2 ;:BEFORE THE POWER DOWN WAS COMPLETE
333 000762 00000C      $SAVR6: 0 ;:PUT THE SP HERE

```

```
334 .SBTTL COMMON TAGS
335
336 ;:*****
337 ;:THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
338 ;:USED IN THE PROGRAM.
339
340 001100 .=1100
341 001100 SCMTAG: ;;START OF COMMON TAGS
342 001100 000000 .WORD 0
343 001102 000 .STSTNM: .BYTE 0 ;;CONTAINS THE TEST NUMBER
344 001103 000 .SERFLG: .BYTE 0 ;;CONTAINS ERROR FLAG
345 001104 000000 .SICNT: .WORD 0 ;;CONTAINS SUBTEST ITERATION COUNT
346 001106 000000 .SLPADR: .WORD 0 ;;CONTAINS SCOPE LOOP ADDRESS
347 001110 000000 .SLPERR: .WORD 0 ;;CONTAINS SCOPE RETURN FOR ERRORS
348 001112 000000 .SERTTL: .WORD 0 ;;CONTAINS TOTAL ERRORS DETECTED
349 001114 000 .SITEMB: .BYTE 0 ;;CONTAINS ITEM CONTROL BYTE
350 001115 001 .SERMAX: .BYTE 1 ;;CONTAINS MAX. ERRORS PER TEST
351 001116 000000 .SERRPC: .WORD 0 ;;CONTAINS PC OF LAST ERROR INSTRUCTION
352 001120 000000 .SGDADR: .WORD 0 ;;CONTAINS ADDRESS OF 'GOOD' DATA
353 001122 000000 .SBDADR: .WORD 0 ;;CONTAINS ADDRESS OF 'BAD' DATA
354 001124 000000 .SGDDAT: .WORD 0 ;;CONTAINS 'GOOD' DATA
355 001126 000000 .SBDDAT: .WORD 0 ;;CONTAINS 'BAD' DATA
356 001130 000000 .WORD 0 ;;RESERVED--NOT TO BE USED
357 001132 000000 .WORD 0
358 001134 000 .SAUTOB: .BYTE 0 ;;AUTOMATIC MODE INDICATOR
359 001135 000 .SINTAG: .BYTE 0 ;;INTERRUPT MODE INDICATOR
360 001136 000000 .WORD 0
361 001140 177570 .SWR: .WORD DSWR ;;ADDRESS OF SWITCH REGISTER
362 001142 177570 .DISPLAY: .WORD DDISP ;;ADDRESS OF DISPLAY REGISTER
363 001144 177560 .STKS: 177560 ;;TTY KBD STATUS
364 001146 177562 .STKB: 177562 ;;TTY KBD BUFFER
365 001150 177564 .STPS: 177564 ;;TTY PRINTER STATUS REG. ADDRESS
366 001152 177566 .STPB: 177566 ;;TTY PRINTER BUFFER REG. ADDRESS
367 001154 000 .SNULL: .BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
368 001155 002 .SFILLS: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
369 001156 012 .SFILLC: .BYTE 12 ;;INSERT FILL CHARS. AFTER A "LINE FEED"
370 001157 000 .STPFLG: .BYTE 0 ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
371 001160 000000 .STMP0: .WORD 0 ;;USER DEFINED
372 001162 000000 .STMP1: .WORD 0 ;;USER DEFINED
373 001164 000000 .STMP2: .WORD 0 ;;USER DEFINED
374 001166 000000 .STMP3: .WORD 0 ;;USER DEFINED
375 001170 000000 .STIMES: 0 ;;MAX. NUMBER OF ITERATIONS
376 001172 000000 .SESCAPE: 0 ;;ESCAPE ON ERROR ADDRESS
377 001174 177607 000377 .SBELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
378 001200 077 .SQUES: .ASCII /?/ ;;QUESTION MARK
379 001201 015 .SCRLF: .ASCII <15> ;;CARRIAGE RETURN
380 001202 000012 .SLF: .ASCIZ <12> ;;LINE FEED
381 ;:*****
382 .SBTTL APT MAILBOX-ETABLE
383
384 ;:*****
385 .EVEN
386 001204 .SMAIL: ;;APT MAILBOX
387 001204 000000 .SMSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
388 001206 000000 .SFATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
389 001210 000000 .STESTN: .WORD ATESTN ;;TEST NUMBER
```

390	001212	000000	\$PASS:	.WORD	APASS	:: PASS COUNT
391	001214	000000	\$DEVCT:	.WORD	ADEVCT	:: DEVICE COUNT
392	001216	000000	\$UNIT:	.WORD	AUNIT	:: I/O UNIT NUMBER
393	001220	000000	\$MSGAD:	.WORD	AMSGAD	:: MESSAGE ADDRESS
394	001222	000000	\$MSGLG:	.WORD	AMSLG	:: MESSAGE LENGTH
395	001224		\$ETABLE:			:: APT ENVIRONMENT TABLE
396	001224	000	\$ENV:	.BYTE	AENV	:: ENVIRONMENT BYTE
397	001225	000	\$ENVM:	.BYTE	AENVM	:: ENVIRONMENT MODE BITS
398	001226	000000	\$SWREG:	.WORD	ASWREG	:: APT SWITCH REGISTER
399	001230	000000	\$USWR:	.WORD	AUSWR	:: USER SWITCHES
400	001232	000000	\$CPUOP:	.WORD	ACPUOP	:: CPU TYPE, OPTIONS
401			*			BITS 15-11=CPU TYPE
402			*			11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
403			*			11/70=06, PDQ=07, Q=10
404			*			BIT 10=REAL TIME CLOCK
405			*			BIT 9=FLOATING POINT PROCESSOR
406			*			BIT 8=MEMORY MANAGEMENT
407	001234	000	\$MAMS1:	.BYTE	AMAMS1	:: HIGH ADDRESS, M.S. BYTE
408	001235	000	\$MTYP1:	.BYTE	AMTYP1	:: MEM. TYPE, BLK#1
409			*			MEM. TYPE BYTE -- (HIGH BYTE)
410			*			900 NSEC CORE=001
411			*			300 NSEC BIPOLAR=002
412			*			500 NSEC MOS=003
413	001236	000000	\$MADR1:	.WORD	AMADR1	:: HIGH ADDRESS, BLK#1
414			*			MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
415	001240	000	\$MAMS2:	.BYTE	AMAMS2	:: HIGH ADDRESS, M.S. BYTE
416	001241	000	\$MTYP2:	.BYTE	AMTYP2	:: MEM. TYPE, BLK#2
417	001242	000000	\$MADR2:	.WORD	AMADR2	:: MEM. LAST ADDRESS, BLK#2
418	001244	000	\$MAMS3:	.BYTE	AMAMS3	:: HIGH ADDRESS, M.S. BYTE
419	001245	000	\$MTYP3:	.BYTE	AMTYP3	:: MEM. TYPE, BLK#3
420	001246	000000	\$MADR3:	.WORD	AMADR3	:: MEM. LAST ADDRESS, BLK#3
421	001250	000	\$MAMS4:	.BYTE	AMAMS4	:: HIGH ADDRESS, M.S. BYTE
422	001251	000	\$MTYP4:	.BYTE	AMTYP4	:: MEM. TYPE, BLK#4
423	001252	000000	\$MADR4:	.WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
424	001254	000000	\$VECT1:	.WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
425	001256	000000	\$VECT2:	.WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
426	001260	000000	\$BASE:	.WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
427	001262	000000	\$DEVN:	.WORD	ADEVN	:: DEVICE MAP
428	001264	000000	\$CDW1:	.WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
429	001266	000000	\$CDW2:	.WORD	ACDW2	:: CONTROLLER DESCRIPTION WORD#2
430	001270	000000	\$DDW0:	.WORD	ADDW0	:: DEVICE DESCRIPTOR WORD#0
431	001272	000000	\$DDW1:	.WORD	ADDW1	:: DEVICE DESCRIPTOR WORD#1
432	001274	000000	\$DDW2:	.WORD	ADDW2	:: DEVICE DESCRIPTOR WORD#2
433	001276	000000	\$DDW3:	.WORD	ADDW3	:: DEVICE DESCRIPTOR WORD#3
434	001300	000000	\$DDW4:	.WORD	ADDW4	:: DEVICE DESCRIPTOR WORD#4
435	001302	000000	\$DDW5:	.WORD	ADDW5	:: DEVICE DESCRIPTOR WORD#5
436	001304	000000	\$DDW6:	.WORD	ADDW6	:: DEVICE DESCRIPTOR WORD#6
437	001306	000000	\$DDW7:	.WORD	ADDW7	:: DEVICE DESCRIPTOR WORD#7
438	001310	000000	\$DDW8:	.WORD	ADDW8	:: DEVICE DESCRIPTOR WORD#8
439	001312	000000	\$DDW9:	.WORD	ADDW9	:: DEVICE DESCRIPTOR WORD#9
440	001314	000000	\$DDW10:	.WORD	ADDW10	:: DEVICE DESCRIPTOR WORD#10
441	001316	000000	\$DDW11:	.WORD	ADDW11	:: DEVICE DESCRIPTOR WORD#11
442	001320	000000	\$DDW12:	.WORD	ADDW12	:: DEVICE DESCRIPTOR WORD#12
443	001322	000000	\$DDW13:	.WORD	ADDW13	:: DEVICE DESCRIPTOR WORD#13
444	001324	000000	\$DDW14:	.WORD	ADDW14	:: DEVICE DESCRIPTOR WORD#14
445	001326	000000	\$DDW15:	.WORD	ADDW15	:: DEVICE DESCRIPTOR WORD#15


```

502 ;*****
503 ;*THE FOLLOWING TAGS ARE USER DEFINED
504 ;*****
505 001514 000000 $VERPC: .WORD 0 ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
506 001516 070032 RESRVD: .WORD 070032 ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
507 ;NOTE: FOR MSI' MEMORY WITH PARITY, CHANGE TO 077772.
508 001520 000000 LMAP: .WORD 0 ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
509 001522 000000 LDDISP: .WORD 0 ;CONTAINS DISPLAY REGISTER IMAGE
510 001524 000000 MEMMAP: .WORD 0 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
511 001524 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
512 001526 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
513 001530 000000 TSTMAP: .WORD 0 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
514 001530 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
515 001532 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
516 001534 000000 SAVTST: .WORD 0 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
517 ;TEST EACH BANK ONCE.
518 001534 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
519 001536 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
520 001540 000000 PMEMAP: .WORD 0 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
521 001540 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
522 001542 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
523 001544 000000 BITPT: .WORD 0 ;POINTER TO CURRENT 4K BANK OF MEMORY
524 001544 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
525 001546 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
526 001550 000000 TMPPT: .WORD 0 ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
527 001550 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
528 001552 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
529 001554 000000 MMORE: .WORD 0 ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
530 ;SET UP BY "INITMM" AND "INITDN" ROUTINES.
531 ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
532 001556 000 SELFGL: .BYTE 0 ;OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
533 001557 000 FLAGBK: .BYTE 0 ;BK BLOCK INDICATOR. USED IN "INITMM" AND "MMUP".
534 001560 000 OEFLG: .BYTE 0 ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
535 001562 001562 .EVEN
536 001562 000000 FSTADR: .WORD 0 ;FIRST VIRTUAL ADDRESS TO BE TESTED.
537 ;FIRST ADDRESS IS USER SELECTABLE.
538 001564 000000 TMPFAD: .WORD 0 ;ADJUSTED FIRST ADDRESS.
539 001566 000000 FADMSK: .WORD 0 ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
540 ;TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
541 001570 000000 000000 FADMAP: .WORD 0,0 ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
542 001574 000000 LSTADR: .WORD 0 ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
543 ;LAST ADDRESS IS USER SELECTABLE.
544 001576 000000 TMLPAD: .WORD 0 ;ADJUSTED LAST ADDRESS.
545 001600 000000 LADMSK: .WORD 0 ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
546 ;TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
547 001602 000000 000000 LADMAP: .WORD 0,0 ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
548 001606 000000 BLKMSK: .WORD 0 ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
549 001610 000000 .CONST: .WORD 0 ;USER SELECTABLE CONSTANT DATA.
550
551 ;*****
552 ;* RELATIVE ADDRESSING TABLE.
553 ;* THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
554 ;* RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMENT TAGS.
555 ;*****
556 001612 001100 RADTAB:
557 ;STACK: STACK ;STACK POINTER INITIAL ADDRESS.

```


APT STATISTICS TABLE

558	001614	001516	.RESRV: RESRVD	; PARITY REGISTER RESERVED BIT MASK ADDRESS.
559	001616	002114	.MPRO: MPRO	; MEMORY PARITY REGISTER TABLE ADDRESS.
560	001620	002314	.MPRX: MPRX	; MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
561	001622	014326	.BRGOB: BRGOB	; "BRANCH GOBBLE" ROUTINE ADDRESS.
562	001624	014400	.BGERR: BGERR	; "BRANCH GOBBLE" ERROR ROUTINE ADDRESS.
563	001626	014512	.BGEXI: BGEXIT	; "BRANCH GOBBLE" EXIT ROUTINE ADDRESS.
564	001630	012732	.PBTRP: PBTRP	; PARITY BYTE TEST TRAP ROUTINE ADDRESS.
565	001632	002066	.MPPAT: MPPATS	; MEMORY PARITY PATTERN TABLE ADDRESS.
566	001634	020060	.PESRV: PESRV	; MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
567	001636	002356	.ERRTB: \$ERRTB	; ERROR TYPEOUT TABLE PONTER.
568	001640	000010	.EIGHT: 8.	; DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
569	001642	014560	.TST32: TST32	; SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.

 * DATA CONTAINERS FOR ERROR PRINTOUT.

573	001644	001116	001120	001124	DT1: \$ERRPC,\$GDADR,\$GDDAT,\$BDDAT,0
574	001652	001126	000000		
575	001656	001514	001116	001120	DT2: \$VERPC,\$ERRPC,\$GDADR,\$GDDAT,\$BDDAT,0
576	001664	001124	001126	000000	
577	001672	001514	001116	001120	DT12: \$VERPC,\$ERRPC,\$GDADR,\$GDDAT,0
578	001700	001124	000000		
579	001704	001514	001116	001160	DT14: \$VERPC,\$ERRPC,\$TMPO,\$GDADR,0
580	001712	001120	000000		
581	001716	001514	001116	001120	DT15: \$VERPC,\$ERRPC,\$GDADR,\$TMPO,\$GDDAT,\$BDDAT,0
582	001724	001160	001124	001126	
583	001732	000000			
584	001734	001514	001116	001160	DT21: \$VERPC,\$ERRPC,\$TMPO,\$GDADR,\$GDDAT,\$BDDAT,0
585	001742	001120	001124	001126	
586	001750	000000			
587	001752	001160	001162	001164	DT22: \$TMPO,\$TMP1,\$TMP2,\$TMP3,\$GDADR,\$GDDAT,\$BDDAT,0
588	001760	001166	001120	001124	
589	001766	001126	000000		
590	001772	001514	001116	001120	DT23: \$VERPC,\$ERRPC,\$GDADR,\$BDADR,\$GDDAT,\$BDDAT,0
591	002000	001122	001124	001126	
592	002006	000000			
593	002010	001514	001116	001122	DT24: \$VERPC,\$ERRPC,\$BDADR,0
594	002016	000000			
595	002020	001514	001116	001122	DT25: \$VERPC,\$ERRPC,\$BDADR,\$TMPO,\$TMP1,0
596	002026	001160	001162	000000	
597	002034	001514	001116	001160	DT26: \$VERPC,\$ERRPC,\$TMPO,\$TMP1,0
598	002042	001162	000000		
599	002046	001160	001162	001120	DT30: \$TMPC,\$TMP1,\$GDADR,\$BDDAT,0
600	002054	001126	000000		
601	002060	001166	000000		
602	002064	177777			DT31: \$TMP3,0 .WORD -1 ;TABLE TERMINATOR.

603
 604 .SBTTL MEMORY PARITY PATTERNS TABLE
 605 ;*****
 606 ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
 607 ;*****

609	002066	125325	MPPATS: 125325	; EVEN, ODD
610	002070	152652	152652	; ODD, EVEN
611	002072	052452	052452	; EVEN, ODD
612	002074	025125	025125	; ODD, EVEN
613	002076	102070	102070	; EVEN, EVEN

614 002100 072527
 615 002102 177777
 616 002104 107030
 617 002106 152525
 619 002110 000000
 620 002112 000000
 621
 622
 623
 624
 625
 626
 627
 628
 629
 630
 631
 632
 633
 634
 635 002114 172101
 636 002116 000000
 637 002120 000000
 638 002122 000000
 639 002124 172103
 640 002126 000000
 641 002130 000000
 642 002132 000000
 643 002134 172105
 644 002136 000000
 645 002140 000000
 646 002142 000000
 647 002144 172107
 648 002146 000000
 649 002150 000000
 650 002152 000000
 651 002154 172111
 652 002156 000000
 653 002160 000000
 654 002162 000000
 655 002164 172113
 656 002166 000000
 657 002170 000000
 658 002172 000000
 659 002174 172115
 660 002176 000000
 661 002200 000000
 662 002202 000000
 663 002204 172117
 664 002206 000000
 665 002210 000000
 666 002212 000000
 667 002214 172121
 668 002216 000000
 669 002220 000000

072527
 177777
 107030
 152525
 0
 MPEND: 0

; ODD, ODD
 ; EVEN, EVEN
 ; ODD, ODD
 ; ODD, EVEN
 ; EXTRA PATTERN HOLDER FOR
 ; FUTURE USE
 ; TABLE TERMINATOR

.SBTTL MEMORY PARITY REGISTER ADDRESS TABLE

////////////////////////////////////
 * THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
 * FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
 *
 * THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
 * IF THE CONTROL IS FOUND NOT TO BE PRESENT. THE MEMORY PRESENT UNDER
 * THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
 * THE DEVICE ADDRESS, EACH BIT REPRESENTING A 4K BLOCK. I.E.
 * FIRST WORD BIT0 = 0 - 4K, BIT1 = 4 - 8K ... BIT15 = 60 - 64K
 * SECOND WORD BIT0 = 64 - 68K, ... BIT14 = 120 - 124K.
 //////////////////////////////////////

MPRO: 172100 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR1: 172102 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR2: 172104 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR3: 172106 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR4: 172110 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR5: 172112 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR6: 172114 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR7: 172116 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)
 0 ; MASK FOR MOS, CORE, MS11-K
 MPR8: 172120 +1 ; PARITY STATUS REGISTER
 0 ; CONTROL MAP (LOW 64K)
 0 ; CONTROL MAP (HIGH 64K)


```

703 .SBTTL ERROR POINTER TABLE
704
705 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
706 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
707 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
708 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
709 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
710
711 ;* EM ::POINTS TO THE ERROR MESSAGE
712 ;* DH ;;POINTS TO THE DATA HEADER
713 ;* DT ;;POINTS TO THE DATA
714 ;* DF ;;POINTS TO THE DATA FORMAT
715
716
717 002356 $ERRTB:
718 ;* ITEM 1
719 002356 027071 DM1 ;PARITY REGISTER DATA ERROR.
720 002360 030473 DH1 ;PC,REG,S/B,WAS
721 002362 001644 DT1 ;$ERRPC,$GDADR,$GDDAT,$BDDAT
722 002364 031103 DF1 ;16,18,16,16
723 ;* ITEM 2
724 002366 027125 DM2 ;ADDRESS TEST ERROR(TST1-5).
725 002370 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
726 002372 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
727 002374 031107 DF2 ;16,18,18,16,16
728 ;* ITEM 3
729 002376 027125 DM2 ;ADDRESS TEST ERROR(TST1-5).
730 002400 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
731 002402 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
732 002404 031114 DF3 ;16,18,18,8,8
733 ;* ITEM 4
734 002406 027161 DM4 ;CONSTANT DATA ERROR(TST6-10).
735 002410 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
736 002412 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
737 002414 031107 DF2 ;16,18,18,16,16
738 ;* ITEM 5
739 002416 027217 DM5 ;ROTATING BIT ERROR(TST11-12).
740 002420 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
741 002422 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
742 002424 031107 DF2 ;16,18,18,16,16
743 ;* ITEM 6
744 002426 027255 DM6 ;1 XOR 8 PATTERN ERROR (TST13).
745 002430 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
746 002432 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
747 002434 031107 DF2 ;16,18,18,16,16
748 ;* ITEM 7
749 002436 027313 DM7 ;3 XOR 9 PATTERN ERROR(TST14-17).
750 002440 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
751 002442 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
752 002444 031107 DF2 ;16,18,18,16,16
753 ;* ITEM 10
754 002446 027354 DM10 ;8 XOR 13 PATTERN ERROR(TST20).
755 002450 030512 DH2 ;V/PC,P/PC,MA,S/B,WAS
756 002452 001656 DT2 ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
757 002454 031107 DF2 ;16,18,18,16,16
758 ;* ITEM 11

```

759	002456	027413	DM11	; PARITY MEMORY ADDRESS ERROR(TST21).
760	002460	030512	DH2	; V/PC, P/PC, MA, S/B, WAS
761	002462	001656	DT2	; \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
762	002464	031114	DF3	; 16, 18, 18, 8, 8
763			* ITEM 12	
764	002466	027457	DM12	; DATIP WITH WRONG PARITY DIDN'T TRAP(TST21).
765	002470	030537	DH12	; V/PC, P/PC, MA, S/B
766	002472	001672	DT12	; \$VERPC, \$ERRPC, \$GDADR, \$GDDAT
767	002474	031114	DF3	; 16, 18, 18, 8
768			* ITEM 13	
769	002476	027533	DM13	; WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
770	002500	030537	DH12	; V/PC, P/PC, MA, S/B
771	002502	001672	DT12	; \$VERPC, \$ERRPC, \$GDADR, \$GDDAT
772	002504	031114	DF3	; 16, 18, 18, 8
773			* ITEM 14	
774	002506	027623	DM14	; PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).
775	002510	030550	DH14	; V/PC, P/PC, REG, MA
776	002512	001704	DT14	; \$VERPC, \$ERRPC, \$TMP0, \$GDADR
777	002514	031121	DF14	; 16, 18, 18, 18
778			* ITEM 15	
779	002516	027071	DM1	; PARITY REGISTER DATA ERROR.
780	002520	030601	DH15	; V/PC, P/PC, MAUT, REG, S/B, WAS
781	002522	001716	DT15	; \$VERPC, \$ERRPC, \$GDADR, \$TMP0, \$GDDAT, \$BDDAT
782	002524	031121	DF14	; 16, 18, 18, 18, 16, 16
783			* ITEM 16	
784	002526	027722	DM16	; MORE THAN ONE REGISTER INDICATED PARITY ERROR.
785	002530	030560	DH14	; V/PC, P/PC, REG, MA
786	002532	001704	DT14	; \$VERPC, \$ERRPC, \$TMP0, \$GDADR
787	002534	031121	DF14	; 16, 18, 18, 18
788			* ITEM 17	
789	002536	030001	DM17	; DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
790				; TRAPPED(TST21).
791	002540	030512	DH2	; V/PC, P/PC, MA, S/B, WAS
792	002542	001656	DT2	; \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
793	002544	031114	DF3	; 16, 18, 18, 8, 8
794			* ITEM 20	
795	002546	030077	DM20	; RANDOM DATA ERROR(TST22).
796	002550	030512	DH2	; V/PC, P/PC, MA, S/B, WAS
797	002552	001656	DT2	; \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
798	002554	031107	DF2	; 16, 18, 18, 16, 16
799			* ITEM 21	
800	002556	030131	DM21	; INSTRUCTION EXECUTION ERROR(TST23-30).
801	002560	030634	DH21	; V/PC, P/PC, IUT, MA, S/B, WAS
802	002562	001734	DT21	; \$VERPC, \$ERRPC, \$TMP0, \$GDADR, \$GDDAT, \$BDDAT
803	002564	031127	DF21	; 16, 18, 16, 18, 16, 16
804			* ITEM 22	
805	002566	030200	DM22	; "BRANCH GOBBLE" ERROR(TST31).
806	002570	030665	DH22	; V/PC, P/PC, PS, S/B, PS, WAS, MA, S/B, WAS
807	002572	001752	DT22	; \$TMP0, \$TMP1, \$TMP2, \$TMP3, \$GDADR, \$GDDAT, \$BDDAT
808	002574	031135	DF22	; 16, 18, 16, 16, 18, 16, 16
809			* ITEM 23	
810	002576	030236	DM23	; PROGRAM CODE CHANGED WHEN RELOCATED.
811	002600	030730	DH23	; V/PC, P/PC, SRC, MA, DST, MA, S/B, WAS
812	002602	001772	DT23	; \$VERPC, \$ERRPC, \$GDADR, \$BDADR, \$GDDAT, \$BDDAT
813	002604	031121	DF14	; 16, 18, 18, 18, 16, 16
814			* ITEM 24	

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMC.P11 02-DEC-76 08:47 ERROR POINTER TABLE

815	002606	030303	DM24	; TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
816	002610	030770	DH24	; V/PC, P/PC, TRP/PC
817	002612	002010	DT24	; \$VERPC, \$ERRPC, \$BDADR
918	002614	031121	DF14	; 16, 18, 18
919			* ITEM 25	
820	002616	030357	DM25	; TRAPPED TO 114.
821	002620	031011	DH25	; V/PC, P/PC, TRP/PC, REG, WAS
822	002622	002020	DT25	; \$VERPC, \$ERRPC, \$BDADR, \$TMPO, \$TMP1
823	002624	031121	DF14	; 16, 18, 18, 16
824			* ITEM 26	
825	002626	030377	DM26	; FAILED TO TRAP.
826	002630	031042	DH26	; V/PC, P/PC, REG, WAS
827	002632	002034	DT26	; \$VERPC, \$ERRPC, \$TMPO, \$TMP1
828	002634	031107	DF2	; 16, 18, 18, 16
829			* ITEM 27	
930	002636	030417	DM27	; (ACTION ENABLE WASN'T SET).
831	002640	031042	DH26	; V/PC, P/PC, REG, WAS
832	002642	002034	DT26	; \$VERPC, \$ERRPC, \$TMPO, \$BDADR
833	002644	031107	DF2	; 16, 18, 18, 16
834			* ITEM 30	
835	002646	000000	0	; NO MESSAGE.
836	002650	031064	DH30	; REG, WAS, MA, WAS
837	002652	002046	DT30	; \$TMPO, \$TMP1, \$GDADR, \$BDADR
938	002654	031144	DF30	; 18, 16, 18, 8
839			* ITEM 31	
840	002656	030453	DM31	; TRAPPED TO 4
841	002660	000000	0	; NO HEADER
842	002662	002060	DT31	; \$TMP3
943	002664	031144	DF30	; 18


```

900 003140 010700          MOV    PC,      RO      ;GET CURRENT PROGRAM COUNTER.
901 003142 022700 003142  CMP    #,      RO      ;CHECK IF THE PROGRAM IS RELOCATED.
902 003146 001402          BEQ    10$,     ;BR IF PROGRAM NOT RELOCATED.
903 003150 000167 175124  JMP    RESTAR    ;GO TRY TO RELOCTED BEFORE CONTINUING.
904 003154 012767 000003 175420 10$:  MOV    #3,      PRGMAP  ;INITIALIZE PROGRAM MAP....LO 64K.
905 003162 005067 175416  CLR    PRGMAP+2  ;...HI 64K.
906 003166 005067 175406  CLR    RELOCF    ;INIT THE RELOCATION FACTOR.
907
908
909 003172 005767 176322 ;* ROUTINE TO SAVE THE LOADERS AT THE END OF BK.
910 003176 001024          TST    LMAP      ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
911 003200 012700 040000  BNE    14$,     ;BRANCH IF ALREADY SAVED
912 003204 010001          MOV    #40000, RO    ;GET END OF BK
913 003206 012737 003220 000004  MOV    RO,      R1    ;GET END OF BK
914 003214 011020          MOV    #12$, 2*ERRVEC ;SET UP TIMEOUT VECTOR
915 003216 000776          MOV    (RO),   (RO)+ ;SEARCH FOR END OF MEMORY
916 003220 022626          BR    11$,     ;KEEP SEARCHING
917 003222 012737 025200 000004 11$:  CMP    (SP)+,  (SP)+  ;RESTORE STACK POINTER
918 003230 010046          MOV    #ERRTRAP,2*ERRVEC ;RESET TIMEOUT VECTOR.
919 003232 012702 002734  MOV    RO,      -(SP) ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
920 003236 014041          MOV    #1500., R2    ;SET UP WORD COUNTER
921 003240 005302          MOV    -(RO),  -(R1) ;SAVE THE LOADERS
922 003242 001375          DEC    R2        ;COUNT THE WORDS
923 003244 012667 176250  BNE    13$,     ;BRANCH IF MORE WORDS
924          MOV    (SP)+, LMAP ;SAVE LAST MEMORY ADDRESS
925
926 003250 005067 175332 14$:  ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
927 003254 032777 010000 175656 CLR    MMAVA     ;CLEAR MEM MGMT AVAILABLE FLAG
928 003262 001014          BIT    #SW12, 2SWR   ;CHECK FOR INHIBIT KT11 SWITCH
929 003264 012737 003314 000004  BNE    IMPCK     ;BRANCH IF SET
930 003272 005037 177572          MOV    #IMPCK,2*ERRVEC ;SET UP TIMEOUT TRAP VECTOR
931 003276 004767 011512          CLR    2*SRO     ;CLEAR MEM MGMT STATUS REG
932 003302 005267 175300          JSR    PC,      MMINIT ;MEM MGMT INITIALIZATION ROUTINE.
933 003306 004567 020264          INC    MMAVA     ;SET MEM MGMT AVAILABLE FLAG
934 003312 025440          JSR    R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
935          .WORD    MMAMES ;ADDRESS OF MESSAGE TO BE TYPED
936          ;"KT11 AVAILABLE"
937
938 003314 012706 001100 14$:  ;* CHECK IF 11/60 CACHE PRESENT, IF SO TURN IT OFF!!!
939 003320 012737 003334 000004 IMPCK: MOV    #STACK, SP
940 003326 052767 000014 174412 MOV    #MAPMEM,2*ERRVEC
941          BIS    #14,  IMPCHE
942
943 ;*****
944 ;* ROUTINE TO MAP ALL OF MEMORY.
945 ;* ONLY FULL 4K BANKS W.L. BE RECOGNIZED.
946 ;* R0 = MEMMAP POINTER...LO 64K.
947 ;* R1 = MEMMAP POINTER...HI 64K.
948 ;* R2 = ADDRESS POINTER
949 ;* R3 = BANK POINTER...LO 64K.
950 ;* R4 = BANK POINTER...HI 64K.
951 ;* R5 = SCRATCH REGISTER.
952 ;*****
953 MAPMEM: MOV    #STACK, SP ;RESET THE STACK
954          MOV    #MEMMAP, RO ;SET UP MEMORY MAP POINTER...LO 64K.
955          MOV    #MEMMAP+2, R1 ;...HI 64K.
956          CLR    (RO) ;CLR MEMORY MAP...LO 64K.

```


956	003352	005011			CLR	(R1)		; HI 64K.
957	003354	005002			CLR	R2		; SET ADDRESS POINTER TO 0
958	003356	012703	000001		MOV	#1, R3		; SETUP 4K BANK POINTER...LO 64K.
959	003362	005004			CLR	R4		; HI 64K.
960	003364	005067	175576		CLR	\$TMP3		; INIT TEMPORARY HIGH ADDRESS BITS.
961	003370	004567	020202		JSR	R5, \$PRINT		; GO PRINT OUT THE FOLLOWING MESSAGE.
962	003374	025505			.WORD	MEMMES		; ADDRESS OF MESSAGE TO BE TYPED
963								; "MEMORY MAP:"
964	003376	012737	003512	000004	MOV	#25, @#ERRVEC		; SET UP TIMEOUT VECTOR
965	003404	011222			MOV	(R2), (R2)+		; READ+WRITE ALL MEMORY
966	003406	032702	017777		BIT	#MASK4K, R2		; CHECK FOR 4K BOUNDARY
967	003412	001374			BNE	1\$; BRANCH IF MORE IN BANK
968	003414	050310			BIS	R3, (R0)		; SET FLAG FOR BANK...LO 64K.
969	003416	050411			BIS	R4, (R1)		; HI 64K.
970	003420	010267	175540		MOV	R2, \$TMP2		; SAVE ADDRESS POINTER.
971	003424	005367	175534		DEC	\$TMP2		; ADJUST TO LAST ADR, LAST BANK.
972	003430	005767	175152		TST	MMAVA		; CHECK FOR MEM MGMT.
973	003434	001432			BEQ	3\$; BR IF NO MEM MGMT.
974	003436	042767	160000	175520	BIC	#160000, \$TMP2		; CLEAR BANK BITS ON RELATIVE ADDRESS.
975	003444	013705	172344		MOV	@#KIPAR2, R5		; SAVE KIPAR2.
976	003450	005067	175512		CLR	\$TMP3		; MAKE SURE HI BITS ARE INIT.
977	003454	006305			ASL	R5		; SHIFT IT 6 PLACES.
978	003456	006305			ASL	R5		
979	003460	006305			ASL	R5		
980	003462	006305			ASL	R5		
981	003464	006305			ASL	R5		
982	003466	006167	175474		ROL	\$TMP3		
983	003472	006305			ASL	R5		
984	003474	006167	175466		ROL	\$TMP3		
985	003500	060567	175460		ADD	R5, \$TMP2		; MAKE LAST ADR PHYSICAL.
986	003504	005567	175456		ADC	\$TMP3		
987	003510	000404			BR	3\$; GO TO UPDATE POINTERS.
988								
989								
990	003512	022626			2\$:	* TIMEOUT TRAPS TO HERE		
991	003514	052702	017777		CMP	(SP)+, (SP)+		; RESTORE THE STACK POINTER
992	003520	005202			BIS	#MASK4K, R2		; LAST ADDRESS OF 4K BANK
993	003522	005767	175060		INC	R2		; FIRST ADDRESS OF NEXT BANK.
994	003526	001411			3\$:	TST	MMAVA	; CHECK FOR MEM MGMT
995	003530	062737	000200	172344	BEQ	4\$; BRANCH IF NO MEM MGMT
996	003536	012702	040000		ADD	#200, @#KIPAR2		; UPDATE THIRD PAR
997	003542	006303			MOV	#40000, R2		; POINT TO START OF THIRD PAR
998	003544	006104			ASL	R3		; UPDATE LO BANK POINTER.
999	003546	100316			ROL	R4		; UPDATE HI BANK POINTER
1000	003550	000402			BPL	1\$; BRANCH IF MORE MEMORY TO MAP.
1001					BR	5\$; EXIT WHEN DONE.
1002	003552	106303			4\$:	ASLB	R3	; UPDATE MAP POINTER
1003	003554	100313			BPL	1\$; BRANCH IF NOT YET DONE
1004	003556	012737	025200	000004	5\$:	MOV	#ERRTRP, @#ERRVEC	; RESET TIMEOUT VECTOR
1005	003564	004767	015252		JSR	PC, TYPMAP		; GO TYPE THE MAP.
1006	003570	004567	020002		JSR	R5, \$PRINT		; GO PRINT OUT THE FOLLOWING MESSAGE.
1007	003574	001201			.WORD	\$CALF		; ADDRESS OF MESSAGE TO BE TYPED
1008	003576	011067	175732		MOV	(R0), SAVTST		; SET UP TEST MAP...LO 64K.
1009	003602	011167	175730		MOV	(R1), SAVTST+2		; HI 64K.
1010	003606	011000			MOV	(R0), R0		; GET LOW MEM MAP
1011	003610	042700	177760		BIC	#177760, R0		; MASK ALL BUT BOTTOM 4 BANKS

M08

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
 DZQMC.P11 02-DEC-76 08:47

MACY11 27(1006) 02-DEC-76 09:00 PAGE 21

SEQ 0103

```

1012 003614 020027 000017          CMP    R0    #17    ;CHECK THAT BOTTOM 16K IS ALL THERE!
1013 003620 001530          BEQ    GMPR   ;BRANCH IF BOTTOM 16K EXISTS
1014 003622 004567 017750          JSR    R5    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1015 003626 025610          .WORD  INSUFF ;ADDRESS OF MESSAGE TO BE TYPED
1016                                ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1017 003630 000000          6$:   HALT      ;FATAL ERROR HALT...
1018                                ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
1019                                ;*****
1020                                ;* SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
1021                                ;* DATI, DATIP, DATO, AND DATOB.
1022                                ;*****
1023 003632 012706 001100          TIMEOUT: MOV   #STACK, SP ;SET UP THE STACK POINTER.
1024 003636 005067 174744          CLR   MMAVA ;CLEAR MEM MGMT AVAILABLE FLAG.
1025 003642 032777 010000 175270          BIT   #SW12, $SWR ;CHECK IF MEM MGMT TO BE INHIBITED.
1026 003650 001011          BNE   1$    ;BR IF NO MEM MGMT.
1027 003652 012737 003674 000004          MOV   #1$, $#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
1028 003660 005037 177572          CLR   $#SR0 ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1029 003664 004767 011124          JSR   PC    MMINIT ;INIT ALL MEM MGMT REGISTERS.
1030 003670 005267 174712          INC   MMAVA ;SET MEM MGMT AVAILABLE FLAG.
1031                                1$:
1032 003674 004567 017676          JSR   R5    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1033 003700 025523          .WORD  BYTMES ;ADDRESS OF MESSAGE TO BE TYPED
1034                                ;"BYTE MEMORY MAP:"
1035                                CLR   R0    ;SET UP TYPE OUT FLAG.
1036 003704 005002          CLR   R2    ;SET ADDRESS POINTER TO ZERO.
1037 003706 012737 003752 000004          MOV   #20$, $#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
1038 003714 105712          10$:  TSTB  (R2) ;DO DATI ONLY.
1039 003716 032702 000001          BIT   #BIT0, R2 ;CHECK FOR WORD ADDRESS.
1040 003722 001001          BNE   11$   ;BR IF ODD BYTE ADDRESS.
1041 003724 011212          MOV   (R2), (R2) ;DO DATI, DATO...NOP FOR READ ONLY MAP.
1042 003726 151212          11$:  BISB  (R2), (R2) ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
1043 003730 005700          TST   R0    ;CHECK FOR PREVIOUS TYPOUT.
1044 003732 001023          BNE   30$   ;BR IF ALREADY TYPED "FROM".
1045 003734 004567 017636          JSR   R5    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1046 003740 025573          .WORD  FROM   ;ADDRESS OF MESSAGE TO BE TYPED
1047                                ;"FROM"
1048 003742 010246          MOV   R2, -(SP) ;PUT THE DATA ON THE STACK.
1049 003744 004767 021266          JSR   PC    $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1050 003750 000413          BR    29$   ;GO TO ADDRESS POINTER UPDATE.
1051                                ;* TIME OUTS COME HERE.
1052 003752 022626          20$:  CMP   (SP)+, (SP)+ ;POP TWO OFF STACK.
1053 003754 005700          TST   R0    ;CHECK FOR PREVIOUS TYPOUT.
1054 003756 001411          BEQ   30$   ;BR IF ALREADY TYPED "TO".
1055 003760 004567 017612          JSR   R5    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1056 003764 025603          .WORD  TO    ;ADDRESS OF MESSAGE TO BE TYPED
1057                                ;"TO"
1058 003766 005302          DEC   R2    ;BACK UP ONE BYTE.
1059 003770 010246          MOV   R2, -(SP) ;PUT THE DATA ON THE STACK.
1060 003772 004767 021240          JSR   PC    $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1061 003776 005202          JNC   R2    ;RESET ADDRESS POINTER.
1062 004000 005100          29$:  COM   R0    ;RESET PREVIOUS TYPOUT FLAG.
1063 004002 005202          30$:  INC   R2    ;UPDATE ADDRESS POINTER TO NEXT BYTE.
1064 004004 001423          BEQ   31$   ;EXIT IF ZERO REACHED.
1065 004006 032702 017777          BIT   #MASK4K, R2 ;CHECK FOR 4K BANK BOUNDARY.
1066 004012 001340          BNE   10$   ;BR IF MORE THIS 4K BANK.
1067 004014 005767 174566          TST   MMAVA ;CHECK IF MEM MGMT IS AVAILABLE.

```

```

1068 004020 001735      BEQ      10$      ;BR IF NO MEM MGMT.
1069 004022 022737 007600 172346    CMP      #7600, @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
1070 004030 001411      BEQ      31$      ;EXIT WHEN ALL DONE.
1071 004032 012702 060000      MOV      #60000, R2 ;RESET VIRTUAL ADDRESS POINTER.
1072 004036 013737 172346 172344    MOV      @#KIPAR3, @#KIPAR2 ;SAVE MEM MGMT REG FOR TYPEOUT.
1073 004044 062737 000200 172346    ADD      #200, @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1074 004052 000720      BR      10$      ;BR BACK TO DO NEXT BANK.
1075 004054 005700      31$:    TST      R0      ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1076 004056 001407      BEQ      32$      ;BR TO EXIT IF TYPING ALL DONE.
1077 004060 004567 017512      JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1078 004064 025603      .WORD   TO      ;ADDRESS OF MESSAGE TO BE TYPED
1079                                     "TO"
1080                                     ;BACK ADDRESS POINTER UP ONE BYTE.
1081 004066 005302      DEC      R2
1082 004070 010246      MOV      R2, -(SP) ;PUT THE DATA ON THE STACK.
1083 004072 004767 021140      JSR      PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1084 004076 000000      32$:    HALT      ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1085 004100 000654      BR      TIMEOUT ;* TO RUN THE MAIN PROGRAM RESTART AT 200 CR 204.
1086                                     ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1087                                     .SBTTL MAP PARITY REGISTERS
1088                                     ;*****
1089                                     ;* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1090                                     ;* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1091                                     ;*****
1092
1093 004102 012704 002314      GMPR:    MOV      #MPRX, R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
1094 004106 032777 000100 175024    BIT      #SW06, @SWR      ;CHECK FOR INHIBIT PARITY SWITCH.
1095 004114 001036      BNE      GMPRD      ;BR IF INHIBIT PARITY.
1096 004116 012703 002114      MOV      #GMPRD, R3      ;SET UP TABLE POINTER
1097 004122 012737 004144 000004    MOV      #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1098 004130 042713 000001      GMPRA:  BIC      #1, (R3)      ;CLEAR FLAG BIT IN TABLE
1099 004134 005773 000000      TST      @#(R3)      ;DOES THIS MEMORY PARITY REGISTER EXIST.
1100                                     ;* IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1101 004140 012324      MOV      (R3)+, (R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
1102 004142 000403      BR      GMPRC      ;SKIP TIMEOUT SERVICE CODE
1103                                     ;* TIMEOUT COMES HERE
1104 004144 022626      GMPRB:  CMP      (SP)+, (SP)+ ;RESTORE STACK POINTER
1105 004146 052723 000001      BIS      #1, (R3)+      ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1106 004152 005023      GMPRC:  CLR      (R3)+      ;CLEAR THE MAP...LO 64K.
1107 004154 005023      CLR      (R3)+      ;...HI 64K.
1108 004156 005023      CLR      (R3)+      ;...AND THE MASK.
1109 004160 020327 002314      CMP      R3, #MPRX      ;HAVE WE CHECKED ALL REGISTERS?
1110 004164 103761      BLO      GMPRA      ;NO - GO BACK TO CHECK NEXT ONE
1111 004166 005014      CLR      (R4)      ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1112 004170 012737 025200 000004    MOV      #ERRTRAP, @#ERRVEC ;RESTORE TRAPCATCHER
1113 004176 005767 176112      TST      MPRX      ;ANY PARITY REGISTERS PRESENT?
1114 004202 001006      BNE      MPAMEM      ;YES - GO TEST CONTROLS PRESENT
1115 004204 004567 017366      JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1116 004210 025671      .WORD   MTR      ;ADDRESS OF MESSAGE TO BE TYPED
1117                                     "NO MEMORY PARITY REGISTERS FOUND"
1118 004212 005014      GMPRD:  CLR      (R4)      ;MAKE SURE TABLE IS CLEAR.
1119 004214 000167 001050      JMP      MANUAL      ;AND SKIP ALL CONTROLS TESTING
1120

```

```

1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133 004220 004767 014474
1134 004224 012767 000001 175312
1135 004232 005067 175312
1136 004236 012702 014000
1137 004242 005767 174340
1138 004246 001404
1139 004250 012702 054000
1140 004254 004767 010534
1141
1142
1143
1144
1145
1146
1147
1148 004260 005067 175254
1149 004264 005067 175252
1150 004270 012703 002114
1151 004274 032713 000001
1152 004300 001052
1153 004302 012773 000004 000000
1154
1155 004310 011212
1156 004312 005712
1157 004314 042773 000004 000000
1158 004322 005773 000000
1159
1160
1161 004326 100014
1162 004330 032773 007740 000000
1163 004336 001404
1164 004340 012763 070032 000006
1165 004346 000413
1166 004350 012763 077772 000006 5$:
1167 004356 000407
1168 004360 032773 007740 000000 6$:
1169 004366 001417
1170 004370 012763 070000 000006
1171 004376 056763 175142 000002 7$:
1172 004404 056763 175136 000004
1173 004412 056767 175126 175120
1174 004420 056767 175122 175114
1175 004426 062703 000010
1176 004432 020327 002314

```

```

.SBTTL MAP PARITY MEMORY
;*****
;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
; - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
; - PARITY GENERATE OR DETECT LOGIC FAILED
; - PARITY ERROR BIT FAILED TO SET
; - PARITY BITS IN MEMORY LOCATION FAILED
; - I.E. BIT STUCK AT GOOD PARITY VALUE
;*****
MPAMEM: JSR PC, CLRPAR ; INITIALIZE ALL PARITY REGISTERS
        MOV #1, BITPT ; INITIALIZE 4K POINTER
        CLR BITPT+2 ; CLEAR HI 64K POINTER
        MOV #14000, R2 ; SET ADR POINTER TO 14000.
        TST MMAVA ; CHECK FOR MEM MGMT
        BEQ MAPRB ; BRANCH IF NO MEM MGMT
        MOV #54000, R2 ; SET ADR POINTER TO PAR2
        JSR PC, MMINIT ; SET UP ALL MEMORY MGMT REGISTERS.
;*****
;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
;* THEN WRITE TEST LOCATION VIA DAT0 & READ TEST LOCATION VIA DAT1
;* THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
;*****
MAPRB: CLR PMEMAP ; CLEAR THE PARITY MEMORY MAP
        CLR PMEMAP+2
1$: MOV #MPRO, R3 ; INITIALIZE TABLE ADDRESS
2$: BIT #1, (R3) ; IS THIS REGISTER PRESENT?
        BNE 3$ ; NO - GET THE NEXT ONE
        MOV #WWP, 2(R3) ; YES - SET WRITE WRONG PARITY
        ; AND CLEAR REST OF REGISTER
        MOV (R2), (R2) ; WRITE WRONG PARITY
        TST (R2) ; READ WRONG PARITY
        BIC #WWP, 2(R3) ; CLEAR WRITE WRONG PARITY
        TST 2(R3) ; OTHERWISE, CHECK TO SEE IF THIS
        ; CONTROL REGISTER GOT A PARITY
        ; ERROR
        BPL 6$ ; BRANCH IF IT DIDN'T AND CHECK
        BIT #7740, 2(R3) ; IS IT A CORE PAR. REG.
        BEQ 5$ ; BRANCH IF NOT.
        MOV #70032, 6(R3) ; IF IT IS SET UP MASK
        BR 7$ ; AND BRANCH TO SET BITS.
5$: MOV #77772, 6(R3) ; IF MOS SET UP MASK
        BR 7$ ; AND BRANCH TO SET BIT.
6$: BIT #7740, 2(R3) ; IF ANY BITS ARE SET
        BEQ 3$ ; THEN CSR IS MS11-K.
        MOV #70000, 6(R3) ; IF MS11-K SET MASK.
7$: BIS BITPT, 2(R3) ; SET FLAG IN MAP FOR THIS PARITY REGISTER
        BIS BITPT+2, 4(R3)
        BIS BITPT, PMEMAP ; SET FLAG IN PARITY MAP
        BIS BITPT+2, PMEMAP+2
3$: ADD #10, R3 ; STEP UP TO NEXT REGISTER
        CMP R3, #MPRX ; ARE WE DONE WITH TABLE?

```

```

1177 00443E 103716          BLO      2$          ;GO BACK TO CHECK FOR ANY MORE!
1178 004440 011212          MOV      (R2),      (R2)      ;CLEAR BAD PARITY
1179 004442 005767 174140        TST      MMAVA      ;CHECK FOR MEM MGMT
1180 004446 001425          BEQ      10$          ;BR IF NO MEM MGMT
1181 004450 062737 000200 172344 4$:  ADD      #200,      @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1182 004456 006367 175062          ASL      BITPT      ;UPDATE BANK POINTER...LO 64K.
1183 004462 006167 175060          ROL      BITPT+2    ;...HI 64K.
1184 004466 100422          BMI      TMAP        ;BR IF ALL DONE.
1185 004470 036767 175050 175026        BIT      BITPT,     MEMMAP    ;CHECK IF BANK EXISTS...LO 64K.
1186 004476 001274          BNE      1$          ;BR IF BANK EXISTS.
1187 004500 036767 175042 175020        BIT      BITPT+2,   MEMMAP+2 ;...HI 64K.
1188 004506 001270          BNE      1$          ;BR IF BANK EXISTS.
1189 004510 000757          BR       4$          ;BR IF BANK DOESN'T EXIST.
1190 004512 036767 175026 175004 11$:  BIT      BITPT,     MEMMAP    ;CHECK IF BANK EXISTS.
1191 004520 001263          BNE      1$          ;BR IF BANK EXISTS.
1192 004522 062702 020000 10$:  ADD      #20000,    R2        ;UPDATE ADDRESS POINTER TO NEXT BANK.
1193 004526 106367 175012          ASLB     BITPT      ;MOVE POINTER TO NEXT BANK.
1194 004532 100367          BPL      11$         ;BR IF MORE TO LOOK FOR.
1195
1196 ;*****
1197 ;* ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
1198 ;* AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
1199 ;*****
1200
1201 004534 004767 014160        TMAP:    JSR      PC,      CLPAR  ;INITIALIZE ALL PARITY REGISTERS PRESENT
1202 004540 004567 017032          JSR      R5,      $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1203 004544 025546          .WORD    MTHAP            ;ADDRESS OF MESSAGE TO BE TYPED
1204 ;"PARITY MEMORY MAP:"
1205 004546 012703 002114          MOV      #MPRO,    R3        ;INITIALIZE TABLE POINTER
1206 004552 032713 000001 1$:  BIT      #BIT0,    (R3)      ;CHECK IF THIS REGISTER IS PRESENT.
1207 004556 001046          BNE      2$          ;BR IF NOT PRESENT.
1208 004560 022763 070032 000006        CMP      #70032,   6(R3)
1209 004566 001004          BNE      3$          ;
1210 004570 004567 017002          JSR      R5,      $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1211 004574 026205          .WORD    MX3              ;ADDRESS OF MESSAGE TO BE TYPED
1212 ;"CORE PARITY"
1213 004576 000417          BR       5$          ;
1214 004600 022763 077772 000006 3$:  CMP      #77772,   6(R3)
1215 004606 001004          BNE      4$          ;
1216 004610 004567 016762          JSR      R5,      $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1217 004614 026224          .WORD    MX4              ;ADDRESS OF MESSAGE TO BE TYPED
1218 ;"MOS PARITY"
1219 004616 000407          BR       5$          ;
1220 004620 022763 070000 000006 4$:  CMP      #70000,   6(R3)
1221 004626 001003          BNE      5$          ;
1222 004630 004567 016742          JSR      R5,      $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1223 004634 026242          .WORD    MX5              ;ADDRESS OF MESSAGE TO BE TYPED
1224 ;"MS11-K CSR"
1225 004636          5$:  JSR      R5,      $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1226 004636 004567 016734          .WORD    MX1              ;ADDRESS OF MESSAGE TO BE TYPED
1227 004642 026155          ;"REGISTER AT"
1228 ;SAVE (R3) FOR TYPEOUT
1229 004644 011346          MOV      (R3),    -(SP)
1230 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
1231 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1232 004646 013746 177776          MOV      @#PSW,   -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK

```

```

1233 004652 004767 020116 JSR PC, $STYPOC ;GO TO THE SUBROUTINE
1234 004656 004567 016714 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1235 004662 026172 .WORD MX2 ;ADDRESS OF MESSAGE TO BE TYPED
1236 ;"CONTROLS"
1237 004664 010300 MOV R3, R0 ;SET UP R0 FOR TYPMAP ROUTINE.
1238 004666 005720 TST (R0)+ ;UPDATE POINTER TO MAP.
1239 004670 004767 014146 JSR PC, TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1240 004674 062703 000010 2$: ADD #10, R3 ;UPDATE TO NEXT REGISTER IN TABLE.
1241 004700 020327 002314 CMP R3, #MPRX ;ARE WE ALL DONE WITH TABLE?
1242 004704 103722 BLO 1$ ;BRANCH IF MORE REGISTERS
1243 004706 004567 016664 JSR R5, $SPRINT ;THE REASON I'M OUTPUTTING THIS CRLF
1244 004712 001201 $CRLF ;IS TO GIVE THE PRINTER ENOUGH TIME TO
1245 ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
1246 004714 026737 063060 002312 CMP 70000, @#MPR15+6 ;DO WE HAVE A MS11-K
1247 004722 001027 BNE CTRLS ;IF NO GO TO TESTS NOW.
1248 004724 043737 002306 001540 BIC @#MPR15+2,@#PMEMAP ;IF YES I AM GOING TO
1249 004732 043737 002310 001542 BIC @#MPR15+4,@#PMEMAP+2 ;CLEAR THE PARITY INDICATORS
1250 004740 012705 002314 MOV #MPRX, R5 ;FOR THAT PORTION OF MEMORY.
1251 004744 022537 002304 6$: CMP (R5)+, @#MPR15 ;SEARCH FOR MS11-K CSR IN
1252 004750 001375 BNE 6$ ;THE AVAILABILITY TABLE.
1253 004752 005045 CLR -(R5) ;AND CLEAR ITS ADDRESS FROM THE TABLE
1254 004754 052737 000001 002304 BIS #1, @#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
1255 004762 004567 016610 JSR R5, $SPRINT ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1256 004766 026260 .WORD MX6
1257 004770 005737 002314 TST @#MPRX ;ARE THERE ANY PARITY REGISTERS TO TEST?
1258 004774 001002 BNE CTRLS ;IF SO TEST THE BITS IN THE REGISTERS.
1259 004776 000167 000266 JMP MANUAL ;IF NO JUMP OVER REGISTER TESTS.
1260
1261 .SBTTL TEST PARITY REGISTERS
1262 ;*****
1263 ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1264 ;* CAN BE SET AND CLEARED.
1265 ;* THIS IS A ONCE ONLY TEST.
1266 ;*****
1267
1268 005002 012703 002114 CTRLS. MOV #MPRO, R3 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1269 005006 011302 1$: MOV (R3), R2 ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
1270 005010 062703 000010 ADD #10, R3 ;UPDATE POINTER TO NEXT PAR. REG. ADD.
1271 005014 032702 000001 BIT #1, R2 ;IS THIS REGISTER BEING USED?
1272 005020 001372 BNE 1$ ;GO TO NEXT IF NOT
1273 005022 020327 002314 CMP R3, #MPRX ;ARE WE AT END OF TABLE
1274 005026 003052 BGT RESCHK ;GO TO NEXT TEST IF YES
1275 005030 016367 177776 174460 MOV -2(R3), RESRVD ;GET MASK FOR REGISTER WE ARE WORKING ON
1276 005036 012700 000001 MOV #1, R0 ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
1277 005042 005012 CLR (R2) ;INITIALIZE THE PARITY REGISTER
1278 005044 011201 MOV (R2), R1 ;READ THE CONTENTS OF THE PARITY REGISTER
1279 005046 046701 174444 BIC RESRVD, R1 ;CLEAR BITS WHICH ARE RESERVED
1280 005052 001405 BEQ 2$ ;CHECK OTHER BITS - BRANCH IF OK
1281 005054 004767 013662 64$: JSR PC, SPRT ;SET UP VALUES FOR ERROR PRINTING.
1282 005060 004767 015164 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1283 005064 000001 .WORD 1 ;ERROR TYPE CODE.
1284 005066 030067 174424 2$: BIT R0, RESRVD ;IS THIS BIT RESERVED?
1285 005072 001025 BNE 3$ ;YES - DON'T TEST IT
1286 005074 010012 MOV R0, (R?) ;NO - SET THIS BIT IN THE PARITY REGISTER
1287 005076 011201 MOV (R2), R1 ;READ & SAVE CONTENTS OF THE PARITY REGISTER
1288 005100 005012 CLR (R2) ;CLEAR THE PARITY REGISTER

```

```

1289 005102 046701 174410      BIC    RESRVD, R1      ;CLEAR BIT LOCATIONS THAT ARE RESERVED
1290 005106 020001              CMP    RD, R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
1291 005110 001405              BEQ    66$           ;BRANCH OVER ERROR CALL IF GOOD DATA.
1292 005112 004767 013654      65$: JSR    PC, SPRNTD  ;SET UP VALUES FOR ERROR PRINTING.
1293 005116 004767 015126      JSR    PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1294 005122 000001              .WORD 1              ;ERROR TYPE CODE.
1295 005124              66$:
1296 005124 011201              MOV    (R2), R1      ;READ THE CONTENTS OF THE PARITY REGISTER
1297 005126 046701 174364      BIC    RESRVD, R1    ;CLEAR BITS WHICH ARE RESERVED
1298 005132 001405              BEQ    3$            ;CHECK OTHER BITS - BRANCH IF OK
1299 005134 004767 013602      67$: JSR    PC, SPRNT    ;SET UP VALUES FOR ERROR PRINTING.
1300 005140 004767 015104      JSR    PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1301 005144 000001              .WORD 1              ;ERROR TYPE CODE.
1302 005146 006300      3$: ASL    RD           ;ROTATE TO GET NEXT BIT TO BE TESTED
1303 005150 103346              BCC    2$           ;BRANCH IF NOT DONE WITH ALL BITS
1304 005152 000715              BR     1$           ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.
1305
1306 ;*****
1307 ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
1308 ;* THIS IS A ONCE ONLY TEST.
1309 ;*****
1310
1311 005154 012704 002114      RESCHK: MOV    #MPRO, R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1312 005160 010403      1$: MOV    R4, R3
1313 005162 062704 000010      ADD    #10, R4
1314 005166 032713 000001      BIT    #1, (R3)        ;IS THIS REGISTER BEING USED
1315 005172 001372              BNE    1$            ;BRANCH IF NO
1316 005174 012773 177777 000000      MOV    #-1, 2(R3)     ;SET ALL BITS TO A 1
1317 005202 022704 002314      CMP    #MPRX, R4      ;ARE WE AT THE END OF THE TABLE
1318 005206 002764              BLT    1$            ;IF YES THEN WE ARE READY TO TEST
1319 005210 000005              RESET              ;RESET THE WORLD
1320 005212 012703 002114      MOV    #MPRO, R3      ;LOAD INITIAL ADDRESS FOR POINTER
1321 005216 011302      2$: MOV    (R3), R2     ;STORE PARITY REGISTER ADDRESS
1322 005220 062703 000010      ADD    #10, R3
1323 005224 032702 000001      BIT    #1, R2
1324 005230 001372              BNE    2$           ;
1325 005232 022703 002314      CMP    #MPRX, R3
1326 005236 002014              BGE    MANUAL
1327 005240 011201              MOV    (R2), R1      ;GET CONTENTS OF REGISTER
1328 005242 005012              CLR    (R2)
1329 005244 042701 077772      BIC    #77772, R1     ;CLEAR BITS NOT EFFECTED BY RESET
1330 005250 005701              TST    R1            ;CHECK IF REST WERE CLEARED BY RESET
1331 005252 001405              BEQ    65$           ;BRANCH OVER ERROR CALL IF GOOD DATA.
1332 005254 004767 013462      64$: JSR    PC, SPRNT    ;SET UP VALUES FOR ERROR PRINTING.
1333 005260 004767 014764      JSR    PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1334 005264 000001              .WORD 1              ;ERROR TYPE CODE.
1335 005266              65$:
1336 005266 000753              BR     2$           ;BRANCH BACK TO CHECK NEXT REGISTER
1337
1338
1339 005270 012700 000014      MANJAL: MOV    #12, RD     ;SET COUNTER TO CLEAR 12 WORDS.
1340 005274 012701 001562      1$: MOV    #FSTADR, R1   ;STARTING AT FSTADR.
1341 005300 005021              CLR    (R1)+         ;CLEAR THE LOCATIONS.
1342 005302 005300              DEC    RD            ;COUNT.
1343 005304 001375              BNE    1$           ;BR IF MORE.
1344 005306 105767 174244      TSTB   SELFLG        ;CHECK FOR SELECT PARAMETERS STARTUP.

```



```

1348
1349
1350
1351
1352 005326 012700 000001
1353 005332 005001
1354 005334 005002
1355 005336 005003
1356 005340 004567 016232
1357 005344 026372
1358
1359
1360
1361 005346 013746 177776
1362 005352 004767 016046
1363 005356 042716 000001
1364 005362 005067 174146
1365 005366 005067 174144
1366 005372 062702 020000
1367 005376 005503
1368 005400 020367 016170
1369 005404 103403
1370 005406 101006
1371 005410 020216
1372 005412 101004
1373 005414 006300
1374 005416 006101
1375 005420 100364
1376 005422 000507
1377 005424 030067 174074
1378 005430 001003
1379 005432 030167 174070
1380 005436 001501
1381 005440 016704 016130
1382 005444
1383 005444 004567 016126
1384 005450 026457
1385
1386
1387
1388 005452 013746 177776
1389 005456 004767 015742
1390 005462 005716
1391 005464 001010
1392 005466 005767 016102
1393 005472 001005
1394 005474 016716 173464
1395 005500 016767 173462 016066
1396 005506 012667 174062
1397 005512 020467 016056
1398 005516 101352
1399 005520 103403
1400 005522 021667 174046
1401 005526 101346
1402 005530 032716 017777
1403 005534 001404

```

```

.SBTTL USER PARAMETER SELECTION SECTION
:*****
:* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
:*****
MANUL1: MOV #BIT0, R0 ;SET UP BANK POINTER.
        CLR R1 ;...HI 64K.
        CLR R2 ;CLEAR ADDRESS POINTER.
        CLR R3 ;...HI ADDRESS BITS.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD FADMES ;ADDRESS OF MESSAGE TO BE TYPED
        ;"FIRST ADDRESS:"
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $RDOCT ;GO TO THE SUBROUTINE
        BIC #BIT0, (SP) ;MAKE SURE ADDRESS IS ON A WORD BOUNDARY.
        CLR SAVTST ;INIT TEST MAP...LO 64K.
        CLR SAVTST+2 ;...HI 64K.
1$: ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
    ADC R3
    CMP R3, $HIOCT ;CHECK HI ADDRESS BITS.
    BLO 2$ ;BR IF NOT HI ENOUGH YET.
    BHI 3$ ;BR IF PAST SELECTED ADDRESS.
    CMP R2, (SP) ;CHECK THE LO ADDRESS BITS.
    BHI 3$ ;BR IF PAST SELECTED ADDRESS.
2$: ASL R0 ;UPDATE POINTER...LO 64K.
    ROL R1 ;...HI 64K.
    BPL 1$ ;BR BACK TO CHECK NEXT BANK.
    BR 17$ ;BR IF OVERFLOW.
3$: BIT R0, MEMMAP ;CHECK IF BANK EXISTS.
    BNE 4$ ;BR IF BANK EXISTS.
    BIT R1, MEMMAP+2 ;CHECK HI 64K.
    BEQ 17$ ;BR IF ADDRESS IN UN-MAPPED BANK.
4$: MOV $HIOCT, R4 ;SAVE FIRST ADR HI BITS.
10$: JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    .WORD LADMES ;ADDRESS OF MESSAGE TO BE TYPED
    ;"LAST ADDRESS:"
:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $RDOCT ;GO TO THE SUBROUTINE
        TST (SP) ;CHECK IF ADR 0 SELECTED (DEFAULT).
        BNE 11$ ;BR IF NOT 0 (DEFAULT)
        TST $HIOCT ;CHECK HI BITS.
        BNE 11$ ;BR IF NOT 0 (DEFAULT).
        MOV $TMP2, (SP) ;SET UP DEFAULT LAST ADR.
11$: MOV $TMP3, $HIOCT
    MOV (SP)+, LSTADR ;GET THE DATA.
    CMP R4, $HIOCT ;CHECK FOR LAST ADR BELOW FIRST ADR.
    BHI 10$ ;BR IF LAST BELOW FIRST.
    BLO 12$ ;BR IF LAST ABOVE FIRST.
    CMF (SP), LSTADR ;CHECK FOR LAST BELOW FIRST.
    BHI 10$ ;BR IF LAST BELOW FIRST.
12$: BIT #MASK4K, (SP) ;CHECK IF FIRST ADR ON BANK BOUNDARY.
    BEQ 13$ ;BR IF ON BOUNDARY.

```

```

1404 005536 010067 174026      MOV      R0,      FADMAP ;SET UP FIRST ADDRESS MAP.
1405 005542 010167 174024      MOV      R1,      FADMAP+2
1406 005546 050067 173762      13$:    BIS      R0,      SAVTST ;SET FLAG IN TEST MAP...LO 64K.
1407 005552 050167 173760      BIS      R1,      SAVTST+2 ;...HI 64K.
1408 005556 020367 016012      14$:    CMP      R3,      $HIOCT ;CHECK FOR PAST LAST ADR.
1409 005562 103404                BLO      15$      ;BR IF BELOW LAST ADR.
1410 005564 101020                BHI      16$      ;BR IF GONE PAST LAST ADR.
1411 005566 020267 174002      CMP      R2,      LSTADR ;CHECK FOR PAST LAST ADR.
1412 005572 101015                BHI      16$      ;BR IF GONE PAST LAST ADR.
1413 005574 062702 020000      15$:    ADD      #20000, R2     ;UPDATE ADDRESS POINTER.
1414 005600 005503                ADC      R3                ;...HI BITS.
1415 005602 006300                ASL      R0                ;UPDATE BANK POINTER...LO 64K.
1416 005604 006101                ROL      R1                ;...HI 64K.
1417 005606 100415                BMI      17$      ;BR IF OVERFLOW.
1418 005610 030067 173710      BIT      R0,      MEMMAP ;CHECK IF THIS BANK EXISTS.
1419 005614 001354                BNE      13$      ;BR IF BANK EXISTS.
1420 005616 030167 173704      BIT      R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1421 005622 001351                BNE      13$      ;BR IF BANK EXISTS.
1422 005624 000754                BR       14$      ;BR IF BANK DOESN'T EXIST.
1423 005626 030067 173672      16$:    BIT      R0,      MEMMAP ;CHECK IF THIS BANK EXISTS.
1424 005632 001010                BNE      20$      ;BR IF IT EXISTS.
1425 005634 030167 173666      BIT      R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1426 005640 001005                BNE      20$      ;BR IF IT EXISTS.
1427 005642 005726      17$:    TST      (SP)+      ;ADJUST THE STACK.
1428 005644 004567 015726      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1429 005650 026502                .WORD   BADADR           ;ADDRESS OF MESSAGE TO BE TYPED
1430                                ;"?ADDRESS IN UNMAPPED BANK?"
1431 005652 000606                BR       MANUAL         ;LOOP BACK TO THE BEGINNING.
1432 005654 010067 173722      20$:    MOV      R0,      LADMAP ;SET UP MAP FOR LAST ADDRESS.
1433 005660 010167 173720      MOV      R1,      LADMAP+2
1434 005664 005767 172716      21$:    TST      MMAVA        ;CHECK FOR MEMORY MANAGEMENT.
1435 005670 001404                BEQ      22$      ;BR IF NO MEM MGMT.
1436 005672 042716 160000      BIC      #160000, (SP)   ;ADJUST FSTADR TO VITRUAL BANK 0.
1437 005676 062716 040000      ADD      #40000, (SP)   ;...TO VIRTUAL BANK 2.
1438 005702 012667 173654      22$:    MOV      (SP)+, FSTADR ;SAVE FISRT ADDRESS OFF THE STACK.
1439 005706                30$:
1440 005706 004567 015664      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1441 005712 026537                .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
1442                                ;"SELECT CONSTANT:"
1443                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1444                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
1445 005714 013746 177776      MOV      @#PSW, -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
1446 005720 004767 015500      JSR      PC,      $RDOCT ;GO TO THE SUBROUTINE
1447 005724 012667 173660      MOV      (SP)+, .CONST ;SAVE THE CONSTANT
1448 005730 005767 172652      MANUL2: TST      MMAVA        ;CHECK IF MEM MGMT IS AVAILABLE.
1449 005734 001406                BEQ      31$      ;BR IF NO MEM MGMT.
1450 005736 042767 160000 173630      BIC      #160000, LSTADR ;ADJUST LSTADR TO VIRTUAL BANK 0.
1451 005744 062767 040000 173622      ADD      #40000, LSTADR ;... VIRTUAL BANK 2.
1452 005752 062767 000002 173614      31$:    ADD      #2, LSTADR     ;ADJUST LAST ADDRESS UP ONE WORD.
1453 005760 042767 000001 173606      BIC      #BIT0, LSTADR  ;MAKE SURE IT IS A WORD ADDRESS.
1454 005766 032767 017777 173600      BIT      #MASK4K, LSTADR ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
1455 005774 001004                BNE      START1        ;BR IF NOT ON BOUNDRY.
1456 005776 005067 173600      CLR      LADMAP        ;CLEAR OUT THE LAST ADDRESS MAP.
1457 006002 005067 173576      CLR      LADMAP+2
1458

```


SECTION 1: MEMORY ADDRESS TESTS

```

1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484 006054
1485 006054 004567 013200
1486 006060 000001
1487
1488 006062 000167 006472
1489
1490
1491 006066 004467 007050
1492 006072 004767 010472
1493 006076 010012
1494 006100 012201
1495 006102 020001
1496 006104 001405
1497 006106 004767 012704
1498 006112 004767 014132
1499 006116 000002
1500 006120
1501 006120 062700 000002
1502 006124 030502
1503 006126 001363
1504 006130 004767 007564
1505
1506
1507
1508 006134 004467 007440
1509 006140 004767 010424
1510 006144 162700 000002
1511 006150 014201
1512 006152 020001
1513 006154 001405
1514 006156 004767 012610
1515 006162 004767 014062
1516 006166 000002
1517 006170
1518 006170 030502
1519 006172 001364
1520 006174 004767 010210

```

```

.SBTTL SECTION 1: MEMORY ADDRESS TESTS
*****
*TEST 1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDRY BIT MASK.
*****
TST1:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   1          ;MINIMUM BLOCK SIZE OF 1 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST32     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
;* UPWARDS WORD ADDRESSING.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOV      R0,      (R2)   ;WRITE VALUE OF ADDRESS INTO ADDRESS
      MOV      (R2)+,  R1     ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   2          ;ERROR TYPE CODE.
65$:  ADD      #2,      R0     ;ADD #2 TO PHYSICAL ADDRESS
      BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE      2$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
;* CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
;* DOWNWARDS WORD ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$:   SUB      #2,      R0     ;DEC DATA BY 2
      MOV      -(R2),  R1     ;GET THE DATA FROM MEMORY
      CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      67$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:  JSR      PC,      SPRNT0 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   2          ;ERROR TYPE CODE.
67$:  BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE      4$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

```

1521
1522
1523
1524
1525
1526
1527
1528
1529
1530 006200
1531 006200 004567 013054
1532 006204 000000
1533
1534 006206 004467 006730
1535 006212 004767 010352
1536 006216 110022
1537 006220 005200
1538 006222 030502
1539 006224 001374
1540 006226 004767 007466
1541
1542
1543
1544 006232 004467 007342
1545 006236 004767 010326
1546 006242 005300
1547 006244 114201
1548 006246 120001
1549 006250 001405
1550 006252 004767 012514
1551 006256 004767 013766
1552 006262 000003
1553 006264
1554 006264 030502
1555 006266 001365
1556 006270 004767 010114
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567 006274
1568 006274 004567 012760
1569 006300 000000
1570
1571 006302 004467 007272
1572 006306 004767 010256
1573 006312 005100
1574 006314 062700 000002
1575 006320 010042
1576 006322 030502

```

```

*****
; *TEST 2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
; * R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
; * R1 = DATA READ FROM MEMORY (WAS)
; * R2 = VIRTUAL ADDRESS
; * R3 = NOT USED
; * R4 = NOT USED
; * R5 = BLOCK BOUNDARY BIT MASK.
*****
TST2:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * UPWARDS BYTE ADDRESSING.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOVB     R0,      (R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS
      INC     R0 ;ADD ONE TO PHYSICAL ADDRESS
      BIT     R5,      R2 ;CHECK FOR END OF A BLOCK.
      BNE     2$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

; * CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
; * DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$:   DEC     R0 ;DEC DATA BY 1
      MOVB     -(R2), R1 ;GET THE DATA FROM MEMORY
      CMPB    R0,      R1 ;CHECK THE DATA...LO BYTE ONLY VALID.
      BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   3 ;ERROR TYPE CODE.
65$:  BIT     R5,      R2 ;CHECK FOR END OF A BLOCK.
      BNE     4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

*****
; *TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
; * R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
; * R1 = DATA READ FROM MEMORY (WAS)
; * R2 = VIRTUAL ADDRESS
; * R3 = NOT USED
; * R4 = NOT USED
; * R5 = BLOCK BOUNDARY BIT MASK.
*****
TST3:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * DOWNWARDS WORD ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
      COM     R0 ;COMPLEMENT THE ADR
2$:   ADD     #2,      R0 ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
      MOV     R0,      -(R2) ;PUT DATA INTO MEMORY
      BIT     R5,      R2 ;CHECK FOR END OF A BLOCK.

```

```

1577 006324 001373          BNE    2$          ;BRANCH IF MORE IN CURRENT BLOCK.
1578 006326 004767 010056 JSR    PC,        MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.
1579
1580 ;* CHECK COMPLEMENT DATA WRITTEN DOWN
1581 ;* UPWARDS WORD ADDRESSING.
1582 006332 004467 006604 JSR    R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1583 006336 004767 010226 3$: JSR    PC,        PHYADR ;GET PHYSICAL ADDRESS INTO R0
1584 006342 005100          COM    R0          ;COMPLEMENT IT
1585 006344
1586 006344 012201          MOV    (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1587 006346 020001          CMP    R0,        R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1588 006350 001405          BEQ    65$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
1589 006352 004767 012440 64$: JSR    PC,        SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
1590 006356 004767 013666 JSR    PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1591 006362 000002          .WORD 2           ;ERROR TYPE CODE.
1592 006364
1593 006364 162700 000002          SUB    #2,        R0      ;COUNT DOWN WITH ADDRESS
1594 006370 030502          BIT    R5,        R2      ;CHECK FOR END OF A BLOCK.
1595 006372 001364          BNE    4$          ;BRANCH IF MORE IN CURRENT BLOCK.
1596 006374 004767 007320 JSR    PC,        MMUP   ;FIND NEXT BLOCK AND LOOP TO 3$.
1597
1598 ;*****
1599 ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1600 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1601 ;* R1 = DATA READ FROM MEMORY (WAS)
1602 ;* R2 = VIRTUAL ADDRESS
1603 ;* R3 = NOT USED
1604 ;* R4 = NOT USED
1605 ;* R5 = BLOCK BOUNDARY BIT MASK.
1606 ;*****
1607 006400
1608 006400 004567 012654 1$T4: JSR    R5,        $SCOPE ;GO TO SCOPE ROUTINE.
1609 006404 000000          .WORD 0           ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1610 ;* UPWARDS BYTE ADDRESSING.
1611 006406 004467 006530 JSR    R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1612 006412 004767 010226 1$: JSR    PC,        BANKNO ;GET THE BANK NUMBER INTO R0
1613 006416 110022 2$: MOVB  R0,        (R2)+ ;WRITE BANK # INTO ALL ADDRESSES
1614 006420 030502          BIT    R5,        R2      ;CHECK FOR END OF A BLOCK.
1615 006422 001375          BNE    2$          ;BRANCH IF MORE IN CURRENT BLOCK.
1616 006424 004767 007270 JSR    PC,        MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
1617
1618 ;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1619 ;* UPWARDS BYTE ADDRESSING.
1620 006430 004467 006506 JSR    R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1621 006434 004767 010204 3$: JSR    PC,        BANKNO ;GET THE BANK NUMBER INTO R0
1622 006440 112201 4$: MOVB  (R2)+,    R1      ;READ THE DATA OUT OF MEMORY
1623 006442 020001          CMP    R0,        R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1624 006444 001405          BEQ    65$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
1625 006446 004767 012326 64$: JSR    PC,        SPRT1  ;SET UP VALUES FOR ERROR PRINTING.
1626 006452 004767 013572 JSR    PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1627 006456 000003          .WORD 3           ;ERROR TYPE CODE.
1628 006460
1629 006460 030502          BIT    R5,        R2      ;CHECK FOR END OF A BLOCK.
1630 006462 001366          BNE    4$          ;BRANCH IF MORE IN CURRENT BLOCK.
1631 006464 004767 007230 JSR    PC,        MMUP   ;FIND NEXT BLOCK AND LOOP TO 3$.
1632

```

```

1633
1634
1635
1636
1637
1638
1639
1640
1641
1642 006470
1643 006470 004567 012564
1644 006474 000000
1645
1646 006476 004467 007076
1647 006502 004767 010136
1648 006506 005100
1649 006510 110042
1650 006512 030502
1651 006514 001375
1652 006516 004767 007666
1653
1654
1655
1656 006522 004467 007052
1657 006526 004767 010112
1658 006532 005100
1659 006534 114201
1660 006536 020001
1661 006540 001405
1662 006542 004767 012224
1663 006546 004767 013476
1664 006552 000003
1665 006554
1666 006554 030502
1667 006556 001366
1668 006560 004767 007624

```

```

*****
;TEST 5 WRITE 1'S COMPLEMENT OF BANK #.
;* RO = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
*****
TSTS.
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
;* DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      BANKNO ;GET THE BANK # INTO R0
      COM      R0 ;SET 1'S COMPLEMENT OF BANK #
2$:   MOVB     R0,      -(R2) ;READ DATA OUT OF MEMORY
      BIT      R5,      R2 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BNE     2$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR      PC,      MMDOWN ;SET UP VALUES FOR ERROR PRINTING.
      ;*** ERROR *** (GO TYPE A MESSAGE)
      ;ERROR TYPE CODE.
;* CHECK THAT DATA WRITTEN CAN BE READ.
;* DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR      PC,      BANKNO ;GET THE BANK # INTO R0
      COM      R0 ;SET 1'S COMPLEMENT OF BANK #
4$:   MOVB     -(R2),   R1 ;READ DATA OUT OF MEMORY
      CMP      R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   3 ;ERROR TYPE CODE.
65$:  BIT      R5,      R2 ;CHECK FOR END OF A BLOCK.
      BNE     4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

```

1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684 006564
1685 006564 004567 012470
1686 006570 000000
1687 006572 016700 173012
1688 006576 004467 006390
1689 006602 010022
1690 006604 030502
1691 006606 001375
1692 006610 004767 007104
1693
1694
1695
1696
1697
1698 006614
1699 006614 004567 012440
1700 006620 000000
1701 006622 016700 172762
1702 006626 004467 006310
1703 006632
1704 006632 012201
1705 006634 020001
1706 006636 001405
1707 006640 004767 012152
1708 006644 004767 013400
1709 006650 000004
1710 006652
1711 006652 030502
1712 006654 001366
1713 006656 004767 007036
1714
1715
1716
1717 006662 032777 000400 172250
1718 006670 001416
1719 006672 017746 172242
1720 006676 042716 177740
1721 006702 022726 000006
1722 006706 001007
1723 006710 162767 000001 172164
1724 006716 162767 000030 172162

```

```

.SBTTL SECTION 2: WORST CASE NOISE TESTS
*****
* THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
* MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
*****
*TEST 6 WRITE A CONSTANT INTO MEMORY.
* THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST6: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
TST6A: MOV CONST, R0 ;GET USER CONSTANT
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R0, (R2)+ ;WRITE CONSTANT INTO MEMORY.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
*TEST 7 READ MEMORY AND COMPARE TO CONSTANT.
* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
*****
TST7: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV CONST, R0 ;GET USER CONSTANT
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 4 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
* BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
BIT #SW08, @SWR ;CHECK THAT LOOP ON TEST BIT SET
BEQ TST10 ;BRANCH IF NOT LOOP ON TEST
MOV @SWR, -(SP) ;GET SWITCH REGISTER DATA.
BIC #177740, (SP) ;CLEAR NON-TEST-NUMBER SWITCHES.
CMP #6, (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
BNE TST10 ;BRANCH IF NOT TEST 6
SUB #1, $TSTNM ;RESET TEST NUM
SJB #TST7-TST6, $LPADR ;RESET LOOP ADR

```


B10

MP.NDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCC.P11 02-DEC-76 08:47

T7 READ MEMORY AND COMPARE TO CONSTANT.

MACY11 27(1006) 02-DEC-76 09:00 PAGE 36

SEQ 0119

```

1725 006724 000722
1726
1727
1728
1729
1730
1731 006726
1732 006726 004567 012326
1733 006732 000000
1734 006734 016704 172672
1735 006740 004767 011322
1736 006744 012400
1737 006746 001420
1738 006750 004467 006166
1739 006754 010012
1740 006756 012201
1741 006760 020001
1742 006762 001405
1743 006764 004767 012026
1744 006770 004767 013254
1745 006774 000004
1746 006776
1747 006776 030502
1748 007000 001365
1749 007002 004767 006712
1750 007006 000754

```

BR TST6A ;GO TO TEST 6

```

;*****
;*TEST 10 WORSE CASE NOISE (PARITY) WORD TESTING
;* CHECK MEMORY WITH A SERIES OF PATTERNS
;*****
TST10:

```

```

JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.MOV .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
.MPPAT, R4 ;INITIALIZE PATTERN TABLE POINTER
1$: JSR PC, CKPMER ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
MOV (R4)+, R0 ;GET THE DATA PATTERN.
BEQ TST11 ;BR IF END OF TABLE.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2$: MOV R0, (R2) ;PUT DATA PATTERN INTO MEMORY.
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 4 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 2$.
BR 1$ ;BR BACK TO DO NEXT PATTERN

```

C10

T11 ROTATE A "0" BIT THROUGH A FIELD OF ONES.

```

1751
1752
1753
1754 007010
1755 007010 004567 012244
1756 007014 000000
1757 007016 012700 177777
1758 007022 004767 007656
1759 007026 004467 006110
1760 007032 000241
1761 007034 004767 007664
1762 007040 016201 177776
1763 007044 103402
1764 007046 020001
1765 007050 001405
1766 007052 004767 011740
1767 007056 004767 013166
1768 007062 000005
1769 007064
1770 007064 030502
1771 007066 001361
1772 007070 004767 006624
1773
1774
1775
1776
1777 007074
1778 007074 004567 012160
1779 007100 000000
1780 007102 005000
1781 007104 004767 007574
1782 007110 004467 006026
1783 007114 000261
1784 007116 004767 007602
1785 007122 016201 177776
1786 007126 103002
1787 007130 020001
1788 007132 001405
1789 007134 004767 011656
1790 007140 004767 013104
1791 007144 000005
1792 007146
1793 007146 030502
1794 007150 001361
1795 007152 004767 006542
  
```

```

*****
*TEST 11 ROTATE A "0" BIT THROUGH A FIELD OF ONES.
*****
†ST11:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      MOV     #-1,     R0      ;SET CHECK WORD
      JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
      JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   CLC
      JSR     PC,     ROTATE  ;CLEAR CARRY BIT IN PSW
      MOV     -2(R2), R1      ;GET RESULT
      BCC    63$,     ;BRANCH IF 'C' BIT WAS SET
      CMP    R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    64$,     ;BRANCH OVER ERROR CALL IF GOOD DATA.
63$:  JSR     PC,     SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   5          ;ERROR TYPE CODE.
64$:  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
      BNE    1$,     ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,     MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

```

*****
*TEST 12 ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
*****
†ST12:
      JSR     R5,     $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      CLR    R0        ;SET CHECK WORD
      JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
      JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   SEC
      JSR     PC,     ROTATE  ;SET 'C' BIT IN PSW
      MOV     -2(R2), R1      ;GO ROTATE '1' BIT
      BCC    63$,     ;GET RESULT
      CMP    R0,     R1      ;BRANCH IF 'C' IS CLEAR
      BEQ    64$,     ;COMPARE THE CHECK WORD WITH THE DATA READ.
63$:  JSR     PC,     SPRT2   ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR     PC,     $ERROR  ;SET UP VALUES FOR ERROR PRINTING.
      .WORD   5          ;*** ERROR *** (GO TYPE A MESSAGE)
      ;ERROR TYPE CODE.
64$:  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
      BNE    1$,     ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,     MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```


E10

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMC.P11 02-DEC-76 08:47

T13 1 XOR 8 TEST PATTERN

MACY11 27(1006) 02-DEC-76 09:00 PAGE 39

SEQ 0121

```

1852 007322 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1853 007324 020001      CMP      R0, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
1854 007326 001405      BEQ     67$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
1855 007330 004767 011462 66$: JSR     PC, SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
1856 007334 004767 012710 JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1857 007340 000006      .WORD   6             ;ERROR TYPE CODE.
1858 007342 67$:
1859 007342 005100      COM     R0             ;
1860 007344 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1861 007346 020001      CMP      R0, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
1862 007350 001405      BEQ     69$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
1863 007352 004767 011440 68$: JSR     PC, SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
1864 007356 004767 012666 JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1865 007362 000006      .WORD   6             ;ERROR TYPE CODE.
1866 007364 69$:
1867 007364 005100      COM     R0             ;
1868 007366 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1869 007370 020001      CMP      R0, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
1870 007372 001405      BEQ     71$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
1871 007374 004767 011416 70$: JSR     PC, SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
1872 007400 004767 012644 JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1873 007404 000006      .WORD   6             ;ERROR TYPE CODE.
1874 007406 71$:
1875 007406 005100      COM     R0             ;COMPLEMENT TEST DATA
1876 007410 005304      DEC     R4             ;DECREMENT 128. WORD COUNTER
1877 007412 001332      BNE     12$,           ;
1878 007414 005100      COM     R0             ;COMPLEMENT TEST DATA
1879 007416 030502      BIT     R5, R2         ;CHECK FOR END OF A BLOCK.
1880 007420 001325      BNE     11$,           ;BRANCH IF MORE IN CURRENT BLOCK.
1881 007422 004767 006272 JSR     PC, MMUP       ;FIND NEXT BLOCK AND LOOP TO 11$.
1882
1883 ;*****
1884 ;* COMPLEMENT 1 XOR 8 PATTERN WRITTEN ABOVE.
1885 ;*****
1886 007426 004467 005510 JSR     R4, INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1887 007432 012704 000040 21$: MOV     #32, R4       ;SET UP 128. WORD BLOCK COUNTER
1888 007436 005122 22$: COM     (R2)+       ;COMPLEMENT PATTERN
1889 007440 005122      COM     (R2)+
1890 007442 005122      COM     (R2)+
1891 007444 005122      COM     (R2)+
1892 007446 005304      DEC     R4             ;CHECK FOR 128. WORDS DONE.
1893 007450 001372      BNE     22$,           ;BRANCH IF MORE
1894 007452 030502      BIT     R5, R2         ;CHECK FOR END OF A BLOCK.
1895 007454 001366      BNE     21$,           ;BRANCH IF MORE IN CURRENT BLOCK.
1896 007456 004767 006236 JSR     PC, MMUP       ;FIND NEXT BLOCK AND LOOP TO 21$.
1897
1898 ;*****
1899 ;* CHECK COMPLEMENTED 1 XOR 8 TEST PATTERN.
1900 ;*****
1901 007462 012700 177777 MOV     #-1, R0        ;INITIALIZE TEST WORD.
1902 007466 004467 005450 JSR     R4, INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1903 007472 012704 000040 31$: MOV     #32, R4       ;SET 128. WORD COUNTER
1904 007476 32$:
1905 007476 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1906 007500 020001      CMP      R0, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
1907 007502 001405      BEQ     73$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.

```

F10

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 27(1006) 02-DEC-76 09:00 PAGE 40
 DZQMC.F11 02-DEC-76 08:47 T13 1 XOR 8 TEST PATTERN

SEQ 0122

1908	007504	004767	011306	72\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1909	007510	004767	012534		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1910	007514	003006			.WORD	6		;ERROR TYPE CODE.
1911	007516			73\$:				
1912	007516	005100			COM	R0		
1913	007520	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1914	007522	020001			CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1915	007524	001405			BEQ	75\$;BRANCH OVER ERROR CALL IF GOOD DATA.
1916	007526	004767	011264	74\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1917	007532	004767	012512		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1918	007536	000006			.WORD	6		;ERROR TYPE CODE.
1919	007540			75\$:				
1920	007540	005100			COM	R0		
1921	007542	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1922	007544	020001			CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1923	007546	001405			BEQ	77\$;BRANCH OVER ERROR CALL IF GOOD DATA.
1924	007550	004767	011242	76\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1925	007554	004767	012470		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1926	007560	000006			.WORD	6		;ERROR TYPE CODE.
1927	007562			77\$:				
1928	007562	005100			COM	R0		
1929	007564	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1930	007566	020001			CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1931	007570	001405			BEQ	79\$;BRANCH OVER ERROR CALL IF GOOD DATA.
1932	007572	004767	011220	78\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1933	007576	004767	012446		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1934	007602	000006			.WORD	6		;ERROR TYPE CODE.
1935	007604			79\$:				
1936	007604	005100			COM	R0		;COMPLEMENT TEST DATA
1937	007606	005304			DEC	R4		;DECREMENT 128. WORD COUNTER
1938	007610	001332			BNE	32\$		
1939	007612	005100			COM	R0		;COMPLEMENT TEST DATA
1940	007614	030502			BIT	R5,	R2	;CHECK FOR END OF A BLOCK.
1941	007616	001325			BNE	31\$;BRANCH IF MORE IN CURRENT BLOCK.
1942	007620	004767	006074		JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO 31\$.

G10

```

1943
1944
1945
1946 007624
1947 007624 004567 011430
1948 007630 000777
1949
1950 007632 000167 000312
1951
1952 007636 005000
1953 007640 012703 177777
1954 007644 004467 005272
1955 007650 004767 007116
1956 007654 030502
1957 007656 001374
1958 007660 004767 006034
1959
1960
1961
1962
1963 007664 005000
1964 007666 004467 005250
1965 007672 012704 000100
1966 007676
1967 007676 012201
1968 007700 020001
1969 007702 001405
1970 007704 004767 011106
1971 007710 004767 012334
1972 007714 000007
1973 007716
1974 007716 012201
1975 007720 020001
1976 007722 001405
1977 007724 004767 011066
1978 007730 004767 012314
1979 007734 000007
1980 007736
1981 007736 012201
1982 007740 020001
1983 007742 001405
1984 007744 004767 011046
1985 007750 004767 012201
1986 007754 000007
1987 007756
1988 007756 012201
1989 007760 020001
1990 007762 001405
1991 007764 004767 011026
1992 007770 004767 012254
1993 007774 000007
1994 007776
1995 007776 005100
1996 010000 005304
1997 010002 001335
1998 010004 005100

```

```

*****
*TEST 14      3 XOR 9 TEST PATTERN.
*****
TST14:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD    777          ;MINIMUM BLOCK SIZE OF 256. WORDS
                               ;REQUIRED FOR THIS TEST.
      JMP      TST15      ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                               ;AVAILABLE FOR TEST.
.3X9: CLR      R0          ;SET UP TEST DATA
      MOV      #-1,      R3 ;SET COM DATA REG
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      BIT      R5,      R2   ;CHECK FOR END OF A BLOCK.
      BNE     1$,      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
* CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
*****
      CLR      R0          ;SET CHECK WORD
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$:  MOV      #64.,     R4   ;SET 256. WORD COUNTER
12$:
      MOV      (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,      R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7           ;ERROR TYPE CODE.
65$:
      MOV      (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,      R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     67$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7           ;ERROR TYPE CODE.
67$:
      MOV      (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,      R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     69$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7           ;ERROR TYPE CODE.
69$:
      MOV      (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,      R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     71$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$:  JSR      PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7           ;ERROR TYPE CODE.
71$:
      COM      R0          ;COMPLEMENT CHECK WORD
      DEC     R4          ;DECREMENT 256. WORD COUNTER
      BNE     12$,      ;
      COM     R0          ;COMPLEMENT CHECK WORD

```

H10

```
1999 010006 030502 BIT R5 R2 ;CHECK FOR END OF A BLOCK.
2000 010010 001330 BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.
2001 010012 004767 005732 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
2002
2003 ::*****
2004 :* CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
2005 ::*****
2006 010016 005000 CLR R0
2007 010020 004467 005116 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2008 010024 012704 000100 21$: MOV #64., R4 ;SET 256. WORD COUNTER
2009 010030 012703 000004 22$: MOV #4, R3 ;SET 4 WORD COUNTER
2010 010034 23$:
2011 010034 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2012 010036 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2013 010040 001405 BEQ 73$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2014 010042 004767 010750 72$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2015 010046 004767 012176 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2016 010052 000007 .WORD 7 ;ERROR TYPE CODE.
2017 010054 73$:
2018 010054 005100 COM R0 ;COMPLEMENT CHECK WORD
2019 010056 005142 COM -(R2) ;COMPLEMENT TEST DATA
2020 010060 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2021 010062 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2022 010064 001405 BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2023 010066 004767 010724 74$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2024 010072 004767 012152 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2025 010076 000007 .WORD 7 ;ERROR TYPE CODE.
2026 010100 75$:
2027 010100 005100 COM R0 ;COMPLEMENT CHECK WORD
2028 010102 005142 COM -(R2) ;COMPLEMENT TEST DATA
2029 010104 012201 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2030 010106 020001 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2031 010110 001405 BEQ 77$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2032 010112 004767 010700 76$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2033 010116 004767 012126 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2034 010122 000007 .WORD 7 ;ERROR TYPE CODE.
2035 010124 77$:
2036 010124 005303 DEC R3 ;DECREMENT 4 WORD COUNTER
2037 010126 001342 BNE 23$ ;BR IF NOT DONE.
2038 010130 005100 COM R0 ;COMPLEMENT CHECK WORD
2039 010132 005304 DEC R4 ;DECREMENT 256. WORD COUNTER
2040 010134 001335 BNE 22$ ;BR IF NOT DONE.
2041 010136 005100 COM R0 ;COMPLEMENT CHECK WORD
2042 010140 030502 BIT R5 R2 ;CHECK FOR END OF A BLOCK.
2043 010142 001330 BNE 21$ ;BRANCH IF MORE IN CURRENT BLOCK.
2044 010144 004767 00555C JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```

2045 ;*****
2046 ;*TEST 15 COMPLEMENT 3 XOR 9 TEST PATTERN
2047 ;*****
2048 †TST15:
2049 JSR R5 $SCOPE ;GO TO SCOPE ROUTINE.
2050 .WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS
2051 ; REQUIRED FOR THIS TEST.
2052 JMP TST16 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2053 ; AVAILABLE FOR TEST.
2054 MOV #-1, R0 ;SET UP TEST DATA
2055 CLR R3 ;SET COM DATA REG
2056 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2057 1$: JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2058 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2059 BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
2060 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
2061
2062
2063 ;*****
2064 ;* CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
2065 ;*****
2066 MOV #-1, R0 ;SET CHECK WORD
2067 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2068 11$: MOV #64, R4 ;SET 256. WORD COUNTER
2069 12$:
2070 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2071 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2072 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2073 64$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2074 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2075 .WORD 7 ;ERROR TYPE CODE.
2076 65$:
2077 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2078 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2079 BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2080 66$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2081 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2082 .WORD 7 ;ERROR TYPE CODE.
2083 67$:
2084 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2085 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2086 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2087 68$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2088 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2089 .WORD 7 ;ERROR TYPE CODE.
2090 69$:
2091 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2092 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2093 BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2094 70$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2095 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2096 .WORD 7 ;ERROR TYPE CODE.
2097 71$:
2098 COM R0 ;COMPLEMENT CHECK WORD
2099 DEC R4 ;DECREMENT 256. WORD COUNTER
2100 BNE 12$

```


J10

```

2101 010332 005100          COM      R0          ;COMPLEMENT CHECK WORD
2102 010334 030502          BIT      R5          R2      ;CHECK FOR END OF A BLOCK.
2103 010336 001330          BNE     11$         ;BRANCH IF MORE IN CURRENT BLOCK.
2104 010340 004767 005354    JSR     PC,        MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2105
2106
2107
2108
2109 010344 012700 177777      MOV     #-1,       R0      ;SET UP CHECK WORD.
2110 010350 004467 004566      JSR     R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2111 010354 012704 000100      21$:   MOV     #64.,  R4      ;SET 256. WORD COUNTER
2112 010360 012703 000004      22$:   MOV     #4,    R3      ;SET 4 WORD COUNTER
2113 010364
2114 010364 012201          MOV     (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2115 010366 020001          CMP     R0,       R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2116 010370 001405          BEQ     73$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2117 010372 004767 010420      72$:   JSR     PC,        SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2118 010376 004767 011646      JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2119 010402 000007          .WORD  7             ;ERROR TYPE CODE.
2120 010404      73$:
2121 010404 005100          COM     R0          ;COMPLEMENT CHECK WORD
2122 010406 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2123 010410 012201          MOV     (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2124 010412 020001          CMP     R0,       R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2125 010414 001405          BEQ     75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2126 010416 004767 010374      74$:   JSR     PC,        SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2127 010422 004767 011622      JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2128 010426 000007          .WORD  7             ;ERROR TYPE CODE.
2129 010430      75$:
2130 010430 005100          COM     R0          ;COMPLEMENT CHECK WORD
2131 010432 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2132 010434 012201          MOV     (R2)+,    R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2133 010436 020001          CMP     R0,       R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2134 010440 001405          BEQ     77$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2135 010442 004767 010350      76$:   JSR     PC,        SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2136 010446 004767 011576      JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2137 010452 000007          .WORD  7             ;ERROR TYPE CODE.
2138 010454      77$:
2139 010454 005303          DEC     R3          ;DECREMENT 4 WORD COUNTER
2140 010456 001342          BNE     23$         ;BR IF NOT DONE.
2141 010460 005100          COM     R0          ;COMPLEMENT CHECK WORD
2142 010462 005304          DEC     R4          ;DECREMENT 256. WORD COUNTER
2143 010464 001335          BNE     22$         ;BR IF NOT DONE.
2144 010466 005100          COM     R0          ;COMPLEMENT CHECK WORD
2145 010470 030502          BIT     R5          R2      ;CHECK FOR END OF A BLOCK.
2146 010472 001330          BNE     21$         ;BRANCH IF MORE IN CURRENT BLOCK.
2147 010474 004767 005220      JSR     PC,        MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
    
```

K10

```

2148 ;*****
2149 ;*TEST 16      MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2150 ;*****
2151 TST16:
2152 010500      004567  010554      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
2153 010504      000777          .WORD      777          ;MINIMUM BLOCK SIZE OF 256. WORDS
2154          ;REQUIRED FOR THIS TEST.
2155 010506      000167  000610      JMP      TST17          ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2156          ;AVAILABLE FOR TEST.
2157 010512      012700  000401      MOV      #401,     R0      ;SET UP PARITY "ALL ZEROS" PATTERN
2158 010516      012703  177777      MOV      #-1,     R3      ;SET COM DATA REG
2159 010522      004467  004414      JSR      R4,      INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2160 010526      004767  006240      1$:     JSR      PC,      W3X9   ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2161 010532      030502          BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
2162 010534      001374          BNE     1$,      ;BRANCH IF MORE IN CURRENT BLOCK.
2163 010536      004767  005156      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
2164
2165 ;*****
2166 ;* CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2167 ;*****
2168 010542      012700  000401      MOV      #401,     R0      ;RESET PARITY "ALL ZEROS" PATTERN.
2169 010546      012703  177777      MOV      #-1,     R3      ;RESET PARITY ALL ONES PATTERN.
2170 010552      004467  004364      JSR      R4,      INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2171 010556      012704  000100      11$:    MOV      #64.,    R4      ;SET 256. WORD COUNTER
2172 010562          12$:
2173 010562      012201          MOV      (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2174 010564      020001          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2175 010566      001405          BEQ     65$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2176 010570      004767  010222      64$:    JSR      PC,      SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2177 010574      004767  011450      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2178 010600          .WORD      7          ;ERROR TYPE CODE.
2179 010602          65$:
2180 010602      012201          MOV      (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2181 010604      020001          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2182 010606      001405          BEQ     67$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2183 010610      004767  010202      66$:    JSR      PC,      SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2184 010614      004767  011430      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2185 010620          .WORD      7          ;ERROR TYPE CODE.
2186 010622          67$:
2187 010622      012201          MOV      (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2188 010624      020001          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2189 010626      001405          BEQ     69$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2190 010630      004767  010162      68$:    JSR      PC,      SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2191 010634      004767  011410      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2192 010640          .WORD      7          ;ERROR TYPE CODE.
2193 010642          69$:
2194 010642      012201          MOV      (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2195 010644      020001          CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2196 010646      001405          BEQ     71$,      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2197 010650      004767  010142      70$:    JSR      PC,      SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2198 010654      004767  011370      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2199 010660          .WORD      7          ;ERROR TYPE CODE.
2200 010662          71$:
2201 010662      010046          MOV      R0,      -(SP)  ;SAVE R0
2202 010664      010300          MOV      R3,      R0      ;PUT R3 INTO R0
2203 010666      012603          MOV      (SP)+,   R3      ;PUT SAVED R0 INTO R3
    
```

L10

```

2204 010670 005304          DEC      R4          ;COUNT 256. WORDS
2205 010672 001333          BNE     12$         ;BRANCH IF MORE
2206 010674 010046          MOV     R0,        -(SP) ;SAVE R0
2207 010676 010300          MOV     R3,        R0    ;PUT R3 INTO R0
2208 010700 012603          MOV     (SP)+,    R3    ;PUT SAVED R0 INTO R3
2209 010702 030502          BIT     R5,        R2    ;CHECK FOR END OF A BLOCK.
2210 010704 001324          BNE     11$         ;BRANCH IF MORE IN CURRENT BLOCK.
2211 010706 004767 005006     JSR     PC,        MMUP  ;FIND NEXT BLOCK AND LOOP TO 11$.
2212
2213
2214
2215
2216 010712 012700 000401     MOV     #401,     R0    ;SET UP PARITY "ALL ZEROS" PATTERN.
2217 010716 012703 177777     MOV     #-1,     R3    ;SET UP ALL ONES PATTERN.
2218 010722 004467 004214     JSR     R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2219 010726 012704 000100     21$:   MOV     #64.,R4    ;SET 256. WORD COUNTER
2220 010732
2221 010732 012201           MOV     (R2)+,    R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2222 010734 020001           CMP     R0,        R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2223 010736 001405           BEQ     73$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2224 010740 004767 010052     72$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2225 010744 004767 011300     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2226 010750 000007           .WORD   7            ;ERROR TYPE CODE.
2227 010752
2228 010752 005100           COM     R0          ;COMPLEMENT CHECK WORD
2229 010754 005142           COM     -(R2)       ;COMPLEMENT TEST DATA
2230 010756 012201           MOV     (R2)+,    R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2231 010760 020001           CMP     R0,        R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2232 010762 001405           BEQ     75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2233 010764 004767 010026     74$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2234 010770 004767 011254     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2235 010774 000007           .WORD   7            ;ERROR TYPE CODE.
2236 010776
2237 010776 005100           COM     R0          ;COMPLEMENT CHECK WORD
2238 011000 005142           COM     -(R2)       ;RESTORE DATA
2239 011002 012201           MOV     (R2)+,    R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2240 011004 020001           CMP     R0,        R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2241 011006 001405           BEQ     77$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2242 011010 004767 010002     76$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2243 011014 004767 011230     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2244 011020 000007           .WORD   7            ;ERROR TYPE CODE.
2245 011022
2246 011022 012201           MOV     (R2)+,    R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2247 011024 020001           CMP     R0,        R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2248 011026 001405           BEQ     79$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2249 011030 004767 007762     78$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2250 011034 004767 011210     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2251 011040 000007           .WORD   7            ;ERROR TYPE CODE.
2252 011042
2253 011042 005100           COM     R0          ;COMPLEMENT CHECK WORD
2254 011044 005142           COM     -(R2)       ;COMPLEMENT TEST DATA
2255 011046 012201           MOV     (R2)+,    R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2256 011050 020001           CMP     R0,        R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2257 011052 001405           BEQ     81$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2258 011054 004767 007736     80$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2259 011060 004767 011164     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)

```


N10

```

2316 011256 001405          BEQ      95$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2317 011260 004767 007532 94$:  JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2318 011264 004767 010760      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2319 011270 000007          .WORD   7           ;ERROR TYPE CODE.
2320 011272
2321 011272 010046          MOV      R0,      -(SP)   ;SAVE R0
2322 011274 010300          MOV      R3,      R0     ;PUT R3 INTO R0
2323 011276 012603          MOV      (SP)+,   R3     ;PUT SAVED R0 INTO R3
2324 011300 005304          DEC      R4           ;DECREMENT 256. WORD COUNTER
2325 011302 001213          BNE     22$         ;BRANCH IF MORE.
2326 011304 010046          MOV      R0,      -(SP)   ;SAVE R0
2327 011306 010300          MOV      R3,      R0     ;PUT R3 INTO R0
2328 011310 012603          MOV      (SP)+,   R3     ;PUT SAVED R0 INTO R3
2329 011312 030502          BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
2330 011314 001204          BNE     21$         ;BRANCH IF MORE IN CURRENT BLOCK.
2331 011316 004767 004376      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 21$.
2332
2333      ;*****
2334      ;*TEST 17      COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2335      ;*****
2336 011322
2337 011322 004567 007732      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
2338 011326 000777          .WORD   777         ;MINIMUM BLOCK SIZE OF 256. WORDS
2339                                ;REQUIRED FOR THIS TEST.
2340 011330 000167 000610      JMP      TST20      ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2341                                ;AVAILABLE FOR TEST.
2342 011334 012700 177777          MOV      #-1,     R0     ;SET UP ALL ONES PATTERN
2343 011340 012703 000401          MOV      #401,    R3     ;SET UP PARITY "ALL ZEROS" PATTERN
2344 011344 004467 003572          JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2345 011350 004767 005416 1$:  JSR      PC,      W3X9   ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2346 011354 030502          BIT      R5,      R2     ;CHECK FOR END OF A BLOCK.
2347 011356 001374          BNE     1$         ;BRANCH IF MORE IN CURRENT BLOCK.
2348 011360 004767 004334      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
2349
2350      ;*****
2351      ;* CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2352      ;*****
2353 011364 012700 177777          MOV      #-1,     R0     ;SET UP ALL ONES PATTERN
2354 011370 012703 000401          MOV      #401,    R3     ;SET UP PARITY "ALL ZEROS" PATTERN
2355 011374 004467 003542          JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2356 011400 012704 000100 11$: MOV      #64.,    R4     ;SET 256. WORD COUNTER
2357 011404 12$:
2358 011404 012201          MOV      (R2)+,   R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2359 011406 020001          CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2360 011410 001405          BEQ     65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
2361 011412 004767 007400 64$: JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2362 011416 004767 010626      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2363 011422 000007          .WORD   7           ;ERROR TYPE CODE.
2364 011424
2365 011424 012201          MOV      (R2)+,   R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2366 011426 020001          CMP      R0,      R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2367 011430 001405          BEQ     67$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
2368 011432 004767 007360 66$: JSR      PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2369 011436 004767 010606      JSR      PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2370 011442 000007          .WORD   7           ;ERROR TYPE CODE.
2371 011444 67$:

```

B11

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER
DZQMCC.P11 02-DEC-76 08:47

T17 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

MACY11 27(1006) 02-DEC-76 09:00 PAGE 49

SEQ 0131

```

2372 011444 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2373 011446 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2374 011450 001405      BEQ     69$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2375 011452 004767 007340 68$:     JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2376 011456 004767 010566      JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
2377 011462 000007      .WORD  7              ;ERROR TYPE CODE.
2378 011464      69$:
2379 011464 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2380 011466 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2381 011470 001405      BEQ     71$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2382 011472 004767 007320 70$:     JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2383 011476 004767 010546      JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
2384 011502 000007      .WORD  7              ;ERROR TYPE CODE.
2385 011504      71$:
2386 011504 010046      MOV      RO, -(SP)     ;SAVE RO
2387 011506 010300      MOV      R3, RO        ;PUT R3 INTO RO
2388 011510 012603      MOV      (SP)+, R3     ;PUT SAVED RO INTO R3
2389 011512 005304      DEC     R4             ;COUNT 256. WORDS
2390 011514 001333      BNE     12$,           ;BRANCH IF MORE
2391 011516 010046      MOV      RO, -(SP)     ;SAVE RO
2392 011520 010300      MOV      R3, RO        ;PUT R3 INTO RO
2393 011522 012603      MOV      (SP)+, R3     ;PUT SAVED RO INTO R3
2394 011524 030502      BIT     R5, R2         ;CHECK FOR END OF A BLOCK.
2395 011526 001324      BNE     11$,           ;BRANCH IF MORE IN CURRENT BLOCK.
2396 011530 004767 004164      JSR     PC, MMUP       ;FIND NEXT BLOCK AND LOOP TO 11$.
2397
2398
2399
2400
2401 011534 012700 177777      ;*****
2402 011540 012703 000401      ;* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
2403 011544 004467 003372      ;*****
2404 011550 012704 000100      ;*****
2405 011554      21$:
2406 011554 012201      MOV      (R2)+, R1      ;SET UP ALL ONES PATTERN
2407 011556 020001      CMP      RO, R1        ;SET UP PARITY "ALL ZEROS" PATTERN
2408 011560 001405      BEQ     73$,           ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2409 011562 004767 007230 22$:     JSR     PC, INITMM     ;SET 256. WORD COUNTER
2410 011566 004767 010456      MOV     #64.,R4
2411 011572 000007      .WORD  7
2412 011574      72$:
2413 011574 005100      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2414 011576 005142      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2415 011600 012201      BEQ     73$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2416 011602 020001      JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2417 011604 001405      JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
2418 011606 004767 007204 73$:     .WORD  7              ;ERROR TYPE CODE.
2419 011612 004767 010432      COM     RO              ;COMPLEMENT CHECK WORD
2420 011616 000007      COM     -(R2)           ;COMPLEMENT TEST DATA
2421 011620      74$:
2422 011620 005100      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2423 011622 005142      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2424 011624 012201      BEQ     75$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2425 011626 020001      JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2426 011630 001405      JSR     PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
2427 011632 004767 007160      .WORD  7              ;ERROR TYPE CODE.
75$:
2428 011632 005100      COM     RO              ;COMPLEMENT CHECK WORD
2429 011634 005142      COM     -(R2)           ;RESTORE DATA
2430 011636 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2431 011638 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2432 011640 001405      BEQ     77$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2433 011642 004767 007160      JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.

```

C11

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 27(1006) 02-DEC-76 09:00 PAGE 50
 DZQMCC.P11 02-DEC-76 08:47 T17 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0132

2428	011636	004767	010406		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2429	011642	000007			.WORD	7		;ERROR TYPE CODE.
2430	011644			77\$:				
2431	011644	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2432	011646	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2433	011650	001405			BEQ	79\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2434	011652	004767	007140	78\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2435	011656	004767	010366		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2436	011662	000007			.WORD	7		;ERROR TYPE CODE.
2437	011664			79\$:				
2438	011664	005100			COM	RO		;COMPLEMENT CHECK WORD
2439	011666	005142			COM	-(R2)		;COMPLEMENT TEST DATA
2440	011670	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2441	011672	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2442	011674	001405			BEQ	81\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2443	011676	004767	007114	80\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2444	011702	004767	010342		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2445	011706	000007			.WORD	7		;ERROR TYPE CODE.
2446	011710			81\$:				
2447	011710	005100			COM	RO		;COMPLEMENT CHECK WORD
2448	011712	005142			COM	-(R2)		;RESTORE DATA
2449	011714	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2450	011716	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2451	011720	001405			BEQ	83\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2452	011722	004767	007070	82\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2453	011726	004767	010316		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2454	011732	000007			.WORD	7		;ERROR TYPE CODE.
2455	011734			83\$:				
2456	011734	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2457	011736	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2458	011740	001405			BEQ	85\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2459	011742	004767	007050	84\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2460	011746	004767	010276		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2461	011752	000007			.WORD	7		;ERROR TYPE CODE.
2462	011754			85\$:				
2463	011754	005100			COM	RO		;COMPLEMENT CHECK WORD
2464	011756	005142			COM	-(R2)		;COMPLEMENT TEST DATA
2465	011760	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2466	011762	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2467	011764	001405			BEQ	87\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2468	011766	004767	007024	86\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2469	011772	004767	010252		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2470	011776	000007			.WORD	7		;ERROR TYPE CODE.
2471	012000			87\$:				
2472	012000	005100			COM	RO		;COMPLEMENT CHECK WORD
2473	012002	005142			COM	-(R2)		;RESTORE DATA
2474	012004	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2475	012006	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2476	012010	001405			BEQ	89\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2477	012012	004767	007000	88\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2478	012016	004767	010226		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2479	012022	000007			.WORD	7		;ERROR TYPE CODE.
2480	012024			89\$:				
2481	012024	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2482	012026	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2483	012030	001405			BEQ	91\$;BRANCH OVER ERROR CALL IF GOOD DATA.

D11

2484	012032	004767	006760	90\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2485	012036	004767	010206		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2486	012042	000007			.WORD	7		;ERROR TYPE CODE.
2487	012044			91\$:				
2488	012044	005100			COM	RO		;COMPLEMENT CHECK WORD
2489	012046	005142			COM	-(R2)		;COMPLEMENT TEST DATA
2490	012050	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2491	012052	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2492	012054	001405			BEQ	93\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2493	012056	004767	006734	92\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2494	012062	004767	010162		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2495	012066	000007			.WORD	7		;ERROR TYPE CODE.
2496	012070			93\$:				
2497	012070	005100			COM	RO		;COMPLEMENT CHECK WORD
2498	012072	005142			COM	-(R2)		;RESTORE DATA
2499	012074	012201			MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2500	012076	020001			CMP	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2501	012100	001405			BEQ	95\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2502	012102	004767	006710	94\$:	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2503	012106	004767	010136		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2504	012112	000007			.WORD	7		;ERROR TYPE CODE.
2505	012114			95\$:				
2506	012114	010046			MOV	RO,	-(SP)	;SAVE RO
2507	012116	010300			MOV	R3,	RO	;PUT R3 INTO RO
2508	012120	012603			MOV	(SP)+,	R3	;PUT SAVED RO INTO R3
2509	012122	005304			DEC	R4		;DECREMENT 256. WORD COUNTER
2510	012124	001213			BNE	22\$;BRANCH IF MORE.
2511	012126	010046			MOV	RO,	-(SP)	;SAVE RO
2512	012130	010300			MOV	R3,	RO	;PUT R3 INTO RO
2513	012132	012603			MOV	(SP)+,	R3	;PUT SAVED RO INTO R3
2514	012134	030502			BIT	R5,	R2	;CHECK FOR END OF A BLOCK.
2515	012136	001204			BNE	21\$;BRANCH IF MORE IN CURRENT BLOCK.
2516	012140	004767	003554		JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO 21\$.

E11

```

2517
2518
2519
2520 012144
2521 012144 004567 007110
2522 012150 017777
2523
2524 012152 000167 000320
2525
2526 012156 005000
2527 012160 004467 002756
2528 012164 012704 000040
2529 012170 005100
2530 012172 012703 000200
2531 012176 005100
2532 012200 010022
2533 012202 005303
2534 012204 001375
2535 012206 005304
2536 012210 001370
2537 012212 030502
2538 012214 001363
2539 012216 004767 003476
2540
2541
2542
2543
2544 012222 005000
2545 012224 004467 002712
2546 012230 012704 000040
2547 012234 005100
2548 012236 012703 000100
2549 012242 005100
2550 012244
2551 012244 012201
2552 012246 020001
2553 012250 001405
2554 012252 004767 006540
2555 012256 004767 007766
2556 012262 000010
2557 012264
2558 012264 012201
2559 012266 020001
2560 012270 001405
2561 012272 004767 006520
2562 012276 004767 007746
2563 012302 000010
2564 012304
2565 012304 005303
2566 012306 001356
2567 012310 005304
2568 012312 001351
2569 012314 030502
2570 012316 001344
2571 012320 004767 003374
2572

```

```

*****
*TEST 20      8 XOR 13 TEST PATTERN
*****
TST20:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD    17777          ;MINIMUM BLOCK SIZE OF 4096. WORDS
                               ;REQUIRED FOR THIS TEST.
      JMP      TST21        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                               ;AVAILABLE FOR TEST.

      .8X13:  CLR      R0
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    MOV      #32.,   R4      ;SET UP 4K LOOP COUNTER
      COM      R0
2$:    MOV      #128.,  R3      ;EACH SMALL LOOP WRITES 128 WORDS
      COM      R0
3$:    MOV      R0,     (R2)+   ;WRITE INTO MEMORY ADDRESSES
      DEC      R3          ;DECREMENT WORD COUNT
      BNE     R3,      3$
      DEC      R4          ;DECREMENT 128. WORD COUNT
      BNE     R4,      2$
      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
      BNE     R5,     1$     ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,     MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
* CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
*****
      CLR      R0          ;SET UP CHECK WORD.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$:   MOV      #32.,   R4      ;SET 4K WORD COUNTER
      COM      R0          ;COMPLEMENT TEST WORD
12$:   MOV      #64.,   R3      ;SET 128 WORD COUNTER
      COM      R0          ;COMPLEMENT TEST WORD
13$:   MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     R0,     65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC,     SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10          ;ERROR TYPE CODE.
65$:   MOV      (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     R0,     67$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:   JSR      PC,     SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    10          ;ERROR TYPE CODE.
67$:   DEC      R3          ;DECREMENT 128 WORD COUNTER
      BNE     R3,     13$     ;BR IF MORE.
      DEC      R4          ;DECREMENT 4096. WORD COUNTER
      BNE     R4,     12$     ;BR IF MORE.
      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
      BNE     R5,     1$     ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,     MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.

```

F11

```

2573      ;*****
2574      ;*  COMPLEMENT 8 XOR 13 TEST PATTERN.
2575      ;*****
2576 012324 016746 167256      MOV      BLKMSK, -(SP)      ;SAVE BLOCK MASK TEMPORARILY.
2577 012330 012767 017777 167250  MOV      *MASK4K, BLKMSK      ;SET BLOCK MASK TO 4K.
2578 012336 004467 002600      JSR      R4, INITMM      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2579 012342 012704 004000 21$:  MOV      #2048., R4      ;SET 4096. WORD COUNTER
2580 012346 005122 22$:  COM      (R2)+      ;COMPLEMENT TEST PATTERN
2581 012350 005122      COM      (R2)+
2582 012352 005304      DEC      R4      ;COUNT 4K WORDS
2583 012354 001374      BNE     22$      ;BR IF MORE.
2584 012356 030502      BIT     R5, R2      ;CHECK FOR END OF A BLOCK.
2585 012360 001370      BNE     21$      ;BRANCH IF MORE IN CURRENT BLOCK.
2586 012362 004767 003332      JSR     PC, MMUP      ;FIND NEXT BLOCK AND LOOP TO 21$.
2587 012366 012667 167214      MOV     (SP)+, BLKMSK ;RESTORE BLOCK MASK.
2588
2589      ;*****
2590      ;*  CHECK COMPLEMENTED 8 XOR 13 TEST PATTERN WRITTEN ABOVE.
2591      ;*****
2592 012372 012700 177777      MOV     #-1., R0      ;SET UP CHECK WORD.
2593 012376 004467 002540      JSR     R4, INITMM      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2594 012402 012704 000040 31$:  MOV     #32., R4      ;SET 4K WORD COUNTER
2595 012406 005100      COM     R0      ;COMPLEMENT TEST WORD
2596 012410 012703 000100 32$:  MOV     #64., R3      ;SET 128 WORD COUNTER
2597 012414 005100      COM     R0      ;COMPLEMENT TEST WORD
2598 012416 33$:
2599 012416 012201      MOV     (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2600 012420 020001      CMP     R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2601 012422 001405      BEQ     69$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2602 012424 004767 006366 68$:  JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2603 012430 004767 007614      JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2604 012434 000010      .WORD  10      ;ERROR TYPE CODE.
2605 012436 69$:
2606 012436 012201      MOV     (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2607 012440 020001      CMP     R0, R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2608 012442 001405      BEQ     71$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2609 012444 004767 006346 70$:  JSR     PC, SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
2610 012450 004767 007574      JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
2611 012454 000010      .WORD  10      ;ERROR TYPE CODE.
2612 012456 71$:
2613 012456 005303      DEC     R3      ;DECREMENT 128 WORD COUNTER
2614 012460 001356      BNE     33$      ;BR IF MORE.
2615 012462 005304      DEC     R4      ;DECREMENT 4096. WORD COUNTER
2616 012464 001351      BNE     32$      ;BR IF MORE.
2617 012466 030502      BIT     R5, R2      ;CHECK FOR END OF A BLOCK.
2618 012470 001344      BNE     31$      ;BRANCH IF MORE IN CURRENT BLOCK.
2619 012472 004767 003222      JSR     PC, MMUP      ;FIND NEXT BLOCK AND LOOP TO 31$.
  
```

```

2620 .....
2621 *TEST 21      WORSE CASE NOISE PARITY BYTE TESTING
2622 * CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
2623 * 1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
2624 * 2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
2625 * 3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
2626 * 4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
2627 .....
2628 †ST21:
2629 012476 004567 006556      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
2630 012502 000000      .WORD    0           ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2631 012504 005767 167604      WWPB0:  TST      MPRX   ;CHECK FOR ANY PARITY MEMORY.
2632 012510 001404      BEQ      1$        ;BR IF NO PARITY MEMORY.
2633 012512 032777 000100 166420  BIT      #SW06,  @SWR   ;CHECK FORINHIBIT PARITY SWITCH.
2634 012520 001402      BEQ      2$        ;BR IF NOT SET.
2635 012522 000167 000614      1$:     JMP      TST22    ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2636 012526 005000      2$:     CLR      R0           ;ZERO TO BE PUT IN ALL MEMORY.
2637 012530 004767 004150      JSR      PC,      SETCON ;ROUTINE TO LOAD ALL MEMORY.
2638 012534 004467 002402      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2639 012540 036767 167000 166772  WWPBYT: BIT      BITPT,  PMEMAP ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
2640 012546 001010      BNE     2$        ;BR IF PARITY MEM.
2641 012550 036767 166772 166764  BIT      BITPT+2, PMEMAP+2 ;...HI 64K.
2642 012556 001004      BNE     2$        ;BR IF PARITY MEM.
2643 012560 050502      BIS      R5,      R2     ;POINT TO END OF BLOCK.
2644 012562 005202      INC      R2           ;FIRST ADR OF NEXT BLOCK.
2645 012564 000167 000532      JMP      WWPB5      ;BR TO FIND NEXT BLOCK.
2646 012570 004767 005436      2$:     JSR      PC,      SETAE ;SET ACTION ENABLE (EVEN IF BANK0.)
2647 012574 004767 005466      JSR      PC,      CKPMER ;CHECK FOR ANY NON TRAP PARITY ERRORS.
2648 012600 020227 000114      WWPB1:  CMP      R2,      #114 ;CKECK IF POINTING TO PARITY ERROR VECTOR.
2649 012604 001004      BNE     3$        ;BR IF NOT AT VECTOR.
2650 012606 062702 000004      ADD      #4,      R2     ;SKIP PARITY VECTOR.
2651 012612 000167 000504      JMP      WWPB5      ;CHECK FOR BLOCK END.
2652 012616 111201      3$:     MOVVB  (R2),    R1     ;CHECK IF BYTE STILL CLEARED.
2653 012620 001405      BEQ      65$       ;BRANCH OVER ERROR CALL IF GOOD DATA.
2654 012622 004767 006114      64$:    JSR      PC,      SPRT  ;SET UP VALUES FOR ERROR PRINTING.
2655 012626 004767 007416      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2656 012632 000011      .WORD    11         ;ERROR TYPE CODE.
2657 65$:
2658 012634 105067 166720      CLRB    OEFLG      ;CLEAR ODD/EVEN FLAG.
2659 012640 112700 000252      MOVVB  #252,    R0     ;SET UP DATA...EVEN, SETS PARITY BIT.
2660 012644 110012      WWPB2:  MOVVB  R0,      (R2) ;MOV DATA INTO TEST LOCATION.
2661 012646 016703 166746      MOV     .MPRX,    R3     ;GET PARITY REGISTER TABLE POINTER.
2662 012652 052733 000005      10$:   BIS      #WWP+AE, @ (R3)+ ;SET WRITE WRONG PARITY.
2663 012656 005713      TST     (R3)      ;CHECK FOR TABLE TERMINATOR.
2664 012660 001374      BNE     10$       ;BR IF MORE REGS IN TABLE.
2665 ;* SET WRONG PARITY IN LOCATION UNDER TEST.
2666 012662 110012      MOVVB  R0,      (R2) ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
2667 012664 016703 166730      MOV     .MPRX,    R3     ;GET PARITY REG TABLE POINTER.
2668 012670 042733 000004      11$:   BIC      #WWP,    @ (R3)+ ;CLEAR WRITE WRONG PARITY.
2669 012674 005713      TST     (R3)      ;CHECK FOR TABLE TERMINATOR.
2670 012676 001374      BNE     11$       ;BR IF MORE PARITY REGISTERS.
2671 012700 016737 166724 000114  MOV     .PBTRP,  @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
2672 ;* DETECT WRONG PARITY VIA DATIP: DATOB SHOULDN'T EXECUTE.
2673 012706 105412      NEGB   (R2)      ;DATIP (DATOB AND COM PARITY BIT.)
2674 ;* SHOULD HAVE TRAPPED TO PBTRP.
2675 012710 016737 166720 000114  MOV     .PESRV,  @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.

```

H11

MAINDEC-11-DZQMC-C-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 27(1006) 02-DEC-76 09:00 PAGE 55
DZQMC.P11 02-DEC-76 08:47 T21 WORSE CASE NOISE PARITY BYTE TESTING

SEQ 0137

```
2676 012716 004767 006050      64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2677 012722 004767 007322      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2678 012726 000012      .WORD 12 ;ERROR TYPE CODE.
2679 012730 000562      BR WWPB4 ;SKIP TRAP SERVICE.
2680
2681 ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
2682 012732 016737 166676 000114 PBTRP: MOV .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2683 012740 022626      CMP (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
2684 012742 016703 166650      MOV .MPRO, R3 ;GET PARITY REG AND MAP TABLE POINTER.
2685 012746 032713 000001      21$: BIT #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
2686 012752 001003      BNE 22$ ;BR IF IT DOESN'T EXIST.
2687 012754 017301 000000      MOV @R3, R1 ;GET THE CONTENTS.
2688 012760 100413      BMI 23$ ;BR IF ERROR FLAG SET.
2689 012762 062703 000010      22$: ADD #10, R3 ;MOVE POINTER TO NEXT REG.
2690 012766 020367 166626      CMP R3, .MPRX ;CHECK FOR END OF TABLE.
2691 012772 103765      BLC 21$ ;BR IF MORE REGISTERS.
2692 012774 004767 005772      64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2693 013000 004767 007244      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2694 013004 000013      .WORD 13 ;ERROR TYPE CODE.
2695 013006 000533      BR WWPB4 ;EXIT AFTER ERROR.
2696 013010 036763 166530 000002      23$: BIT BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
2697 013016 001011      BNE 24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2698 013020 036763 166522 000004      BIT BITPT+2, 4(R3) ;CHECK THE HI 64K.
2699 013026 001005      BNE 24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2700 013030 004767 005732      65$: JSR PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2701 013034 004767 007210      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2702 013040 000014      .WORD 14 ;ERROR TYPE CODE.
2703
2704 013042      24$:
2705 013042 010046      MOV RO, -(SP) ;: PUSH RO ON STACK
2706 013044 010200      MOV R2, RO ;GET THE ADDRESS POINTER.
2707 013046 042700 003777      BIC #3777, RO ;CLEAR LOW ADDRESS BITS.
2708 013052 000300      SWAB RO ;SHIFT 6 PLACES RIGHT.
2709 013054 006300      ASL RO
2710 013056 006300      ASL RO
2711 013060 005767 165522      TST MMAVA ;CHECK FOR MEM MGMT.
2712 013064 001404      BEQ 25$ ;BR IF NO MEM MGMT.
2713 013066 042700 177600      BIC #177600, RO ;CLEAR BANK BITS
2714 013072 063700 172344      ADD @#KIPAR2, RO ;ADD MEM MGMT OFFSET.
2715 013076 052700 100001      25$: BIS #BIT15+BIT0, RO ;SET ERROR AND AE BIT IN CHECK WORD.
2716 013102 016367 000006 166406      MOV 6(R3), RESRVD ;GET APPROPRIATE MASK.
2717 013110 046700 166402      BIC RESRVD, RO ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2718 013114 046701 166376      BIC RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2719 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2720 013120 020001      CMP RO, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2721 013122 001405      BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2722 013124 004767 005636      66$: JSR PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2723 013130 004767 007114      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2724 013134 000015      .WORD 15 ;ERROR TYPE CODE.
2725 013136 005073 000000      67$: CLR @R3 ;CLEAR REG INCLUDING ACTION ENABLE.
2726 013142 010346      MOV R3, -(SP) ;: PUSH R3 ON STACK
2727 013144 062703 000010      26$: ADD #10, R3 ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2728 013150 020367 166444      CMP R3, .MPRX ;CHECK FOR END OF TABLE.
2729 013154 101014      BHI WWPB3 ;BR IF END OF TABLE REACHED.
2730 013156 032713 000001      BIT #BIT0, (R3) ;CHECK IF NEXT REG EXISTS.
2731 013162 001370      BNE 26$ ;BR IF THIS PARITY REG DOESN'T EXIST.
```

2732	013164	017301	000000		MOV	2(R3), R1		;SAVE AND CHECK FOR ERROR FLAG.
2733	013170	100365			BPL	26\$;BR IF NO ERROR FLAG.
2734	013172	004767	005570	68\$:	JSR	PC,	SPRNTF	;SET UP VALUES FOR ERROR PRINTING.
2735	013176	004767	007046		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2736	013202	000016			.WORD	16		;ERROR TYPE CODE.
2737	013204	000757			BR	26\$;BR AFTER ERROR.
2738	013206	111204		WWPB3:	MOVB	(R2), R4		;GET THE DATA FOR CHECKING.
2739					;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT			
2740					;* ACTION ENABLE IS NOT SET IN CONTROLING REG, SO NO TRAP SHOULD OCCURE.			
2741	013210	111212			MOVB	(R2), (R2)		;RESTORE RIGHT PARITY
2742				:NOTE:	THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS			
2743				:	WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.			
2744	013212	012603			MOV	(SP)+,R3		;POP STACK INTO R3
2745	013214	017301	000000		MOV	2(R3), R1		;READ THE PARITY REGISTER TO CHECK IT AGAIN.
2746	013220	046701	166272		BIC	RESRVD, R1		;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2747				:NOTE:	THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.			
2748	013224	042700	000001		BIC	#AE, R0		;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
2749	013230	020001			CMP	R0, R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2750	013232	001405			BEQ	65\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2751	013234	004767	005526	64\$:	JSR	PC,	SPRNTF	;SET UP VALUES FOR ERROR PRINTING.
2752	013240	004767	007004		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2753	013244	000015			.WORD	15		;ERROR TYPE CODE.
2754	013246			65\$:				
2755	013246	012773	000001	000000	MOV	#1, 2(R3)		;CLEAR ALL BUT ACTION ENABLE.
2756	013254	010401			MOV	R4, R1		;GET DATA READ FROM MEMORY FOR TESTING.
2757	013256	012600			MOV	(SP)+,R0		;POP STACK INTO R0
2758	013260	120001			CMPB	R0, R1		;CHECK THE DATA.
2759	013262	001405			BEQ	67\$;BRANCH OVER ERROR CALL IF GOOD DATA.
2760	013264	004767	005502	66\$:	JSR	PC,	SPRNTD	;SET UP VALUES FOR ERROR PRINTING.
2761	013270	004767	006754		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2762	013274	000017			.WORD	17		;ERROR TYPE CODE.
2763	013276			67\$:				
2764	013276	110012		WWPB4:	MOVB	R0, (R2)		;RESTORE DATA.
2765	013300	105712			TSTB	(R2)		;DO A DATI TO BE SURE RIGHT PARITY.
2766	013302	012700	000253		MOV	#253, R0		;SET ODD PARITY DATA.
2767	013306	105167	166246		COMB	OEFLG		;CHECK IF DONE BOTH ODD AND EVEN PARITY.
2768	013312	100002			BPL	27\$;BR IF DONE BOTH EVEN AND ODD.
2769	013314	000167	177324		JMP	WWPB2		;LOOP BACK AND DO ODD(PARITY BIT CLR).
2770	013320	005202		27\$:	INC	R2		;MOVE POINTER TO NEXT MEMORY BYTE.
2771	013322	030502		WWPB5:	BIT	R5, R2		;CHECK FOR END OF BLOCK.
2772	013324	001402			BEQ	30\$;BR IF END OF BLOCK FOUND.
2773	013326	000167	177246		JMP	WWPB1		;LOOP BACK TO TEST NEXT BYTE.
2774	013332	004767	002362	30\$:	JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO WWPB5T
2775	013336	004767	004624		JSR	PC,	MAMF	;GO RESET PARITY REGISTERS.

```

2776
2777
2778
2779 013342
2780 013342 004567 005712
2781 013346 000000
2782 013350 010703
2783 013352 042703 007777
2784 013356 004467 001560
2785 013362 010246
2786 013364 010346
2787 013366 012322
2788 013370 032703 007777
2789 013374 001002
2790 013376 162703 010000
2791 013402 030502
2792 013404 001370
2793 013406 012603
2794 013410 012602
2795 013412 012300
2796 013414 012201
2797 013416 020001
2798 013420 001405
2799 013422 004767 005370
2800 013426 004767 006616
2801 013432 000020
2802 013434
2803 013434 032703 007777
2804 013440 001002
2805 013442 162703 010000
2806 013446
2807 013446 030502
2808 013450 001360
2809 013452 004767 002242

```

```

*****
;TEST 22 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
*****
†ST22:
      JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
RANTST: MOV    PC,    R3 ;GET CURRENT PROGRAM COUNTER.
      BIC    #7777, R3 ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
      JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV    R2,    -(SP) ;SAVE MEMORY POINTER.
      MOV    R3,    -(SP) ;SAVE "DATA" POINTER.
2$:   MOV    (R3)+, (R2)+ ;MOV CODE INTO TEST MEMORY.
      BIT    #7777, R3 ;CHECK FOR END OF "DATA TABLE"
      BNE    3$ ;BRANCH IF MORE
      SUB    #10000, R3 ;RESET POINTER TO START OF "RANDOM DATA"
3$:   BIT    R5,    R2 ;CHECK FOR END OF BLOCK
      BNE    2$ ;BRANCH IF MORE.
      MOV    (SP)+, R3 ;RESET "DATA" POINTER.
      MOV    (SP)+, R2 ;RESET MEMORY POINTER.
4$:   MOV    (R3)+, R0 ;GET S/B DATA.
      MOV    (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP    R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR    PC,    SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 20 ;ERROR TYPE CODE.
65$:  BIT    #7777, R3 ;CHECK FOR END OF "DATA TABLE"
      BNE    5$ ;BR IF MORE.
      SUB    #10000, R3 ;RESET POINTER TO TOP OF "DATA TABLE".
5$:   BIT    R5,    R2 ;CHECK FOR END OF A BLOCK.
      BNE    4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,    MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836 013456
2837 013456 004567 005576
2838 013462 000003
2839
2840 013464 000167 000056
2841
2842 013470 012703 010412
2843 013474 012704 000205
2844 013500 010400
2845 013502 004467 001434
2846 013506 010322
2847 013510 010412
2848 013512 004542
2849 013514 012201
2850 013516 020001
2851 013520 001405
2852 013522 004767 005264
2853 013526 004767 005516
2854 013532 000021
2855 013534
2856 013534 010322
2857 013536 030502
2858 013540 001363
2859 013542 004767 002152

```

.SBTTL SECTION 3: INSTRUCTION EXECUTION TESTS.
;*****
;TEST 23 EXECUTE DATI, DATO THRU MEMORY.
; EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
; AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
; CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
; THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
;
;          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
;          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
;
; 1ST PASS / 40000          010412          000205
; THRU TEST / 40002          000205          000205
;
; 2ND PASS / 40002          010412          000205
; THRU TEST / 40004          000205          000205
;
;          ETC., ETC., ETC.
;
; RD = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
; R1 = DATA READ FROM MEMORY (WAS).
; R2 = ADDRESS OF IUT/DATA.
; R3 = INSTRUCTION UNDER TEST (IUT).
; R4 = RTS R5 (CODE 205).
; R5 = BLOCK BOUNDARY BIT MASK.
;*****
TST23:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
;          ;REQUIRED FOR THIS TEST.
      JMP      TST24     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
;          ;AVAILABLE FOR TEST.
DIDO: MOV      #010412,R3 ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
      MOV      #205,   R4 ;GET 'RTS R5'
      MOV      R4,     R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,     INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV      R3,     (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV      R4,     (R2) ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,     -(R2) ;GO EXECUTE THE IUT.
      MOV      (R2)+,  R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,     R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,     SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21        ;ERROR TYPE CODE.
65$:  MOV      R3,     (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,     R2 ;CHECK FOR END OF A BLOCK.
      BNE     2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,     MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903
2904
2905
2906
2907
2908

013546
013546 004567 005506
013552 000003
013554 000167 000060
013560 012703 110412
013564 012704 000205
013570 012700 110605
013574 004467 001342
013600 010322
013602 010412
013604 004542
013606 012201
013610 020001
013612 001405
013614 004767 005172
013620 004767 006424
013624 000021
013626
013626 010322
013630 030502
013632 001363
013634 004767 002060

```
*****
*TEST 24 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
*          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000        110412          110605
* THRU TEST / 40002        000205          000205
*
* 2ND PASS / 40002        110412          110605
* THRU TEST / 40004        000205          000205
*
*          ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE- IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†ST24:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
                        ;REQUIRED FOR THIS TEST.
      JMP      TST25     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.
DIDBL: MOV      #110412,R3 ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
      MOV      #205,R4    ;GET 'RTS R5'
      MOV      #110605,R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV      R4,      (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2) ;GO EXECUTE THE IUT.
      MOV      (R2)+,   R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$       ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:  MOV      R3,      (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2    ;CHECK FOR END OF A BLOCK.
      BNE      2$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.

```


M11

```

2909 *****
2910 *TEST 25 EXECUTE DATI, DATOR (HIGH BYTE) THRU MEMORY.
2911 * EXECUTES THE INSTRUCTION 'MOVB R3, -(R2)' THROUGHOUT MEMORY.
2912 * AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
2913 * CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2914 * THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2915 *
2916 *          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
2917 *          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
2918 *
2919 * 1ST PASS / 40000          110342          161342
2920 * THRU TEST / 40002          000205          000205
2921 *
2922 * 2ND PASS / 40002          110342          161342
2923 * THRU TEST / 40004          000205          000205
2924 *
2925 *          ETC., ETC., ETC.
2926 *
2927 * R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2928 * R1 = DATA READ FROM MEMORY (WAS).
2929 * R2 = ADDRESS OF IUT/DATA.
2930 * R3 = INSTRUCTION UNDER TEST (IUT).
2931 * R4 = RTS R5 (CODE 205).
2932 * R5 = BLOCK BOUNDARY BIT MASK.
2933 *****
2934 TST25:
2935 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2936 .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
2937 ;REQUIRED FOR THIS TEST.
2938 JMP TST26 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2939 ;AVAILABLE FOR TEST.
2940 DIDBH: MOV #110342, R3 ;GET 'MOVB R3, -(R2)' INSTRUCTION (IUT).
2941 MOV #205, R4 ;GET 'RTS R5'
2942 MOV #161342, R0 ;SET UP S/B DATA AFTER EXECUTION.
2943 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2944 1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2945 2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
2946 JSR R5, -2(R2) ;GO EXECUTE THE IUT.
2947 DEC R2 ;ADJUST R2 TO POINT TO MAUT.
2948 MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2949 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2950 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2951 64$: JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
2952 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2953 .WORD 21 ;ERROR TYPE CODE.
2954 65$:
2955 MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
2956 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2957 BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
2958 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972
2973
2974
2975
2976
2977
2978
2979
2980
2981
2982
2983
2984 013736
2985 013736 004567 005316
2986 013742 000003
2987
2988 013744 000167 000060
2989
2990 013750 012703 005412
2991 013754 012704 000205
2992 013760 012700 172366
2993 013764 004467 001152
2994 013770 010322
2995 013772 010412
2996 013774 004542
2997 013776 012201
2998 014000 020001
2999 014002 001405
3000 014004 004767 005002
3001 014010 004767 006234
3002 014014 000021
3003 014016
3004 014016 010322
3005 014020 030502
3006 014022 001363
3007 014024 004767 001670

```
*****
*TEST 26 EXECUTE DATI, DATIP, DATO THRU MEMORY.
* EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
*          LOCATION              PLACED THERE      AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000              005412          172366
* THRU TEST / 40002            000205          000205
*
* 2ND PASS / 40002              005412          172366
* THRU TEST / 40004            000205          000205
*
*          ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†TST26:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST27     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
DIP00: MOV      #005412,R3 ;GET 'NEG (R2)' INSTRUCTION (IUT).
      MOV      #205, R4   ;GET 'RTS R5'
      MOV      #172366,R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV      R4,      (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2) ;GO EXECUTE THE IUT.
      MOV      (R2)+, R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:  MOV      R3,      (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2   ;CHECK FOR END OF A BLOCK.
      BNE      2$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.

```

3008
3009
3010
3011
3012
3013
3014
3015
3016
3017
3018
3019
3020
3021
3022
3023
3024
3025
3026
3027
3028
3029
3030
3031
3032
3033
3034
3035
3036
3037
3038
3039
3040
3041
3042
3043
3044
3045
3046
3047
3048
3049
3050
3051
3052
3053
3054
3055
3056

014030
014030 004567 005224
014034 000003
014036 000167 000060
014042 012703 142242
014046 012704 000205
014052 012700 142000
014056 004467 001060
014062 010322
014064 010412
014066 004542
014070 012201
014072 020001
014074 001405
014076 004767 004710
014102 004767 006142
014106 000021
014110
014110 010322
014112 030502
014114 001363
014116 004767 001576

```
*****
*TEST 27 EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION BICB (R2)+, -(R2) THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE BICB INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 142242 142000
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 142242 142000
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
```

```
TST27: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST30 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DPDBL: MOV #142242, R3 ;GET BICB (R2)+, -(R2) INSTRUCTION (IUT).
        MOV #205, R4 ;GET 'RTS R5'
        MOV #142000, R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) ;GO EXECUTE THE IUT.
        MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT R5, R2 ;CHECK FOR END OF A BLOCK.
        BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

3057
3058
3059
3060
3061
3062
3063
3064
3065
3066
3067
3068
3069
3070
3071
3072
3073
3074
3075
3076
3077
3078
3079
3080
3081
3082
3083
3084
3085
3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104
3105
3106

014122
014122 004567 005132
014126 000003
014130 000167 000062
014134 012703 152212
014140 012704 000205
014144 012700 157212
014150 004467 000766
014154 010322
014156 010412
014160 004542
014162 005302
014164 012201
014166 020001
014170 001405
014172 004767 004614
014176 004767 006046
014202 000021
014204
014204 010322
014206 030502
014210 001362
014212 004767 001502

```
*****
*TEST 30 EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 152212 157212
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 152212 157212
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST30: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST31 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DPDBH: MOV #152212,R3 ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
        MOV #205,R4 ;GET 'RTS R5'
        MOV #157212,R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) ;GO EXECUTE THE IUT.
        DEC R2 ;RESET R2 TO POINT TO IUT.
        MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT R5, R2 ;CHECK FOR END OF A BLOCK.
        BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```

3107
3108
3109
3110
3111
3112
3113
3114
3115
3116
3117
3118 014215
3119 014216 004567 005036
3120 014222 000077
3121
3122 014224 000167 000330
3123
3124 014230
3125 014230 004467 000706
3126 014234 012746 000024
3127 014240 010246
3128 014242 016700 165354
3129 014246 012701 000025
3130 014252 012022
3131 014254 030502
3132 014256 001404
3133 014260 0053C'
3134 014262 001373
3135 014264 005742
3136 014266 000765
3137 014270 005301
3138 014272 001407
3139 014274 162701 000025
3140 014300 005042
3141 014302 005201
3142 014304 001375
3143 014306 016722 000064
3144 014312 011602
3145 014314 016700 165304
3146 014320 016701 165302
3147 014324 000112
3148
3149
3150
3151 014326 010703
3152 014330 062703 000024
3153 014334 010304
3154 014336 005204
3155 014340 005013
3156 014342 000261
3157 014344 105513
3158 014346 100403
3159 014350 105214
3160 014352 000773
3161 014354 000
3162 014355 000

```

```

*****
*TEST 31 "BRANCH GOBBLE" TEST
* THIS TEST LOADS THE ROUTINE FOUND AT "BRGOB" BELOW INTO THE BEGINNING
* OF MEMORY UNDER TEST. IF THERE IS ROOM, IT LOADS THE ROUTINE AGAIN
* STARTING AT THE LAST INSTRUCTION OF THE FIRST COPY, THUS OVERLAYING
* THE EXIT INSTRUCTION (JMP (R1)) OF THE FIRST COPY. THIS CONTINUES
* UNTIL THE CURRENT BANK(S) ARE FULL. THE CODE IS THEN EXECUTED.
* THIS ENTIRE PROCESS IS THEN REPEATED STARTING AT THE SECOND ADDRESS
* OF THE BANK UNDER TEST, THEN THE THIRD, ETC., UNTIL EACH INSTRUCTION
* OF THE ROUTINE HAS BEEN LOADED INTO AND EXECUTED OUT OF EVERY LOCATION.
*****
TST31: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 77 ;MINIMUM BLOCK SIZE OF 32. WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST32 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.

BRGOBL: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
BGTOP: MOV #BGLEN-1, -(SP) ;SAVE ROUTINE LENGTH
BGL0OP: MOV R2, -(SP) ;SAVE 1ST BLOCK ADDRESS.
5$: MOV .BRGOB, R0 ;GET FIRST LOC OF "BRANCH GOBBLE" ROUTINE.
      MOV #BGLEN, R1 ;SET UP LENGTH COUNTER
6$: MOV (R0)+, (R2)+ ;MOV ROUTINE INTO MEMORY
      BIT R5, R2 ;CHECK FOR END OF BLOCK.
      BEQ 7$ ;BRANCH WHEN DONE.
      DEC R1 ;COUNT WORDS IN ROUTINE.
      BNE 6$ ;BR IF MORE.
      TST -(R2) ;BACK UP ONE TO OVERLAY RETURN.
7$: BR 5$ ;BR TO START ANOTHER ROUTINE.
      DEC R1 ;CHECK FOR EXACT FINISH.
      BEQ 9$ ;BR IF ALL OK.
8$: SUB #BGLEN, R1 ;SET COUNTER FOR BACKING UP.
      CLR -(R2) ;PUT HALTS IN EXTRA MEMORY.
9$: INC R1 ;COUNT NUMBER OF WORDS TO BACK JP..
      BNE 8$ ;BR IF MORE.
      MOV BGEND, (R2)+ ;INSERT RETURN INSTRUCTION.
      MOV (SP), R2 ;RESET POINTER TO BEGINNING OF BLOCK.
      MOV .BGERR, R0 ;POINT TO ERROR ROUTINE.
      MOV .BGEXI, R1 ;POINT TO EXIT ROUTINE.
      JMP (R2) ;GO TO BRANCH GOBBLE ROUTINES IN MEMORY.

;* "BRANCH GOBBLE" ROUTINE WHICH IS RELOCATED INTO MEMORY UNDER TEST
;* AND THEN EXECUTED THERE.
BRGOB: MOV PC, R3 ;GET CURRENT ROUTINE LOCATION.
BG1: ADD #BG3-BG1, R3 ;POINT TO DATA WORD "BG3"
      MOV R3, R4 ;COPY
      INC R4 ;POINT TO DATA BYTE "BG4"
      CLR (R3) ;CLR DATA WORD "BG3" & "BG4"
BG2: SEC ;SET CARRY... TO BE ADDED TO "BG3"
      ADCB (R3) ;ADD THE CARRY... AS IF INC INST
      BMI BG5 ;BRANCH WHEN BIT 7 IS SET
      INCB (R4)
      BR BG2 ;BR BACK AND CONTINUE TO COUNT.
BG3: .BYTE 0 ;DATA BYTE POINTED TO BY R3
BG4: .BYTE 0 ;DATA BYTE POINTED TO BY R4

```

```

3163 014356 102401      BG5:  BVS      BG6      ;BRANCH IF V-BIT SET...IT SHOULD BE.
3164 014360 004510      JSR      RS.      (R0)  ;ERROR!!!          V-BIT NOT SET.
3165 014362 000242      BG6:  CLV                      ;CLR V-BIT
3166 014364 105214      INCB     (R4)      ;ONE MORE INC ON "BG4"
3167 014366 103402      BCS      BG7      ;BR IF C SET...IT SHOULD NOT BE.
3168 014370 102001      BVC      BG7      ;BR IF V SET...IT SHOULD NOT BE.
3169 014372 100401      BMI      BGEND     ;BR IF N SET...IT SHOULD BE.
3170 014374 004510      BG7:  JSR      RS.      (R0)  ;ERROR!!!          CC NOT = 2
3171 014376 000111      BGEND: JMP      (R1)     ;EXIT CODE...ONLY AT END OF BLOCK.
3172      000025      BGLLEN= <.-BRGOB>/2 ;CALCULATED LENGTH OF ROUTINE IN WORDS.
3173
3174 014400 013767 177776 164560 BGERR:  MOV      @#PSW, $TMP3 ;SAVE PSW
3175 014406 010567 164546      MOV      RS,      $TMP0 ;SAVE ERROR PC
3176 014412 160367 164542      SUB      R3,      $TMP0 ;OFFSET IT TO THE DATA POINTER
3177 014416 062767 014352 164534      ADD      #BG3-2, $TMP0 ;MAKE IT VIRTUAL SO IT CAN BE FOUND IN LISTING
3178 014424 010567 164532      MOV      RS,      $TMP1 ;GET ERROR PC
3179 014430 162767 000002 164524      SUB      #2,      $TMP1 ;MAKE IT PHYSICAL
3180 014436 012767 000012 164520      MOV      #12,     $TMP2 ;SET UP S/B PSW
3181 014444 010367 164450      MOV      R3,      $GDADR ;GET PHYSICAL DATA LOCATION
3182 014450 011367 164452      MOV      (R3),    $BDDAT ;GET THE DATA
3183 014454 026727 164500 014374      CMP      $TMP0,   #BG7   ;CHECK WHICH ERROR
3184 014462 001404      BEQ      1$      ;BRANCH IF SECOND ERROR
3185 014464 012767 077600 164432      MOV      #077600,$GDDAT ;SET UP S/B DATA
3186 014472 000403      BR       2$      ;SKIP
3187 014474 012767 100200 164422 1$:  MOV      #100200,$GDDAT ;SET UP S/B DATA
3188 014502      2$:
3189 014502 004767 005542      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3190 014506 000022      .WORD    22      ;ERROR TYPE CODE.
3191 014510 000205      RTS      RS      ;CONTINUE TESTING.
3192
3193      ;* BRANCH GOBBLE EXIT ROUTINE
3194 014512 012602      BGEXIT: MOV      (SP)+, R2 ;GET FIRST ADDRESS OF BLOCK (SHIFTED).
3195 014514 005722      TST      (R2)+   ;SHIFT IT ONE WORD.
3196 014516 005767 164470      TST      $PASS   ;CHECK FOR PASS 0.
3197 014522 001406      BEQ      1$      ;BR IF FIRST PASS
3198 014524 032777 004000 164406      BIT      #4000,  @SWR  ;CHECK FOR INHIBIT ITERATIONS
3199 014532 001002      BNE      1$      ;BRANCH IF SET
3200 014534 005316      DEC      (SP)    ;COUNT TIMES SHIFTED
3201 014536 001240      BNE      BGLoop  ;BR IF NOT DONE ENOUGH SHIFTS.
3202 014540 005726      1$:  TST      (SP)+   ;ADJUST STACK POINTER.
3203 014542 050502      BIS      R5,     R2 ;RESET ADR POINTER TO TOP OF BLOCK.
3204 014544 005202      INC      R2      ;GO TO NEXT BLOCK
3205 014546 004767 001146      JSR      PC,     MMUP ;GO FIND NEXT BLOCK AND LOOP TO BGLoop.

```

F12

0-124K MEMORY EXERCISER, 16K VER
02-DEC-76 08:47

0-124K MEMORY EXERCISER, 16K VER
02-DEC-76 08:47

MACY11 27(1006) 02-DEC-76 09:00 PAGE 66

SEQ 0149

.SBTTL DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.

DONE:

3206	014552	004567	004502
3207	014552	000000	
3208	014552	004567	004502
3209	014556	000000	
3210	014560	005067	164404
3211	014564	105067	164312
3212	014570	036767	164006 164736
3213	014576	001004	
3214	014600	036767	164000 164730
3215	014606	001426	
3216	014610	032777	000200 164322
3217	014616	001022	
3218	014620	022767	000003 163754
3219	014626	001007	
3220	014630	005737	000042
3221	014634	001006	
3222	014636	004767	002332
3223	014642	000167	171140
3224			
3225	014646	004767	002724
3226	014652	004767	003126
3227	014656	004557	006714
3228	014662	001201	

```

JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.DWORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
TST32: CLR $TIMES ;RESET ITERATION COUNTER FOR RESTARTING TEST.
CLR $STNM ;RESET TEST NUMBER.
1$: BIT PRGMAP, SAVTST ;CHECK IF PROGRAM IS IN TEST AREA.
BNE 2$ ;BR IF IT PROG IN MEM TO BE TESTED.
BIT PRGMAP+2, SAVTST+2 ;CHECK HI 64K
BEQ $EOP ;BR IF PROG NOT IN MEM TO BE TESTED.
2$: BIT #SW07, $SWR ;CHECK FOR INHIBIT RELOCATION SWITCH.
BNE $EOP ;SKIP RELOCATION IF SWITCH SET.
CMP #3, PRGMAP ;CHECK IF PROGRAM IN FIRST BK.
BNE 4$ ;BR IF NOT IN FIRST BK.
TST $#42 ;CHECK FOR A MONITOR.
BNE 5$ ;BR IF A MONITOR.
3$: JSR PC, RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.
JMP START1 ;LOOP BACK AND RUN ALL TESTS AGAIN.

4$: JSR PC, RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
5$: JSR PC, RESLDR ;RESTORE LOADERS.
JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED

```

```

3229
3230
3231
3232
3233
3234
3235
3236
3237
3238 014664
3239 014664 000240
3240 014666 005067 164276
3241 014672 005267 164314
3242 014676 042767 100000 164306
3243 014704 005327
3244 014706 000001
3245 014710 003027
3246 014712 012737
3247 014714 000001
3248 014716 014706
3249 014720 004567 006652
3250 014724 014774
3251 014726 016746 164260
3252
3253
3254 014732 013746 177776
3255 014736 004767 007554
3256 014742 004567 006630
3257 014746 015011
3258 014750
3259
3260 014750 016700 163066
3261 014754 001405
3262 014756 000005
3263 014760 004710
3264 014762 000240
3265 014764 000240
3266 014766 000240
3267 014770
3268 014770 000167 171012
3269 014774 005015 047105 020104
3270 015002 040520 051523 021440
3271 015010 000
3272 015011 377 377 000
3273
3274
3275
3276
3277
3278
3279
3280
3281
3282 015014
3283 015014 012737 077406 172300
3284 015022 012737 077406 172302

```

```

:*****
.SBTTL END OF PASS ROUTINE

;*INCREMENT THE PASS NUMBER ($PASS)
;*TYPE "END PASS *XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO START1

$EOP:
      NOP
      CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
      INC $PASS ;:INCREMENT THE PASS NUMBER
      BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
      DEC (PC)+ ;:LOOP?
$EOPCT: .WORD 1
      BGT $DOAGN ;:YES
      MOV (PC)+,(PC)+ ;:RESTORE COUNTER
$ENDCT: .WORD 1
      $EOPCT
      JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD $ENDMG ;ADDRESS OF MESSAGE TO BE TYPED
      MOV $PASS,-(SP) ;:SAVE $PASS FOR TYPEOUT
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
      MOV @#PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
      JSR PC,$TYPDS ;GO TO THE SUBROUTINE
      JSR RS,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD $ENULL ;ADDRESS OF MESSAGE TO BE TYPED
$GET42:
      MOV 42,RO ;:GET MONITOR ADDRESS
      BEQ $DOAGN ;:BRANCH IF NO MONITOR
      RESET ;:CLEAR THE WORLD
$ENDAD: JSR PC,(RO) ;:GO TO MONITOR
      NOP ;:SAVE ROOM
      NOP ;:FOR
      NOP ;:ACT11
$DOAGN: JMP START1 ;:RETURN*****
$ENDMG: .ASCIZ <15><12>/END PASS #/
$ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
.SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
.SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
:*****
;* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
;* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
;* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
;* PARS 4, 5, AND 6 ARE UNUSED.
:*****
MINIT:
      MOV #200-1*400+UP+RW,@#KIPDR0 ;:SET KIPDR0 = RW UP 200 BLOCKS
      MOV #200-1*400+UP+RW,@#KIPDR1 ;:SET KIPDR1 = RW UP 200 BLOCKS

```


MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.

```

3285 015030 012737 077406 172304 MOV #200-1*400+UP+RW, @#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
3286 015036 012737 077406 172306 MOV #200-1*400+UP+RW, @#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
3287 015044 005037 172310 CLR @#KIPDR4
3288 015050 005037 172312 CLR @#KIPDR5
3289 015054 005037 172314 CLR @#KIPDR6
329C 015060 012737 077406 172316 MOV #200-1*400+UP+RW, @#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
3291 015066 005037 172340 CLR @#KIPAR0 ;MAP PAR0 INTO BANK0
3292 015072 012737 000200 172342 MOV #200, @#KIPAR1 ;MAP PAR1 INTO BANK1
3293 015100 005037 172344 CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
3294 015104 005037 172346 CLR @#KIPAR3
3295 015110 005037 172350 CLR @#KIPAR4
3296 015114 005037 172352 CLR @#KIPAR5
3297 015120 005037 172354 CLR @#KIPAR6
3298 015124 012737 007600 172356 MOV #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
3299 015132 012737 000001 177572 MOV #1, @#SRO ;ENABLE MEMORY MANAGEMENT
3300 015140 000207 RTS PC ;RETURN

```

```

3301
3302
3303 ;:*****

```

```

;* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
;:*****

```

```

3304 INITMM: MOV #BIT0, BITPT ;SET POINTER TO BANK0
3305 CLR BITPT+2 ;CLEAR HI 64K BANK POINTERS
3306 015142 012767 000001 164374 CLR R2 ;SET ADDRESS POINTER TO 0
3307 015150 005067 164372 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3308 015154 005002 CLR MMAVA ;CHECK FOR MEM MGMT AVAILABLE
3309 015156 016705 164424 TST #10$ ;BRANCH IF NO MEM MGMT
3310 015162 005767 163420 BEQ @#KIPAR2 ;SET UP 3RD PAR TO BANK0
3311 015166 001514 CLR #40000, R2 ;RESET VIRTUAL ADR POINTER
3312 015170 005037 172344 MOV BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED
3313 015174 012702 040000 164322 1$: BNE 2$ ;BRANCH IF MATCH
3314 015200 036767 164340 BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
3315 015206 001015 BNE 2$ ;BRANCH IF MATCH
3316 015210 036767 164332 164314 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3317 015216 001011 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3318 015220 062737 000200 172344 ROL BITPT+2 ;...HI POINTER.
3319 015226 006367 164312 BPL 1$ ;BR IF MORE.
3320 015232 006167 164310 HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
3321 015236 100360 BIT BITPT, LADMAP ;CHECK IF LAST BANK.
3322 015240 000000 BNE 3$ ;BR IF LAST BANK.
3323 015242 036767 164276 164332 2$: BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
3324 015250 001004 BEQ 4$ ;BR IF NOT LAST BANK.
3325 015252 036767 164270 164324 MOV LADMSK, R5 ;SET MASK TO FIND LAST ADR.
3326 015260 001405 BIC #20000, TMAPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
3327 015262 016705 164312 3$: MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3328 015266 042767 020000 164302 MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3329 015274 013737 172344 172346 4$: MOV BITPT+2, TMPPT+2 ;...HI 64K.
3330 015302 016767 164236 164240 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3331 015310 016767 164232 164234 BEQ 21$ ;BRANCH IF NOT 8K.
3332 015316 032705 020000 ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3333 015322 001505 ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3334 015324 062737 000200 172346 5$: ROL TMPPT+2 ;...HI POINTER.
3335 015332 006367 164212 BMI 20$ ;BR IF NO MORE.
3336 015336 006167 164210 BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3337 015342 100473 BNE 6$ ;BRANCH IF A MATCH.
3338 015344 036767 164200 164156 BIT TMPPT+2, TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3339 015352 001004
3340 015354 036767 164172 164150

```

```

3341 015362 001760          BEQ      5$          ;BRANCH IF NO MEMORY
3342 015364 036767 164160 164210 6$: BIT      TMPPT, LADMAP ;CHECK IF LAST BANK.
3343 015372 001004          BNE      7$          ;BRANCH IF A MATCH
3344 015374 036767 164152 164202 BIT      TMPPT+2,LADMAP+2 ;CHECK HI 64K
3345 015402 001455          BEQ      21$         ;BR IF NOT LAST BANK.
3346 015404 016705 164170          MOV      LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3347 015410 052767 020000 164160 7$: BIS      #20000, TEMPLAD ;MAKE SURE LAST ADDRESS IS IN BANK 3.
3348 015416 000447          BR       21$         ;BR TO FINISH UP.
3349
3350 015420 036767 164120 164102 10$: BIT      BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED.
3351 015426 001006          BNE      11$         ;BR IF MATCH.
3352 015430 062702 020000          ADD      #20000, R2 ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3353 015434 106367 164104          ASLB    BITPT ;UPDATE BANK POINTER TO NEXT BANK.
3354 015440 100367          BPL     10$         ;BR IF MORE BANKS.
3355 015442 000000          HALT    ;FATAL ERROR!!! NO 4K BANK FOUND?
3356 015444 016767 164074 164076 11$: MOV      BITPT, TMPPT ;COPY BANK POINTER.
3357 015452 036767 164066 164122 BIT      BITPT, LADMAP ;CHECK IF LAST BANK.
3358 015460 001021          BNE      12$         ;BR IF LAST BANK.
3359 015462 032705 020000          BIT      #BIT13, R5 ;CHECK FOR 8K BLOCK SIZE.
3360 015466 001423          BEQ      21$         ;BRANCH IF SMALLER BLOCK SIZE.
3361 015470 106367 164054          ASLB    TMPPT ;POINT TO NEXT BANK.
3362 015474 100416          BMI     20$         ;BRANCH IF OVERFLOW.
3363 015476 036767 164046 164024 BIT      TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3364 015504 001412          BEQ      20$         ;BRANCH IF NOT TO BE TESTED.
3365 015506 112767 000011 164043 MOV      #11, FLAG8K ;SET 8K BLOCK SIZE FLAG.
3366 015514 036767 164030 164060 BIT      TMPPT, LADMAP ;CHECK FOR LAST BANK.
3367 015522 001403          BEQ      20$         ;BR IF NOT LAST BANK.
3368 015524 016705 164050          12$: MOV      LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3369 015530 000402          BR       21$         ;SKIP MASK RESET.
3370 015532 012705 017777          20$: MOV      #MASK4K, R5 ;RESET MASK TO 4K BLOCK SIZE.
3371 015536 056767 164002 164004 21$: BIS      BITPT, TMPPT ;SET TMPPT FOR FLAGING LAST BANK.
3372 015544 056767 163776 164000 BIS      BITPT+2, TMPPT+2
3373 015552 036767 163766 164010 BIT      BITPT, FADMAP ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3374 015560 001004          BNE      22$         ;BR IF FIRST BANK.
3375 015562 036767 163760 164002 BIT      BITPT+2, FADMAP+2 ;CHECK HI 64K.
3376 015570 001450          BEQ      INITEX ;BR IF NOT FIRST BANK.
3377 015572 016702 163766          22$: MOV      TMPFAD, R2 ;RESET ADDRESS POINTER TO FIRST ADR.
3378 015576 000445          BR       INITEX
3379
3380 015600 016705 164002          INITDN: MOV      BLKMSK, R5 ;RESET R5 TO CURRENT BLOCK MASK.
3381 015604 005002          CLR     R2 ;INIT ADDRESS POINTER.
3382 015606 005767 162774          TST     MMAPA ;CHECK FOR MEM MGMT
3383 015612 001411          BEQ      31$         ;BRANCH IF NO MEM MGMT
3384 015614 012767 100000 163724 MOV      #BIT15, BITPT+2 ;SET POINTER TO TOP BIT
3385 015622 005067 163716          CLR     BITPT
3386 015626 012737 007600 172347 MOV      #7600, @#KIPAR2 ;SET PAR TO TOP OF MEM
3387 015634 000403          BR       32$         ;BRANCH TO COMMON AREA
3388
3389 015636 012767 000400 163700 31$: MOV      #BIT8, BITPT ;SET UP BANK POINTER
3390 015644 012767 015666 163702 32$: MOV      #33$, MMORE ;SET "MMDOWN" EXIT ADDRESS.
3391 015652 066767 152722 163674 ADD      RELOC, MMORE ;ADD OFFSET
3392 015660 004767 000524          JSR     PC, MMDOWN ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3393 015664 000000          HALT    ;FATAL ERROR!!! NO MEM INDICATED IN MEM MAP ABOVE 8K!
3394 015666 036767 163652 163706 33$: BIT      BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADR.
3395 015674 001004          BNE      34$         ;BR IF LAST BANK FLAG FOUND.
3396 015676 036767 163644 163700 BIT      BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADR.

```

```

3397 015704 001402          BEQ      INITEX      ;BR IF NO LAD FLG FOUND.
3398 015706 016702 163662 34$:  MOV      LSTADR, R2  ;SET UP R2.
3399 015712 010467 163636 INITEX: MOV      R4,    MMORE ;PUT RETURN PC INTO "MMORE"
3400 015716 000204          RTS      R4      ;RETURN
3401
3402
3403
3404
3405
3406
3407
3408 015720 036767 163624 163654 MMUP:  BIT      TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3409 015726 001122          BNE      10$      ;BR IF LAST BANK.
3410 015730 036767 163616 163646 BIT      TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3411 015736 001116          BNE      10$      ;BR IF LAST BANK.
3412 015740 016705 163642          MOV      BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3413 015744 005767 162636          TST      MMAPA    ;CHECK FOR MEM MGMT AVAILABLE
3414 015750 001515          BEQ      20$      ;BRANCH IF NO MEM MGMT
3415 015752 012702 040000          MOV      #40000, R2 ;RESET VIRTUAL ADR POINTER
3416 015756 062737 000200 172344 1$:  ADD      #200, 2#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3417 015764 006367 163554          ASL      BITPT    ;UPDATE LO POINTER TO NEXT BANK.
3418 015770 006167 163552          ROL      BITPT+2  ;...HI POINTER.
3419 015774 100577          BMI      32$     ;BR IF ALL DONE.
3420 015776 036767 163542 163524 BIT      BITPT,  TSTMAP ;CHECK IF THIS BANK EXISTS
3421 016004 001004          BNE      2$      ;BRANCH IF MATCH
3422 016006 036767 163534 163516 BIT      BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3423 016014 001760          BEQ      1$      ;BRANCH IF NO MATCH
3424 016016 036767 163522 163556 2$:  BIT      BITPT,  LADMAP ;CHECK FOR LAST BANK FLAG.
3425 016024 001004          BNE      3$      ;BRANCH IF LAST BANK FLAG.
3426 016026 036767 163514 163550 BIT      BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3427 016034 001405          BEQ      4$      ;BR IF NOT LAST BANK.
3428 016036 016705 163536          MOV      LADMSK, R5 ;RESET MASK.
3429 016042 042767 020000 163526 BIC      #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3430 016050 016767 163470 163472 4$:  MOV      BITPT,  TMPPT ;COPY BITPT...LO 64K.
3431 016056 016767 163464 163466 MOV      BITPT+2,TMPPT+2 ;...HI 64K.
3432 016064 032705 020000          BIT      #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3433 016070 001530          BEQ      31$     ;BRANCH IF NOT.
3434 016072 013737 172344 172346 MOV      2#KIPAR2,2#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3435 016100 062737 000200 172346 5$:  ADD      #200, 2#KIPAR3 ;UP DATE FORTH PAR.
3436 016106 006367 163436          ASL      TMPPT    ;UPDATE LO POINTER TO NEXT 4K BANK.
3437 016112 006167 163434          ROL      TMPPT+2  ;...HI POINTER.
3438 016116 100513          BMI      30$     ;BR IF NO MORE.
3439 016120 036767 163424 163402 6$:  BIT      TMPPT,  TSTMAP ;CHECK IF BANK TO BE TESTED.
3440 016126 001004          BNE      7$      ;BRANCH IF A MATCH.
3441 016130 036767 163416 163374 BIT      TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3442 016136 001760          BEQ      5$      ;BRANCH IF NO MEMORY
3443 016140 036767 163404 163434 7$:  BIT      TMPPT,LADMAP ;CHECK FOR LAST BANK FLAG.
3444 016146 001004          BNE      8$      ;BRANCH IF A MATCH
3445 016150 036767 163376 163426 BIT      TMPPT+2,LADMAP+2 ;CHECK HI 64K
3446 016156 001475          BEQ      31$     ;BR IF NO LAST BANK FLAG.
3447 016160 016705 163414          MOV      LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.
3448 016164 052767 020000 163404 BIS      #20000, TEMPLAD ;SET VIRTUAL ADR TO BANK 3.
3449 016172 000467          BR       31$
3450
3451 016174 026702 163376 10$:  CMP      TEMPLAD, R2 ;CHECK IF LAST ADR REACHED.
3452 016200 001064          BNE      31$     ;BR IF MORE.

```

```

3453 016202 000474 BR 32$ ;BR IF ALL DONE.
3454
3455 016204 106267 163347 20$: ASRB FLAG8K ;SHIFT 8K FLAG
3456 016210 001407 BEQ 22$ ;BR IF NOT 8K BLOCK.
3457 016212 103455 BCS 30$ ;BR IF ANOTHER 4K.
3458 016214 105067 163337 CLR8 FLAG8K ;CLEAR OUT ALL FLAGS.
3459 016220 162702 040000 SUB #40000, R2 ;BACK UP 8K.
3460 016224 062702 020000 21$: ADD #20000, R2 ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3461 016230 106367 163310 22$: ASLB BITPT ;UPDATE POINTER.
3462 016234 100457 BMI 32$ ;BRANCH WHEN END IS REACHED.
3463 016236 036767 163302 163264 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS.
3464 016244 001767 BEQ 21$ ;BRANCH IF NO MATCH.
3465 016246 036767 163272 163326 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3466 016254 001402 BEQ 23$ ;BR IF NO MATCH.
3467 016256 016705 163316 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3468 016262 016767 163256 163260 23$: MOV BITPT, TMPPT ;SET UP TMP POINTER.
3469 016270 032705 020000 BIT #BIT13, R5 ;CHECK FOR 8K BLOCK SIZE.
3470 016274 001426 BEQ 31$ ;BRANCH IF SMALLER BLOCK SIZE.
3471 016276 106367 163246 ASLB TMPPT ;POINT TO NEXT BANK.
3472 016302 100421 BMI 30$ ;BRANCH IF OVERFLOW.
3473 016304 036767 163240 163216 BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3474 016312 001415 BEQ 30$ ;BRANCH IF NOT TO BE TESTED.
3475 016314 036767 163224 163260 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3476 016322 112767 000011 163227 MOV8 #11, FLAG8K ;SET 8K BLOCK FLAG.
3477 016330 036767 163210 163244 BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3478 016336 001403 BEQ 30$ ;BR IF NO FLAG.
3479 016340 016705 163234 MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADR.
3480 016344 000402 BR 31$
3481 016346 012705 017777 30$: MOV #MASK4K, R5 ;SET MASK TO 4K.
3482 016352 056767 163166 163170 31$: BIS BITPT, TMPPT ;SET TMPPT FOR FINDING LAST ADR.
3483 016360 056767 163162 163164 BIS BITPT+2, TMPPT+2
3484 016366 016716 163162 MOV MMORE, (SP) ;FUDGE RETURN ADDRESS TO LOOP.
3485 016372 000207 RTS PC ;RETURN
3486 ;* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3487 016374 005767 163714 32$: TST MPRX ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3488 016400 001402 BEQ 33$ ;BR IF NONE.
3489 016402 004767 001660 JSR PC, CKPMER ;CHECK FOR PARITY MEMORY ERRORS.
3490 016406 000207 33$: RTS PC ;STRAIGHT RETURN.
3491
3492 ;*****
3493 ;* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
3494 ;* FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
3495 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3496 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3497 ;*****
3498 016410 036767 163130 163152 MMDOWN: BIT BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
3499 016416 001004 BNE 1$ ;BR IF FIRST ADR IN THIS BANK.
3500 016420 036767 163122 163144 BIT BITPT+2, FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
3501 016426 001404 BEQ 2$ ;BR IF NO FLAG
3502 016430 026702 163130 1$: CMP TMPFAD, R2 ;CHECK IF FIRST ADDRESS REACHED.
3503 016434 001052 BNE 9$ ;BR IF MORE.
3504 016436 000453 BR 10$ ;BR IF ALL DONE.
3505 016440 005767 162142 2$: TST MMAVA ;CHECK IF MEM MGMT IS AVAILABLE
3506 016444 001425 BEQ 6$ ;BRANCH IF NOT
3507 016446 162737 000200 172344 3$: SUB #200, #KIPAR2 ;LOWER MEM MGMT PAR BY 4K
3508 016454 006067 163066 ROR BITPT+2 ;MOV POINTER TO NEXT LOWER BANK...HI MAP.

```


.SBTTL SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.

* SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN RO (BOTTOM 16 BITS).
* BITS 16 AND 17 ARE IN \$TMPD.

3529
3530
3531
3532
3533
3534 016570 010200
3535 016572 005067 162362
3536 016576 005767 162004
3537 016602 001417
3538 016604 010146
3539 016606 013701 172344
3540 016612 006301
3541 016614 006301
3542 016616 006301
3543 016620 006301
3544 016622 006301
3545 016624 006167 162330
3546 016630 006301
3547 016632 006167 162322
3548 016636 060100
3549 016640 012601
3550 016642 000207

PHYADR: MOV R2, RO ;VITRUAL INTO RO
CLR \$TMPD ;CLEAR TEMP SAVE OF HIGH BITS
TST MAVA ;CHECK FOR MEM MGMT AVAILABLE
BEQ 1\$;BRANCH IF NO MEM MGMT
MOV R1, -(SP) ;PUSH R1 ON STACK
MOV @#KIPAR2, R1 ;GET PAR TO BE ADDED TO VIRTUAL
ASL R1 ;SHIFT IT 6 TIMES
ASL R1
ASL R1
ASL R1
ROL \$TMPD ;SAVE EXTRA BITS
ASL R1
ROL \$TMPD
ADD R1, RO ;ADD SHIFTED PAR TO VIRTUAL
MOV (SP)+, R1 ;POP STACK INTO R1
1\$: RTS PC ;RETURN

* SUBROUTINE TO PUT BANK NUMBER INTO RO.

3551
3552
3553
3554
3555 016644 005000
3556 016646 010146
3557 016650 010246
3558 016652 016701 162666
3559 016656 016702 162664
3560 016662 006202
3561 016664 006001
3562 016666 103403
3563 016670 105200
3564 016672 100373
3565 016674 000000
3566 016676
3567 016676 012602
3568 016700 012601
3569 016702 000207

BANKNO: CLR RO ;INIT RO
MOV R1, -(SP) ;PUSH R1 ON STACK
MOV R2, -(SP) ;PUSH R2 ON STACK
MOV BITPT, R1 ;GET BANK MAP POINTER...LO 64K.
MOV BITPT+2, R2 ;...HI 64K.
1\$: ASR R2 ;SHIFT POINTER...HI
ROR R1 ;...LO
BCS 2\$;BR WHEN POINTER FOUND.
INCB RO ;COUNT BANKS.
BPL 1\$;BR IF NOT OVERFLOW.
HALT ;FATAL ERROR!!! NO POINTER FOUND.
2\$: MOV (SP)+, R2 ;POP STACK INTO R2
MOV (SP)+, R1 ;POP STACK INTO R1
RTS PC ;RETURN

* SUBROUTINE TO WRITE THE CONSTANT IN RO INTO ALL OF MEMORY.

3570
3571
3572
3573
3574 016704
3575 016704 004467 176232
3576 016710 010022
3577 016712 030502
3578 016714 001375
3579 016716 004767 176776
3580 016722 000207

SETCON: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2\$: MOV RO, (R2)+ ;MOV CONSTANT INTO MEMORY
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2\$;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1\$.
RTS PC ;RETURN

```

3581
3582
3583
3584 016724 106112
3585 016726 106112
3586 016730 106112
3587 016732 106112
3588 016734 106112
3589 016736 106112
3590 016740 106112
3591 016742 106112
3592 016744 106122
3593 016746 106112
3594 016750 106112
3595 016752 106112
3596 016754 106112
3597 016756 106112
3598 016760 106112
3599 016762 106112
3600 016764 106112
3601 016766 106122
3602 016770 000207
3603
3604
3605
3606
3607 016772 012704 000020
3608
3609 016776 010022
3610 017000 010022
3611 017002 010022
3612 017004 010022
3613
3614 017006 010322
3615 017010 010322
3616 017012 010322
3617 017014 010322
3618
3619 017016 010022
3620 017020 010022
3621 017022 010022
3622 017024 010022
3623
3624 017026 010322
3625 017030 010322
3626 017032 010322
3627 017034 010322
3628
3629 017036 005304
3630 017040 001356
3631 017042 010046
3632 017044 010300
3633 017046 012603
3634 017050 000207

```

```

;*****
;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
;*****
ROTATE: ROLB (R2) ;(R2)=177776 OR 000001
        ROLB (R2) ;(R2)=177775 OR 000002
        ROLB (R2) ;(R2)=177773 OR 000004
        ROLB (R2) ;(R2)=177767 OR 000010
        ROLB (R2) ;(R2)=177757 OR 000020
        ROLB (R2) ;(R2)=177737 OR 000040
        ROLB (R2) ;(R2)=177677 OR 000100
        ROLB (R2) ;(R2)=177777 OR 000000
        ROLB (R2)+ ;(R2)=177577 OR 000200
        ROLB (R2) ;(R2)=177377 OR 000400
        ROLB (R2) ;(R2)=176777 OR 001000
        ROLB (R2) ;(R2)=175777 OR 002000
        ROLB (R2) ;(R2)=173777 OR 004000
        ROLB (R2) ;(R2)=167777 OR 010000
        ROLB (R2) ;(R2)=157777 OR 020000
        ROLB (R2) ;(R2)=137777 OR 040000
        ROLB (R2) ;(R2)=077777 OR 100000
        ROLB (R2)+ ;(R2)=177777 OR 000000
        RTS PC ;RETURN

```

```

;*****
;* SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
;*****
W3X9: MOV #16.,R4 ;EACH LOOP WRITES 256. WORDS
25: MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R0,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    MOV R3,(R2)+
    DEC R4
    BNE 25
    MOV R0,-(SP) ;SAVE R0
    MOV R3,R0 ;PUT R3 INTO R0
    MOV (SP)+,R3 ;PUT SAVED R0 INTO R3
    RTS PC ;RETURN

```

RELOCATION SUBROUTINES.

```

3635
3636
3637
3638
3639 017052
3640 017052 010246
3641 017054 010346
3642 017056 010446
3643 017060 012502
3644 017062 012503
3645 017064 012704 020000
3646 017070 012223
3647 017072 005304
3648 017074 001375
3649 017076 012704 020000
3650 017102 024243
3651 017104 001417
3652 017106 011267 162012
3653 017112 011367 162010
3654 017116 010267 161776
3655 017122 010367 161774
3656 017126 004767 003116
3657 017132 000023
3658 017134 000000
3659 017136 162705 000004
3660 017142 000746
3661 017144 005304
3662 017146 001355
3663 017150 004567 004422
3664 017154 026623
3665
3666 017156 010346
3667 017160 004767 006052
3668 017164 012604
3669 017166 012603
3670 017170 012602
3671 017172 000205
3672
3673
3674
3675 017174 022767 000003 161400
3676 017202 001401
3677 017204 000000
3678 017206
3679 017206 010046
3680 017210 010146
3681 017212 005767 161370
3682 017216 001465
3683 017220 012737 007600 172346
3684 017226 005000
3685 017230 012701 100000
3686 017234 162737 000200 172346
3687 017242 006001
3688 017244 006000
3689 017246 103500
3690 017250 030167 162252

```

```

.SBTTL RELOCATION SUBROUTINES.
;*****
;* ROUTINE TO RELOCATE PROGRAM CODE
;*****
RELOC:
MOV R2,-(SP) ;:PUSH R2 ON STACK
MOV R3,-(SP) ;:PUSH R3 ON STACK
MOV R4,-(SP) ;:PUSH R4 ON STACK
4$: MOV (R5)+, R2 ;:GET FIRST LOCATION.
MOV (R5)+, R3 ;:GET FIRST LOCATION OF DESTINATION.
MOV #20000, R4 ;:SET UP BK COUNTER.
1$: MOV (R2)+, (R3)+ ;:MOV THE DATA.
DEC R4 ;:COUNT THE WORDS.
BNE 1$ ;:BR IF MORE.
MOV #20000, R4 ;:RESET THE COUNTER.
2$: CMP -(R2), -(R3) ;:CHECK THE DATA JUST MOVED.
BEQ 3$ ;:BR IF DATA OK.
MOV (R2), $GDDAT ;:GET SOURCE DATA.
MOV (R3), $BDDAT ;:GET DESTINATION DATA.
MOV R2, $GDADR ;:GET SOURCE ADDRESS.
MOV R3, $BDADR ;:GET DESTINATION ADDRESS.
JSR PC, $ERROR ;:*** ERROR *** (GO TYPE A MESSAGE)
.WORD 23 ;:ERROR TYPE CODE.
HALT ;:FATAL ERROR!!! RELOCATION FAILED.
SUB #4, R5 ;:ADJUST RETURN POINTER.
BR 4$ ;:GO BACK AND TRY AGAIN.
3$: DEC R4 ;:COUNT WORDS.
BNE 2$ ;:BR IF MORE.
JSR R5, $PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD PRELOC ;:ADDRESS OF MESSAGE TO BE TYPED
;:"PROGRAM RELOCATED TO "
MOV R3, -(SP) ;:PUT THE DATA ON THE STACK.
JSR PC, $TYPAD ;:DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
MOV (SP)+, R4 ;:POP STACK INTO R4
MOV (SP)+, R3 ;:POP STACK INTO R3
MOV (SP)+, R2 ;:POP STACK INTO R2
RTS R5 ;:RETURN
;*****
;* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
;*****
RELTOP: CMP #3, PRGMAP ;:CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
BEQ 1$ ;:BR IF OK
HALT ;:FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
1$: MOV R0,-(SP) ;:PUSH R0 ON STACK
MOV R1,-(SP) ;:PUSH R1 ON STACK
TST MMVA
BEQ 10$
MOV #7600, @#KIPAR3 ;:SET PAR TO TOP OF MEM
CLR R0 ;:INIT BANK POINTER...LO 64K
MOV #BIT15, R1 ;:...HI 64K.
2$: SUB #200, @#KIPAR3 ;:BACK DOWN ONE BANK.
ROR R1 ;:MOVE POINTER...HI 64K.
ROR R0 ;:...LO 64K.
BCS 90$
BIT R1, MEMMAP+2 ;:CHECK FOR BANK EXISTS.

```



```

3691 017254 001003      BNE      3$          ;BR IF AVAILABLE
3692 017256 030067 162242  BIT      RO,      MEMMAP ;CHECK FOR BANK EXISTS.
3693 017262 001764      BEQ      2$          ;BR IF NO BANK FOUND.
3694 017264 013737 172346 172344 3$:  MOV      2*#KIPAR3,2*#KIPAR2 ;COPY PAR
3695 017272 010046      MOV      RO,-(SP)    ;PUSH RO ON STACK
3696 017274 010146      MOV      R1,-(SP)    ;PUSH R1 ON STACK
3697 017276 162737 000200 172344 4$:  SUB      #200, 2*#KIPAR2 ;BACK DOWN WITH LOW PAR.
3698 017304 006001      ROR      R1          ;SHIFT POINTER.
3699 017306 006000      ROR      RO          ;...LO 64K.
3700 017310 103457      BCS      90$        ;BR IF OVERFLW.
3701 017312 030167 162210      BIT      R1,      MEMMAP+2 ;CHECK IF BANK EXISTS...HI 64K.
3702 017316 001003      BNE      6$          ;BR IF BANK EXISTS.
3703 017320 030067 162200      BIT      RO,      MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
3704 017324 001764      BEQ      4$          ;BR IF BANK DOESN'T EXIST.
3705 017326 052601      6$:  BIS      (SP)+, R1    ;GET SECOND BANK POINTER.
3706 017330 052600      BIS      (SP)+, RO    ;...LO 64K.
3707 017332 030067 161244      BIT      RO,      PRGMAP ;CHECK FOR CONFLICT.
3708 017336 001044      BNE      90$        ;ABORT IF DESTINATION OVERLAYS SOURCE.
3709 017340 004567 177506      JSR      R5,      RELOC ;GO RELOCATE PROGRAM.
3710 017344 000000      .WORD   0          ;SOURCE FIRST ADDRESS.
3711 017346 040000      .WORD   40000      ;DESTINATION FIRST ADDRESS.
3712 017350 013737 172344 172340  MOV      2*#KIPAR2,2*#KIPAR0 ;RELOCATE LO BANK
3713 017356 013737 172346 172342  MOV      2*#KIPAR3,2*#KIPAR1 ;RELOCATE HI BANK.
3714                                ;* PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3715 017364 010167 161214      MOV      R1,      PRGMAP+2 ;RESET PROGRAM MAP.
3716 017370 000473      BR       30$        ;BR TO COMMON EXIT.
3717
3718 017372 012700 000400 10$:  MOV      #BIT8, RO    ;SET BANK POINTER TO TOP OF MEM.
3719 017376 005001      CLR      R1          ;SET ADDRESS POINTER TO TOP.
3720 017400 162701 020000 11$:  SUB      #20000, R1   ;BACK DOWN ONE BANK.
3721 017404 006200      ASR      RO          ;MOVE POINTER DOWN ONE BANK.
3722 017406 103420      BCS      90$        ;BR IF OVERFLOW.
3723 017410 030067 162110      BIT      RO,      MEMMAP ;CHECK IF THIS BANK EXISTS.
3724 017414 001771      BEQ      11$        ;BR IF NON-EXISTANT BANK.
3725 017416 162701 020000      SUB      #20000, R1   ;BACK DOWN TO NEXT BANK.
3726 017422 006200      ASR      RO          ;MOV POINTER DOWN ONE BANK.
3727 017424 103411      BCS      90$        ;BR IF OVERFLOW.
3728 017426 030067 162072      BIT      RO,      MEMMAP ;CHECK IF THIS BANK EXISTS.
3729 017432 001762      BEQ      11$        ;BR TO START OVER IF NO LOWER BANK.
3730 017434 010046      MOV      RO,      -(SP) ;SAVE THE POINTER.
3731 017436 006300      ASL      RO          ;RESET POINTER TO HI BANK.
3732 017440 052600      BIS      (SP)+, RO    ;SET BIT FOR LO BANK.
3733 017442 030067 161134      BIT      RO,      PRGMAP ;CHECK FOR A PROGRAM CONFLICT.
3734 017446 001401      BEQ      12$        ;BR IF NO CONFLICT.
3735 017450 90$:
3736 017450 000000      HALT     ;FATAL ERROR!!! NOT ENOUGH MEMORY??
3737 017452 010167 000006 12$:  MOV      R1,      13$  ;SET DATA FOR RELOCATION SUBROUTINE.
3738 017456 004567 177370      JSR      R5,      RELOC ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3739 017462 000000      .WORD   0          ;SOURCE STARTING ADDRESS.
3740 017464 000000 13$:  .WORD   0          ;DESTINATION STARTING ADDRESS.
3741 017466 010167 161106      MOV      R1,      RELOC ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3742 017472 060107      ADD      R1,      PC  ;JUMP TO RELOCATED PROGRAM
3743                                ;* PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3744 017474 060106      ADD      R1,      SP  ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3745 017476 010167 161076      MOV      R1,      RELOC ;SET THE RELOCATION FACTOR.
3746 017502 060137 000004      ADD      R1,      2*#ERRVEC ;ADJUST ERROR VECTOR.
    
```

```

3747 017506 060137 000024      ADD    R1,    @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3748 017512 060137 000114      ADD    R1,    @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3749 017516 026727 161416 177570  CMP    SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
3750 017524 001404          BEQ    14$      ;BR IF HARDWARE SWITCH REGISTER.
3751 017526 060167 161406      ADD    R1,    SWR    ;ADJUST SOFTWARE SWITCH REGISTER.
3752 017532 060167 161404      ADD    R1,    DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3753 017536 062701 001612 14$:  ADD    #RADTAB,R1 ;POINT TO THE RELATIVE RELOCATION TABLE.
3754 017542 066721 161032 15$:  ADD    RELOCF, (R1)+ ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3755 017546 005721          16$:  TST    (R1)+    ;CHECK FOR INTERUM TERMINATOR.
3756 017550 001776          BEQ    16$      ;BR SO AS TO NOT MODIFY ZERO.
3757 017552 024127 177777      CMP    -(R1), #-1   ;CHECK FOR END OF TABLE.
3758 017556 001371          BNE    15$      ;BR IF MORE IN TABLE.
3759 017560 010067 161016 30$:  MOV    R0,    PRGMAP ;SET NEW PROGRAM MAP...LO 64K.
3760 017564 012601          MOV    (SP)+,R1    ;POP STACK INTO R1
3761 017566 012600          MOV    (SP)+,R0    ;POP STACK INTO R0
3762 017570 066716 161004      ADD    RELOCF, (SP) ;ADJUST RETURN ADDRESS.
3763 017574 000207          RTS    PC        ;RETURN
3764
3765
3766
3767
3768 017576 032767 000003 160776 RELO: BIT    #3,    PRGMAP ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3769 017604 001401          BEQ    1$      ;BR IF NO CONFLICT.
3770 017606 000000          HALT          ;FATAL ERROR!!! PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3771 017610 005767 160772 1$:  TST    MMAVA    ;CHECK FOR MEM MGMT.
3772 017614 001417          BEQ    10$     ;BR IF NO MEMMGMT.
3773 017616 005037 172344      CLR    @#KIPAR2  ;SET PAR 2 TO BANK 0.
3774 017622 012737 000200 172346  MOV    #200,    @#KIPAR3 ;SET PAR 3 TO BANK 1.
3775 017630 004567 177216      JSR    R5,    RELOC ;GO MOVE BK INTO BANKS 0 AND 1.
3776 017634 000000          .WORD    0      ;SOURCE STARTING ADDRESS.
3777 017636 040000          .WORD    40000  ;DESTINATION STARTING ADDRESS.
3778 017640 005037 172340      CLR    @#KIPAR0  ;RESTORE PAR 0 TO BANK0.
3779 017644 012737 000200 172342  MOV    #200,    @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
3780
3781 017652 000444          ;* PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3782          BR    30$   ;BR TO COMMON EXIT.
3783 017654 016746 160720 10$:  MOV    RELOCF, -(SP) ;PUT RELOCATION FACTOR ONTO THE STACK.
3784 017660 011667 000004      MOV    (SP),  20$   ;SET DATA FOR RELOC SUBROUTINE.
3785 017664 004567 177162      JSR    R5,    RELOC ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3786 017670 000000 20$:  .WORD    0      ;SOURCE STARTING ADDRESS.
3787 017672 000000          .WORD    0      ;DESTINATION STARTING ADDRESS.
3788 017674 161607          SUB    (SP),  PC    ;JUMP TO RELOCATED PROGRAM.
3789          ;* THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3790          SUB    (SP),  SP    ;RESET THE STACK POINTER.
3791 017700 010046          MOV    R0, -(SP)  ;PUSH R0 ON STACK
3792 017702 012700 001612      MOV    #RADTAB,R0 ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3793 017706 166620 000002 21$:  SUB    2(SP),  (R0)+ ;RESET ADDRESSES TO UNRELOCATED VALUES.
3794 017712 005720 22$:  TST    (R0)+    ;CHECK FOR TERMINATORS.
3795 017714 001776          BEQ    22$      ;BR OVER TERMINATORS.
3796 017716 024027 177777      CMP    -(R0), #-1  ;CHECK FOR END OF TABLE INDICATOR.
3797 017722 001371          BNE    21$      ;BR IF MORE ADDRESSES IN TABLE.
3798 017724 012600          MOV    (SP)+,R0  ;POP STACK INTO R0
3799 017726 161637 000004      SUB    (SP),    @#ERRVEC ;ADJUST ERROR VECTOR.
3800 017732 161637 000024      SUB    (SP),    @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3801 017736 161637 000114      SUB    (SP),    @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3802 017742 026727 161172 177570  CMP    SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.

```

```

3803 017750 001404 BEQ 23$ ;BR IF HARDWARE SWITCH REGISTER.
3804 017752 161667 161162 SUB (SP), SWR ;ADJUST SOFTWARE SWITCH REGISTER.
3805 017756 161667 161160 SUB (SP), DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3806 017762 162616 23$: SUB (SP)+ (SP) ;ADJUST RETURN ADDRESS.
3807 017764 005067 160610 30$: CLR RELOCF ;RESET RELOCATION FACTOR.
3808 017770 012767 000003 160604 MOV #3, PRGMAP ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3809 017776 005067 160602 CLR PRGMAP+2 ;...HI 64K.
3810 020002 000207 RTS PC ;RETURN.

```

```

3811
3812 ::*****
3813 * THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3814 * WHENCE IT CAME. THE LOADER AREA IS SAVED AT THE END OF THE BK OF
3815 * PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3816 ::*****

```

```

3817 020004 016700 161510 RESLDR: MOV LMA0, R0 ;CHECK IF THE LOADERS WERE SAVED.
3818 020010 001001 BNE 1$ ;BR IF LOADER AREA WAS SAVED.
3819 020012 000000 HALT ;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3820 020014 005767 160566 1$: TST MMAVA ;CHECK FOR MEM MGMT.
3821 020020 001402 BEQ 2$ ;SKIP IF NO MEM MGMT.
3822 020022 005037 177572 CLR #SRO ;DISABLE MEM MGMT.
3823 020026 012701 040000 2$: MOV #40000, R1 ;GET END OF BK, ASSUME PROG NOT RELOCATED.
3824 020032 012702 002734 MOV #1500., R2 ;GET COUNTER.
3825 020036 014140 3$: MOV -(R1), -(R0) ;MOVE THE LOADER AREA.
3826 020040 005302 DEC R2 ;COUNT HOW LONG THE AREA IS.
3827 020042 001375 BNE 3$ ;BR IF NOT MORE TO MOVE.
3828 020044 005767 160536 TST MMAVA ;CHECK FOR MEM MGMT.
3829 020050 001402 BEQ 4$ ;BR IF NO MEM MGMT.
3830 020052 005237 177572 INC #SRO ;ENABLE MEM MGMT.
3831 020056 000207 4$: RTS PC ;RETURN.

```

```

3832
3833
3834
3835
3836
3837
3838 020060 011667 161036
3839 020064 004567 003506
3840 020070 026562
3841
3842 020072 010146
3843 020074 010346
3844 020076 016703 161516
3845 020102 005733
3846 020104 100415
3847 020106 005713
3848 020110 001374
3849 020112 004767 003132
3850
3851 020116 000024
3852 020120 000417
3853 020122 005713
3854 020124 001415
3855 020126 005733
3856 020130 100374
3857 020132 004567 003440
3858 020136 026653
3859
3860 020140
3861 020140 004767 000610
3862 020144 004767 002100
3863 020150 000025
3864 020152 004767 000216
3865 020156 000761
3866 020160
3867 020160 012603
3868 020162 012601
3869 020164 000002
3870
3871
3872
3873
3874
3875 020166 005767 162122
3876 020172 001434
3877 020174 032777 000100 160736
3878 020202 001030
3879 020204 005767 160370
3880 020210 001410
3881 020212 032777 000040 160720
3882 020220 001004
3883 020222 026727 161334 001000
3884 020230 103415
3885 020232 016737 161376 000114
3886 020240 005037 000116
3887 020244 010346

```

```

.SBTTL PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
;*****
;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
;* FIND OUT WHICH REGISTER DETECTED THE ERROR.
;* THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
;*****
PESRV: MOV (SP), $BDADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
      JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
      ;"UNEXPECTED MEMORY PARITY TRAP."
      MOV R1,-(SP) ;PUSH R1 ON STACK
      MOV R3,-(SP) ;PUSH R3 ON STACK
      MOV .MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
1$: TST @ (R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
     BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
     TST (R3) ;CHECK FOR TABLE TERMINATOR.
     BNE 1$ ;BR IF MORE REGISTERS.
     JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
;***ERROR*** NO REGISTER INDICATED ERROR
     .WORD 24 ;ERROR TYPE CODE.
     BR 4$ ;EXIT
2$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
     BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
     TST @ (R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
     BPL 2$ ;BR IF NO ERROR FLAG.
     JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
     .WORD MTCE ;ADDRESS OF MESSAGE TO BE TYPED
     ;"MORE THAN ONE ERROR FOUND."
3$: JSR PC, $SPRINTQ ;SET UP VALUES FOR ERROR PRINTING.
5$: JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
     .WORD 25 ;ERROR TYPE CODE.
     JSR PC, $PCAN ;GO SCAN MEMORY FOR BAD PARITY.
     BR 2$ ;GO LOOK FOR MORE ERRORS.
4$: MOV (SP)+,R3 ;POP STACK INTO R3
     MOV (SP)+,R1 ;POP STACK INTO R1
     RTI ;RETURN.
;*****
;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
;*****
MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
      BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
      BIT #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
      BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
      TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK C.
      BEQ SETAE ;BR IF PROG IN BANK D.
      BIT #SW5, @SWR ;CHECK IF VECTORS PROTECTED.
      BNE SETAE ;BR IF VECTOR AREA PROTECTED.
      CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
      BLO MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.
SE*AE: MOV .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
      CLR @#PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
      MOV R3,-(SP) ;PUSH R3 ON STACK

```

```

3888 020246 016703 161346
3889 020252 052733 000001
3890 020256 005713
3891 020260 001374
3892 020262 012603
3893 020264 000207
3894
3895
3896
3897
3898 020266 005767 162022
3899 020272 001437
3900 020274 032777 000100 160636
3901 020302 001033
3902 020304 010346
3903 020306 016703 161306
3904 020312 005733
3905 020314 100023
3906 020316 032773 000001 177776
3907 020324 001010
3908 020326 004767 000422
3909 020332 004767 001712
3910 020336 000026
3911 020340 000411
3912 020342 004767 000026
3913 020346
3914 020346 004767 000402
3915 020352 004767 001672
3916 020356 000027
3917 020360 004767 000010
3918 020364 005713
3919 020366 001351
3920 020370 012603
3921 020372 000207
3922
3923
3924
3925
3926
3927
3928 020374
3929 020374 010046
3930 020376 010146
3931 020400 010246
3932 020402 010346
3933 020404 010446
3934 020406 013746 000114
3935 020412 013746 000116
3936 020416 004567 003154
3937 020422 026717
3938
3939 020424 012700 000001
3940 020430 005001
3941 020432 005002
3942 020434 005004
3943 020436 004767 000256

```

```

MAMF1: MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
        BIS #AE, 2(R3)+ ;SET ACTION ENABLE BIT IN PARITY REG
        TST (R3) ;CHECK FOR END OF TABLE.
        BNE MAMF1 ;BR IF MORE PARITY REGISTERS.
MAMF2: MOV (SP)+,R3 ;POP STACK INTO R3
        RTS PC ;RETURN.

;*****
;* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
;*****
CHKMER: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
        BEQ 4$ ;BR IF NO PARITY REGISTERS.
        BIT #SW6, 2SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
        BNE 4$ ;BR IF PARITY ERROR CHECKING INHIBITED.
        MOV R3, -(SP) ;PUSH R3 ON STACK
        MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
1$: TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
        BPL 3$ ;BR IF NO ERROR
        BIT #BITC, 2-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
        BNE 2$ ;BR IF NO ACTION ENABLE.
64$: JSR PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 26 ;ERROR TYPE CODE.
        BR 3$
        JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
2$:
65$: JSR PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 27 ;ERROR TYPE CODE.
        JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
        BNE 1$ ;BR IF MORE.
        MOV (SP)+,R3 ;POP STACK INTO R3
4$: RTS PC ;RETURN.

;*****
;* THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY.
;* TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
;* LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
;*****
PSCAN: MOV R0, -(SP) ;PUSH R0 ON STACK
        MOV R1, -(SP) ;PUSH R1 ON STACK
        MOV R2, -(SP) ;PUSH R2 ON STACK
        MOV R3, -(SP) ;PUSH R3 ON STACK
        MOV R4, -(SP) ;PUSH R4 ON STACK
        MOV 2#114, -(SP) ;PUSH 2#114 ON STACK
        MOV 2#116, -(SP) ;PUSH 2#116 ON STACK
        JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD SCANM ;ADDRESS OF MESSAGE TO BE TYPED
        ;"SCANNING MEMORY FOR BAD PARITY."
        MOV #BIT0, R0 ;SET BIT POINTER TO FIRST BANK.
        CLR R1 ;CLR HI 64K POINTER.
        CLR R2 ;INIT ADDRESS POINTER.
        CLR R4 ;INIT ERROR DETECTED FLAG.
        JSR PC, CLRPAR ;CLEAR THE PARITY REGISTERS.

```

H13

```

3944 020442 012737 000116 000114 MOV #116, @#114 ;HALT IF ANOTHER PARITY TRAP.
3945 020450 005037 000116 CLR @#116
3946 020454 005767 160126 TST MMAVA ;CHECK FOR MEMORY MANAGEMENT.
3947 020460 001406 BEQ 1$ ;BR IF NO MEM MGMT.
3948 020462 013746 172344 MOV @#KIPAR2, -(SP) ;PUSH @#KIPAR2 ON STACK
3949 020466 005037 172344 CLR @#KIPAR2 ;INIT MEM MGMT TO POINT TO BANK 0.
3950 020472 012702 040000 MOV #40000, R2 ;SET ADR POINTER TO PAR2.
3951 020476 030067 161022 1$: BIT R0, MEMMAP ;CHECK IF THIS BANK OF MEM EXISTS.
3952 020502 001003 BNE 2$ ;BR IF THIS BANK EXISTS.
3953 020504 030167 161016 BIT R1, MEMMAP+2 ;CHECK HI 64K MAP.
3954 020510 001442 BEQ 10$ ;BR IF THIS BANK DOESN'T EXIST.
3955 020512 2$:
3956 020512 010146 MOV R1, -(SP) ;PUSH R1 ON STACK
3957 020514 111201 3$: MOV (R2), R1 ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
3958 020516 016703 161076 MOV .MPRX, R3 ;SET UP POINTER TO PARITY REGISTERS.
3959 020522 005733 4$: TST @ (R3)+ ;CHECK FOR THE ERROR FLAG.
3960 020524 100024 BPL 6$ ;BR IF NO ERROR FLAG.
3961 020526 005704 TST R4 ;CHECK IF FIRST ERROR, THIS SCAN.
3962 020530 001003 BNE 5$ ;BR IF MORE THAN ONE ERROR FOUND.
3963 020532 005367 160354 DEC $ERTTL ;ADJUST ERROR COUNT.
3964 020536 005204 INC R4 ;SET FLAG TO INDICATE ERROR FOUND.
3965 020540 5$:
3966 020540 004767 000210 6$: JSR PC, $PRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3967 020544 004767 001500 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3968 020550 000030 .WORD 30 ;ERROR TYPE CODE.
3969 020552 111212 MOV (R2), (R2) ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
3970 020554 005053 CLR @-(R3) ;CLEAR THE ERROR FLAG.
3971 020556 105712 TSTB (R2) ;CHECK IF THE PARITY ERROR WAS CLEARED.
3972 020560 005733 TST @ (R3)+ ;CHECK FOR THE ERROR FLAG.
3973 020562 100005 BPL 6$ ;BR IF IT IS OK.
3974 020564 004567 003006 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3975 020570 026761 .WORD PEWNC ;ADDRESS OF MESSAGE TO BE TYPED
3976 "PARITY ERROR WILL NOT CLEAR."
3977 020572 005073 177776 CLR @-2(R3) ;CLEAR OUT THE PARITY ERROR FLAG.
3978 020576 005713 5$: TST (R3) ;CHECK FOR THE END OF REG ADR TABLE.
3979 020600 001350 BNE 4$ ;BR IF MORE PARITY REGISTERS.
3980 020602 005202 INC R2 ;GO TO NEXT MEMORY ADDRESS.
3981 020604 032702 017777 BIT #MASK4K, R2 ;CHECK FOR END OF 4K BANK.
3982 020610 001341 BNE 3$ ;BR IF MORE MEMORY THIS BANK.
3983 020612 012601 MOV (SP)+, R1 ;POP STACK INTO R1
3984 020614 000402 BR 11$ ;BR TO CHECK FOR NEXT BANK.
3985 020616 062702 020000 10$: ADD #20000, R2 ;SKIP BANKS THAT AREN'T THERE.
3986 020622 005767 157760 11$: TST MMAVA ;CHECK FOR MEM MGMT.
3987 020626 001413 BEQ 12$ ;BR IF NO MEM MGMT.
3988 020630 062737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
3989 020636 012702 040000 MOV #40000, R2 ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
3990 020642 006300 ASL R0 ;JPDTE BANK POINTER.
3991 020644 006101 ROL R1 ;...HI 64K.
3992 020646 100313 BPL 1$ ;BR IF MORE BANKS.
3993 020650 012637 172344 MOV (SP)+, @#KIPAR2 ;POP STACK INTO @#KIPAR2
3994 020654 000402 BR 20$ ;GO CHECK IF ANY ERRORS FOUND.
3995 020656 106300 12$: ASLB R0 ;UPDATE POINTER TO NEXT BANK.
3996 020660 100306 BPL 1$ ;BR IF MORE BANKS.
3997 020662 005704 20$: TST R4 ;CHECK IF ANY PARITY ERRORS DETECTED.
3998 020664 001003 BNE 21$ ;BR IF ERRORS DETECTED.
3999 020666 004567 002704 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
  
```

```

4000 020672 025772          .WORD  NOPE$          ;ADDRESS OF MESSAGE TO BE TYPED
4001 020674                21$:
4002 020674 012637 000116  MOV      (SP)+,2#116    ;;POP STACK INTO 2#116
4003 020700 012637 000114  MOV      (SP)+,2#114    ;;POP STACK INTO 2#114
4004 020704 012604          MOV      (SP)+,R4       ;;POP STACK INTO R4
4005 020706 012603          MOV      (SP)+,R3       ;;POP STACK INTO R3
4006 020710 012602          MOV      (SP)+,R2       ;;POP STACK INTO R2
4007 020712 012601          MOV      (SP)+,R1       ;;POP STACK INTO R1
4008 020714 012600          MOV      (SP)+,R0       ;;POP STACK INTO R0
4009 020716 000207          RTS        PC          ;RETURN.

```

```

4010
4011
4012
4013
4014
4015
4016
4017
4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029

```

```

*****
ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
*****

```

```

CLRPAR:
4014 020720          MOV      R3,-(SP)       ;;PUSH R3 ON STACK
4015 020720 010346          MOV      .MPRX, R3     ;;GET PARITY REGISTER TABLE POINTER.
4016 020722 016703 160672  TST      (R3)          ;;CHECK FOR THE TABLE TERMINATOR.
4017 020726 005713          BEQ      2$           ;;BR IF DONE ALL PARITY REGISTERS.
4018 020730 001402          CLR      2(R3)+       ;;CLEAR THE PARITY REGISTER.
4019 020732 005033          BR       1$           ;;BR FOR MORE
4020 020734 000774
4021 020736          2$:
4022 020736 012603          MOV      (SP)+,R3     ;;POP STACK INTO R3
4023 020740 000207          RTS        PC          ;RETURN.

```

```

.SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
*****
* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.SCMTAG)
* FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSTEMAC**.
*****

```

```

4030 020742 010267 160152  SPRNT:  MOV      R2, $GDADR ;SAVE THE ADDRESS UNDER TEST.
4031 020746 005067 160152  CLR      $GDDAT ;SHOULD BE DATA IS "0".
4032 020752 000430          BR       SPRNTB
4033
4034 020754 014367 160200  SPRNTQ: MOV      -(R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
4035 020760 013367 160176  MOV      2(R3)+, $TMP1 ;GET THE CONTENTS OF THE PARITY REG.
4036 020764 000402          BR       SPRNTQ
4037
4038 020766 011367 160166  SPRNTP: MOV      (R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
4039 020772 010267 160122  SPRNTQ: MOV      R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4040 020776 000414          BR       SPRNTA ;BR TO COMMON SECTION.
4041
4042 021000 010267 160114  SPRNT1: MOV      R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4043 021004 005367 160110  DEC      $GDADR ;ADJUST IT FOR PRINTOUT.
4044 021010 000407          BR       SPRNTA ;BR TO COMMON SECTION.
4045
4046 021012 010367 160142  SPRNT3: MOV      R3, $TMP0 ;GET THE DATA IN R3.
4047 021016 010267 160076  SPRNT2: MOV      R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
4048 021022 162767 000002 160070  SUB      #2, $GDADR ;ADJUST IT FOR PRINTOUT.
4049 021030 010067 160070  SPRNTA: MOV      R0, $GDDAT ;GET WHAT THE DATA SHOULD BE
4050 021034 010167 160066  SPRNTB: MOV      R1, $BDDAT ;GET WHAT THE DATA WAS
4051 021040 000207          RTS        PC          ;RETURN TO ENTER ERROR ROUTINES

```

```

*****
* SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
* R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.

```

```

4052
4053
4054
4055

```



```

4056
4057 021042 005710
4058 021044 001007
4059 021046 005760 000002
4060 021052 001004
4061 021054 004567 002516
4062 021060 026347
4063
4064 021062 000475
4065 021064
4066 021064 010146
4067 021066 010246
4068 021070 010346
4069 021072 010446
4070 021074 012701 000001
4071 021100 005002
4072 021102 012703 177777
4073 021106 010304
4074 021110 030110
4075 021112 001014
4076 021114 030260 000002
4077 021120 001011
4078 021122 105703
4079 021124 001042
4080 021126 162703 000001
4081 021132 005604
4082 021134 004567 002436
4083 021140 025603
4084 021142 000410
4085 021144 105703
4086 021146 001431
4087 021150 062703 000001
4088 021154 005504
4089 021156 004567 002414
4090 021162 025573
4091 021164
4092 021164 010346
4093 021166 010446
4094 021170 006303
4095 021172 006104
4096 021174 006003
4097 021176 010446
4098
4099
4100
4101 021200 013746 177776
4102 021204 004767 003540
4103 021210 003
4104 021211 000
4105 021212 010346
4106
4107
4108
4109 021214 013746 177776
4110 021220 004767 003524
4111 021224 005

```

```

*****
TYPMAP: TST (R0) ;CHECK IF ANY MEMORY IN MAP...LO 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        TST 2(R0) ;...HI 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD NOMEM ;ADDRESS OF MESSAGE TO BE TYPED
        ;"NO MEMORY FOUND."
        BR 6$ ;EXIT
1$:
        MOV R1,-(SP) ;;PUSH R1 ON STACK
        MOV R2,-(SP) ;;PUSH R2 ON STACK
        MOV R3,-(SP) ;;PUSH R3 ON STACK
        MOV R4,-(SP) ;;PUSH R4 ON STACK
        MOV #BIT0, R1 ;SET UP BANK POINTER...LO 64K.
        CLR R2 ;...HI 64K.
        MOV #-1, R3 ;SET UP ADDRESS POINTER TO -1.
        MOV R3, R4 ;HI BITS OF ADDRESS AS WILL.
2$:
        BIT R1, (R0) ;CHECK THE MAP FOR THIS BANK.
        BNE 3$ ;BR IF THIS BANK PRESENT.
        BIT R2, 2(R0) ;CHECK HI 64K MAP.
        BNE 3$ ;BR IF THIS BANK PRESENT.
        TSTB R3 ;CHECK FOR PREVIOUS PRINTOUT.
        BNE 5$ ;BR IF ALREADY TYPED "TO"
        SUB #1, R3 ;BACK UP TO LAST ADR OF PREVIOUS BANK.
        SBC R4 ;...HI ADDRESS BITS.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
        BR 4$ ;GO TO TYPE THE ADDRESS.
3$:
        TSTB R3 ;CHECK FOR PREVIOUS TYPEOUT.
        BEQ 5$ ;BR IF ALREADY TYPE "FROM".
        ADD #1, R3 ;POINT TO FIRST ADDRESS OF THIS BANK.
        ADC R4 ;...HI BITS OF ADDRESS.
        JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED
4$:
        MOV R3,-(SP) ;;PUSH R3 ON STACK
        MOV R4,-(SP) ;;PUSH R4 ON STACK
        ASL R3 ;BIT 15 INTO C-BIT
        ROL R4 ;BIT 15 INTO R4.
        ROR R3 ;RESTORE BITS 14-0.
        MOV R4,-(SP) ;;SAVE R4 FOR TYPEOUT
        ;;TYPE ADDRESS BITS 21-15
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $TYPOS ;GO TO THE SUBROUTINE
        .BYTE 3 ;TYPE 3 DIGIT(S)
        .BYTE 0 ;SUPPRESS LEADING ZEROS
        MOV R3,-(SP) ;SAVE R3 FOR TYPEOUT
        ;;TYPE ADDRESS BITS 14-0
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR PC, $TYPOS ;GO TO THE SUBROUTINE
        .BYTE 5 ;TYPE 5 DIGIT(S)

```



```

4112 021225 001 .BYTE 1 ;:TYPE LEADING ZEROS
4113 021226 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
4114 021230 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
4115 021232 062703 020000 5$: ADD #20000, R3 ;:UPDATE TO NEXT BANK.
4116 021236 005504 ADC R4 ;:...HI ADDRESS BITS.
4117 021240 006301 ASL R1 ;:SHIFT POINTER...LO 64K.
4118 021242 006102 ROL R2 ;:...HI 64K.
4119 021244 103321 BCC 2$ ;:BR IF MORE BANKS.
4120 021246 012604 MOV (SP)+,R4 ;:POP STACK INTO R4
4121 021250 012603 MOV (SP)+,R3 ;:POP STACK INTO R3
4122 021252 012602 MOV (SP)+,R2 ;:POP STACK INTO R2
4123 021254 012601 MOV (SP)+,R1 ;:POP STACK INTO R1
4124 021256 000207 6$: RTS PC ;:RETURN.
4125
4126 .SBTTL SCOPE HANDLER ROUTINE
4127
4128 ;:*****
4129 ;:*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
4130 ;:*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
4131 ;:*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
4132 ;:*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4133 ;:*SW14=1 LOOP ON TEST
4134 ;:*SW11=1 INHIBIT ITERATIONS
4135 ;:*SW09=1 LOOP ON ERROR
4136 ;:*SW08=1 LOOP ON TEST IN SWR<4:0>
4137 ;:*CALL
4138 ;:* SCOPE ;:SCOPE=IOT
4139
4140 021260 $SCOPE:
4141 021260 012504 MOV (R5)+, R4 ;:SAVE MINIMUM BLOCK MASK NEXT TEST.
4142 021262 010516 MOV R5, (SP) ;:PUT RETURN PC ONTO STACK. SIMULATE JSR PC.
4143 021264 032777 040000 157646 1$: BIT #BIT14, $SWR ;:LOOP ON PRESENT TEST?
4144 021272 001117 BNE $OVER ;:YES IF SW14=1
4145 ;:*****START OF CODE FOR THE XOR TESTER*****
4146 021274 000416 $XTSTR: BR 6$ ;:IF RUNNING ON THE "XOR" TESTER CHANGE
4147 ;:THIS INSTRUCTION TO A "NOP" (NOP=240)
4148 021276 013746 000004 MOV @#ERRVEC, -(SP) ;:SAVE THE CONTENTS OF THE ERROR VECTOR
4149 021302 012737 021322 000004 MOV #5$, @#ERRVEC ;:SET FOR TIMEOUT
4150 021310 005737 177060 TST @#177060 ;:TIME OUT ON XOR?
4151 021314 012637 000004 MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
4152 021320 000466 BR $SVLAD ;:GO TO THE NEXT TEST
4153 021322 022626 5$: CMP (SP)+, (SP)+ ;:CLEAR THE STACK AFTER A TIME OUT
4154 021324 012637 000004 MOV (SP)+, @#ERRVEC ;:RESTORE THE ERROR VECTOR
4155 021330 000426 BR 7$ ;:LOOP ON THE PRESENT TEST
4156 021332 6$: ;:*****END OF CODE FOR THE XOR TESTER*****
4157 021332 032777 000400 157600 BIT #BIT08, $SWR ;:LOOP ON SPEC. TEST?
4158 021340 001407 BEQ 2$ ;:BR IF NO
4159 021342 017746 157572 MOV @SWR, -(SP) ;:SET DESIRED TEST NUM. FROM SWR
4160 021346 042716 000340 BIC #5$SWRMK, (SP) ;:STRIP AWAY UNDESIRED BITS
4161 021352 122667 157524 CMPB (SP)+, $TSTNM ;:ON THE RIGHT TEST?
4162 021356 001465 BEQ $OVER ;:BR IF YES
4163 021360 105767 157517 2$: TSTB $ERFLG ;:HAS AN ERROR OCCURRED?
4164 021364 001421 BEQ 3$ ;:BR IF NO
4165 021366 126767 157523 157507 CMPB $ERMAX, $ERFLG ;:MAX. ERRORS FOR THIS TEST OCCURRED?
4166 021374 101015 BHI 3$ ;:BR IF NO
4167 021376 032777 001000 157534 BIT #BIT09, $SWR ;:LOOP ON ERROR?

```

4168	021404	001404				BEQ	4\$::BR IF NO
4169	021406	016767	157476	157472	7\$:	MOV	\$LPERR,\$LPADR	::SET LOOP ADDRESS TO LAST SCOPE
4170	021414	000446				BR	\$OVER	
4171	021416	105067	157461		4\$:	CLRB	\$ERFLG	::ZERO THE ERROR FLAG
4172	021422	005067	157542			CLR	\$TIMES	::CLEAR THE NUMBER OF ITERATIONS TO MAKE
4173	021426	000415				BR	1\$::ESCAPE TO THE NEXT TEST
4174	021430	032777	004000	157502	3\$:	BIT	#BIT11,\$SWR	::INHIBIT ITERATIONS?
4175	021436	001011				BNE	1\$::BR IF YES
4176	021440	005767	157546			TST	\$PASS	::IF FIRST PASS OF PROGRAM
4177	021444	001406				BEQ	1\$::INHIBIT ITERATIONS
4178	021446	005267	157432			INC	\$ICNT	::INCREMENT ITERATION COUNT
4179	021452	026767	157512	157424		CMP	\$TIMES,\$ICNT	::CHECK THE NUMBER OF ITERATIONS MADE
4180	021460	002024				BGE	\$OVER	::BR IF MORE ITERATION REQUIRED
4181	021462	012767	000001	157414	1\$:	MOV	#1,\$ICNT	::REINITIALIZE THE ITERATION COUNTER
4182	021470	016767	000552	157472		MC,	\$MXCNT,\$TIMES	::SET NUMBER OF ITERATIONS TO DO
4183	021476	105267	157400			INCB	\$STNM	::COUNT TEST NUMBERS
4184	021502	116767	157374	157500		MOVB	\$STNM,\$TESTN	::SET TEST NUMBER IN APT MAILBOX
4185	021510	011667	157372			MOV	(SP),\$LPADR	::SAVE SCOPE LOOP ADDRESS
4186	021514	011667	157370			MOV	(SP),\$LPERR	::SAVE ERROR LOOP ADDRESS
4187	021520	005067	157446			CLR	\$ESCAPE	::CLEAR THE ESCAPE FROM ERROR ADDRESS
4188	021524	112767	000001	157363		MOVB	#1,\$ERMAX	::ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4189	021532	016777	157344	157402		\$OVER: MOV	\$STNM,\$DISPLAY	::DISPLAY TEST NUMBER
4190	021540	016716	157342			MOV	\$LPADR,(SP)	::FUDGE RETURN ADDRESS
4191	021544	020515				INSERT: CMP	R5,(SP)	::CHECK FOR LOOP ON TEST.
4192	021546	001402				BEQ	1\$::BR IF START NEXT TEST.
4193	021550	000167	000470			JMP	ENDINS	::JMP IF LOOP ON LAST TEST.
4194	021554	012767	037777	160024	1\$:	MOV	#37777,\$BLKMSK	::SET BK BOUNDARY MASK.
4195	021562	005767	157424			TST	\$PASS	::CHECK FOR PASS 0.
4196	021566	001404				BEQ	2\$::BR IF PASS 0
4197	021570	126727	157306	000023		CMPB	\$STNM,#23	::CHECK IF IN SECTION 3.
4198	021576	103002				BHIS	3\$::BR IF IN SECTION 3.
4199	021600	006267	160002		2\$:	ASR	\$BLKMSK	::RESET BOUNDARY TO 4K.
4200	021604	016767	157752	157752	3\$:	MOV	\$FSTADR,\$TMPFAD	::GET FIRST ADDRESS.
4201	021612	005767	156762			TST	\$RELOC	::CHECK IF PRG RELOCATED.
4202	021616	001430				BEQ	4\$::BR IF NOT RELOCATED.
4203	021620	032777	000040	157312		BIT	#SW05,\$SWR	::CHECK IF LOC 0-776 TO BE PROTECTED.
4204	021626	001424				BEQ	4\$::BR IF SW NOT SET.
4205	021630	026727	157730	001000		CMP	\$TMPFAD,#1000	::CHECK IF NOT BEING TESTED.
4206	021636	103020				BHIS	4\$::BR IF ALREADY PROTECTED.
4207	021640	012767	001000	157716		MOV	#1000,\$TMPFAD	::RESET FIRST ADDRESS.
4208	021646	052767	000001	157714		BIS	#BIT0,\$FADMAP	::SET FLAG IN FIRST BANK.
4209	021654	026727	157714	001000		CMP	\$LSTADR,#1000	::CHECK IF GONE PAST LAST ADR.
4210	021662	101006				BHI	4\$::BR IF ENOUGH MEMORY.
4211	021664	004567	001706			JSR	R5,\$SPRINT	::GO PRINT OUT THE FOLLOWING MESSAGE.
4212	021670	027020				.WORD	NOMTST	::ADDRESS OF MESSAGE TO BE TYPED
4213								::"NO MEMORY TESTED"
4214	021672	016716	157744			MOV	\$TST32,(SP)	::ADJUST RETURN ADR FOR ABORT.
4215	021676	000207				RTS	PC	::ABORT.
4216	021700	016767	157670	157670	4\$:	MOV	\$LSTADR,\$TMPLAD	::GET LAST ADDRESS.
4217	021706	016767	157622	157614		MOV	\$SAVTST,\$TSTMAP	::GET TEST MAP, LO 64K.
4218	021714	016767	157616	157610		MOV	\$SAVTST+2,\$TSTMAP+2	::HI 64K.
4219	021722	046767	156654	157600		BIC	\$PRGMAP,\$TSTMAP	::DON'T TEST OVER THE PROGRAM.
4220	021730	046767	156650	157574		BIC	\$PRGMAP+2,\$TSTMAP+2	
4221	021736	005767	157250			TST	\$PASS	::CHECK FOR FIRST PASS
4222	021742	001011				BNE	10\$::BR IF NOT FIRST PASS.
4223	021744	032767	000003	157556		BIT	#3,\$TSTMAP	::CHECK IF FIRST TWO BANKS AVAILABLE.

```

4224 021752 001405      BEQ      10$      ;NOT TESTING FIRST 2 BANKS.
4225 021754 042767 177774 157546      BIC      #177774,TSTMAP ;CLR ALL BUT FIRST 2 BANKS.
4226 021762 005067 157544      CLR      TSTMAP+2
4227 021766 005704      10$:    TST      R4      ;CHECK FOR A MINIMUM BLOCK SIZE.
4229 021770 001503      BEQ      20$      ;BR IF NO MIN BLOCK SIZE.
4229 021772 030467 157566      BIT      R4,      TMPFAD ;CHECK IF FIRST ADR ON BLOCK BOUNDARY.
4230 021776 001416      BEQ      11$      ;BR IF FIRST ADR ON BLOCK BOUNDARY.
4231 022000 050467 157560      BIS      R4,      TMPFAD ;ADJUST FIRST ADR TO END OF BLOCK.
4232 022004 005267 157554      INC      TMPFAD      ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4233 022010 032767 017777 157546      BIT      #MASK4K,TMPFAD ;CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4234 022016 001006      BNE      11$      ;BR IF NOT ON 4K BOUNDARY.
4235 022020 046767 157544 157502      BIC      FADMAP, TSTMAP ;DON'T TEST FIRST BANK.
4236 022026 046767 157540 157476      BIC      FADMAP+2,TSTMAP+2
4237 022034 030467 157536      11$:    BIT      R4,      TEMPLAD ;CHECK IF LAST ADR ON BLOCK BOUNDARY.
4238 022040 001414      BEQ      12$      ;BR IF ON BLOCK BOUNDARY.
4239 022042 040467 157530      BIC      R4,      TEMPLAD ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4240 022046 032767 017777 157522      BIT      #MASK4K,TEMPLAD ;CHECK IF ADJUSTED TO 4K BOUNDARY.
4241 022054 001006      BNE      12$      ;BR IF NOT ON 4K BOUNDARY.
4242 022056 046767 157520 157444      BIC      LADMAP, TSTM, ;SKIP TESTING LAST BANK.
4243 022064 046767 157514 157440      BIC      LADMAP+2,TSTM+2
4244 022072 036767 157472 157502      12$:    BIT      FADMAP, LADMAP ;CHECK IF FIRST AND LAST IN SAME BANK.
4245 022100 001004      BNE      13$      ;BR IF IN SAME BANK.
4246 022102 036767 157464 157474      BIT      FADMAP+2,LADMAP+2 ;...UPPER 64K.
4247 022110 001404      BEQ      14$      ;BR IF FIRST AND LAST NOT SAME BANK.
4248 022112 026767 157460 157444      13$:    CMP      TEMPLAD, TMPFAD ;CHECK IF ANY MEMORY LEFT.
4249 022120 101406      BLOS     15$      ;BR IF NO MEMORY TO TEST.
4250 022122 005767 157402      14$:    TST      TSTMAP      ;CHECK IF ANY BANKS LEFT TO TEST!!
4251 022126 001017      BNE      16$      ;BR IF TEST MAP NOT EMPTY.
4252 022130 005767 157376      TST      TSTMAP+2    ;CHECK FOR ANY BANKS.
4253 022134 001014      BNE      16$      ;BR IF TEST MAP NOT EMPTY.
4254 022136      15$:
4255 022136 004567 001434      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4256 022142 027044      .WORD    SKPMES      ;ADDRESS OF MESSAGE TO BE TYPED
4257      ;"SKIPPING TEST #"
4258 022144 005046      CLR      -(SP)      ;CLEAR THE WORD ON THE STACK.
4259 022146 116716 156730      MOV      $TSTNM, (SP) ;PUT THE DATA ON THE STACK.
4260      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4261      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4262 022152 013746 177776      MOV      @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4263 022156 004767 002566      JSR      PC,      $TYPOS ;GO TO THE SUBROUTINE
4264 022162 003      .BYTE    3          ;TYPE 3 DIGITS.
4265 022163 001      .BYTE    1          ;TYPE LEADING ZEROS.
4266 022164 000427      BR      ENDINS     ;RETURN TO SKIP TEST.
4267 022166 062716 000004 156706      16$:    ADD      #4,      (SP) ;SKIP THE SKIP ON RETURN.
4268 022172 062767 000004 157360      ADD      #4,      $LPADR ;ADJUST THE LOOP ADR PAST THE SKIP.
4269 022200 012767 017777 157350      20$:    MOV      #MASK4K,FADMSK ;GET 4K MASK.
4270 022206 016705 157352      MOV      TMPFAD, R5 ;GET FIRST ADR.
4271 022212 040567 157350      21$:    BIC      R5,      FADMSK ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4272 022216 006305      ASL      R5      ;MOVE LOWEST BIT UP ONE.
4273 022220 001374      BNE      21$      ;LOOP UNTIL OVERFLOW.
4274 022222 012767 017777 157350      MOV      #MASK4K,LADMSK ;SET MASK BITS
4275 022230 016705 157342      MOV      TEMPLAD, R5 ;GET LAST ADR.
4276 022234 040567 157340      22$:    BIC      R5,      LADMSK ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4277 022240 006305      ASL      R5      ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4278 022242 001374      BNE      22$      ;LOOP UNTIL OVERFLOW.
4279 022244 000207      ENDINS: RTS      PC ;EXIT SCOPE ROUTINE BACK TO TEST.

```

```

4280 022246 000004 $MXCNT: 4 ;;MAX. NUMBER OF ITERATIONS
4281 .SBTTL ERROR HANDLER ROUTINE
4282
4283 ;;*****
4284 ;;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4285 ;;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4286 ;;AND GO TO $ERRTYP ON ERROR
4287 ;;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4288 ;;*SW15=1 HALT ON ERROR
4289 ;;*SW13=1 INHIBIT ERROR TYPEOUTS
4290 ;;*SW10=1 BELL ON ERROR
4291 ;;*SW09=1 LOOP ON ERROR
4292 ;;*CALL
4293 ;* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4294
4295 $ERROR:
4296 022250 062716 000002 ADD #2, (SP) ;;ADJUST POINTER PAST CODE WORD.
4297 022254 105267 156623 7$: INCB $ERFLG ;;SET THE ERROR FLAG
4298 022260 001775 BEQ 7$ ;;DON'T LET THE FLAG GO TO ZERO
4299 022262 016777 156614 156652 MOV $TSTNM, $DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4300 022270 032777 002000 156642 BIT #BIT0, $SWR ;;BELL ON ERROR?
4301 022276 001403 BEQ 1$ ;;NO - SKIP
4302 022300 004567 001272 JSR RS, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4303 022304 001174 .WORD $BELL ;;ADDRESS OF MESSAGE TO BE TYPED
4304 022306 005267 156600 1$: INC $ERTTL ;;COUNT THE NUMBER OF ERRORS
4305 022312 011667 156600 MOV (SP), $ERPPC ;;GET ADDRESS OF ERROR INSTRUCTION
4306 022316 162767 000002 156572 SUB #2, $ERPPC
4307 022324 117767 156566 156562 MOVB $ERPPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CCDE
4308 022332 032777 020000 156600 BIT #BIT13, $SWR ;;SKIP TYPEOUT IF SET
4309 022340 001005 BNE 20$ ;;SKIP TYPEOUTS
4310 022342 004767 000106 JSR PC, $ERRTYP ;;GO TO USER ERROR ROUTINE
4311 022346 004567 001224 JSR RS, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4312 022352 001201 .WORD $CALF ;;ADDRESS OF MESSAGE TO BE TYPED
4313 022354
4314 022354 122767 000001 156642 20$: CMPB #APTENV, $ENV ;;RUNNING IN APT MODE
4315 022362 001007 BNE 2$ ;;NO SKIP APT ERROR REPORT
4316 022364 116767 156524 000004 MOVB $ITEMB, 21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4317 022372 004767 001530 JSR PC, $ATY4 ;;REPORT FATAL ERROR TO APT
4318 022376 000 21$: .BYTE 0
4319 022377 000 .BYTE 0
4320 022400 000777 22$: BR 22$ ;;APT ERROR LOOP
4321 022402 005777 156532 2$: TST $SWR ;;HALT ON ERROR
4322 022406 100001 BPL 3$ ;;SKIP IF CONTINUE
4323 022410 000000 HALT ;;HALT ON ERROR!
4324 022412 032777 001000 156520 3$: BIT #BIT09, $SWR ;;LOOP ON ERROR SWITCH SET?
4325 022420 001402 BEQ 4$ ;;BR IF NO
4326 022422 016716 156462 MOV $LPERR, (SP) ;;FUDGE RETURN FOR LOOPING
4327 022426 005767 156540 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
4328 022432 001402 BEQ 5$ ;;BR IF NONE
4329 022434 016716 156532 MOV $ESCAPE, (SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4330 022440 5$:
4331 022440 022737 014760 000042 CMP #SENDAD, $#42 ;;ACT-11 AUTO-ACCEPT?
4332 022446 001001 BNE 6$ ;;BRANCH IF NO
4333 022450 000000 HALT ;;YES
4334 022452
4335 022452 000207 6$: RTS PC

```

```

4336 ;*****
4337
4338 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
4339
4340 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4341 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4342 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4343
4344 $ERRTYP:
4345 JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4346 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4347 MOV RO,-(SP) ;SAVE RO
4348 CLR RO ;PICKUP THE ITEM INDEX
4349 BISB $ITEMB,RO
4350 BNE 1$ ;IF ITEM NUMBER IS ZERO, JUST
4351 ;TYPE THE PC OF THE ERROR
4352 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
4353 ;ERROR ADDRESS
4354 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4355 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4356 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4357 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4358 BR 10$ ;GET OUT
4359 MOV $ERRPC, $VERPC ;SET UP VIRTUAL PC FOR TYPEOUT.
4360 SUB RELOCF, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
4361 DEC RO ;ADJUST THE INDEX SO THAT IT WILL
4362 ASL RO ; WORK FOR THE ERROR TABLE
4363 ASL RO
4364 ASL RO
4365 ADD .ERRTB, RO ;FORM TABLE POINTER
4366 MOV (RO)+, 2$ ;PICKUP "ERROR MESSAGE" POINTER
4367 BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
4368 JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4369 .WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
4370 JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4371 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4372 MOV (RO)+, 4$ ;PICKUP "DATA HEADER" POINTER
4373 BEQ 5$ ;SKIP TYPEOUT IF 0
4374 JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4375 .WORD 0 ;"DATA HEADER" POINTER GOES HERE
4376 JSR RS, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4377 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4378 MOV R1,-(SP) ;SAVE R1
4379 MOV (RO)+, R1 ;PICKUP "DATA TABLE" POINTER
4380 BEQ 9$ ;BR IF NO DATA TO BE TYPED
4381 ADD RELOCF, R1 ;ADJUST POINTER
4382 MOV (RO)+, RO ;PICKUP "DATA FORMAT" POINTER
4383 ADD RELOCF, RO ;ADJUST POINTER.
4384 TSTB (RO)+ ;CHECK THE FORMAT
4385 BNE 7$ ;BR IF NOT 16-BIT OCTAL
4386 MOV @#(R1)+, -(SP) ;;SAVE @#(R1)+ FOR TYPEOUT
4387 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4388 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4389 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4390 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4391 BR 8$
    
```

ERROR MESSAGE TYPEOUT ROUTINE

```

4392 022646 100406 3$: BMI 17$ ;BRANCH IF NOT DECIMAL
4393 022650 013146 MOV 2(R1)+,-(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4394 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
4395 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
4396 022652 013746 177776 MOV 2#PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4397 022656 004767 001634 JSR PC,$TYPDS ;GO TO THE SUBROUTINE
4398 022662 000417 BR 8$ ;SKIP
4399 022664 122760 177777 177777 17$: CMPB #-1,-1(R0) ;CHECK FOR 18-BIT ADDRESS FORMAT.
4400 022672 001004 BNE 18$ ;BR IF NOT 18-BIT ADDRESS FORMAT.
4401 022674 013146 MOV 2(R1)+,-(SP) ;PUT THE DATA ON THE STACK.
4402 022676 004767 002334 JSR PC,$TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4403 022702 000407 BR 8$ ;SKIP
4404 022704 18$: CLR -(SP) ;CLEAR THE WORD ON THE STACK.
4405 022704 005046 MOVB 2(R1)+,(SP) ;PUT THE DATA ON THE STACK.
4406 022706 113116 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4407 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
4408 MOV 2#PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4409 022710 013746 177776 JSR PC,$TYPOS ;GO TO THE SUBROUTINE
4410 022714 004767 002030 .BYTE 3 ;TYPE 3 DIGITS.
4411 022720 003 .BYTE 1 ;TYPE LEADING ZEROS.
4412 022721 001 9$: TST (R1) ;IS THERE ANOTHER NUMBER?
4413 022722 005711 BEQ 9$ ;BR IF NO
4414 022724 001404 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4415 022726 004567 000644 .WORD 11$ ;ADDRESS OF MESSAGE TO BE TYPED
4416 022732 022752 BR 6$ ;LOOP
4417 022734 000734 9$: MOV (SP)+,R1 ;RESTORE R1
4418 4419 022736 012601 10$: MOV (SP)+,R0 ;RESTORE R0
4420 022740 012600 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4421 022742 004567 000630 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4422 022746 001201 RTS PC ;RETURN
4423 022750 000207 11$: .ASCIZ / / ;TAB CHARACTER.
4424 022752 000011 .EVEN
4425 .SBTTL TTY INPUT ROUTINE
4426 ;*****
4427 .ENABL LSB
4428 ;*****
4429 .DSABL LSB
4430 ;*****
4431 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4432 ;*CALL:
4433 ;* RDCHR ;INPUT A SINGLE CHARACTER FROM THE TTY
4434 ;* RETURN HERE ;CHARACTER IS ON THE STACK
4435 ;* ;WITH PARITY BIT STRIPPED OFF
4436 ;*
4437 ;*
4438 ;*
4439 ;*
4440 ;*
4441 ;*
4442 022754 011646 $RDCHR: MOV (SP)-,(SP) ;PUSH DOWN THE PC
4443 022756 016666 000004 000002 MOV 4(SP),2(SP) ;SAVE THE PS
4444 022764 105777 156154 1$: TSTB 2$TKS ;WAIT FOR
4445 022770 100375 BPL 1$ ;A CHARACTER
4446 022772 117766 156150 000004 MOVB 2$TKB,4(SP) ;READ THE TTY
4447 023000 042766 177600 000004 BIC #177,4(SP) ;GET RID OF JUNK IF ANY

```

```

4448 023006 026627 000004 000023      CMP      4(SP),#23      ;; IS IT A CONTROL-5?
4449 023014 001013                    BNE      3$           ;; BRANCH IF NO
4450 023016 105777 156122      2$:      TSTB     2$TKS      ;; WAIT FOR A CHARACTER
4451 023022 100375                    BPL      2$           ;; LOOP UNTIL ITS THERE
4452 023024 117746 156116      MOVB     2$TKB,-(SP)   ;; GET CHARACTER
4453 023030 042716 177600      BIC      #177,(SP)    ;; MAKE IT 7-BIT ASCII
4454 023034 022627 000021      CMP      (SP)+,#21    ;; IS IT A CONTROL-Q?
4455 023040 001366                    BNE      2$           ;; IF NOT DISCARD IT
4456 023042 030750                    BR       1$           ;; YES, RESUME
4457 023044 026627 000004 000140      3$:      CMP      4(SP),#140  ;; IS IT UPPER CASE?
4458 023052 002407                    BLT      4$           ;; BRANCH IF YES
4459 023054 026627 000004 000175      CMP      4(SP),#175  ;; IS IT A SPECIAL CHAR?
4460 023062 003003                    BGT      4$           ;; BRANCH IF YES
4461 023064 042756 000040 000004      BIC      #40,4(SP)   ;; MAKE IT UPPER CASE
4462 023072 000002      4$:      RTI                    ;; GO BACK TO USER
4463                                     ;; *****
4464                                     ;; *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4465                                     ;; *CALL:
4466                                     ;; *
4467                                     ;; *      RDLIN                    ;; INPUT A STRING FROM THE TTY
4468                                     ;; *      RETURN HERE              ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4469                                     ;; *                               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
4470 023074 010346      $RDLIN: MOV      R3,-(SP)      ;; SAVE R3
4471 023076 005046                    CLR      -(SP)        ;; CLEAR THE RUBOUT KEY
4472 023100 012703 023360      1$:      MOV      #$TTYIN,R3  ;; GET ADDRESS
4473 023104 022703 023370      2$:      CMP      #$TTYIN+8.,R3 ;; BUFFER FULL?
4474 023110 101467                    BLOS     4$           ;; BR IF YES
4475                                     ;; * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4476                                     ;; * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4477 023112 013746 177776      MOV      2$PSW,-(SP)  ;; PUT THE PROCESSOR STATUS ON THE STACK
4478 023116 004767 177632      JSR      PC,$RDCHR   ;; GO TO THE SUBROUTINE
4479 023122 112613                    MOVB     (SP)+,(R3)   ;; GET CHARACTER
4480 023124 122713 000177      10$:     CMPB     #177,(R3)   ;; IS IT A RUBOUT
4481 023130 001024                    BNE      5$           ;; BR IF NO
4482 023132 005716                    TST      (SP)        ;; IS THIS THE FIRST RUBOUT?
4483 023134 001010                    BNE      6$           ;; BR IF NO
4484 023136 112767 000134 000212      MOVB     #' \,9$     ;; TYPE A BACK SLASH
4485 023144 004567 000426      JSR      R5,$PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4486 023150 023356                    .WORD   9$           ;; ADDRESS OF MESSAGE TO BE TYPED
4487 023152 012716 177777      MOV      #-1,(SP)    ;; SET THE RUBOUT KEY
4488 023156 005303      6$:      DEC      R3          ;; BACKUP BY ONE
4489 023160 020327 023360      CMP      R3,$TTYIN  ;; STACK EMPTY?
4490 023164 103441                    BLO      4$           ;; BR IF YES
4491 023166 111367 000164      MOVB     (R3),9$     ;; SETUP TO TYPEOUT THE DELETED CHAR.
4492 023172 004567 000400      JSR      R5,$PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4493 023176 023356                    .WORD   9$           ;; ADDRESS OF MESSAGE TO BE TYPED
4494 023200 000741                    BR       2$           ;; GO READ ANOTHER CHAR.
4495 023202 005716      5$:      TST      (SP)        ;; RUBOUT KEY SET?
4496 023204 001407                    BEQ      7$           ;; BR IF NO
4497 023206 112767 000134 000142      MOVB     #' \,9$     ;; TYPE A BACK SLASH
4498 023214 004567 000356      JSR      R5,$PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4499 023220 023356                    .WORD   9$           ;; ADDRESS OF MESSAGE TO BE TYPED
4500 023222 005016                    CLR      (SP)        ;; CLEAR THE RUBOUT KEY
4501 023224 122713 000025      7$:      CMPB     #25,(R3)   ;; IS CHARACTER A CTRL U?
4502 023230 001004                    BNE      8$           ;; BR IF NO
4503 023232 004567 000340      JSR      R5,$PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.

```


4504	023236	023370				.WORD	\$CNTLU		;; ADDRESS OF MESSAGE TO BE TYPED
4505	023240	000717				BR	1\$;; GO START OVER
4506	023242	122713	000022		8\$:	CMPB	#22,(R3)		;; IS CHARACTER A "r"?
4507	023246	001014				BNE	3\$;; BRANCH IF NO
4509	023250	105013				CLRB	(R3)		;; CLEAR THE CHARACTER
4509	023252	004567	000320			JSR	R5, \$PRINT		;; GO PRINT OUT THE FOLLOWING MESSAGE.
4510	023256	001201				.WORD	\$CRLF		;; ADDRESS OF MESSAGE TO BE TYPED
4511	023260	004567	000312			JSR	.	\$PRINT	;; GO PRINT OUT THE FOLLOWING MESSAGE.
4512	023264	023360				.WORD	\$TTYIN		;; ADDRESS OF MESSAGE TO BE TYPED
4513	023266	000706				BR	2\$;; GO PICKUP ANOTHER CHARACTER
4514	023270				4\$:				
4515	023270	004567	000302			JSR	R5, \$PRINT		;; GO PRINT OUT THE FOLLOWING MESSAGE.
4516	023274	001200				.WORD	\$QUES		;; ADDRESS OF MESSAGE TO BE TYPED
4517	023276	000700				BR	1\$;; CLEAR THE BUFFER AND LOOP
4518	023300	111367	000052		3\$:	MOVB	(R3),9\$;; ECHO THE CHARACTER
4519	023304	004567	000266			JSR	R5, \$PRINT		;; GO PRINT OUT THE FOLLOWING MESSAGE.
4520	023310	023356				.WORD	9\$;; ADDRESS OF MESSAGE TO BE TYPED
4521	023312	122723	000015			CMPB	#15,(R3)+		;; CHECK FOR RETURN
4522	023316	001272				BNE	2\$;; LOOP IF NOT RETURN
4523	023320	105063	177777			CLRB	-1(R3)		;; CLEAR RETURN (THE 15)
4524	023324	004567	000246			JSR	R5, \$PRINT		;; GO PRINT OUT THE FOLLOWING MESSAGE.
4525	023330	001202				.WORD	\$LF		;; ADDRESS OF MESSAGE TO BE TYPED
4526	023332	005726				TST	(SP)+		;; CLEAN RUBOUT KEY FROM THE STACK
4527	023334	012603				MOV	(SP)+,R3		;; RESTORE R3
4528	023336	011646				MOV	(SP),-(SP)		;; ADJUST THE STACK AND PUT ADDRESS OF THE
4529	023340	016666	000004	000002		MOV	4(SP),2(SP)		;; FIRST ASCII CHARACTER ON IT
4530	023346	012766	023360	000004		MOV	#\$TTYIN,4(SP)		
4531	023354	000002				RTI			;; RETURN
4532	023356	000			9\$:	.BYTE	0		;; STORAGE FOR ASCII CHAR. TO TYPE
4533	023357	000				.BYTE	0		;; TERMINATOR
4534	023360	000010			\$TTYIN:	.BLKB	8.		;; RESERVE 8 BYTES FOR TTY INPUT
4535	023370	052536	005015	000	\$CNTLU:	.ASCIZ	/r/<15><12>		;; CONTROL "U"
4536	023375	136	006507	000012	\$CNTLG:	.ASCIZ	/g/<15><12>		;; CONTROL "G"
4537	023402	005015	053523	020122	\$MSWR:	.ASCIZ	<15><12>/SWR = /		
4538	023410	020075	000						
4539	023413	040	047040	053505	\$MNEW:	.ASCIZ	/ NEW = /		
4540	023420	036440	000040						
4541						.SBTTL	READ AN OCTAL NUMBER FROM THE TTY		
4542									
4543						;;	*****		
4544						;;	*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND		
4545						;;	*CHANGE IT TO BINARY.		
4546						;;	*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL		
4547						;;	*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED		
4548						;;	*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST		
4549						;;	*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.		
4550						;;	*CALL:		
4551						;;	* RDOCT		;; READ AN OCTAL NUMBER
4552						;;	* RETURN HERE		;; LOW ORDER BITS ARE ON TOP OF THE STACK
4553						;;	*		;; HIGH ORDER BITS ARE IN \$HDOCT
4554									
4555	023424	011646			\$RDOCT:	MOV	(SP),-(SP)		;; PROVIDE SPACE FOR THE
4556	023426	016666	000004	000002		MOV	4(SP),2(SP)		;; INPUT NUMBER
4557	023434	010046				MOV	R0,-(SP)		;; PUSH R0 ON STACK
4558	023436	010146				MOV	R1,-(SP)		;; PUSH R1 ON STACK
4559	023440	010246				MOV	R2,-(SP)		;; PUSH R2 ON STACK

F14

```

4560 023442 1$:
4561 : * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
4562 : * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4563 023442 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4564 023446 004767 177422 JSR PC, $RDLIN ;GO TO THE SUBROUTINE
4565 023452 012600 MOV (SP)+,R0 ;;GET ADDRESS OF 1ST CHARACTER
4566 023454 010067 000102 MOV R0,$$ ;;AND SAVE IT
4567 023460 005001 CLR R1 ;;CLEAR DATA WORD
4568 023462 005002 CLR R2
4569 023464 112046 2$: MOVB (R0)+,-(SP) ;;PICKUP THIS CHARACTER
4570 023466 001420 BEQ 3$ ;;IF ZERO GET OUT
4571 023470 122716 000060 CMPB #'0,(SP) ;;MAKE SURE THIS CHARACTER
4572 023474 003026 BGT 4$ ;;IS AN OCTAL DIGIT
4573 023476 122716 000067 CMPB #'7,(SP)
4574 023502 002423 BLT 4$
4575 023504 006301 ASL R1 ;;*2
4576 023506 006102 ROL R2 ;;*4
4577 023510 006301 ASL R1 ;;*4
4578 023512 006102 ROL R2 ;;*8
4579 023514 006301 ASL R1 ;;*8
4580 023516 006102 ROL R2
4581 023520 042716 177770 BIC #'C7,(SP) ;;STRIP THE ASCII JUNK
4582 023524 062601 ADD (SP)+,R1 ;;ADD IN THIS DIGIT
4583 023526 000756 BR 2$ ;;LOOP
4584 023530 005726 3$: TST (SP)+ ;;CLEAN TERMINATOR FROM STACK
4585 023532 010166 000012 MOV R1,12(SP) ;;SAVE THE RESULT
4586 023536 010267 000032 MOV R2,$HIOCT
4587 023542 012602 MOV (SP)+,R2 ;;POP STACK INTO R2
4588 023544 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4589 023546 012600 MOV (SP)+,R0 ;;POP STACK INTO R0
4590 023550 000002 RTI ;;RETURN
4591 023552 005726 4$: TST (SP)+ ;;CLEAN PARTIAL FROM STACK
4592 023554 105010 CLRB (R0) ;;SET A TERMINATOR
4593 023556 004567 000014 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4594 023562 000000 5$: .WORD 0
4595 023564 004567 000006 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4596 023570 001200 .WORD $QUES ;ADDRESS OF MESSAGE TO BE TYPED
4597 023572 000723 BR 1$ ;;TRY AGAIN
4598 023574 000000 $HIOCT: .WORD 0 ;;HIGH ORDER BITS GO HERE
4599
4600 :*****
4601 : * SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
4602 : * CALL: JSR R5, $PRINT
4603 : * <MESSAGE VIRTUAL ADDRESS>
4604 :*****
4605 023576 012567 000016 $PRINT: MOV (R5)+, 1$ ;GET THE MESSAGE VIRTUAL ADDRESS.
4606 023602 066767 154772 000010 ADD RELOC#, 1$ ;MAKE IT PHYSICAL.
4607
4608 : * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4609 : * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4610 023610 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4611 023614 004767 000004 JSR PC, $TYPE ;GO TO THE SUBROUTINE
4612 023620 000000 1$: .WORD 0 ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
4613 RTS R5 ;RETURN.
4614
4615 .SBTTL TYPE ROUTINE
  
```

```

4616 .....
4617 *ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4618 *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4619 *NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4620 *NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4621 *NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4622 *
4623 *CALL:
4624 *1) USING A TRAP INSTRUCTION
4625 * TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4626 *OR
4627 * TYPE
4628 * MESADR
4629 *
4630
4631 023624 105767 155327 $TYPE: TSTB $TPFLG ;: IS THERE A TERMINAL?
4632 023630 100002 BPL 1$ ;: BR IF YES
4633 023632 000000 HALT ;: HALT HERE IF NO TERMINAL
4634 023634 000430 BR 3$ ;: LEAVE
4635 023636 010046 1$: MOV RO,-(SP) ;: SAVE RO
4636 023640 017600 000002 MOV 22(SP),RO ;: GET ADDRESS OF ASCIZ STRING
4637 023644 122767 000001 155352 CMPB #APTENV,$ENV ;: RUNNING IN APT MODE
4638 023652 001011 BNE 62$ ;: NO, GO CHECK FOR APT CONSOLE
4639 023654 132767 000100 155343 BITB #APTPOOL,$ENVM ;: SPOOL MESSAGE TO APT
4640 023662 001405 BEQ 62$ ;: NO, GO CHECK FOR CONSOLE
4641 023664 010067 000004 MOV RO,61$ ;: SETUP MESSAGE ADDRESS FOR APT
4642 023670 004767 000222 JSR PC,$ATY3 ;: SPOOL MESSAGE TO APT
4643 023674 000000 61$: .WORD 0 ;: MESSAGE ADDRESS
4644 023676 132767 000040 155321 62$: BITB #APTCSUP,$ENVM ;: APT CONSOLE SUPPRESSED
4645 023704 001003 BNE 60$ ;: YES, SKIP TYPE OUT
4646 023706 112046 2$: MOVB (RO)+,-(SP) ;: PUSH CHARACTER TO BE TYPED ONTO STACK
4647 023710 001005 BNE 4$ ;: BR IF IT ISN'T THE TERMINATOR
4648 023712 005726 TST (SP)+ ;: IF TERMINATOR POP IT OFF THE STACK
4649 023714 012600 60$: MOV (SP)+,RO ;: RESTORE RO
4650 023716 062716 000002 3$: ADD #2,(SP) ;: ADJUST RETURN PC
4651 023722 000002 RTI ;: RETURN
4652 023724 122716 000011 4$: CMPB #HT,(SP) ;: BRANCH IF <HT>
4653 023730 001431 BEQ 9$ ;:
4654 023732 122716 000200 CMPB #CRLF,(SP) ;: BRANCH IF NOT <CRLF>
4655 023736 001007 BNE 5$ ;:
4656 023740 005726 TST (SP)+ ;: POP <CR><LF> EQUIV
4657 023742 004567 177630 JSR RS,$PRINT ;: GO PRINT OUT THE FOLLOWING MESSAGE.
4658 023746 001201 $CRLF
4659 023750 105067 000130 CLRB $CHARCNT ;: CLEAR CHARACTER COUNT
4660 023754 000754 BR 2$ ;: GET NEXT CHARACTER
4661 023756 004767 000056 5$: JSR PC,$TYPEC ;: GO TYPE THIS CHARACTER
4662 023762 126726 155170 5$: CMPB $FILLC,(SP)+ ;: IS IT TIME FOR FILLER CHARS.?
4663 023766 001347 BNE 2$ ;: IF NO GO GET NEXT CHAR.
4664 023770 016746 155160 MOV $NULL,-(SP) ;: GET # OF FILLER CHARS. NEEDED
4665 ;: AND THE NULL CHAR.
4666 023774 105366 000001 7$: DECB 1(SP) ;: DOES A NULL NEED TO BE TYPED?
4667 024000 002770 BLT 6$ ;: BR IF NO--GO POP THE NULL OFF OF STACK
4668 024002 004767 000032 JSR PC,$TYPEC ;: GO TYPE A NULL
4669 024006 105367 000072 DECB $CHARCNT ;: DO NOT COUNT AS A COUNT
4670 024012 000770 BR 7$ ;: LOOP
4671

```

```

4672          :HORIZONTAL TAB PROCESSOR
4673
4674 024014 112716 000040 8$:   MOVB   #' (SP)      ;; REPLACE TAB WITH SPACE
4675 024020 004767 000014 9$:   JSR    PC,$TYPEC    ;; TYPE A SPACE
4676 024024 132767 000007 000052 BITB   #7,$CHARCNT    ;; BRANCH IF NOT AT
4677 024032 001372          BNE    9$          ;; TAB STOP
4678 024034 005726          TST    (SP)+        ;; POP SPACE OFF STACK
4679 024036 000723          BR     2$          ;; GET NEXT CHARACTER
4680 024040 105777 155104 $TYPEC: TSTB   @STPS     ;; WAIT UNTIL PRINTER IS READY
4681 024044 100375          BPL    $TYPEC
4682 024046 116677 000002 155076 MOVB   2(SP),@STPB    ;; LOAD CHAR TO BE TYPED INTO DATA REG.
4683 024054 122766 000015 000002 CMPB   #CR,2(SP)     ;; IS CHARACTER A CARRIAGE RETURN?
4684 024062 001003          BNE    1$          ;; BRANCH IF NO
4685 024064 105067 000014          CLRB   $CHARCNT    ;; YES--CLEAR CHARACTER COUNT
4686 024070 000406          BR     $TYPEX     ;; EXIT
4687 024072 122766 000012 000002 1$:   CMPB   #LF,2(SP)    ;; IS CHARACTER A LINE FEED?
4688 024100 001402          BEQ    $TYPEX     ;; BRANCH IF YES
4689 024102 105227          INCB   (PC)+        ;; COUNT THE CHARACTER
4690 024104 000000 $CHARCNT: .WORD   0    ;; CHARACTER COUNT STORAGE
4691 024106 000207 $TYPEX: RTS      PC
4692
4693          .SBTTL  APT COMMUNICATIONS ROUTINE
4694
4695          ;;*****
4696 024110 112767 000001 000376 $ATY1: MOVB   #1,$FFLG    ;; TO REPORT FATAL ERROR
4697 024116 112767 000001 000366 $ATY3: MOVB   #1,$MFLG    ;; TO TYPE A MESSAGE
4698 024124 000403          BR     $ATYC
4699 024126 112767 000001 000360 $ATY4: MOVB   #1,$FFLG    ;; TO ONLY REPORT FATAL ERROR
4700 024134          $ATYC:
4701 024134 010046          MOV    R0,-(SP)     ;; PUSH R0 ON STACK
4702 024136 010146          MOV    R1,-(SP)     ;; PUSH R1 ON STACK
4703 024140 105767 000346          TSTB   $MFLG        ;; SHOULD TYPE A MESSAGE?
4704 024144 001450          BEQ    5$          ;; IF NOT: BR
4705 024146 122767 000001 155050 CMPB   #APTENV,$ENV    ;; OPERATING UNDER APT?
4706 024154 001031          BNE    3$          ;; IF NOT: BR
4707 024156 132767 000100 155041 BITB   #APTPOOL,$ENVM  ;; SHOULD SPOOL MESSAGES?
4708 024164 001425          BEQ    3$          ;; IF NOT: BR
4709 024166 017600 000004          MOV    @4(SP),R0     ;; GET MESSAGE ADDR.
4710 024172 062766 000002 000004 ADD    #2,4(SP)        ;; BUMP RETURN ADDR.
4711 024200 005767 155000 1$:   TST    $MSGTYPE     ;; SEE IF DONE W/ LAST XMISSION?
4712 024204 001375          BNE    1$          ;; IF NOT: WAIT
4713 024206 010067 155006          MOV    R0,$MSGAD     ;; PUT ADDR IN MAILBOX
4714 024212 105720 2$:   TSTB   (R0)+        ;; FIND END OF MESSAGE
4715 024214 001376          BNE    2$
4716 024216 166700 154776          SUB    $MSGAD,R0     ;; SUB START OF MESSAGE
4717 024222 006200          ASR    R0           ;; GET MESSAGE LNTH IN WORDS
4718 024224 010067 154772          MOV    R0,$MSGGLT    ;; PUT LENGTH IN MAILBOX
4719 024230 012767 000004 154746 MOV    #4,$MSGTYPE    ;; TELL APT TO TAKE MSG.
4720 024236 000413          BR     5$
4721 024240 017667 000004 000016 3$:   MOV    @4(SP),4$     ;; PUT MSG ADDR IN JSR LINKAGE
4722 024246 062766 000002 000004 ADD    #2,4(SP)        ;; BUMP RETURN ADDRESS
4723 024254 016746 153516          MOV    177775,-(SP)  ;; PUSH 177776 ON STACK
4724 024260 004767 177340          JSR    PC,$TYPE     ;; CALL TYPE MACRO
4725 024264 000000 4$:   .WORD   0
4726 024266 5$:   TSTB   $LFLG        ;; SHOULD LOG AN ERROR?
4727 024266 105767 000221

```

```

4728 024272 001422 BEQ 10$ ;;IF NOT: BR
4729 024274 017600 MOV 24(SP),RO ;;GET ERROR #
4730 024300 062766 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4731 024306 012701 MOV #SASTAT,R1 ;;POINT TO TABLE START
4732 024312 005711 6$: TST (R1) ;;END OF TABLE?
4733 024314 100404 BMI 8$ ;;IF SO: BR
4734 024316 020021 CMP RO,(R1)+ ;;PROPER ENTRY?
4735 024320 001406 BEQ 9$ ;;IF SO: BR
4736 024322 005721 TST (R1)+ ;;MOVE PAST COUNTER WORD
4737 024324 000772 BR 6$ ;;KEEP LOOKING
4738 024326 026701 9$: CMP $APTR,R1 ;;TABLE FULL?
4739 024332 001402 BEQ 10$ ;;IF SO: BR -- NO MORE ROOM
4740 024334 010021 MOV RO,(R1)+ ;;SET UP NEW ENTRY
4741 024336 005211 9$: INC (R1) ;;BUMP ERROR COUNT
4742 024340 105767 10$: *STB $FFLG ;;SHOULD REPORT FATAL ERROR?
4743 024344 001416 BEQ 12$ ;;IF NOT: BR
4744 024346 005767 154652 TST $ENV ;;RUNNING UNDER APT?
4745 024352 001413 BEQ 12$ ;;IF NOT: BR
4746 024354 005767 11$: TST $MSGTYPE ;;FINISHED LAST MESSAGE?
4747 024360 001375 BNE 11$ ;;IF NOT: WAIT
4748 024362 017667 000004 154616 MOV 24(SP), $FATAL ;;GET ERROR #
4749 024370 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4750 024376 005267 154602 INC $MSGTYPE ;;TELL APT TO TAKE ERROR
4751 024402 105067 000106 12$: CLRB $FFLG ;;CLEAR FATAL FLAG
4752 024406 105067 000101 CLRB $LFLG ;;CLEAR LOG FLAG
4753 024412 105067 000074 CLRB $MFLG ;;CLEAR MESSAGE FLAG
4754 024416 012601 MOV (SP)+,R1 ;;POP STACK INTO R1
4755 024420 012600 MOV (SP)+,RO ;;POP STACK INTO RO
4756 024422 000207 RTS PC ;;RETURN
4757 024424 $ATY6:
4758 024424 010046 MOV RO, -(SP) ;;PUSH RO ON STACK
4759 024426 016700 155060 MOV $APTR,RO
4760 024432 162700 001344 SUB #SASTAT,RO ;;GET SIZE OF STAT TABLE
4761 024436 005767 154542 1$: TST $MSGTY ;;SEE IF DONE LAST COMMUNICATION
4762 024442 001375 BNE 1$ ;;IF NOT: WAIT
4763 024444 010067 154552 MOV RO,$MSGLG ;;SET MESSAGE LENGTH
4764 024450 012767 001344 154542 MOV #SASTAT,$MSGAD ;;SET MESSAGE ADDR.
4765 024456 012767 000002 154520 MOV #2,$MSGTY ;;TELL APT TO TAKE STATS.
4766 024464 012600 MOV (SP)+,RO ;;POP STACK INTO RO
4767 024466 000207 RTS PC ;;RETURN
4768 024470 $ATY7:
4769 024470 010046 MOV RO, -(SP) ;;PUSH RO ON STACK
4770 024472 012701 001344 MOV #SASTAT,R1 ;;GET START OF TABLE
4771 024476 005721 1$: TST (R1)+ ;;END OF TABLE?
4772 024500 100402 BMI 2$ ;;IF SO: BR
4773 024502 005021 CLR (R1)+ ;;CLEAR ERROR COUNT
4774 024504 000774 BR 1$ ;;KEEP CLEARING
4775 024506 2$:
4776 024506 012600 MOV (SP)+,RO ;;POP STACK INTO RO
4777 024510 000207 RTS PC ;;RETURN
4778 024512 000 $MFLG: .BYTE 0 ;;MESSG. FLAG
4779 024513 000 $LFLG: .BYTE 0 ;;LOG FLAG
4780 024514 000 $FFLG: .BYTE 0 ;;FATAL FLAG
4781 024516 .EVEN
4782 000200 APTSIZE=200
4783 000001 APTENV=001

```

```

4784      000100
4785      000040
4786
4787
4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799  024516
4800  024516  010046
4801  024520  010146
4802  024522  010246
4803  024524  010346
4804  024526  010546
4805  024530  012746  020200
4806  024534  016605  000020
4807  024540  100004
4808  024542  005405
4809  024544  112766  000055  000001
4810  024552  016700  154022
4811  024556  012703  024740
4812  024562  060003
4813  024564  112723  000040
4814  024570  005002
4815  024572  016001  024730
4816  024576  160105
4817  024600  002402
4818  024602  005202
4819  024604  000774
4820  024606  060105
4821  024610  005702
4822  024612  001002
4823  024614  105716
4824  024616  100407
4825  024620  105316
4826  024622  103003
4827  024624  116663  000001  177777
4828  024632  052702  000060
4829  024636  052702  000040
4830  024642  110223
4831  024644  005720
4832  024646  020067  154766
4833  024652  103746
4834  024654  101002
4835  024656  010502
4836  024660  000764
4837  024662  105726
4838  024664  100003
4839  024666  116663  177777  177776

```

```

APTSPool=100
APTCSUP=040
:*****
.SBTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
:
: *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
: *SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
: *NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
: *BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
: *REPLACED WITH SPACES.
: *CALL:
: *      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
: *      TYPDS                      ;;GO TO THE ROUTINE
:
$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5     ;;GET THE INPUT NUMBER
BPL      1$            ;;BR IF INPUT IS POS.
NEG      R5            ;;MAKE THE BINARY NUMBER POS.
MOVB    #'-(SP)       ;;MAKE THE ASCII NUMBER NEG.
1$:      MOV      RELOC, R0  ;;GET RELOCATION FACTOR.
MOV      #DBLK,R3     ;;SETUP THE OUTPUT POINTER
ADD      R0,R3        ;;ADD IN RELOCATION FACTOR.
MOVB    #' ,(R3)+     ;;SET THE FIRST CHARACTER TO A BLANK
2$:      CLR      R2        ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
3$:      SUB      R1,R5     ;;FORM THIS BCD DIGIT
BLT      4$           ;;BR IF DONE
INC      R2           ;;INCREASE THE BCD DIGIT BY 1
BR       3$
4$:      ADD      R1,R5     ;;ADD BACK THE CONSTANT
TST      R2           ;;CHECK IF BCD DIGIT=0
BNE      5$           ;;FALL THROUGH IF 0
TSTB    (SP)         ;;STILL DOING LEADING 0'S?
BMI      7$           ;;BR IF YES
5$:      ASLB    (SP)     ;;MSD?
BCC      6$           ;;BR IF NO
MOVB    1(SP),-1(R3)  ;;YES--SET THE SIGN
6$:      BIS      #'0,R2   ;;MAKE THE BCD DIGIT ASCII
7$:      BIS      #' ,R2   ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB    R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST     (R0)+        ;;JUST INCREMENTING
CMP     R0,.EIGHT    ;;CHECK THE TABLE INDEX
BLO     2$           ;;GO DO THE NEXT DIGIT
BHI     8$           ;;GO TO EXIT
MOV     R5,R2        ;;GET THE LSD
BR      6$           ;;GO CHANGE TO ASCII
8$:     TSTB    (SP)+  ;;WAS THE LSD THE FIRST NON-ZERO?
BPL     9$           ;;BR IF NO
9$:     MOVB   -1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING

```

```

4840 024674 105013          9$: CLRB (R3)          ;; SET THE TERMINATOR
4841 024676 012605          MOV (SP)+,R5        ;; POP STACK INTO R5
4842 024700 012603          MOV (SP)+,R3        ;; POP STACK INTO R3
4843 024702 012602          MOV (SP)+,R2        ;; POP STACK INTO R2
4844 024704 012601          MOV (SP)+,R1        ;; POP STACK INTO R1
4845 024706 012600          MOV (SP)+,R0        ;; POP STACK INTO R0
4846 024710 004567 176662 JSR R5, $PRINT      ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4847 024714 024740          .WORD $DBLK        ;; ADDRESS OF MESSAGE TO BE TYPED
4848 024716 016666 000002 000004 MOV 2(SP),4(SP)     ;; ADJUST THE STACK
4849 024724 012616          MOV (SP)+,(SP)
4850 024726 000002          RTI                ;; RETURN TO USER
4851 024730 023420          $DTBL: 10000.
4852 024732 001750          1000.
4853 024734 000144          100.
4854 024736 000012          10.
4855 024740 000004          $DBLK: .BLKW 4
4856          .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
4857
4858          ;*****
4859          ;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4860          ;OCTAL (ASCII) NUMBER AND TYPE IT.
4861          ;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4862          ;CALL:
4863          ; MOV NUM,-(SP) ;; NUMBER TO BE TYPED
4864          ; TYPOS ;; CALL FOR TYPEOUT
4865          ; .BYTE N ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4866          ; .BYTE M ;; M=1 OR 0
4867          ; ;; 1=TYPE LEADING ZEROS
4868          ; ;; 0=SUPPRESS LEADING ZEROS
4869          ;
4870          ;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4871          ;$TYPOS OR $TYPOC
4872          ;CALL:
4873          ; MOV NUM,-(SP) ;; NUMBER TO BE TYPED
4874          ; TYPON ;; CALL FOR TYPEOUT
4875          ;
4876          ;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
4877          ;CALL:
4878          ; MOV NUM,-(SP) ;; NUMBER TO BE TYPED
4879          ; TYPOC ;; CALL FOR TYPEOUT
4880          ;
4881 024750 017646 000000 $TYPOS: MOV 2(SP),-(SP) ;; PICKUP THE MODE
4882 024754 116667 000001 000213 MOVB 1(SP),$OFILL ;; LOAD ZERO FILL SWITCH
4883 024762 112667 000211 MOVB (SP)+,$OMODE+1 ;; NUMBER OF DIGITS TO TYPE
4884 024766 062716 000002 ADD #2,(SP) ;; ADJUST RETURN ADDRESS
4885 024772 000406 BR $TYPON
4886 024774 112767 000001 000173 $TYPOC: MOVB #1,$OFILL ;; SET THE ZERO FILL SWITCH
4887 025002 112767 000006 000167 MOVB #6,$OMODE+1 ;; SET FOR SIX(6) DIGITS
4888 025010 112767 000005 000156 $TYPON: MOVB #5,$OCNT ;; SET THE ITERATION COUNT
4889 025016 010346 MOV R3,-(SP) ;; SAVE R3
4890 025020 010446 MOV R4,-(SP) ;; SAVE R4
4891 025022 010546 MOV R5,-(SP) ;; SAVE R5
4892 025024 116704 000147 MOVB $OMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
4893 025030 005404 NEG R4
4894 025032 062704 000006 ADD #6,R4 ;; SUBTRACT IT FOR MAX. ALLOWED
4895 025036 110467 000134 MOVB R4,$OMODE ;; SAVE IT FOR LSE

```

BINARY TO OCTAL (ASCII) AND TYPE

```

4896 025042 116704 000127      MOVB  $OFILL,R4      ;; GET THE ZERO FILL SWITCH
4897 025046 016605 000012      MOV   12(SP),R5     ;; PICKUP THE INPUT NUMBER
4898 025052 005003              CLR   R3            ;; CLEAR THE OUTPUT WORD
4899 025054 006105      1$:  ROL   R5         ;; ROTATE MSB INTO "C"
4900 025056 000404              BR    3$           ;; GO DO MSB
4901 025060 006105      2$:  ROL   R5         ;; FORM THIS DIGIT
4902 025062 006105              ROL   R5
4903 025064 006105              ROL   R5
4904 025066 010503              MOV   R5,R3
4905 025070 006103      3$:  ROL   R3            ;; GET LSB OF THIS DIGIT
4906 025072 105367 000100      DECB  $OMODE        ;; TYPE THIS DIGIT?
4907 025076 100017              BPL   7$           ;; BR IF NO
4908 025100 042703 177770      BIC   #177770,R3   ;; GET RID OF JUNK
4909 025104 001002              BNE   4$           ;; TEST FOR 0
4910 025106 005704              TST  R4            ;; SUPPRESS THIS 0?
4911 025110 001403              BEQ   5$           ;; BR IF YES
4912 025112 005204      4$:  INC   R4            ;; DON'T SUPPRESS ANYMORE 0'S
4913 025114 052703 000060      BIS   #'0,R3       ;; MAKE THIS DIGIT ASCII
4914 025120 052703 000040      5$:  BIS   #' ,R3     ;; MAKE ASCII IF NOT ALREADY
4915 025124 110367 000042      MOVB  R3,8$        ;; SAVE FOR TYPING
4916 025130 004567 176442      JSR   R5,8$        $PRINT ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4917 025134 025172              .WORD 8$          ADDRESS OF MESSAGE TO BE TYPED
4918 025136 105367 000032      7$:  DECB  $OCNT        ;; COUNT BY 1
4919 025142 003346              BGT   2$           ;; BR IF MORE TO DO
4920 025144 002402              BLT   6$           ;; BR IF DONE
4921 025146 005204              INC   R4            ;; INSURE LAST DIGIT ISN'T A BLANK
4922 025150 000743              BR    2$           ;; GO DO THE LAST DIGIT
4923 025152 012605      6$:  MOV   (SP)+,R5     ;; RESTORE R5
4924 025154 012604              MOV   (SP)+,R4     ;; RESTORE R4
4925 025156 012603              MOV   (SP)+,R3     ;; RESTORE R3
4926 025160 016666 000002 000004      MOV   2(SP),4(SP)  ;; SET THE STACK FOR RETURNING
4927 025166 012616              MOV   (SP)+,(SP)
4928 025170 000002              RTI
4929 025172 000              8$:  .BYTE 0            ;; RETURN
4930 025173 000              .BYTE 0            ;; STORAGE FOR ASCII DIGIT
4931 025174 000      $OCNT: .BYTE 0      ;; TERMINATOR FOR TYPE ROUTINE
4932 025175 000      $OFILL: .BYTE 0     ;; OCTAL DIGIT COUNTER
4933 025176 000000      $OMODE: .WORD 0     ;; ZERO FILL SWITCH
4934              .ERROR TRAP SERVICE ROUTINE ;; NUMBER OF DIGITS TO TYPE
4935 025200 005727      ERRTRP: TST (PC)+  ;; CHECK IF PREV TRAP TO 4 REPORTED
4936 025202 000000      1$:  .WORD 0            ;; CONTAINS ERROR REPORTED FLAG
4937 025204 001010              BNE   2$           ;; BRANCH IF NOT REPORTED
4938 025206 005267 177770      INC   1$           ;; SET DOUBLE TRAP FLAG.
4939 025212 011667 153750      MOV   (SP), $TMP3  ;; SAVE THE BAD PC FOR TYPING.
4940 025216 004767 175026      JSR   PC, $ERROR   ;; *** ERROR *** (GO TYPE A MESSAGE)
4941 025222 000031              .WORD 31          ;; ERROR TYPE CODE.
4942 025224 000401              BR    3$           ;; SKIP HALT
4943 025226 000000      2$:  HALT            ;; ERROR! SECOND TRAP TO 4 OCCURRED
4944              ;; BEFORE FIRST WAS PRINTED
4945 025230 005067 177746      3$:  CLR   1$           ;; RETURN TO PROGRAM AND TRY TO RECOVER
4946 025234 000002              RTI
4947
4948      .SBTTL PHYSICAL ADDRESS TYPE ROUTINE
4949      ;* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4950 025236
4951 025236 010046      $TYPAD: MOV   R0,-(SP)  ;; PUSH R0 ON STACK

```

```

4952 025240 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
4953 025242 010246      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
4954 025244 010346      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
4955 025246 016602 000012      MOV      12(SP), R2    ;; GET BASE ADDRESS
4956 025252 005003      CLR      R3            ;; WORKING & INDEX REGISTER
4957 025254 005767 153326      TST      MMAVA        ;; CHECK FOR MEM MGMT AVAILABLE
4958 025260 001430      BEQ      1$           ;; BRANCH IF NO MEM MGMT
4959 025262 032737 000001 177572      BIT      #1,          @#SRO ;; CHECK IF MEM MGMT ENABLED
4960 025270 001424      BEQ      1$           ;; BRANCH IF MEM MGMT NOT ENABLED
4961 025272 010201      MOV      R2,          R1 ;; COPY VIRTUAL ADR
4962 025274 006101      ROL      R1           ;; SHUFFLE BITS 13,14,15 INTO 1,2,3
4963 025276 006101      ROL      R1
4964 025300 006101      ROL      R1
4965 025302 006101      ROL      R1
4966 025304 006101      ROL      R1
4967 025306 042701 177761      BIC      #177761, R1    ;; CLR ALL EXCEPT BITS 1,2,3
4968 025312 062701 172340      ADD      #KIPARO, R1   ;; SET TO APPROPRIATE PAR
4969 025316 011101      MOV      (R1),        R1 ;; GET CONTENTS OF PAR
4970 025320 012700 000006      MOV      #6,          RO ;; SET UP COUNTER
4971 025324 006301      4$:      ASL      R1           ;; SHIFT PAR
4972 025326 006103      ROL      R3           ;; SAVE OVERFLOW BITS
4973 025330 077003      SOB      RO,          4$ ;; COUNT SIX SHIFTS
4974 025332 042702 160000      BIC      #160000, R2   ;; SAVE BANK BITS
4975 025336 060102      ADD      R1,          R2 ;; COMPUTE PHYSICAL ADDRESS
4976 025340 005503      ADC      R3           ;; MAKE SURE CARRY ISN'T LOST!
4977 025342 006302      1$:      ASL      R2           ;; FIRST DIGIT TO R3
4978 025344 006103      ROL      R3
4979 025346 012700 000006      MOV      #6,RO        ;; DIGIT COUNT
4980 025352 000404      BR       3$           ;; PRINT FIRST DIGIT
4981 025354 006302      2$:      ASL      R2
4982 025356 006103      ROL      R3
4983 025360 005301      DEC      R1
4984 025362 001374      BNE      2$
4985 025364 012701 000003      3$:      MOV      #3,R1        ;; DIGIT SHIFT COUNT
4986 025370 062703 000060      ADD      #60,         R3 ;; MAKE IT AN ASCII DIGIT
4987 025374 110367 000036      MOV      R3,          8$ ;; LOAD DIGIT INTO MESSAGE
4988 025400 004567 176172      JSR      R5,          $PRINT ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4989 025404 025436      .WORD   8$           ;; ADDRESS OF MESSAGE TO BE TYPED
4990 025406 005003      CLR      R3           ;; CLEAR INDEX
4991 025410 005300      DEC      RO           ;; DEC DIGIT COUNT
4992 025412 001360      BNE      2$
4993 025414 012603      MOV      (SP)+,R3     ;; POP STACK INTO R3
4994 025416 012602      MOV      (SP)+,R2     ;; POP STACK INTO R2
4995 025420 012601      MOV      (SP)+,R1     ;; POP STACK INTO R1
4996 025422 012600      MOV      (SP)+,RO     ;; POP STACK INTO RO
4997 025424 012616      MOV      (SP)+, (SP)  ;; ADJUST THE STACK TO CLEAR DATA
4998 025426 004567 176144      JSR      R5,          $PRINT ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4999 025432 027066      .WORD   FILL2        ;; ADDRESS OF MESSAGE TO BE TYPED
5000 025434 000207      RTS      PC           ;; RETURN
5001 025436      000      8$:      .BYTE   0           ;; ONE DIGIT MESSAGE BUFFER
5002 025437      000      .BYTE   0           ;; MESSAGE TERMINATOR
5003
5004      .SBTTL  STANDARD PROGRAM MESSAGES
5005      ;;*****
5006      ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
5007      ;THE PROGRAM

```


STANDARD PROGRAM MESSAGES

```

5008      ..*****
5009 025440 005015 052113 030461 MMAMES: .ASCIZ <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
5010 025446 024040 042515 047515
5011 025454 054522 046440 047101
5012 025462 043501 046505 047105
5013 025470 024524 040440 040526
5014 025476 046111 041101 042514
5015 025504      000
5016 025505      015 046412 046505 MEMMES: .ASCIZ <15><12>'MEMORY MAP:'
5017 025512 051117 020131 040515
5018 025520 035120      000
5019 025523      015 041012 052131 BYTMES: .ASCIZ <15><12>'BYTE MEMORY MAP:'
5020 025530 020105 042515 047515
5021 025536 054522 046440 050101
5022 025544 000072
5023 025546 005015 040520 044522 MMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'
5024 025554 054524 046440 046505
5025 025562 051117 020131 040515
5026 025570 035120      000
5027 025573      015 043012 047522 FROM: .ASCIZ <15><12>'FROM '
5028 025600 020115      000
5029 025603      040 047524 000040 TO: .ASCIZ ' TO '
5030 025610 005015 047111 052523 INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
5031 025616 043106 041511 042511
5032 025624 052116 046440 046505
5033 025632 051117 027131 027056
5034 025640 044506 051522 020124
5035 025646 033061 020113 047516
5036 025654 020124 046101 020114
5037 025662 044124 051105 020505
5038 025670      000
5039 025671      015 047012 020117 MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'
5040 025676 040520 044522 054524
5041 025704 051040 043505 051511
5042 025712 042524 051522 043040
5043 025720 052517 042116      000
5044 025725      015 051012 051505 PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
5045 025732 040524 052122 047111
5046 025740 020107 043101 042524
5047 025746 020122 020101 047520
5048 025754 042527 020122 040506
5049 025762 046111 051125 006505
5050 025770 000012
5051 025772 005015 047516 050040 NOPE: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
5052 026000 051101 052111 020131
5053 026006 051105 047522 051522
5054 026014 043040 052517 042116
5055 026022 047440 020116 042515
5056 026030 047515 054522 051440
5057 026036 040503 006516 000012
5058 026044 005015 051120 043517 PROCREL: .ASCII <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
5059 026052 040522 020115 047516
5060 026060 020127 042522 044523
5061 026066 042504 020123 040502
5062 026074 045503 040440 020124
5063 026102 020060 047524 034040

```

5064	026110	113							
5065	026111	015	044012	052111		.ASCIZ	<15><12>	'HIT CONTINUE FOR NORMAL RUNNING'	<15><12>
5066	026116	041440	047117	044524					
5067	026124	052516	020105	047506					
5068	026132	020122	047516	046522					
5069	026140	046101	051040	047125					
5070	026146	044516	043516	005015					
5071	026154	000							
5072	026155	122	043505	051511	MX1:	.ASCIZ	'REGISTER AT '		
5073	026162	042524	020122	052101					
5074	026170	000040							
5075	026172	041440	047117	051124	MX2:	.ASCIZ	'CONTROLS '		
5076	026200	046117	020123	000					
5077	026205	015	041412	051117	MX3:	.ASCIZ	<15><12>	'CORE PARITY '	
5078	026212	020105	040520	044522					
5079	026220	054524	000040						
5080	026224	005015	047515	020123	MX4:	.ASCIZ	<15><12>	'MOS PARITY '	
5081	026232	040520	044522	054524					
5082	026240	000040							
5083	026242	005015	051515	030461	MX5:	.ASCIZ	<15><12>	'MS11-K CSR '	
5084	026250	045455	041440	051123					
5085	026256	000040							
5086	026260	051515	030461	045455	MX6:	.ASCIZ	'MS11-K MEMORY PRESENT!! TO CORRECTLY TEST RUN DZMML...'		
5087	026266	046440	046505	051117					
5088	026274	020131	051120	051505					
5089	026302	047105	020524	020041					
5090	026310	047524	041440	051117					
5091	026316	042522	052103	054514					
5092	026324	052040	051505	020124					
5093	026332	052522	020116	055104					
5094	026340	046515	027114	027056					
5095	026346	000							
5096	026347	015	047012	020117	NOMEM:	.ASCIZ	<15><12>	'NO MEMORY FOUND.'	
5097	026354	042515	047515	054522					
5098	026362	043040	052517	042116					
5099	026370	000056							
5100	026372	005015	005012	047111	FADMES:	.ASCII	<15><12><12><12>	'INPUT ALL PARAMETERS IN OCTAL.'	
5101	026400	052520	020124	046101					
5102	026406	020114	040520	040522					
5103	026414	042515	042524	051522					
5104	026422	044440	020116	041517					
5105	026430	040524	027114						
5106	026434	005015	044506	051522		.ASCIZ	<15><12>	'FIRST ADDRESS: '	
5107	026442	020124	042101	051104					
5108	026450	051505	035123	020040					
5109	026456	000							
5110	026457	015	046012	051501	LADMES:	.ASCIZ	<15><12>	'LAST ADDRESS: '	
5111	026464	020124	042101	051104					
5112	026472	051505	035123	020040					
5113	026500	000040							
5114	026502	005015	040477	042104	BADADR:	.ASCIZ	<15><12>	'?ADDRESS IN UNMAPPED BANK?'	
5115	026510	042522	051523	044440					
5116	026516	020116	047125	040515					
5117	026524	050120	042105	041040					
5118	026532	047101	037513	000					
5119	026537	015	051412	046105	CONST:	.ASCIZ	<15><12>	'SELECT CONSTANT: '	

STANDARD PROGRAM MESSAGES

5120	026544	041505	020124	047503	
5121	026552	051516	040524	052116	
5122	026560	000072			
5123	026562	005015	047125	054105	UNEXPT: .ASCIZ <15><12>'UNEXPECTED MEMORY PARITY ERROR'
5124	026570	042520	052103	042105	
5125	026576	046440	046505	051117	
5126	026604	020131	040520	044522	
5127	026612	054524	042440	051122	
5128	026620	051117	000		
5129	026623	015	050012	047522	PRELOC: .ASCIZ <15><12>'PROGRAM RELOCATED TO '
5130	026630	051107	046501	051040	
5131	026636	046105	041517	052101	
5132	026644	042105	052040	020117	
5133	026652	000			
5134	026653	015	046412	051117	MTOE: .ASCIZ <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
5135	026660	020105	044124	047101	
5136	026666	047440	042516	050040	
5137	026674	051101	052111	020131	
5138	026702	051105	047522	020122	
5139	026710	047506	047125	027104	
5140	026716	000			
5141	026717	015	051412	040503	SCANM: .ASCIZ <15><12>'SCANNING MEMORY FOR BAD PARITY.'
5142	026724	047116	047111	020107	
5143	026732	042515	047515	054522	
5144	026740	043040	051117	041040	
5145	026746	042101	050040	051101	
5146	026754	052111	027131	000	
5147	026761	015	050012	051101	PEWNC: .ASCIZ <15><12>'PARITY ERROR WILL NOT CLEAR.'
5148	026766	052111	020131	051105	
5149	026774	047522	020122	044527	
5150	027002	046114	047040	052117	
5151	027010	041440	042514	051101	
5152	027016	000056			
5153	027020	005015	047516	046440	NOMTST: .ASCIZ <15><12>'NO MEMORY TESTED.'
5154	027026	046505	051117	020131	
5155	027034	042524	052123	042105	
5156	027042	000056			
5157	027044	005015	045523	050111	SKPMES: .ASCIZ <15><12>'SKIPPING TEST #'
5158	027052	044520	043516	052040	
5159	027060	051505	020124	000043	
5160	027066	177777	000		FILL2: .ASCIZ <377><377>
5161					
5162					.SBTTL ERROR REPORTING MESSAGES AND TABLES.
5163					*****
5164					* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
5165					*****
5166	027071	120	051101	052111	DM1: .ASCIZ 'PARITY REGISTER DATA ERROR.'
5167	027076	020131	042522	044507	
5168	027104	052123	051105	042040	
5169	027112	052101	020101	051105	
5170	027120	047522	027122	000	
5171	027125	101	042104	042522	DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'
5172	027132	051523	052040	051505	
5173	027140	020124	051105	047522	
5174	027146	024122	051524	030524	
5175	027154	032455	027051	000	

5176	027161	103	047117	052123	DM4:	.ASCIZ	'CONSTANT DATA ERROR(TST6-10).'
5177	027166	047101	020124	040504			
5178	027174	040524	042440	051122			
5179	027202	051117	052050	052123			
5180	027210	026466	030061	027051			
5181	027216	000					
5182	027217	122	052117	052101	DMS:	.ASCIZ	'ROTATING BIT ERROR(TST11-12).'
5183	027224	047111	020107	044502			
5184	027232	020124	051105	047522			
5185	027240	024122	051524	030524			
5186	027246	026461	031061	027051			
5187	027254	000					
5188	027255	061	054040	051117	DM6:	.ASCIZ	'1 XOR 8 PATTERN ERROR(TST13).'
5189	027262	034040	050040	052101			
5190	027270	042524	047122	042440			
5191	027276	051122	051117	052050			
5192	027304	052123	031461	027051			
5193	027312	000					
5194	027313	063	054040	051117	DM7:	.ASCIZ	'3 XOR 9 PATTERN ERROR(TST14-17).'
5195	027320	034440	050040	052101			
5196	027326	042524	047122	042440			
5197	027334	051122	051117	052050			
5198	027342	052123	032061	030455			
5199	027350	024467	000056				
5200	027354	020070	047530	020122	DM10:	.ASCIZ	'8 XOR 13 PATTERN ERROR(TST20).'
5201	027362	031461	050040	052101			
5202	027370	042524	047122	042440			
5203	027376	051122	051117	052050			
5204	027404	052123	030062	027051			
5205	027412	000					
5206	027413	120	051101	052111	DM11:	.ASCIZ	'PARITY MEMORY ADDRESS ERROR(TST21).'
5207	027420	020131	042515	047515			
5208	027426	054522	040440	042104			
5209	027434	042522	051523	042440			
5210	027442	051122	051117	052050			
5211	027450	052123	030462	027051			
5212	027456	000					
5213	027457	104	052101	050111	DM12:	.ASCIZ	'DATIP WITH WRONG PARITY DIDN'T TRAP(TST21).'
5214	027464	053440	052111	020110			
5215	027472	051127	047117	020107			
5216	027500	040520	044522	054524			
5217	027506	042040	042111	023516			
5218	027514	020124	051124	050101			
5219	027522	052050	052123	030462			
5220	027530	027051	000				
5221	027533	127	047522	043516	DM13:	.ASCIZ	'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
5222	027540	050040	051101	052111			
5223	027546	020131	051124	050101			
5224	027554	042520	026104	041040			
5225	027562	052125	047040	020117			
5226	027570	042522	044507	052123			
5227	027576	051105	051440	047510			
5228	027604	051527	042440	051122			
5229	027612	051117	043040	040514			
5230	027620	027107	000				
5231	027623	120	051101	052111	DM14:	.ASCIZ	'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).'

E15

5232	027630	020131	042522	044507		
5233	027636	052123	051105	047040		
5234	027644	052117	046440	050101		
5235	027652	042520	020104	051501		
5236	027660	041440	047117	051124		
5237	027666	046117	044514	043516		
5238	027674	052040	044510	020123		
5239	027702	042101	051104	051505		
5240	027710	024123	051524	031124		
5241	027716	024461	000056			
5242	027722	047515	042522	052040	DM16:	.ASCIZ 'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
5243	027730	040510	020116	047117		
5244	027736	020105	042522	044507		
5245	027744	052123	051105	044440		
5246	027752	042116	041511	052101		
5247	027760	042105	050040	051101		
5248	027766	052111	020131	051105		
5249	027774	047522	027122	000		
5250	030001	104	052101	020101	DM17:	.ASCIZ "'DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST21)."
5251	030006	044123	052517	042114		
5252	030014	023516	020124	040510		
5253	030022	042526	041440	040510		
5254	030030	043516	042105	053440		
5255	030036	042510	020116	040520		
5256	030044	044522	054524	042440		
5257	030052	051122	051117	052040		
5258	030060	040522	050120	042105		
5259	030066	052050	052123	030462		
5260	030074	027051	000			
5261	030077	122	047101	047504	DM20:	.ASCIZ 'RANDOM DATA ERROR(TST22).'
5262	030104	020115	040504	040524		
5263	030112	042440	051122	051117		
5264	030120	052050	052123	031062		
5265	030126	027051	000			
5266	030131	111	051516	051124	DM21:	.ASCIZ 'INSTRUCTION EXECUTION ERROR(TST23-30).'
5267	030136	041525	044524	047117		
5268	030144	042440	042530	052503		
5269	030152	044524	047117	042440		
5270	030160	051122	051117	052050		
5271	030166	052123	031462	031455		
5272	030174	024460	000056			
5273	030200	041042	040522	041516	DM22:	.ASCIZ "'BRANCH GOBBLE" ERROR(TST31).'
5274	030206	020110	047507	041102		
5275	030214	042514	020042	051105		
5276	030222	047522	024122	051524		
5277	030230	031524	024461	000056		
5278	030236	051120	043517	040522	DM23:	.ASCIZ 'PROGRAM CODE CHANGED WHEN RELOCATED.'
5279	030244	020115	047503	042504		
5280	030252	041440	040510	043516		
5281	030260	042105	053440	042510		
5282	030266	020116	042522	047514		
5283	030274	040503	042524	027104		
5284	030302	000				
5285	030303	124	040522	050120	DM24:	.ASCIZ "'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
5286	030310	042105	020054	052502		
5287	030316	020124	047516	051040		

F15

MAINDEC-11-02QMC-C-D: 02-DEC-76

0-124K MEMORY EXERCISER, 16K VER 08:47

MACY11 27(1006)

02-DEC-76 09:00 PAGE 105

SEG 0187

ERROR REPORTING MESSAGES AND TABLES.

5288 030324 043505 051511 042524
5289 030332 020122 040510 020104
5290 030340 051105 047522 020122
5291 030346 044502 020124 042523
5292 030354 027124 000
5293 030357 124 040522 050120
5294 030364 042105 052040 020117
5295 030372 030461 027064 000
5296 030377 106 044501 042514
5297 030404 020104 047524 052040
5298 030412 040522 027120 000
5299 030417 050 041501 044524
5300 030424 047117 042440 040516
5301 030432 046102 020105 040527
5302 030440 047123 052047 051440
5303 030446 052105 027051 000
5304 030453 015 052012 040522
5305 030460 050120 042105 052040
5306 030466 020117 020064 000
5307
5308
5309
5310
5311
5312 030473 120 004503 042522
5313 030500 004507 027523 004502
5314 030506 040527 000123
5315 030512 027526 041520 050011
5316 030520 050057 004503 040515
5317 030526 051411 041057 053411
5318 030534 051501 000
5319 030537 126 050057 004503
5320 030544 027520 041520 046411
5321 030552 004501 027523 000102
5322 030560 027526 041520 050011
5323 030566 050057 004503 042522
5324 030574 004507 040515 000
5325 030601 126 050057 004503
5326 030606 027520 041520 046411
5327 030614 052501 004524 042522
5328 030622 004507 027523 004502
5329 030630 040527 000123
5330 030634 027526 041520 050011
5331 030642 050057 004503 052511
5332 030650 004524 040515 051411
5333 030656 041057 053411 051501
5334 030664 000
5335 030665 126 050057 004503
5336 030672 027520 041520 050011
5337 030700 020123 027523 004502
5338 030706 051520 053440 051501
5339 030714 046411 004501 027523
5340 030722 004502 040527 000123
5341 030730 027526 041520 050011
5342 030736 050057 004503 051123
5343 030744 020103 040515 042011

DM25: .ASCIZ 'TRAPPED TO 114.'
DM26: .ASCIZ 'FAILED TO TRAP.'
DM27: .ASCIZ "(ACTION ENABLE WASN'T SET)."
DM31: .ASCIZ '<15><12>'TRAPPED TO 4 '

; DATA COLUMN HEADINGS

DH1: .ASCIZ 'PC REG S/B WAS'
DH2: .ASCIZ 'V/PC P/PC MA S/B WAS'
DH12: .ASCIZ 'V/PC P/PC MA S/B'
DH14: .ASCIZ 'V/PC P/PC REG MA'
DH15: .ASCIZ 'V/PC P/PC MAUT REG S/B WAS'
DH21: .ASCIZ 'V/PC P/PC IUT MA S/B WAS'
DH22: .ASCIZ 'V/PC P/PC PS S/B PS WAS MA S/B WAS'
DH23: .ASCIZ 'V/PC P/PC SRC MA DST MA S/B WAS'

ERROR REPORTING MESSAGES AND TABLES.

5344	030752	052123	046440	004501						
5345	030760	027523	004502	040527						
5346	030766	000123								
5347	030770	027526	041520	050011	DH24:	.ASCIZ	'V/PC	P/PC	TRP/PC'	
5348	030776	050057	004503	051124						
5349	031004	027520	041520	000						
5350	031011	126	050057	004503	DH25:	.ASCIZ	'V/PC	P/PC	TRP/PC	REG WAS'
5351	031016	027520	041520	052011						
5352	031024	050122	050057	004503						
5353	031032	042522	004507	040527						
5354	031040	000123								
5355	031042	027526	041520	050011	DH26:	.ASCIZ	'V/PC	P/PC	REG	WAS'
5356	031050	050057	004503	042522						
5357	031056	004507	040527	000123						
5358	031064	042522	004507	040527	DH30:	.ASCIZ	'REG	WAS	MA	WAS'
5359	031072	004523	040515	053411						
5360	031100	051501	000							

```

*****
* DATA FORMAT TABLE FOR ERROR PRINTOUT.
*****

```

5365	031103	000	377	000	DF1:	.BYTE	0,-1,0,0
5366	031106	000					
5367	031107	000	377	377	DF2:	.BYTE	0,-1,-1,0,0
5368	031112	000	000				
5369	031114	000	377	377	DF3:	.BYTE	0,-1,-1,-2,-2
5370	031117	376	376				
5371	031121	000	377	377	DF14:	.BYTE	0,-1,-1,-1,0,C
5372	031124	377	000	000			
5373	031127	000	377	000	DF21:	.BYTE	0,-1,0,-1,0,0
5374	031132	377	000	000			
5375	031135	000	377	376	DF22:	.BYTE	0,-1,-2,-2,-1,0,0
5376	031140	376	377	000			
5377	031143	000					
5378	031144	377	000	377	DF30:	.BYTE	-1,0,-1,-2
5379	031147	376					

.EVEN

5381										
5382		032110			.	=	32110			
5383										
5384		0000C1								

;THE LOADERS ARE SAVE HERE TO END OF BK

.END

ABASE = 000000	385	426			
ACDW1 = 000000	385	428			
ACDW2 = 000000	385	429			
ACPUOP = 000000	385	400			
ADWC = 000000	385	430			
ADDW1 = 000000	385	431			
ADDW10 = 000000	385	440			
ADDW11 = 000000	385	441			
ADDW12 = 000000	385	442			
ADDW13 = 000000	385	443			
ADDW14 = 000000	385	444			
ADDW15 = 000000	385	445			
ADDW2 = 000000	385	432			
ADDW3 = 000000	385	433			
ADDW4 = 000000	385	434			
ADDW5 = 000000	385	435			
ADDW6 = 000000	385	436			
ADDW7 = 000000	385	437			
ADDW8 = 000000	385	438			
ADDW9 = 000000	385	439			
ADZVCT = 000000	385	391			
ADZVM = 000000	385	427			
AE = 000001	180*	2662	2748	3889	
RENV = 000000	385	396			
RENYM = 000000	385	397			
AFATAL = 000000	385	388			
AMADR1 = 000000	385	413			
AMADR2 = 000000	385	417			
AMADR3 = 000000	385	420			
AMADR4 = 000000	385	423			
AMAMS1 = 000000	385	407			
AMAMS2 = 000000	385	415			
AMAMS3 = 000000	385	418			
AMAMS4 = 000000	385	421			
AMSGAD = 000000	385	393			
AMSGLG = 000000	385	394			
AMSGTY = 000000	385	387			
AMTYP1 = 000000	385	408			
AMTYP2 = 000000	385	416			
AMTYP3 = 000000	385	419			
AMTYP4 = 000000	385	422			
APASS = 000000	385	390			
APRIOR = 000000	385				
APTCSU = 000040	4644	4785*			
APTENV = 000001	4314	4637	4705	4783*	
APTSIZ = 000200	883	4782*			
APTSP0 = 000100	4639	4707	4784*		
ASWREG = 000000	385	398			
ATESTN = 000000	385	389			
AUNIT = 000000	385	392			
AUSWR = 000000	385	399			
AVECT1 = 000000	385	424			
AVECT2 = 000000	385	425			
BADADR = 026502	1429	5114*			
BANKNO = 016644	1612	1621	1647	1657	3555*
BGENC = 014376	3143	3169	3171*		

E16

\$MTYP2	001241	416*																	
\$MTYP3	001245	419*																	
\$MTYP4	001251	422*																	
\$MXCNT	022246	4182	4280*																
\$NULL	001154	367*	4664	4693															
\$NWTST=	000001	1475*	1477	1521*	1523	1558*	1560	1598*	1600	1633*	1635	1674*	1676	1677					
		1694*	1696	1727*	1729	1751*	1774*	1796*	1943*	2045*	2148*	2333*	2517*	2620*					
		2622	2776*	2811*	2813	2860*	2862	2909*	2911	2959*	2961	3008*	3010	3057*					
		3059	3107*	3109															
\$OCNT	025174	4888*	4918*	4931*															
\$OMODE	025176	4883*	4887*	4892	4895*	4906*	4933*												
\$OVER	021532	4144	4162	4170	4180	4189*													
\$PASS	001212	390*	882*	3196	3241*	3242*	3251	3269	4176	4195	4221	4231							
\$PASTM	001336	467*																	
\$PRINT	023576	326	895	933	961	1006	1014	1032	1045	1055	1077	1115	1202	1210					
		1216	1222	1226	1234	1243	1255	1356	1383	1428	1440	3227	3249	3256					
		3663	3839	3857	3936	3974	3999	4061	4082	4089	4211	4255	4302	4311					
		4345	4368	4370	4374	4376	4415	4421	4485	4492	4498	4503	4509	4511					
		4515	4519	4524	4593	4595	4605*	4657	4846	4916	4988	4998							
\$PWRAD	000752	329*																	
\$PWRDN	000610	296*	324	863															
\$PWRMG	000746	327*																	
\$PWRUP	000662	306	312*																
\$QUES	001200	378*	4336	4516	4535	4596	4599	4693											
\$RDCHR	022754	4442*	4478																
\$RDLIN	023074	4470*	4564																
\$RDOCT	023424	1362	1389	1446	4555*														
\$RDSZ =	000010	4463*																	
\$SAVR6	000762	305*	313	314*	315*	333*													
\$SCOPE	021260	1485	1531	1568	1608	1643	1685	1699	1732	1755	1778	1800	1947	2049					
		2152	2337	2521	2629	2780	2837	2886	2935	2985	3034	3083	3119	3208					
		4140*																	
\$SETUP=	000030	189*	862	863	865	866	893	897	3240	4141	4296	4324	4331	4431					
		4541																	
\$STJP =	177777	189*																	
\$SVLAD	021476	4152	4183*																
\$SVPC =	000010	214*	219																
\$SWR =	167400	1*	12	17	19	19	20	21	22	23	24	330	375	376					
		377	866	1490	1533	1570	1610	1645	1687	1701	1734	1757	1780	1805					
		1952	2054	2157	2342	2526	2631	2782	2842	2891	2940	2990	3039	3088					
		3124	3235	3240	3259	3268	3269	4132	4133	4134	4135	4136	4143	4155					
		4157	4158	4163	4164	4165	4172	4173	4174	4186	4189	4280	4287	4288					
		4289	4290	4291	4300	4308	4321	4324	4336										
\$SWREG	001226	398*	885																
\$SWRMK=	000340	1*	24	25	4136	4137	4159	4160											
\$TESTN	001210	389*	4184*																
\$TIMES	001170	280*	375*	3210*	3240*	4172*	4179	4182*	4280										
\$TKB	001146	364*	4429	4446	4452														
\$TKS	001144	363*	4429	4444	4450														
\$TMPO	001160	371*	579	581	584	587	595	597	599	3175*	3176*	3177*	3183	3535*					
		3545*	3547*	4034*	4038*	4046*													
\$TMP1	001162	372*	587	595	597	599	3178*	3179*	4035*										
\$TMP2	001164	373*	587	970*	971*	974*	985*	1346	1394	3180*									
\$TMP3	001166	374*	587	601	960*	976*	982*	984*	986*	1395	3174*	4929*							
\$*N =	000032	1*	12	1475	1490*	1521	1533*	1558	1570*	1598	1610*	1633	1645*	1674					
		1687*	1694	1696	1701*	1714	1727	1734*	1737	1751	1757*	1774	1780*	1796					

ABORT	1#	261	3322	3355	3393	3565	3658	3677	3735	3770	3819				
CKWD	1#	1290	1495	1512	1587	1623	1660	1705	1741	1764	1787	1845	1853	1861	1869
	1906	1914	1922	1930	1968	1975	1982	1989	2012	2021	2030	2071	2078	2085	2092
	2115	2124	2133	2174	2181	2188	2195	2222	2231	2240	2247	2256	2265	2272	2281
	2290	2297	2306	2315	2359	2366	2373	2380	2407	2416	2425	2432	2441	2450	2457
	2466	2475	2482	2491	2500	2552	2559	2600	2607	2719	2749	2797	2850	2899	2949
	2998	3047	3097												
CKWD2	1#	1494	1585	1703	1740	1843	1852	1860	1868	1904	1913	1921	1929	1966	1974
	1981	1988	2010	2020	2029	2069	2077	2084	2091	2113	2123	2132	2172	2180	2187
	2194	2220	2230	2239	2246	2255	2264	2271	2280	2289	2296	2305	2314	2357	2365
	2372	2379	2405	2415	2424	2431	2440	2449	2456	2465	2474	2481	2490	2499	2550
	2558	2598	2606	2796											
COMMEN	1#	138#	845	1459											
ENDCOM	1#	12	138#	849	1463										
ERROR	32#														
ESCAPE	138#														
GETPRI	138#														
GETSWR	138#	897#													
LDPDR	1#	3282	3284	3285	3286	3290									
MORETA	334#	448													
MULT	138#														
NEWTST	1#	138#	1475	1521	1558	1598	1633	1674	1694	1727	1751	1774	1796	1943	2045
	2149	2333	2517	2620	2776	2811	2860	2909	2959	3008	3057	3107			
PCP	1#	138#	317	318	2744	2757	3549	3566	3668	3760	3798	3866	3892	3920	3983
	3993	4001	4021	4113	4120	4587	4754	4755	4766	4775	4841	4993			
PRINT	1#	933	961	1006	1014	1031	1045	1055	1077	1115	1202	1210	1216	1222	1225
	1234	1356	1382	1428	1439	3227	3663	3839	3857	3936	3974	3999	4061	4211	4254
	4988	4998													
PUSH	1#	138#	298	304	2703	2726	3538	3556	3639	3678	3695	3791	3842	3887	3902
	3928	3948	3955	4014	4065	4091	4557	4700	4702	4723	4757	4768	4799	4950	
RDCHR	1#	4475													
RDDEC	1#														
RDLIN	1#	4560													
RDOCT	1#	1359	1386	1443											
REPORT	1#	138#													
RESREG	1#														
SAVREG	1#														
SCOPE	33#														
SCOPEX	4126#	4191													
SCOPIX	4126#	4141													
SETPRI	138#														
SETUP	1#	138#	854												
SIMTRP	1#	1230	1359	1386	1443	3252	4099	4107	4260	4354	4387	4394	4407	4475	4561
	4607														
SKIP	138#	1737													
SLASH	1#	138#	524	634											
SPACF	138#														
STARS	1#	138#	212	221	226	294	310	336	381	384	450	452	459	472	502
	504	551	555	570	572	605	607	942	951	1019	1022	1088	1091	1122	1131
	1142	1146	1196	1199	1262	1266	1306	1309	1349	1351	1475	1493	1521	1529	1559
	1566	1598	1606	1633	1641	1670	1673	1674	1683	1694	1697	1727	1730	1751	1753
	1774	1776	1796	1798	1837	1839	1883	1885	1898	1900	1943	1945	1960	1962	2003
	2005	2045	2047	2063	2065	2106	2108	2148	2150	2165	2167	2213	2215	2333	2335
	2350	2352	2398	2400	2517	2519	2541	2543	2573	2575	2599	2591	2620	2627	2776
	2778	2811	2835	2860	2884	2909	2933	2959	2983	3008	3032	3057	3081	3107	3117
	3229	3275	3281	3303	3305	3402	3407	3492	3497	3530	3533	3552	3554	3571	3573

.DIDBH	2909#	2911													
.CIDBL	2860#	2862													
.DIDO	2811#	2813													
.DIPDO	2959#	2961													
.DPDBH	3057#	3059													
.DPDBL	3008#	3010													
.EQUAT	1#	28													
.ERROR	1#	1282	1293	1300	1333	1498	1515	1551	1590	1626	1663	1708	1744	1767	1790
	1848	1856	1864	1872	1909	1917	1925	1933	1971	1978	1985	1992	2015	2024	2033
	2074	2081	2088	2095	2118	2127	2136	2177	2184	2191	2198	2225	2234	2243	2250
	2259	2268	2275	2284	2293	2300	2309	2318	2362	2369	2376	2383	2410	2419	2428
	2435	2444	2453	2460	2469	2478	2485	2494	2503	2555	2562	2603	2610	2655	2677
	2693	2701	2722	2735	2752	2761	2800	2853	2902	2952	3001	3050	3100	3188	3656
	3849	3862	3909	3915	3967	4940									
.HEADE	1#	2													
.KT11	1#	140													
.SCOPE	1#	1484	1530	1567	1607	1642	1684	1698	1731	1754	1777	1799	1946	2048	2151
	2336	2520	2628	2779	2836	2885	2934	2984	3033	3092	3118	3207			
.SETUP	1#	189													
.SWRHI	1#	13													
.SWRLO	1#	25#													
.TM7	1694#	1696													
.\$ACT1	1#	210													
.\$APT8	1#	382#													
.\$APTH	1#	448													
.\$APTY	1#	4693													
.\$ASTA	1#	470													
.\$CATC	1#	189													
.\$CMTA	1#	334													
.\$EOP	1#	3229													
.\$ERRO	1#	4281													
.\$ERRT	1#	4336													
.\$POWE	1#	292													
.\$RDOC	1#	4541													
.\$READ	1#	4426													
.\$SCOP	1#	4126													
.\$TYPD	1#	4786													
.\$TYPE	1#	4614													
.\$TYPO	1#	4856													

. ABS. 032110 000

ERRORS DETECTED: 0
DEFAULT GLOBALS GENERATED: 0

DZQMCC.BIN,DZQMCC.LST/SOL/CRF=DZQMCC.P11
RUN-TIME: 84 64 5 SECONDS
RUN-TIME RATIO: 375/154=2.4
CORE USED: 32K (63 PAGES)