MM11/ME11

0 124K MEM PARITY EXER

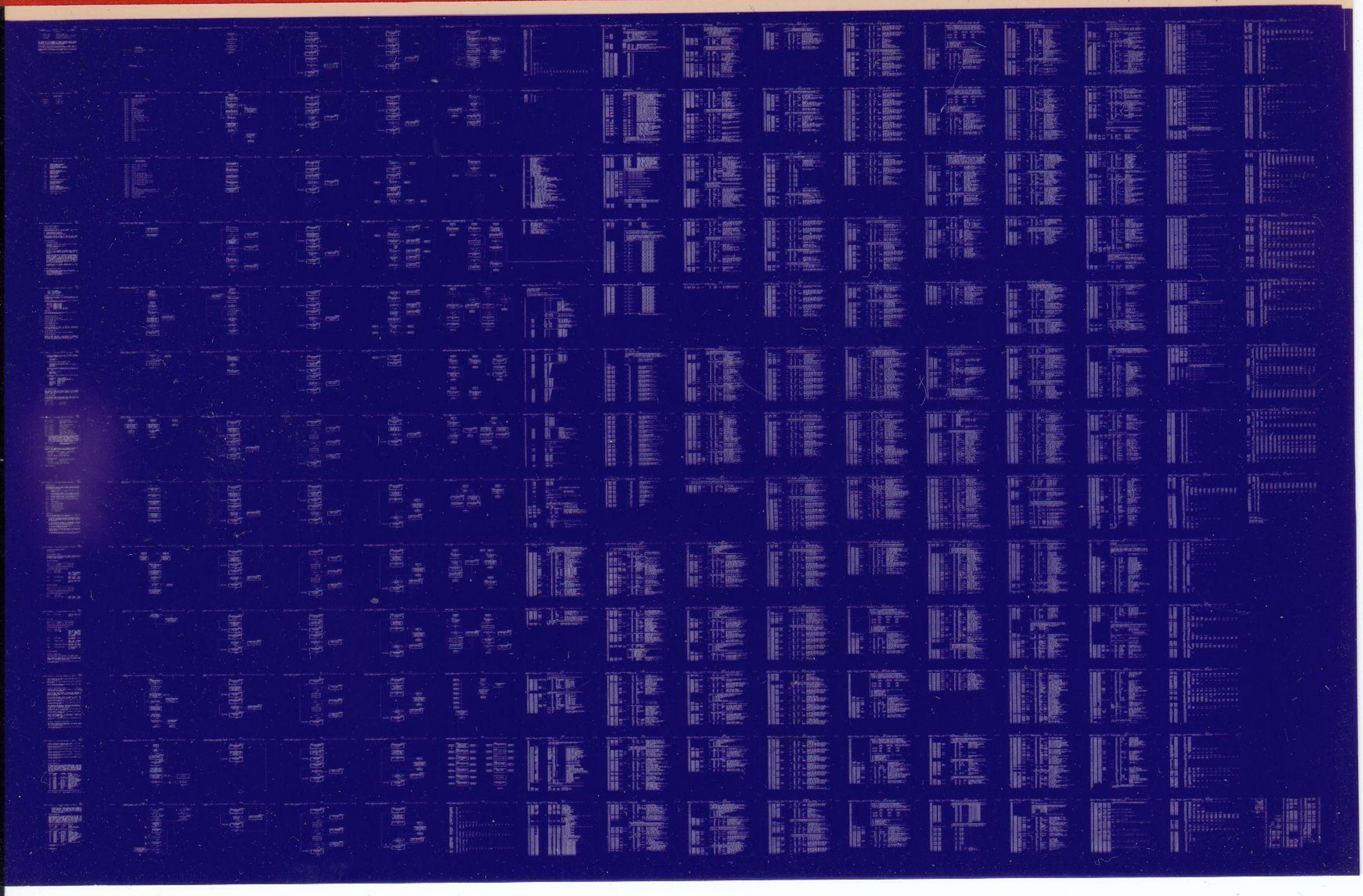
MD-11-DZQMC-C

EP DZQMC C DL B

FICHE 1 OF 1

MAR 1977

MADE IN USA



EOF1DZVSDBSEQ 00010000 770225 PDP10 411 HDR1DZQMCCSEQ 00010000 770225
PDP10 411 SEQ 0001

IDENTIFICATION
==========

PRODUCT CODE: MAINDEC-11-DZQMC-C-D

PRODUCT NAME: 0-124K MEMORY EXERCISER, 16K VERSION.

MAINTAINER: DIAGNOSTIC ENGINEERING, CC301

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)

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 (SR0)
    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

3.0 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:

P/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.

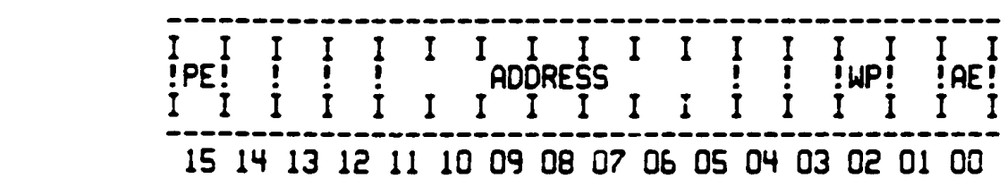
4.0 PERFORMANCE AND PROGRESS REPORTS

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:

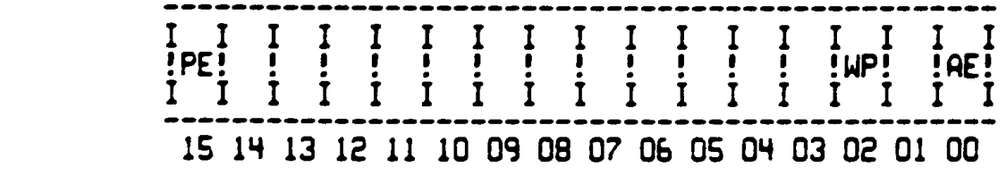
5.1 CORE PARITY REGISTER



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



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
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

5.3 MS11-K CSR

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	DOUBLE ERROR	
BITS 11-5	ERROR ADDRESS	WHEN BIT02 CLEARED CONTAINS HIGH ORDER BITS OF ADDRESS OF PARITY ERROR(BITS 17-11); WHEN BIT02 SET CONTAINS CHECK BITS FOR ECC.
BIT04	SINGLE ERROR	SET WHENEVER SINGLE ERROR OCCURS
BIT02	DIAGNOSTIC CHECK A	WHEN SET ENABLES READ-WRITE OF CHECK BITS(SEE BITS 11-5)
BIT01	DISABLE ERROR CORRECTION	WHEN SET NO ERROR CORRECTION TAKES PLACE
BIT00	DOUBLE ERROR ENABLE	WHEN SET ENABLES TRAP TO VECTOR 114 ON DOUBLE ERROR.

6.0 SUB-TEST SUMMARIES

6.1 SECTION 1: ADDRESS TESTS.

THESE TESTS VERIFY THE UNIQUENESS OF EVERY MEMORY ADDRESS.

TEST 1 WRITES AND READS THE VALUE OF EACH MEMORY WORD ADDRESS INTO THAT MEMORY LOCATION. AFTER ALL MEMORY HAS BEEN WRITTEN, ALL LOCATIONS ARE CHECKED AGAIN.

TEST 2 WRITES THE BYTE VALUE OF EACH ADDRESS INTO THAT BYTE LOCATION AND CHECKS IT.

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 TOGGLING 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
****

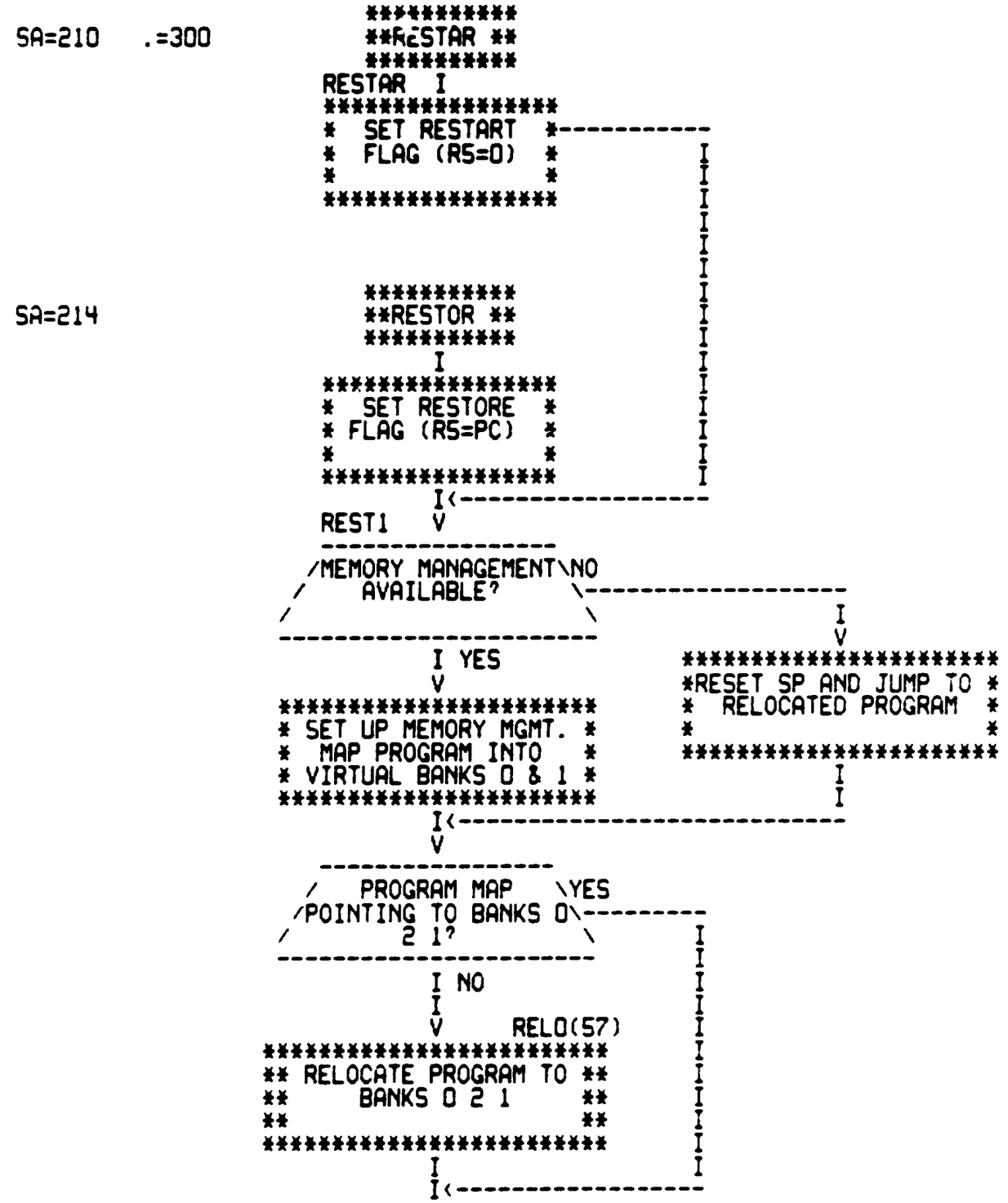
MAINDEC-11-DZQMC-C 0-124K MEMORY EXERCISER. (16K VERSION)

****

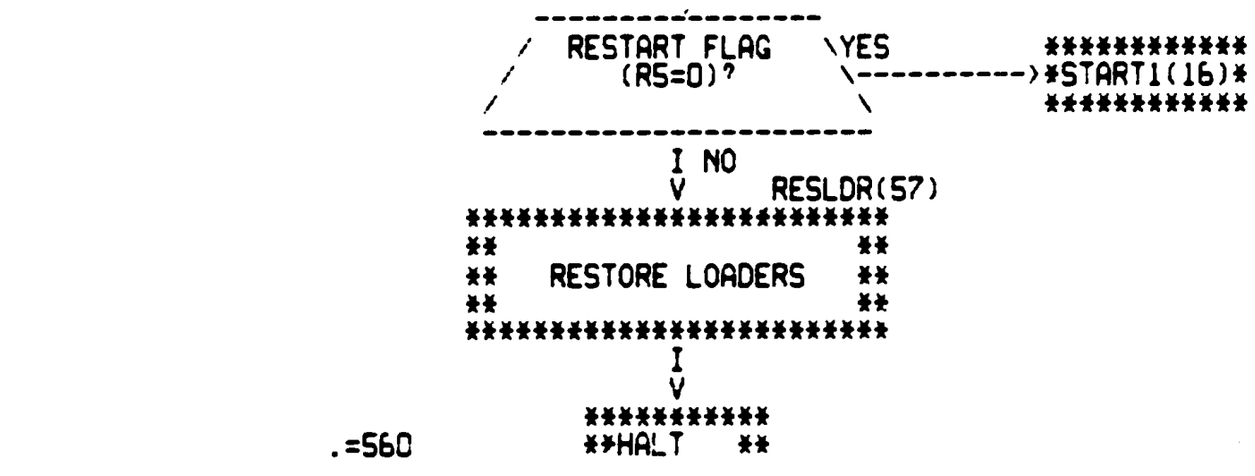
TABLE OF CONTENTS
****

TABLE OF CONTENTS
****

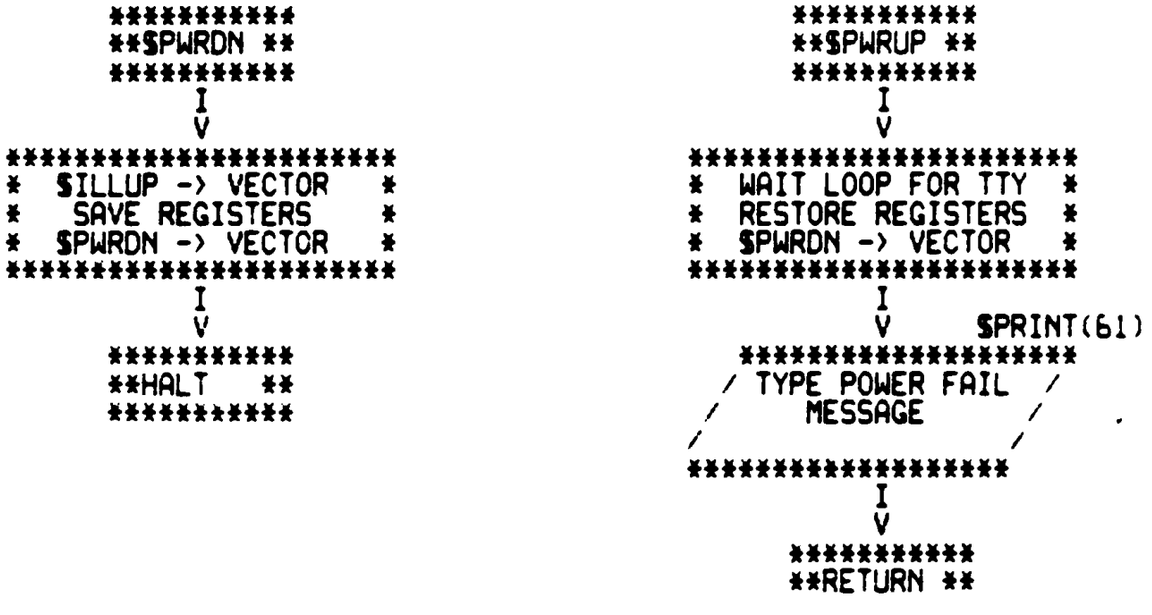
```
                        **************************
                        * SWITCH SETTINGS AND    *
                        *  BASIC DEFINITIONS     *
                        *                        *
                        **************************


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


SA=210
.=300
****
**RESTAR **
****
RESTAR
SET RESTART FLAG (R5=0)
SA=214
****
**RESTOR **
****
SET RESTORE FLAG (R5=PC)
REST1
MEMORY MANAGEMENT AVAILABLE?
NO
YES
SET UP MEMORY MGMT. MAP PROGRAM INTO VIRTUAL BANKS 0 & 1
RESET SP AND JUMP TO RELOCATED PROGRAM
PROGRAM MAP POINTING TO BANKS 0 2 1?
NO
RELO(57)
RELOCATE PROGRAM TO BANKS 0 2 1

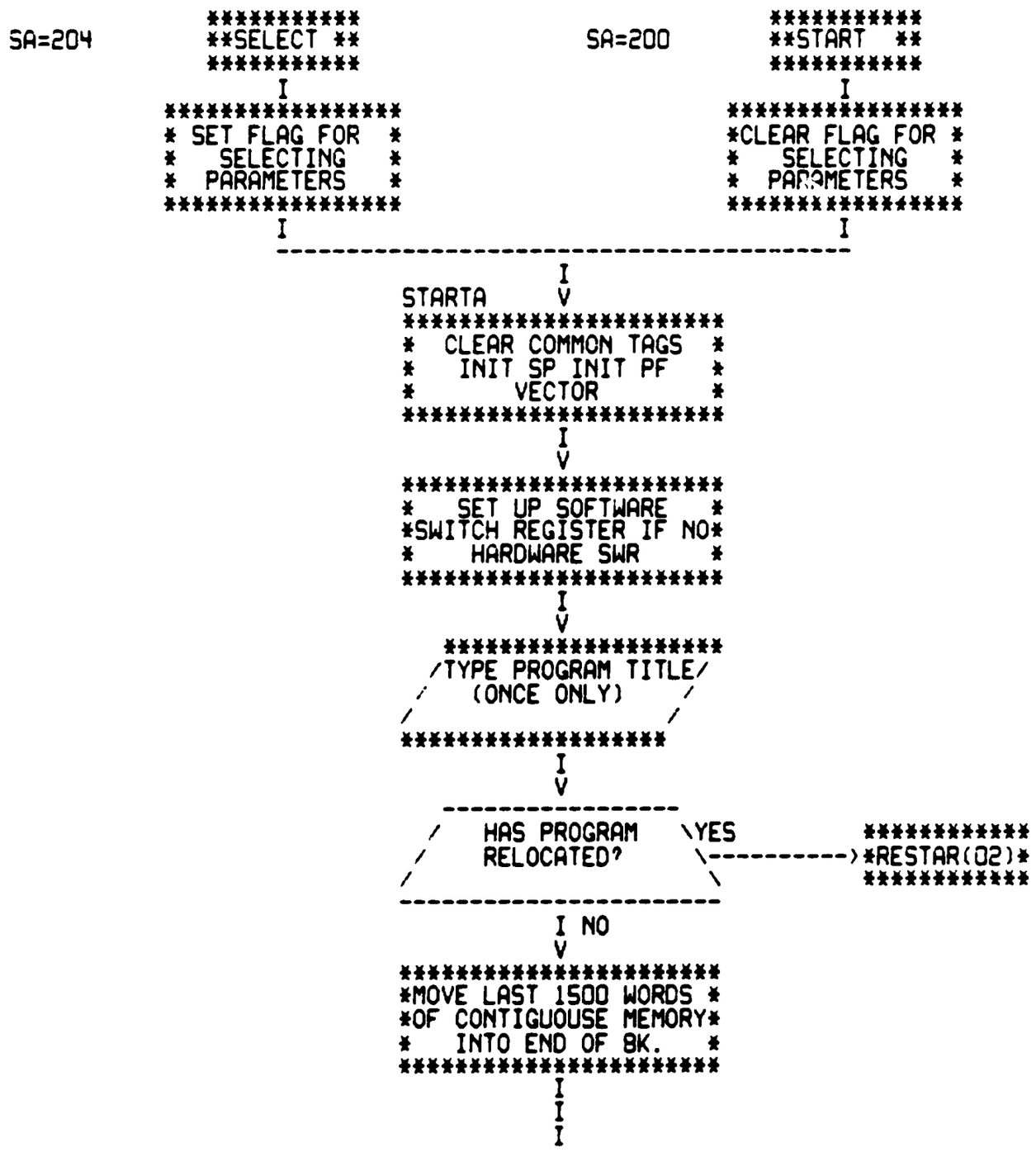
RESTART AND RESTORE ROUTINES

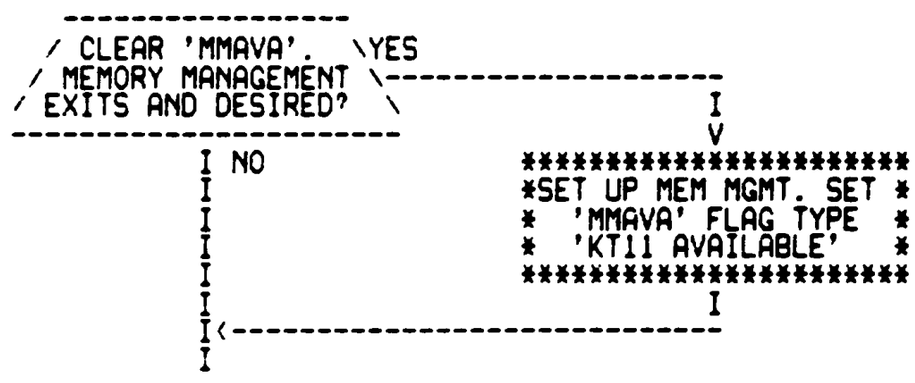


POWER FAIL ROUTINES

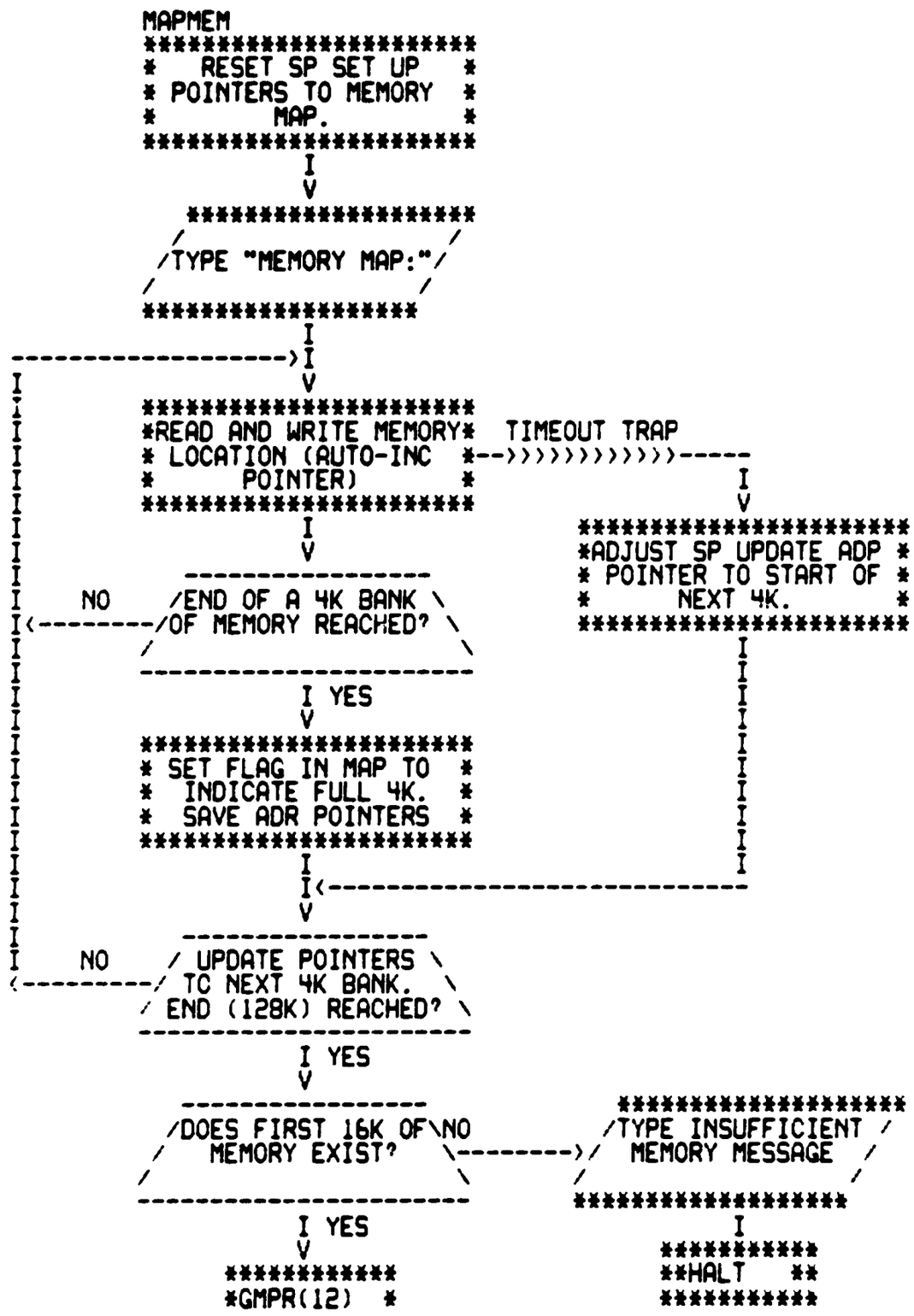


```
                        **************************
                        *  STANDARD 'SYSMAC'     *
.=1100                  *     COMMON TAGS        *
                        *                        *
                        **************************


                        **************************
                        *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           *
                        *                        *
```

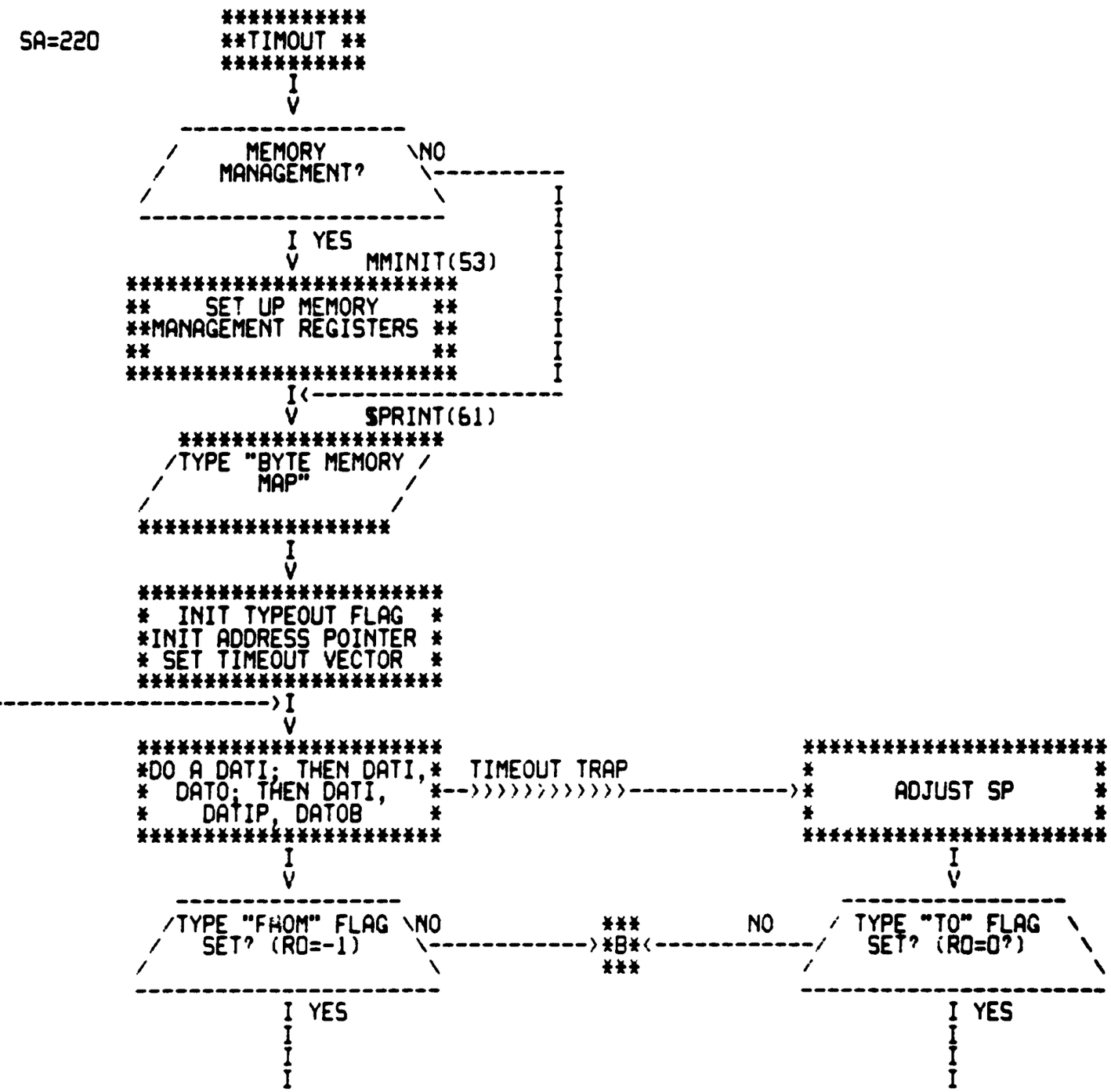

SA=204
**SELECT **
SET FLAG FOR SELECTING PARAMETERS
SA=200
**START **
CLEAR FLAG FOR SELECTING PARAMETERS
STARTA
CLEAR COMMON TAGS
INIT SP INIT PF VECTOR
SET UP SOFTWARE
SWITCH REGISTER IF NO HARDWARE SWR
/TYPE PROGRAM TITLE/
(ONCE ONLY)
HAS PROGRAM RELOCATED?
YES
RESTAR(02)
NO
MOVE LAST 1500 WORDS
OF CONTIGUOUSE MEMORY
INTO END OF 8K.

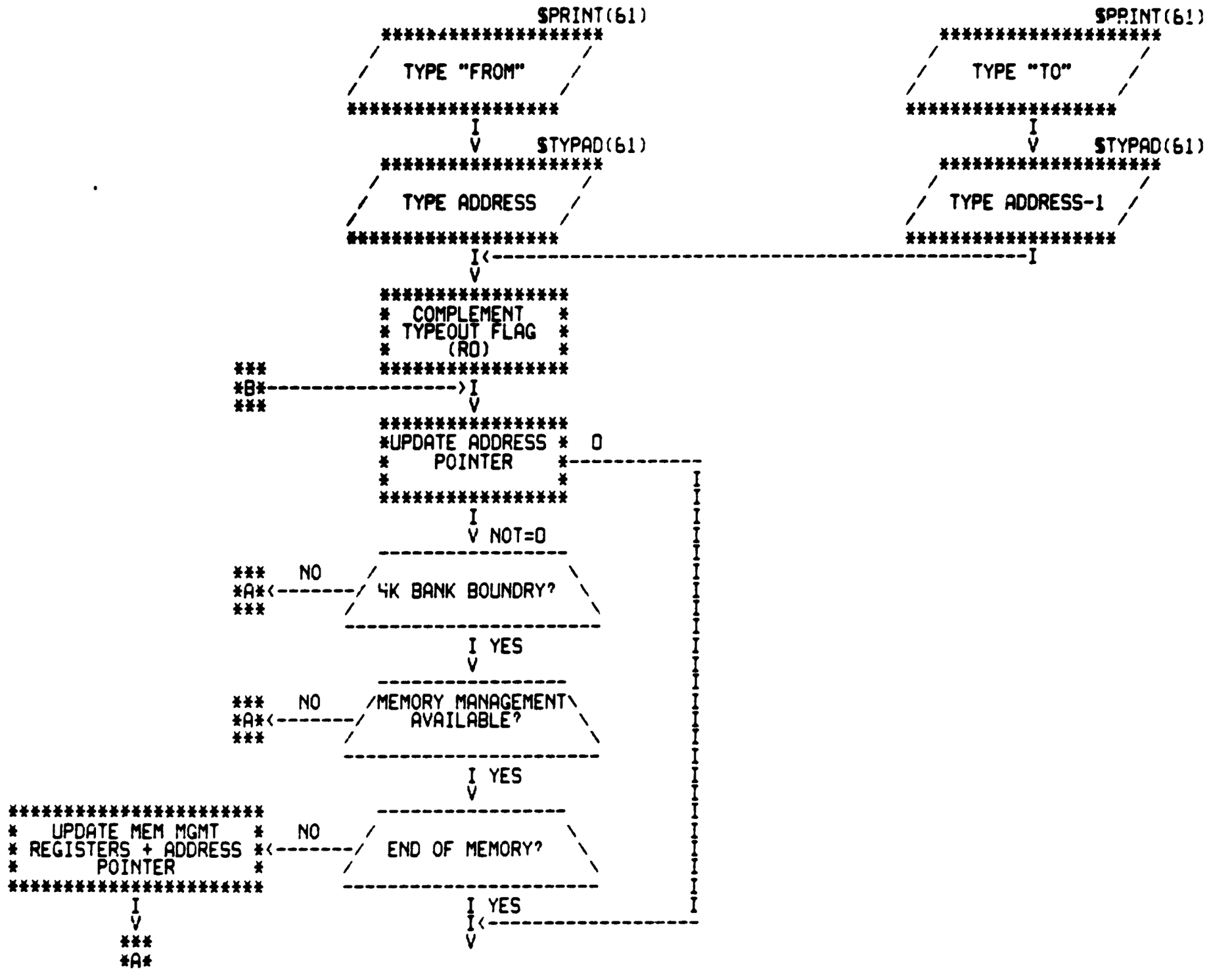

/ CLEAR 'MMAVA'. \YES
/ MEMORY MANAGEMENT \
/ EXITS AND DESIRED? \
NO
*SET UP MEM MGMT. SET *
* 'MMAVA' FLAG TYPE *
* 'KT11 AVAILABLE' *

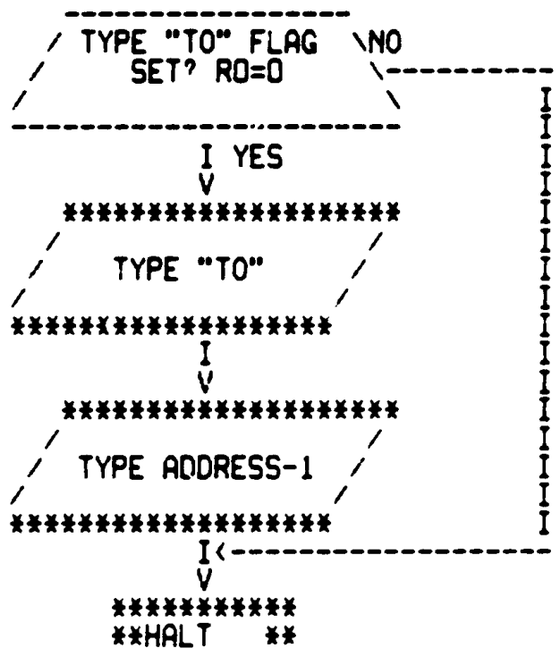
MAP MEMORY


MAPMEM
RESET SP SET UP POINTERS TO MEMORY MAP.
TYPE "MEMORY MAP:"
READ AND WRITE MEMORY LOCATION (AUTO-INC POINTER)
TIMEOUT TRAP
ADJUST SP UPDATE ADP POINTER TO START OF NEXT 4K.
END OF A 4K BANK OF MEMORY REACHED?
NO
YES
SET FLAG IN MAP TO INDICATE FULL 4K. SAVE ADR POINTERS
UPDATE POINTERS TO NEXT 4K BANK. END (128K) REACHED?
NO
YES
DOES FIRST 16K OF MEMORY EXIST?
NO
YES
TYPE INSUFFICIENT MEMORY MESSAGE
GMPR(12)
HALT

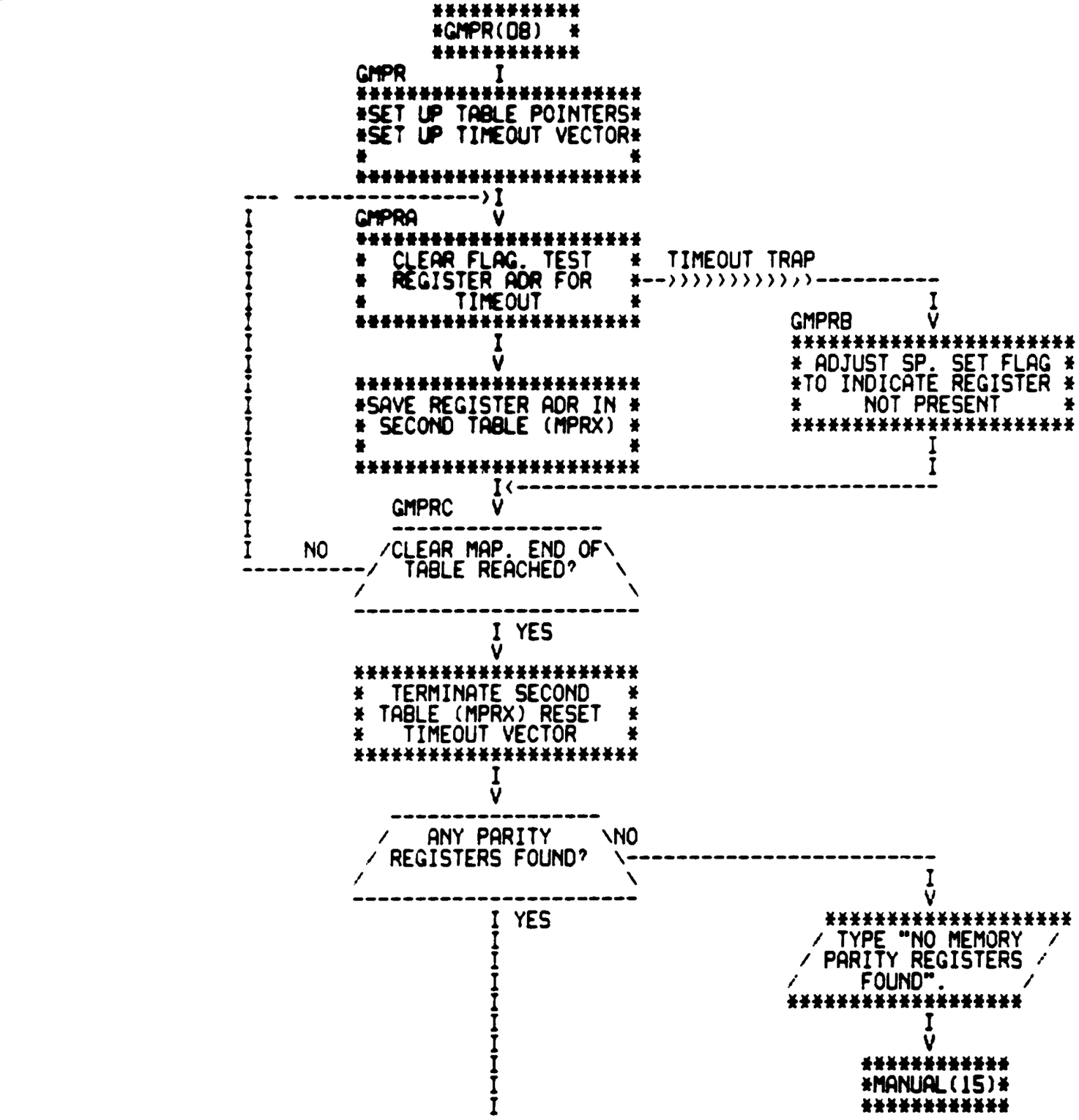
M02

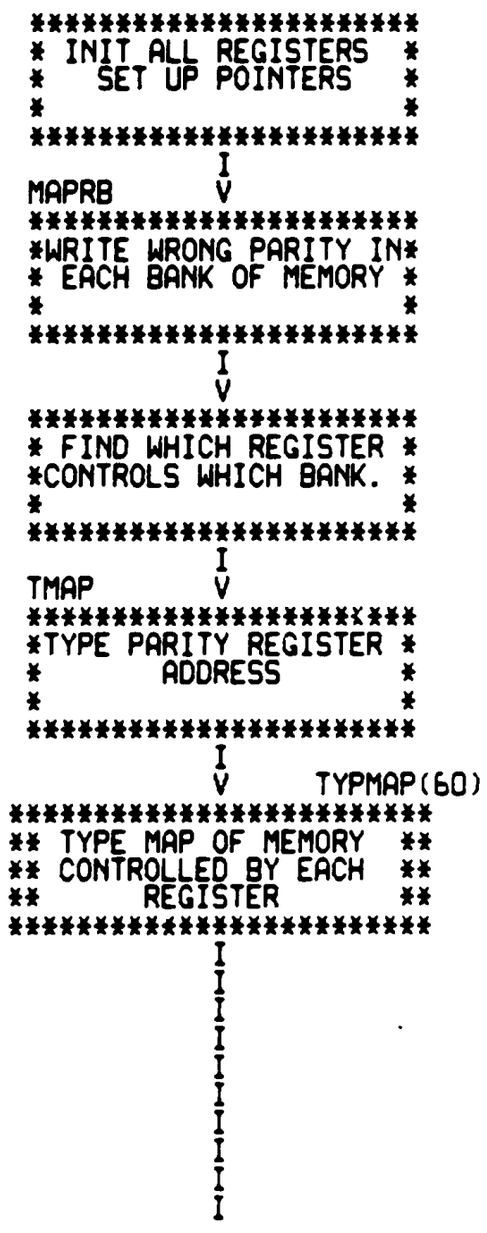

SA=220
**TIMOUT **
MEMORY MANAGEMENT?
NO
YES
MMINIT(53)
SET UP MEMORY MANAGEMENT REGISTERS
$PRINT(61)
TYPE "BYTE MEMORY MAP"
INIT TYPEOUT FLAG
INIT ADDRESS POINTER
SET TIMEOUT VECTOR
A
DO A DATI; THEN DATI, DATO; THEN DATI, DATIP, DATOB
TIMEOUT TRAP
ADJUST SP
TYPE "FROM" FLAG SET? (R0=-1)
TYPE "TO" FLAG SET? (R0=0?)
B

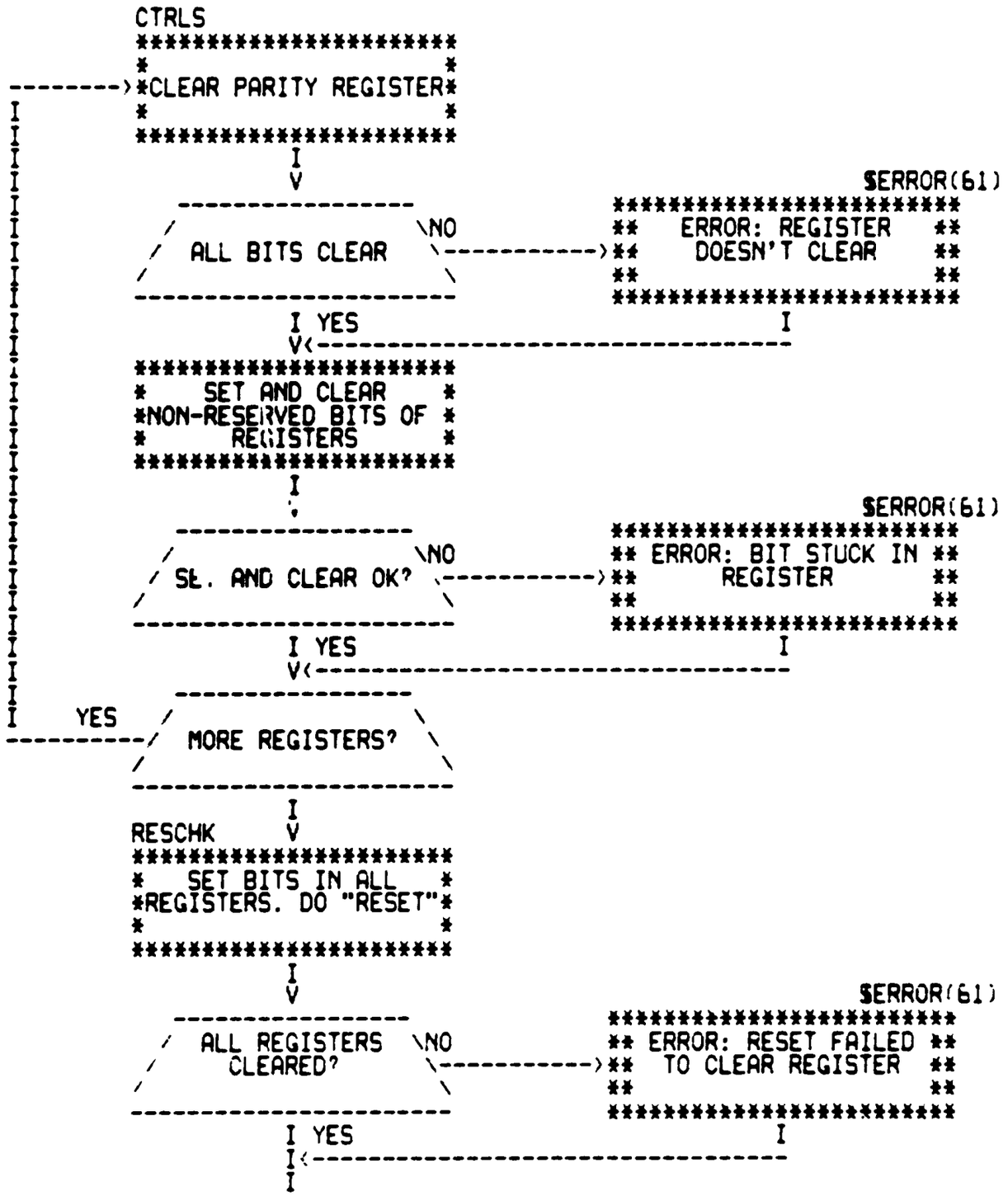

$PRINT(61)
TYPE "FROM"
$TYPAD(61)
TYPE ADDRESS
$PRINT(61)
TYPE "TO"
$TYPAD(61)
TYPE ADDRESS-1
COMPLEMENT TYPEOUT FLAG (RO)
B
UPDATE ADDRESS POINTER
0
NOT=0
NO
A
4K BANK BOUNDRY?
YES
NO
A
MEMORY MANAGEMENT AVAILABLE?
YES
UPDATE MEM MGMT REGISTERS + ADDRESS POINTER
NO
END OF MEMORY?
YES
A


TYPE "TO" FLAG SET? R0=0
NO
YES
TYPE "TO"
TYPE ADDRESS-1
HALT

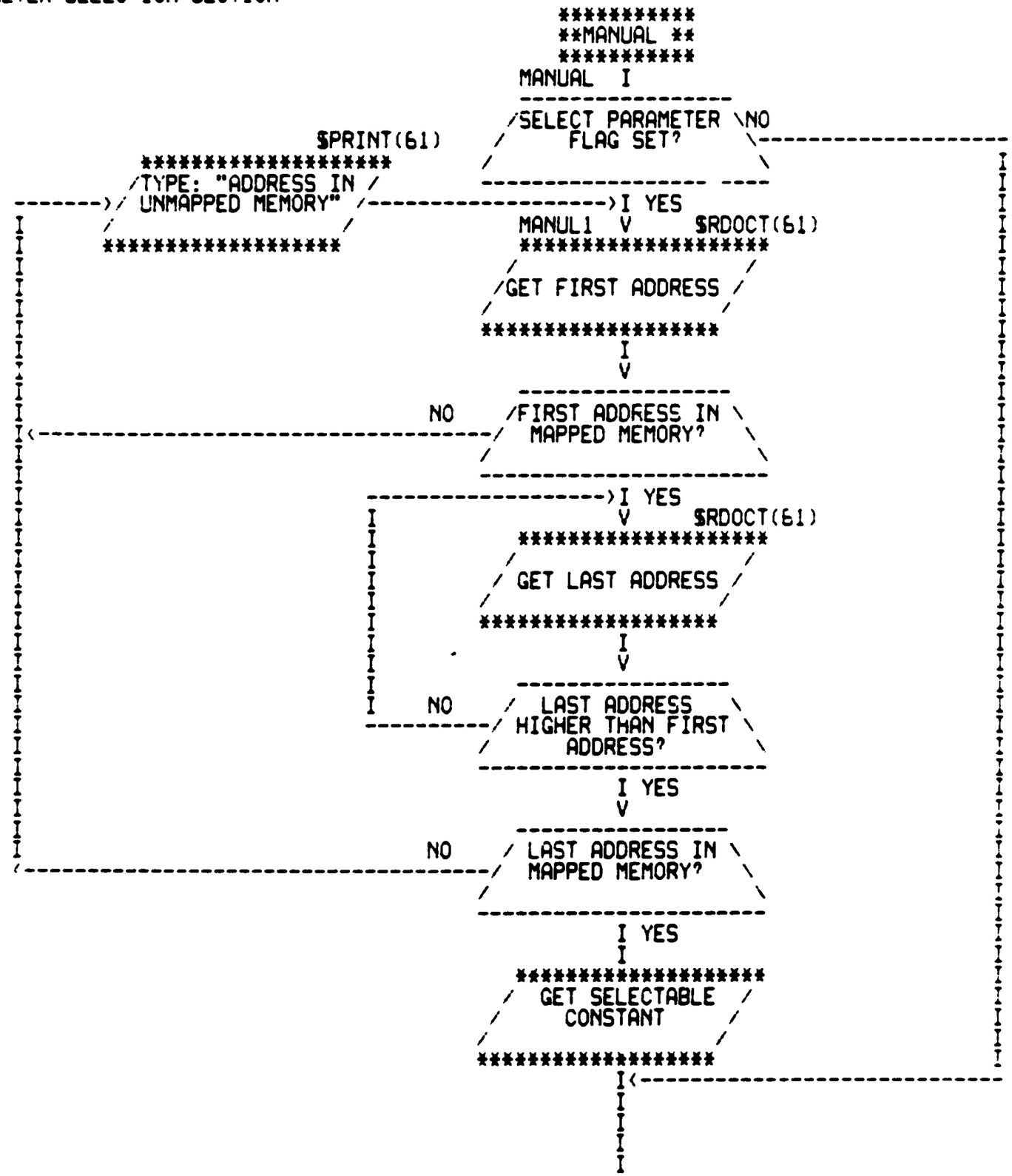
MAP PARITY REGISTERS



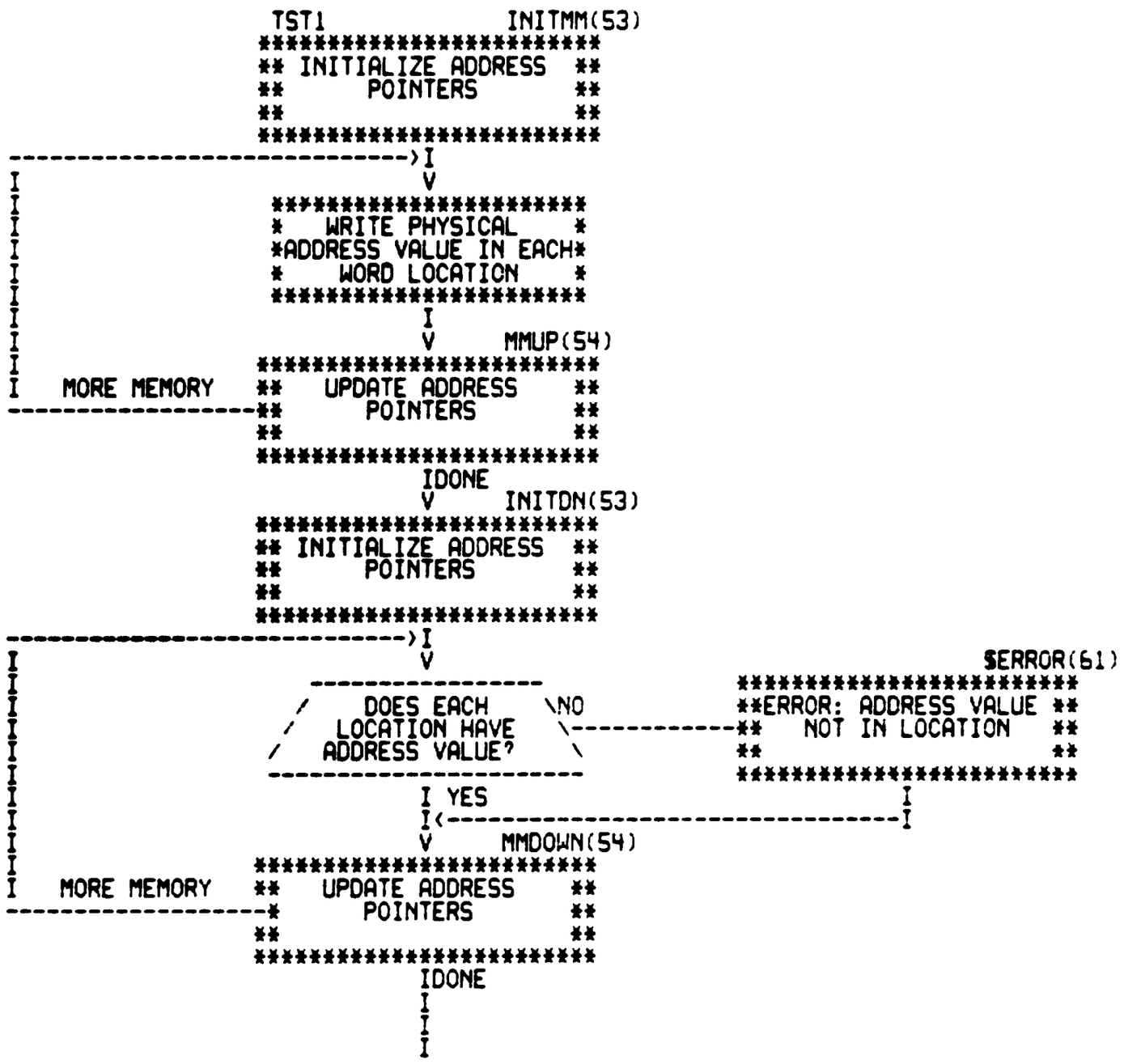

INIT ALL REGISTERS
SET UP POINTERS
MAPRB
WRITE WRONG PARITY IN
EACH BANK OF MEMORY
FIND WHICH REGISTER
CONTROLS WHICH BANK.
TMAP
TYPE PARITY REGISTER
ADDRESS
TYPMAP(60)
TYPE MAP OF MEMORY
CONTROLLED BY EACH
REGISTER

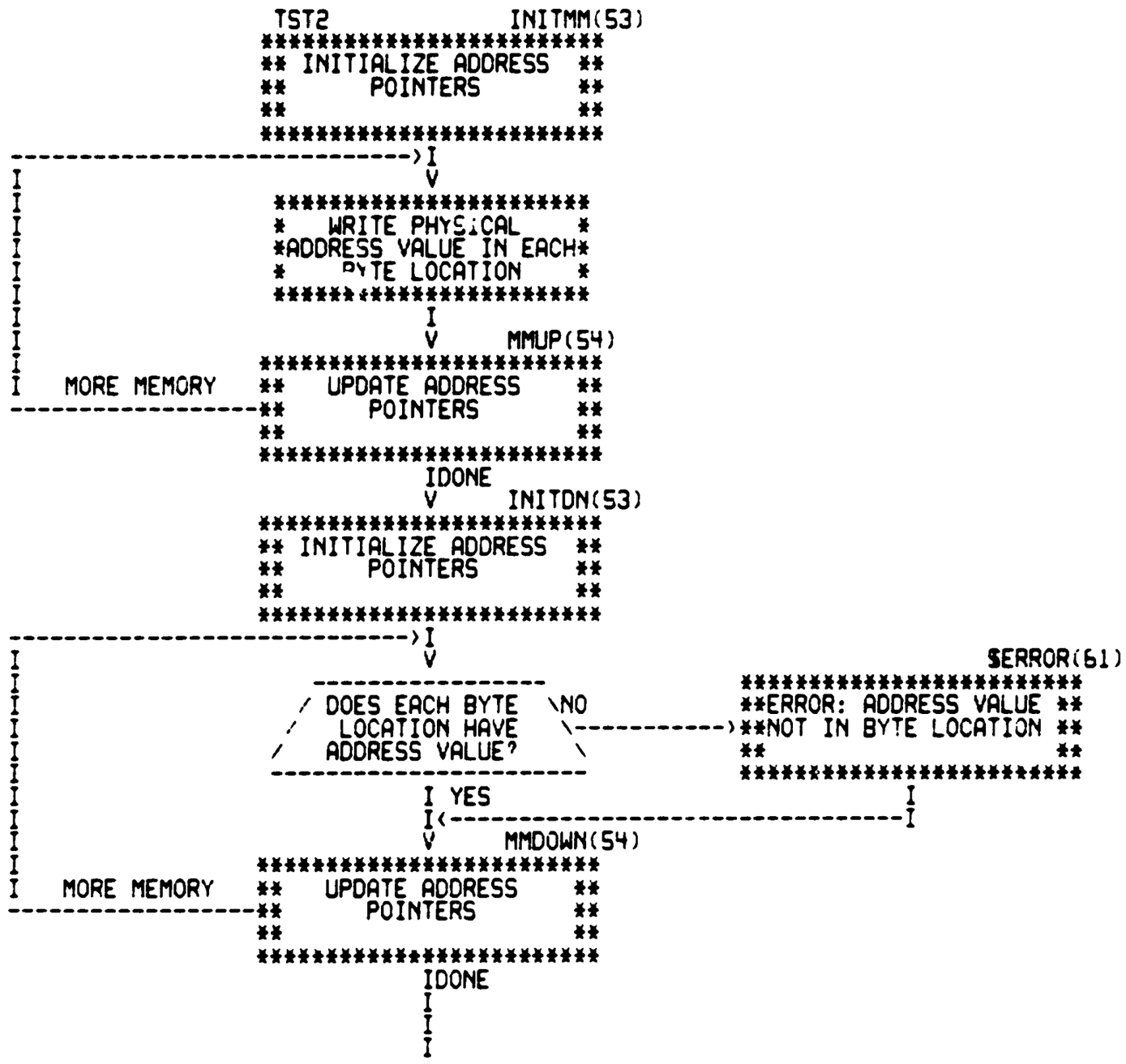

CTRLS
CLEAR PARITY REGISTER
ALL BITS CLEAR
NO
$ERROR(61)
ERROR: REGISTER DOESN'T CLEAR
YES
SET AND CLEAR NON-RESERVED BITS OF REGISTERS
SE. AND CLEAR OK?
NO
$ERROR(61)
ERROR: BIT STUCK IN REGISTER
YES
MORE REGISTERS?
YES
RESCHK
SET BITS IN ALL REGISTERS. DO "RESET"
ALL REGISTERS CLEARED?
NO
$ERROR(61)
ERROR: RESET FAILED TO CLEAR REGISTER
YES

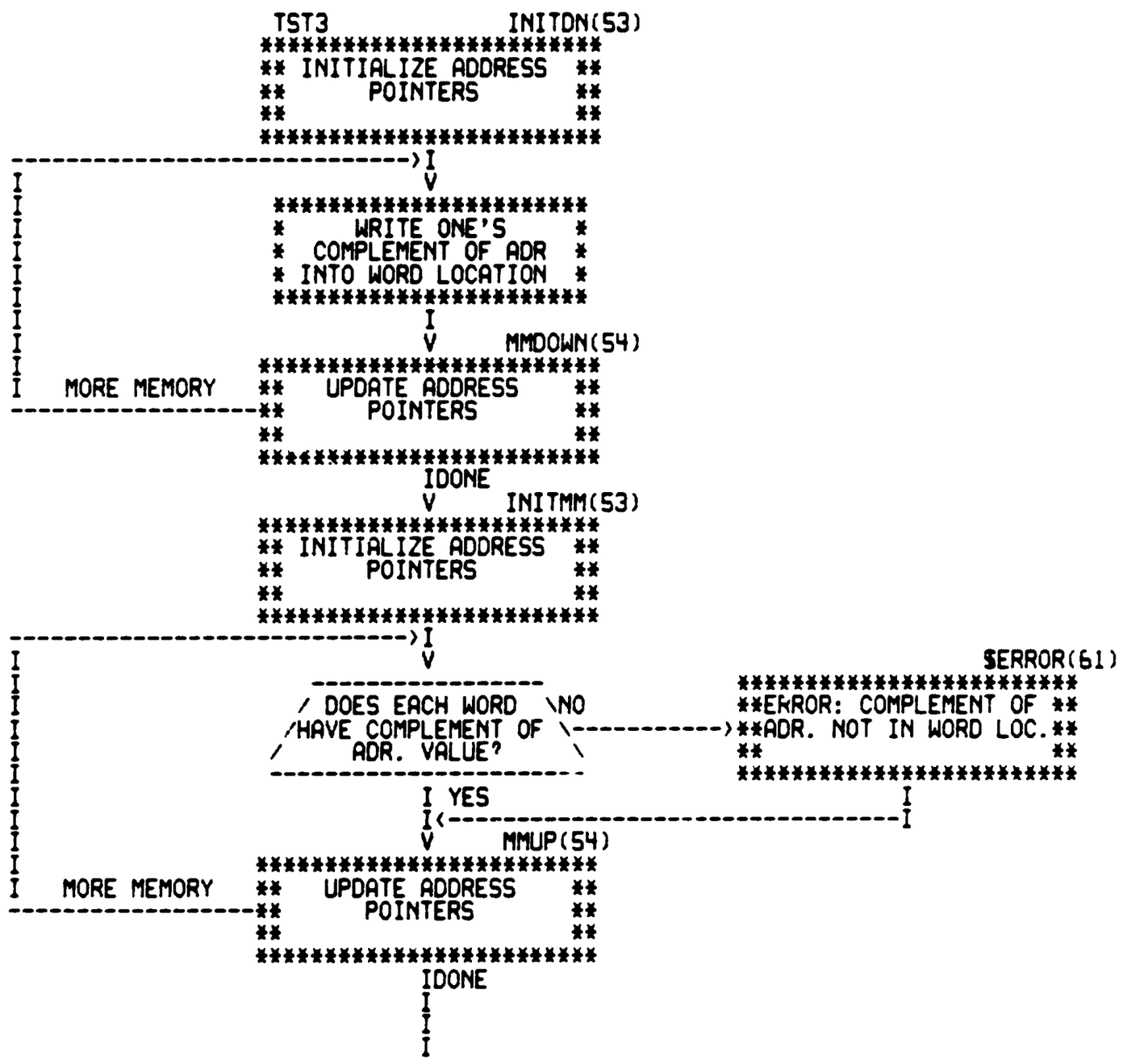
USER PARAMETER SELECTION SECTION

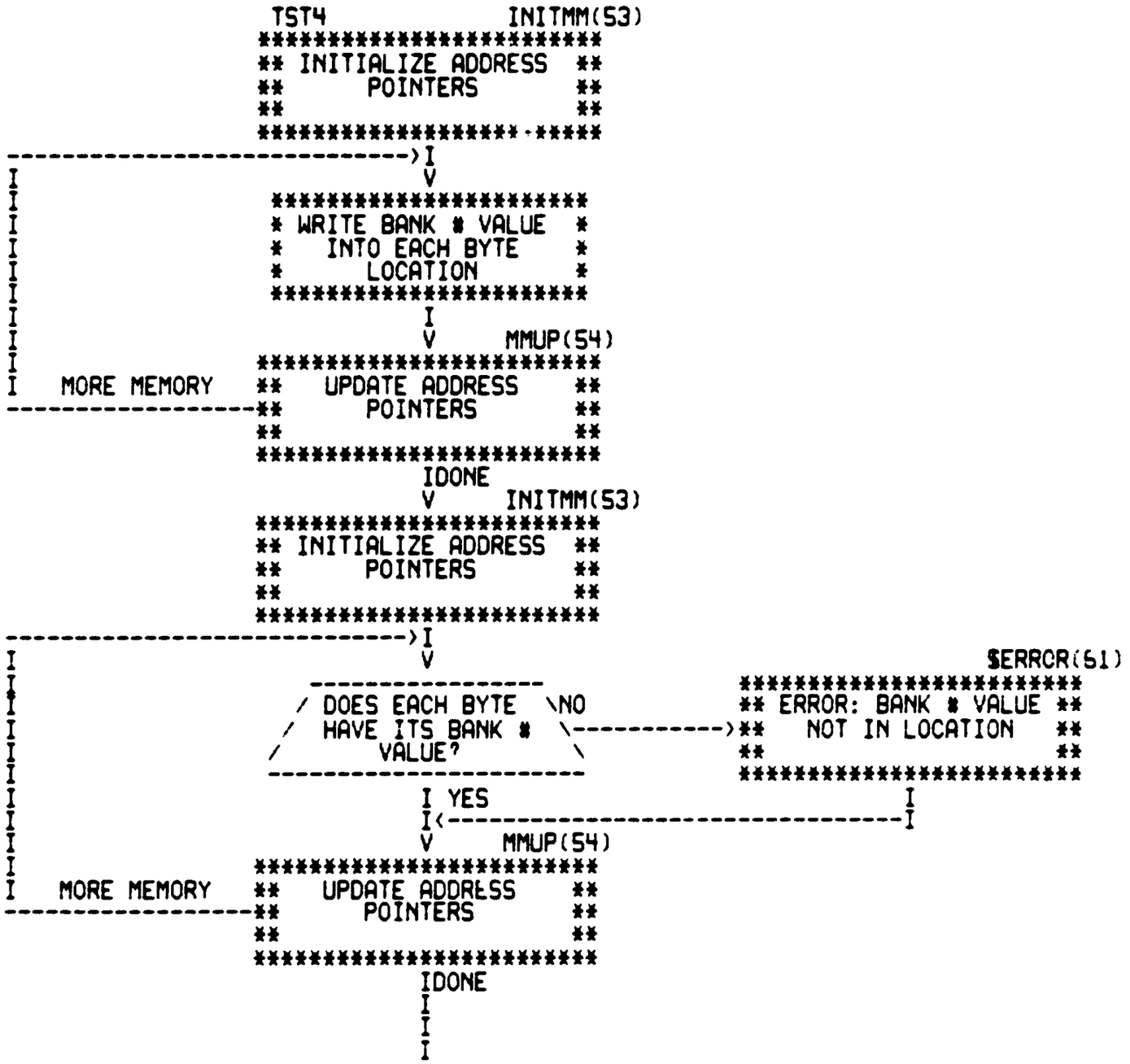


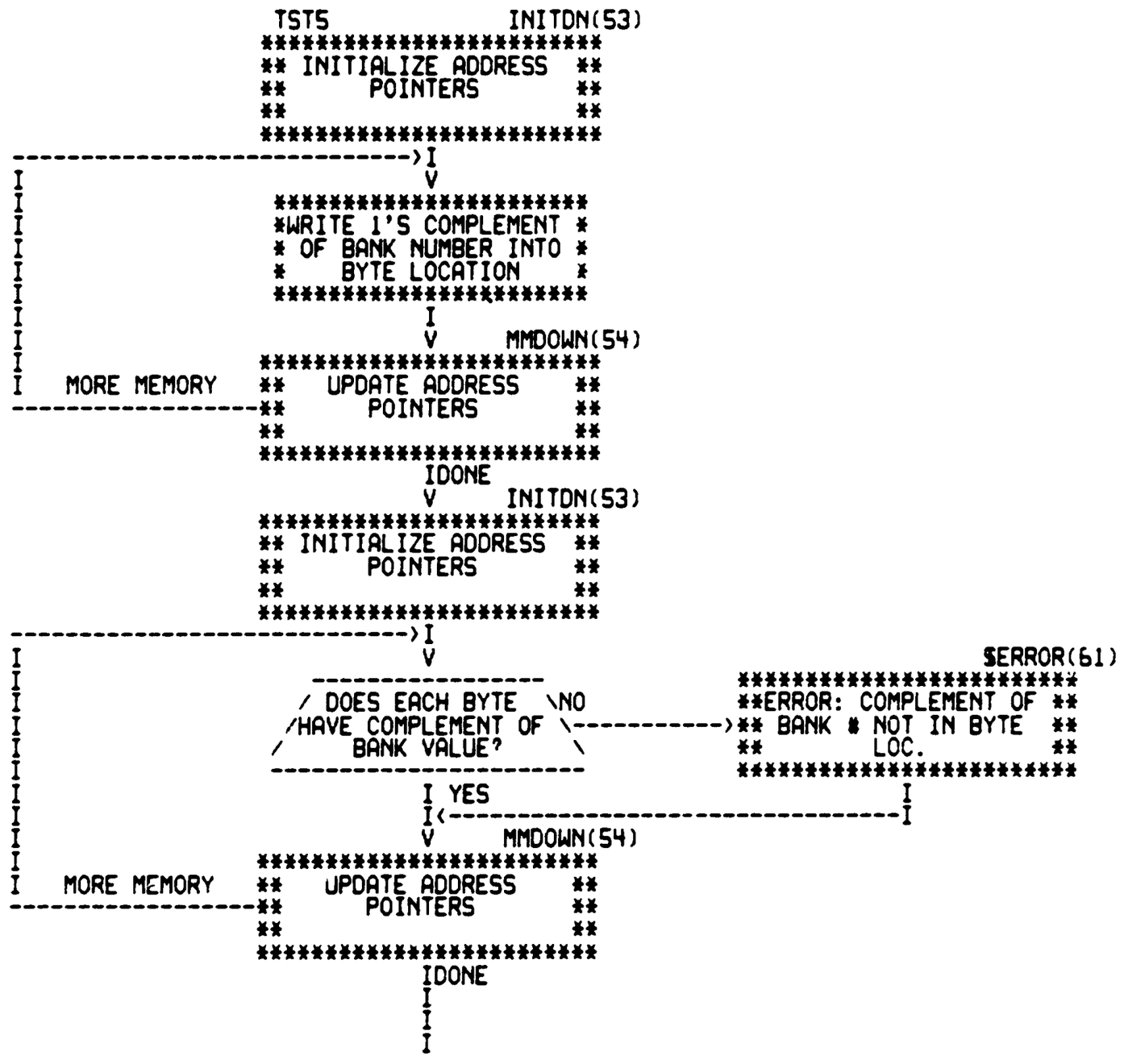
MANUL2
MAKE NECESSARY
ADJUSTMENTS TO FIRST
AND LAST ADDRESSES
START1
START1
START1
INITIALIZE EVERYTING
FOR A NEW PASS

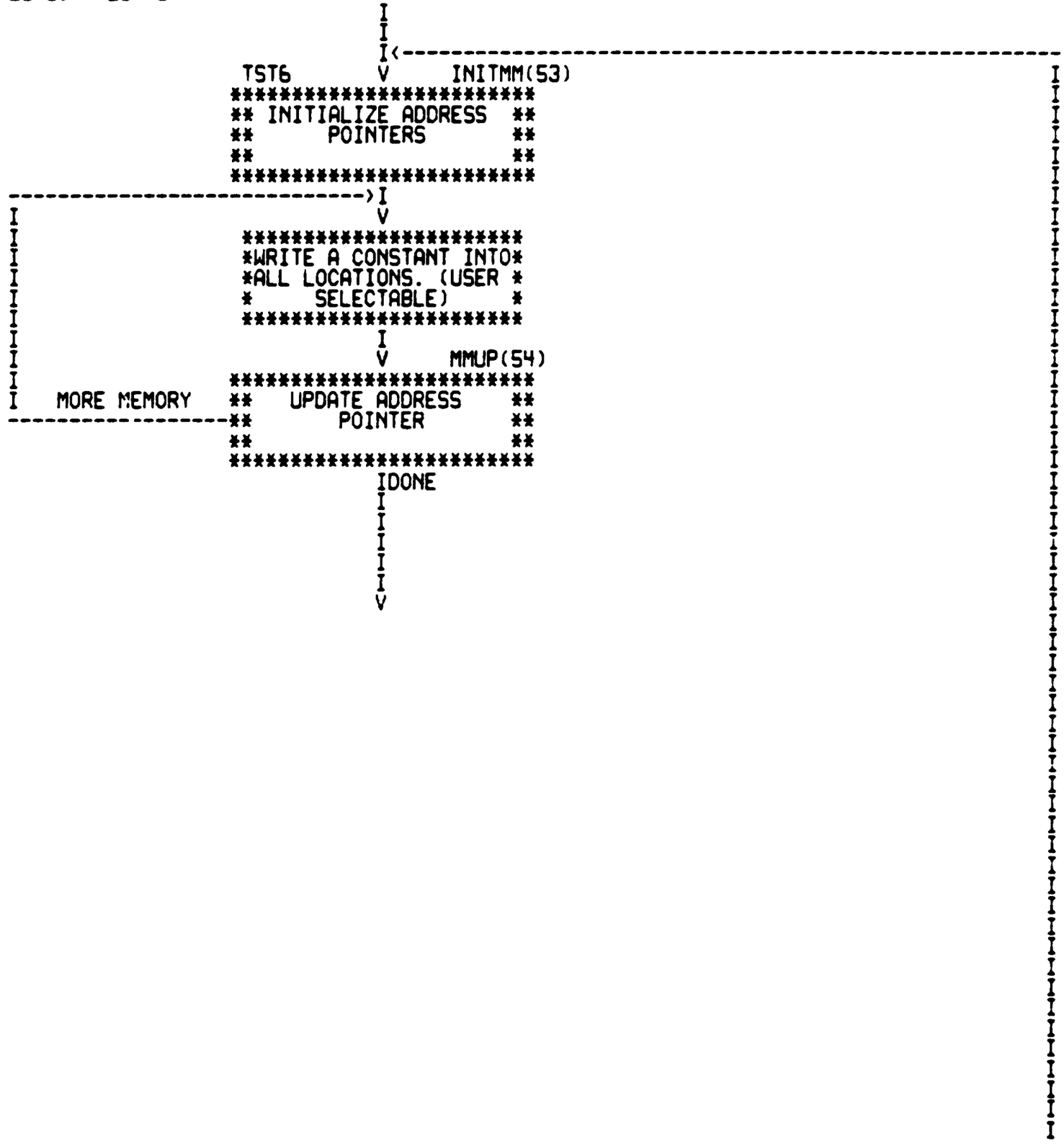

TST1
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE PHYSICAL
ADDRESS VALUE IN EACH
WORD LOCATION
MMUP(54)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE
INITDN(53)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH
LOCATION HAVE
ADDRESS VALUE?
NO
YES
$ERROR(61)
**ERROR: ADDRESS VALUE **
** NOT IN LOCATION **
MMDOWN(54)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE

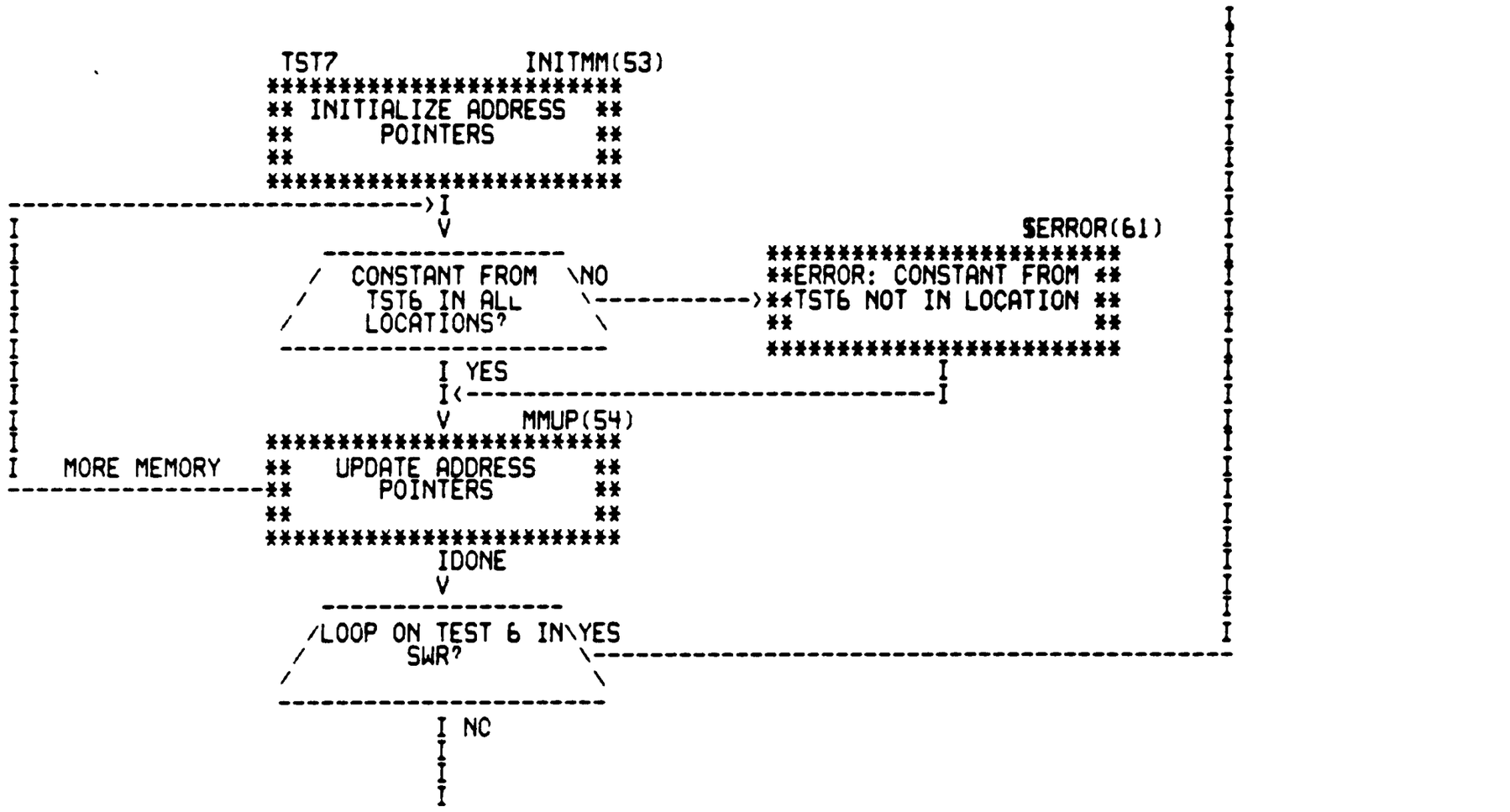

TST2
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE PHYSICAL
ADDRESS VALUE IN EACH
BYTE LOCATION
MMUP(54)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE
INITDN(53)
INITIALIZE ADDRESS
POINTERS
DOES EACH BYTE LOCATION HAVE ADDRESS VALUE?
NO
$ERROR(61)
**ERROR: ADDRESS VALUE **
**NOT IN BYTE LOCATION **
YES
MMDOWN(54)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE

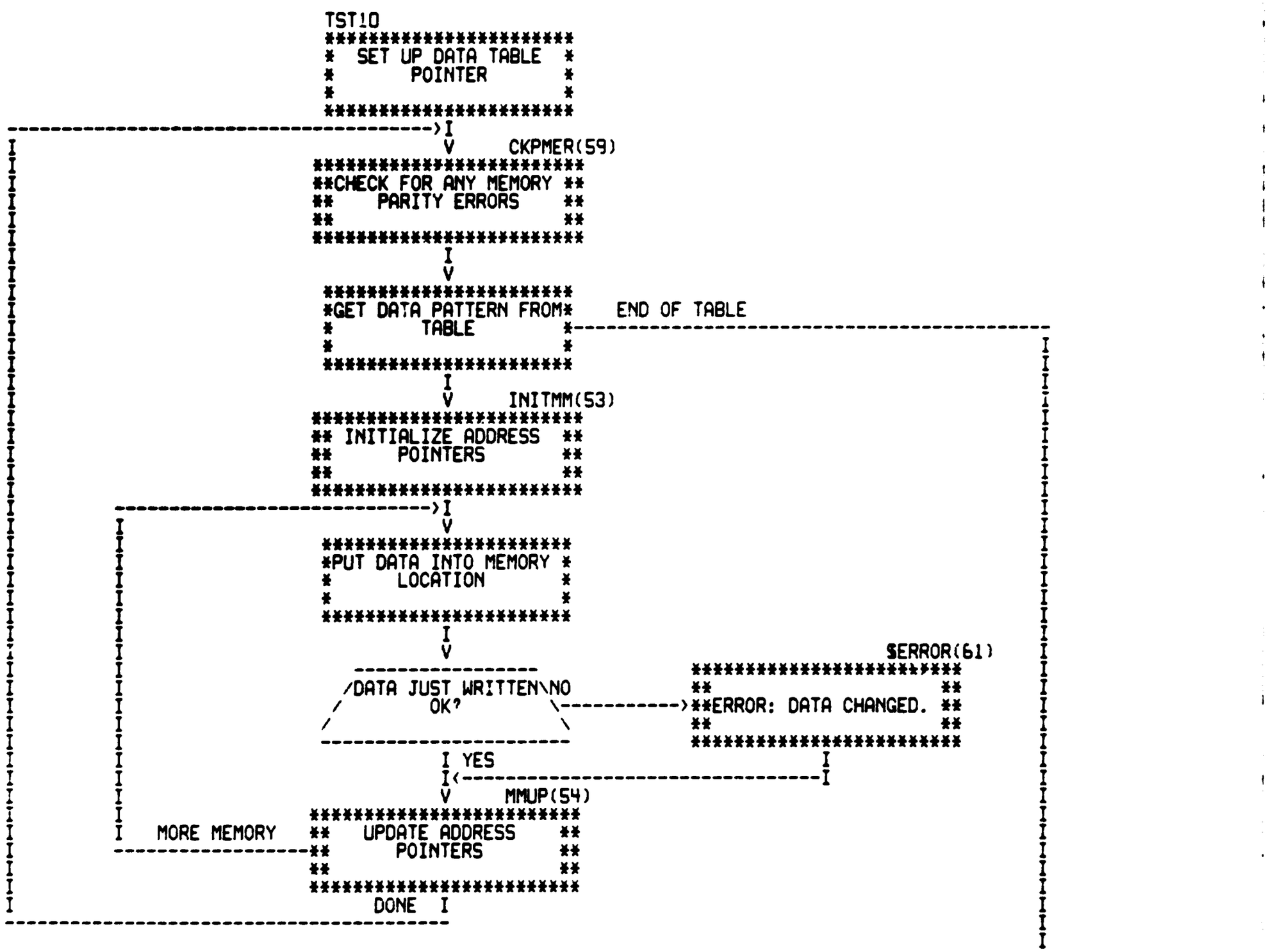

TST3
INITDN(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE ONE'S COMPLEMENT OF ADR INTO WORD LOCATION
MMDOWN(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH WORD HAVE COMPLEMENT OF ADR. VALUE?
NO
$ERROR(61)
**ERROR: COMPLEMENT OF **
ADR. NOT IN WORD LOC.
YES
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

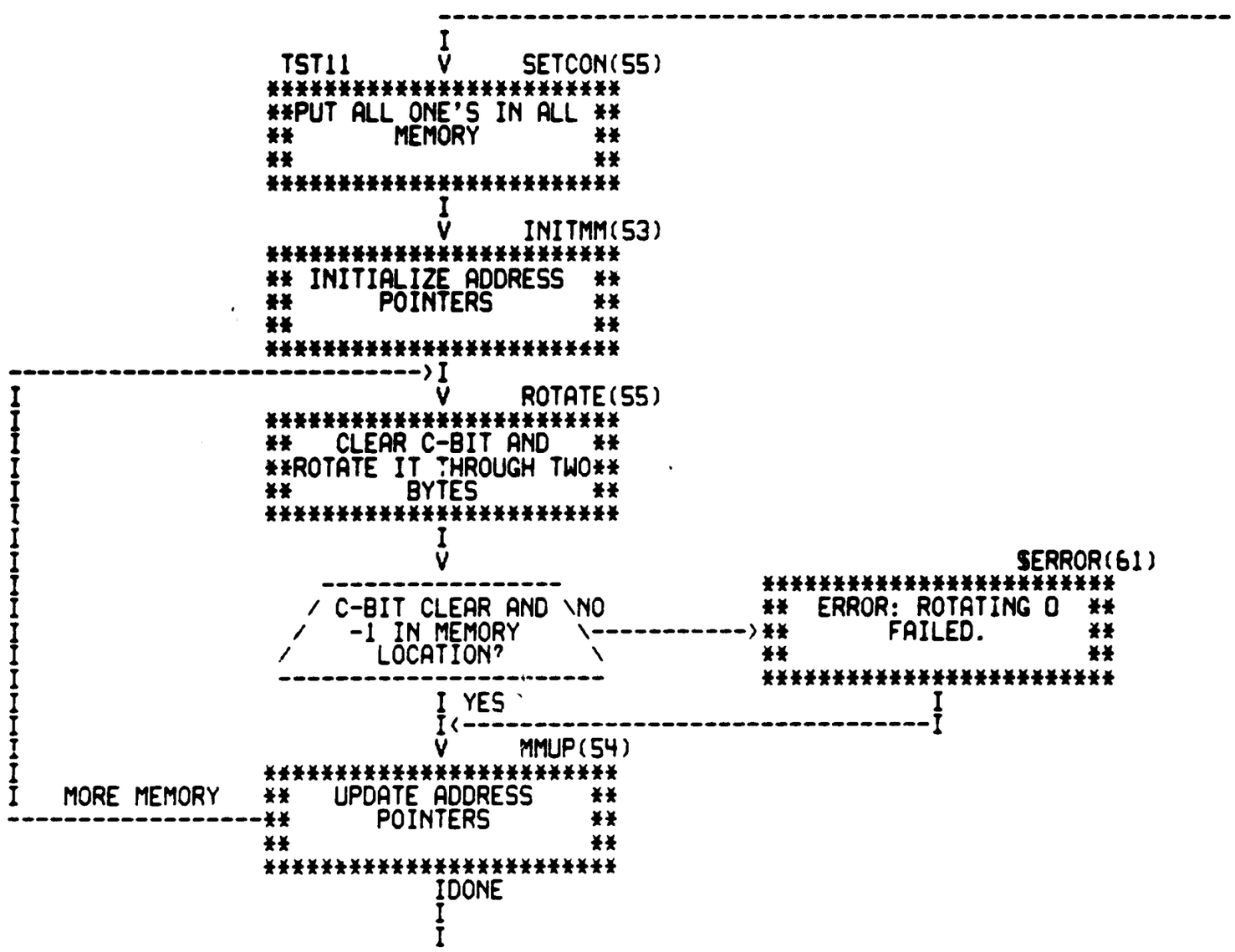

TST4
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE BANK # VALUE
INTO EACH BYTE
LOCATION
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH BYTE HAVE ITS BANK # VALUE?
NO
$ERROR(61)
** ERROR: BANK # VALUE **
** NOT IN LOCATION **
YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

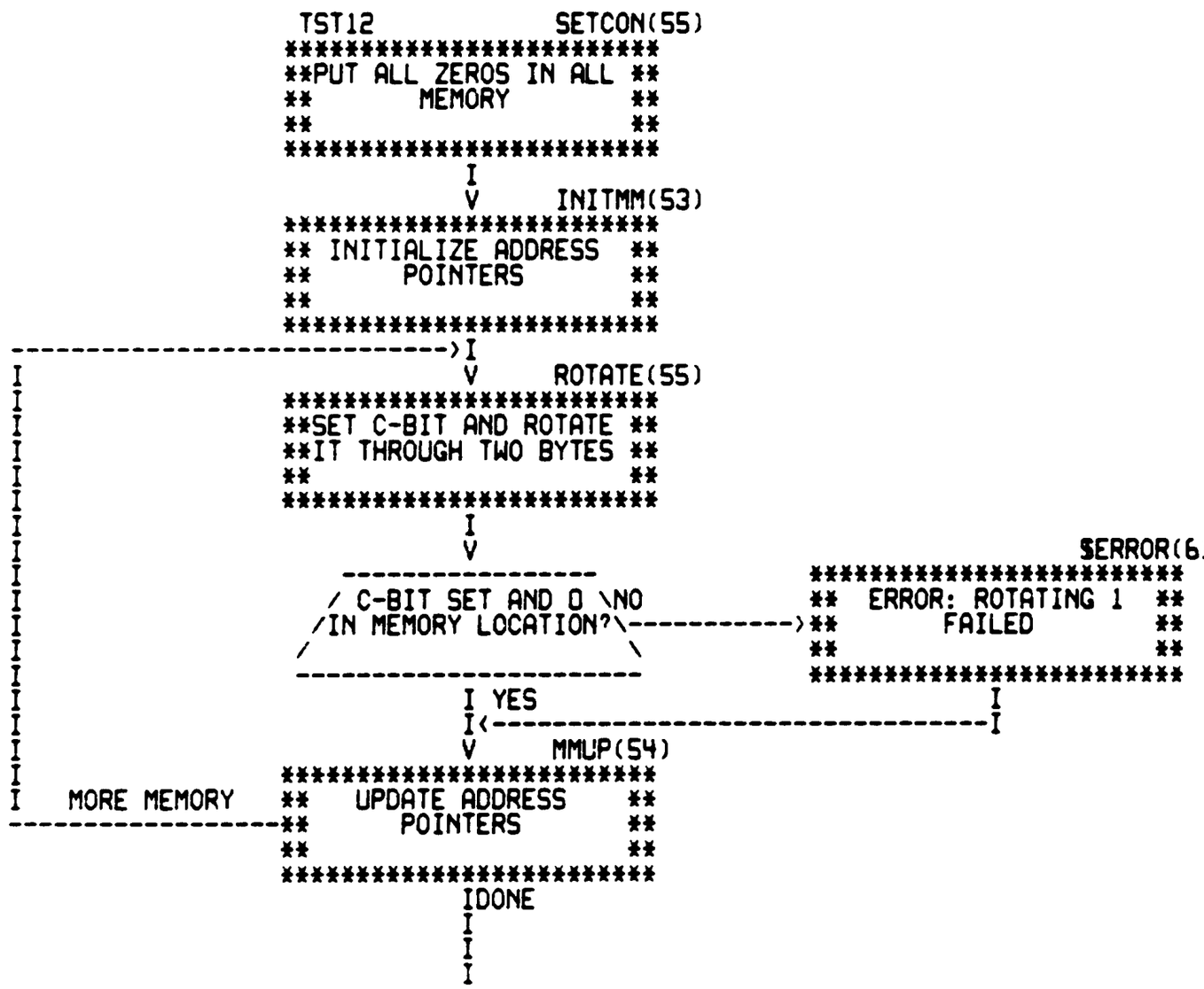

TST5
INITDN(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE 1'S COMPLEMENT OF BANK NUMBER INTO BYTE LOCATION
MMDOWN(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITDN(53)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH BYTE HAVE COMPLEMENT OF BANK VALUE?
NO
$ERROR(61)
**ERROR: COMPLEMENT OF BANK # NOT IN BYTE LOC.
YES
MMDOWN(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

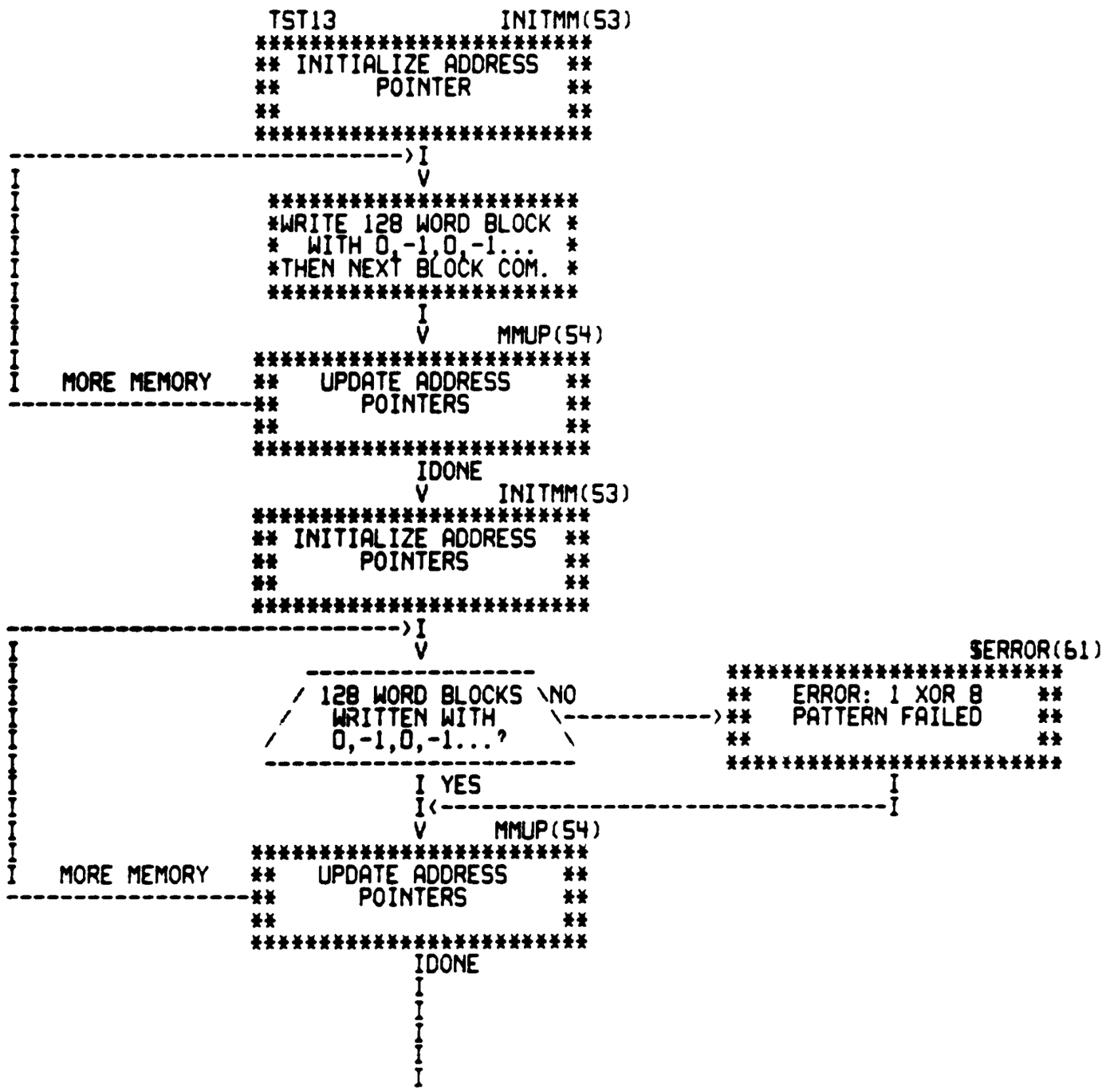

TST6
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE A CONSTANT INTO
*ALL LOCATIONS. (USER *
* SELECTABLE) *
MMUP(54)
** UPDATE ADDRESS **
** POINTER **
MORE MEMORY
IDONE

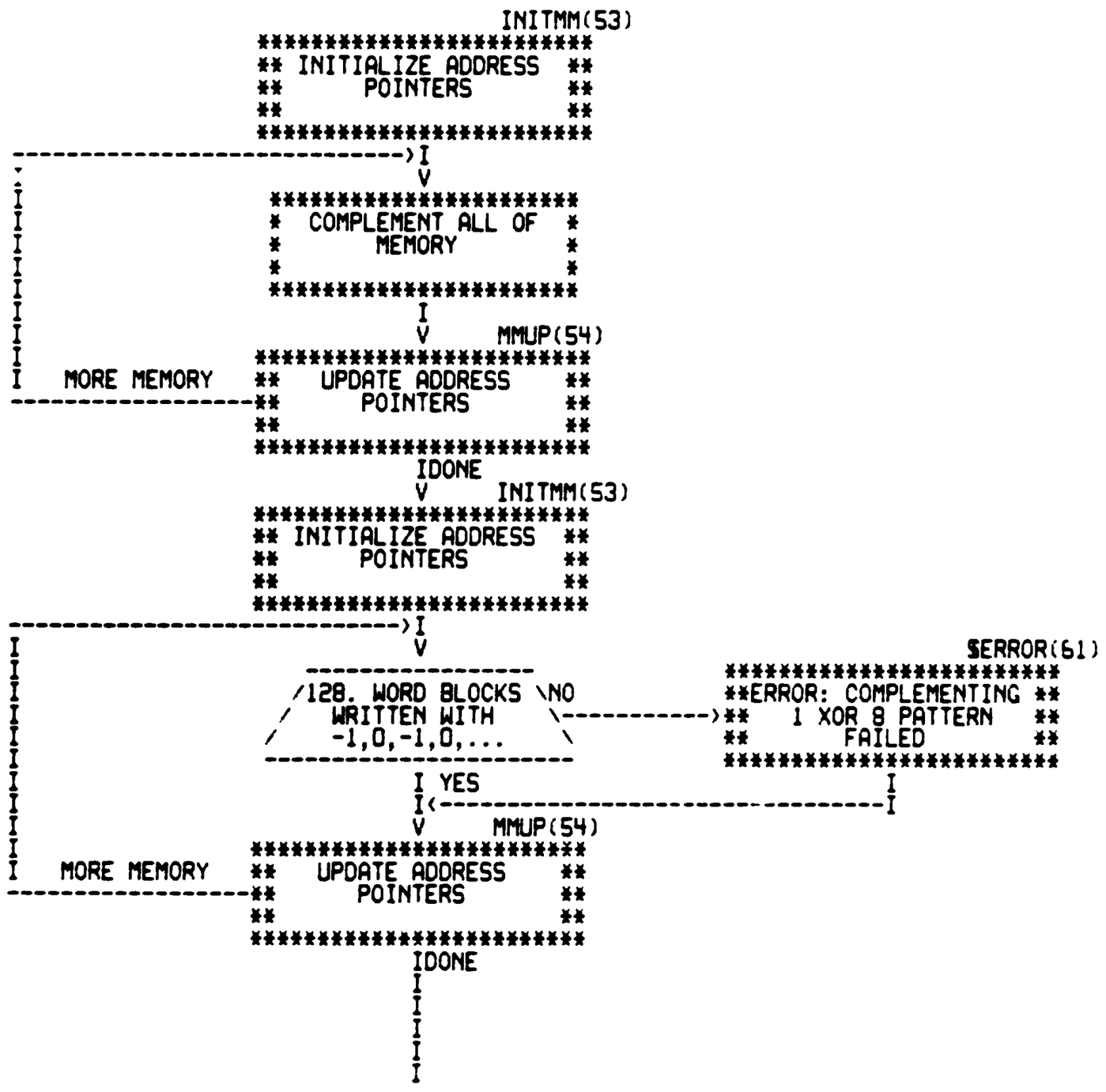

TST7
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
CONSTANT FROM TST6 IN ALL LOCATIONS?
NO
$ERROR(61)
**ERROR: CONSTANT FROM **TST6 NOT IN LOCATION **
YES
MMUP(54)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
IDONE
LOOP ON TEST 6 IN SWR?
YES
NO

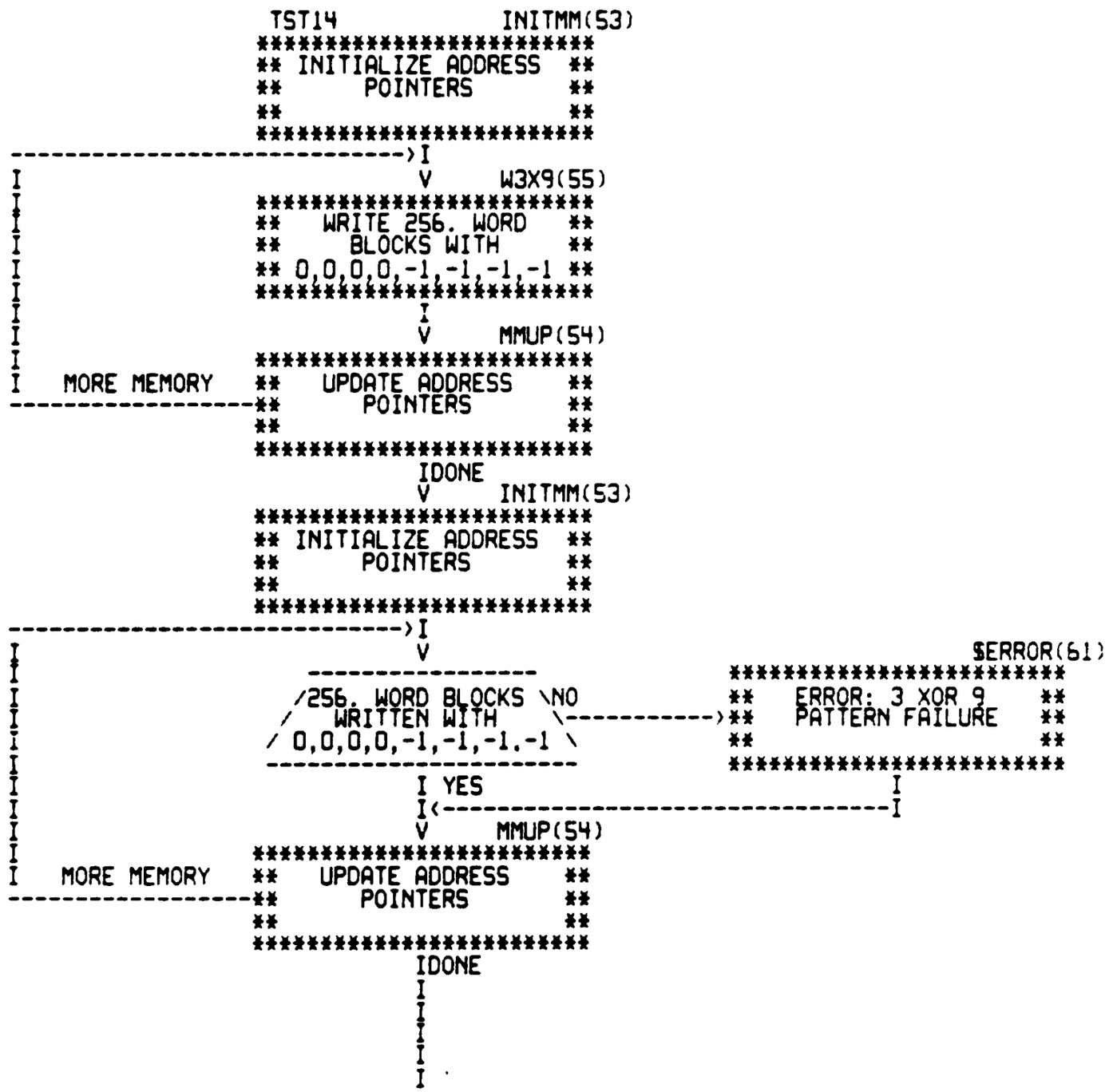

TST10
SET UP DATA TABLE POINTER
CKPMER(59)
CHECK FOR ANY MEMORY PARITY ERRORS
GET DATA PATTERN FROM TABLE
END OF TABLE
INITMM(53)
INITIALIZE ADDRESS POINTERS
PUT DATA INTO MEMORY LOCATION
DATA JUST WRITTEN OK?
NO
$ERROR(61)
ERROR: DATA CHANGED.
YES
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

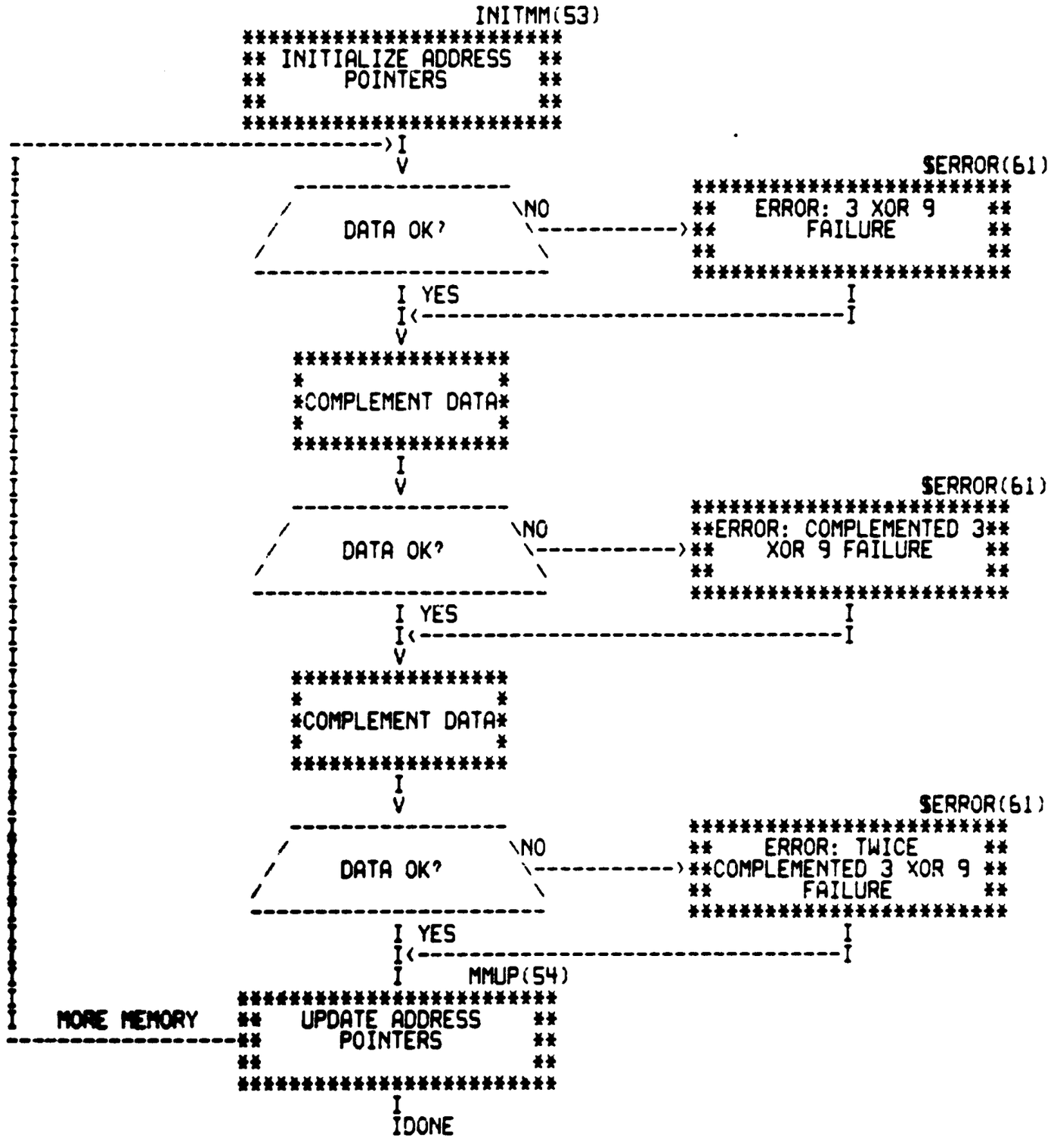

TST11
SETCON(55)
PUT ALL ONE'S IN ALL MEMORY
INITMM(53)
** INITIALIZE ADDRESS POINTERS **
ROTATE(55)
** CLEAR C-BIT AND ROTATE IT THROUGH TWO BYTES **
C-BIT CLEAR AND -1 IN MEMORY LOCATION?
NO
YES
$ERROR(61)
** ERROR: ROTATING 0 FAILED. **
MMUP(54)
** UPDATE ADDRESS POINTERS **
MORE MEMORY
IDONE

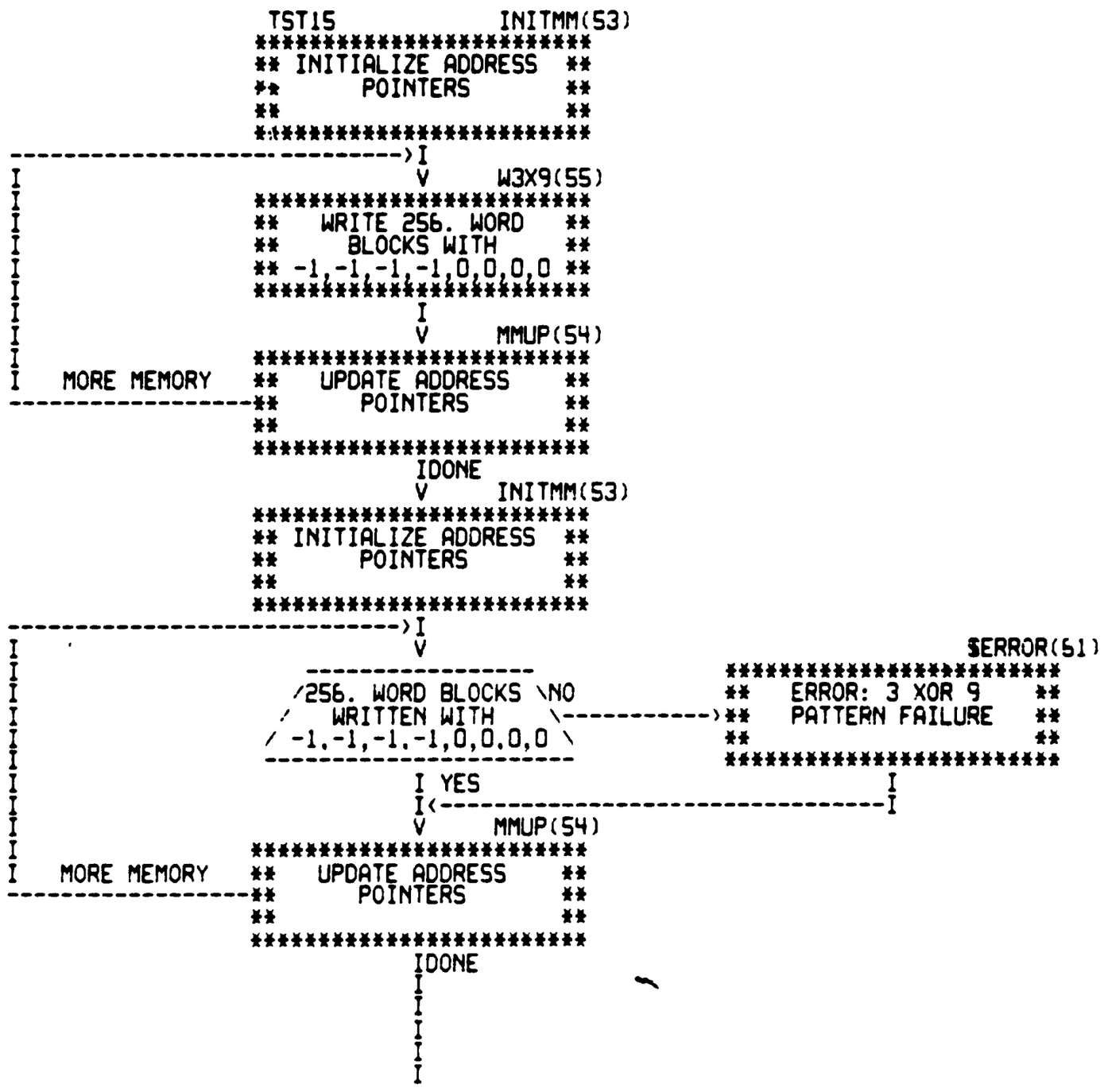

TST12
SETCON(55)
PUT ALL ZEROS IN ALL MEMORY
INITMM(53)
** INITIALIZE ADDRESS POINTERS **
ROTATE(55)
SET C-BIT AND ROTATE IT THROUGH TWO BYTES
C-BIT SET AND 0 IN MEMORY LOCATION?
NO
SERROR(61)
** ERROR: ROTATING 1 FAILED **
YES
MMUP(54)
** UPDATE ADDRESS POINTERS **
MORE MEMORY
IDONE

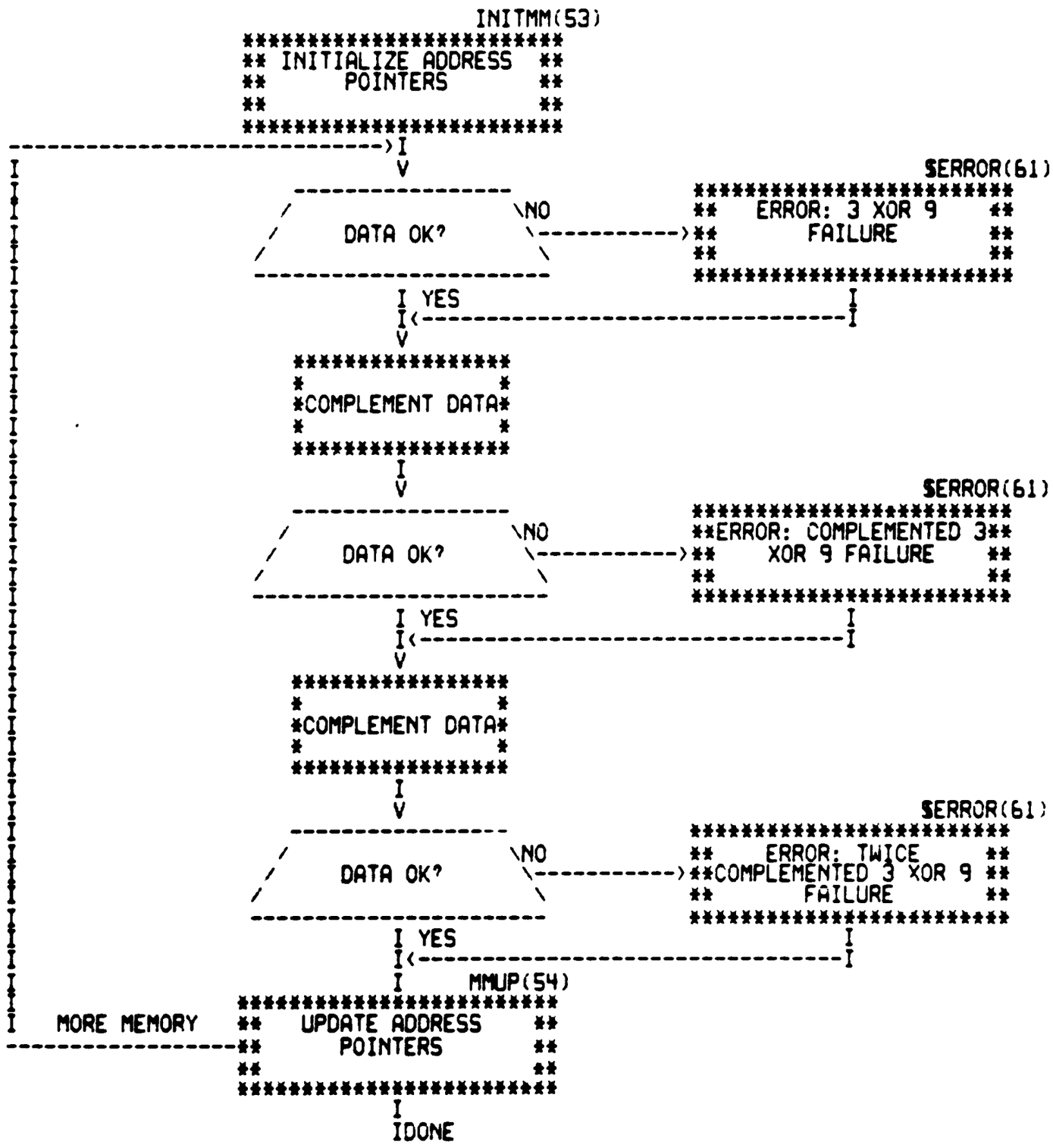

TST13
INITMM(53)
** INITIALIZE ADDRESS **
** POINTER **
*WRITE 128 WORD BLOCK *
* WITH 0,-1,0,-1... *
*THEN NEXT BLOCK COM. *
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
128 WORD BLOCKS WRITTEN WITH 0,-1,0,-1...?
NO
$ERROR(61)
** ERROR: 1 XOR 8 **
** PATTERN FAILED **
YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
IDONE
MORE MEMORY
MORE MEMORY

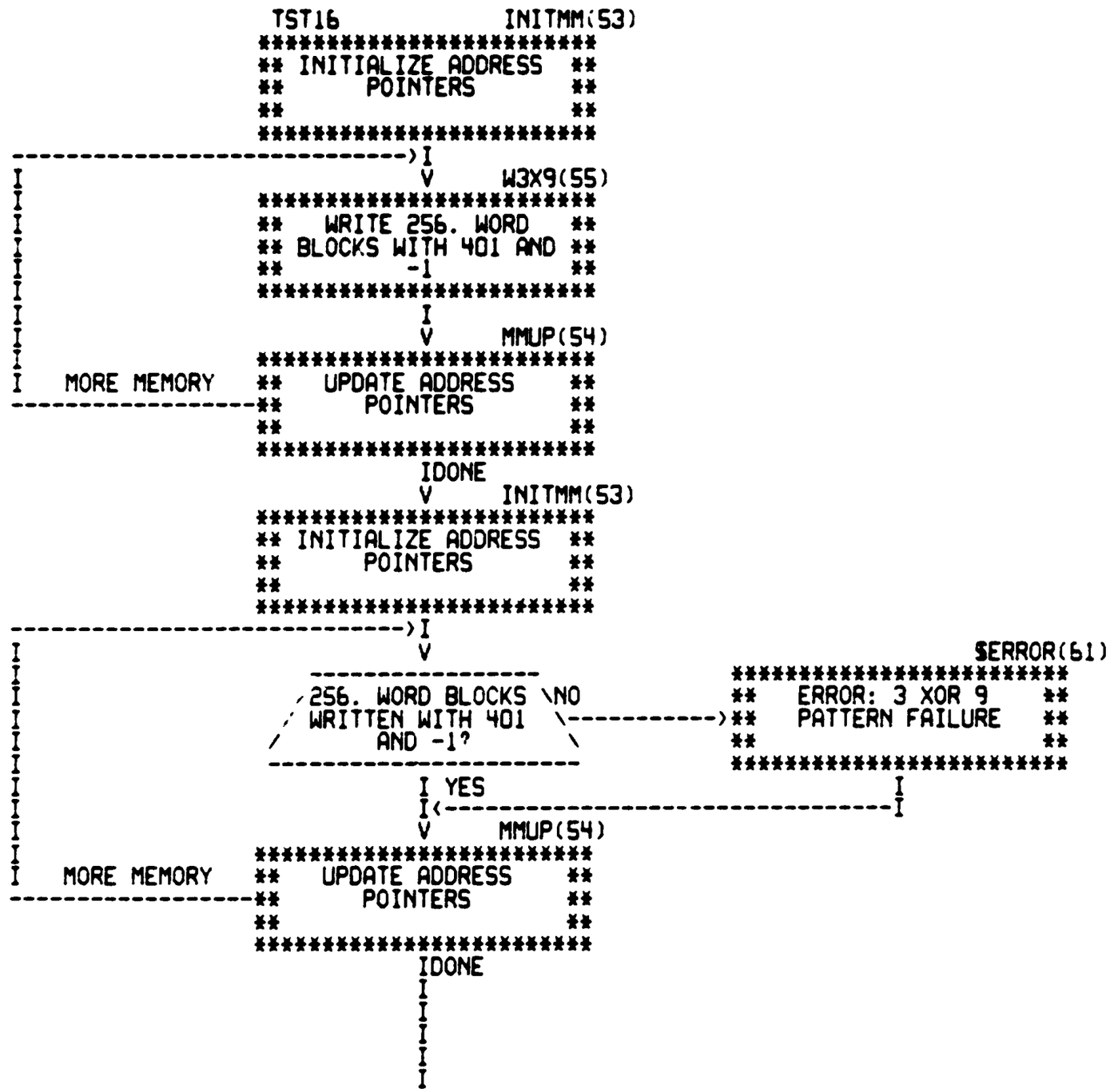

INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
/128. WORD BLOCKS \NO
/ WRITTEN WITH \
/ -1,0,-1,0,... \
I YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(61)
**ERROR: COMPLEMENTING **
** 1 XOR 8 PATTERN **
** FAILED **

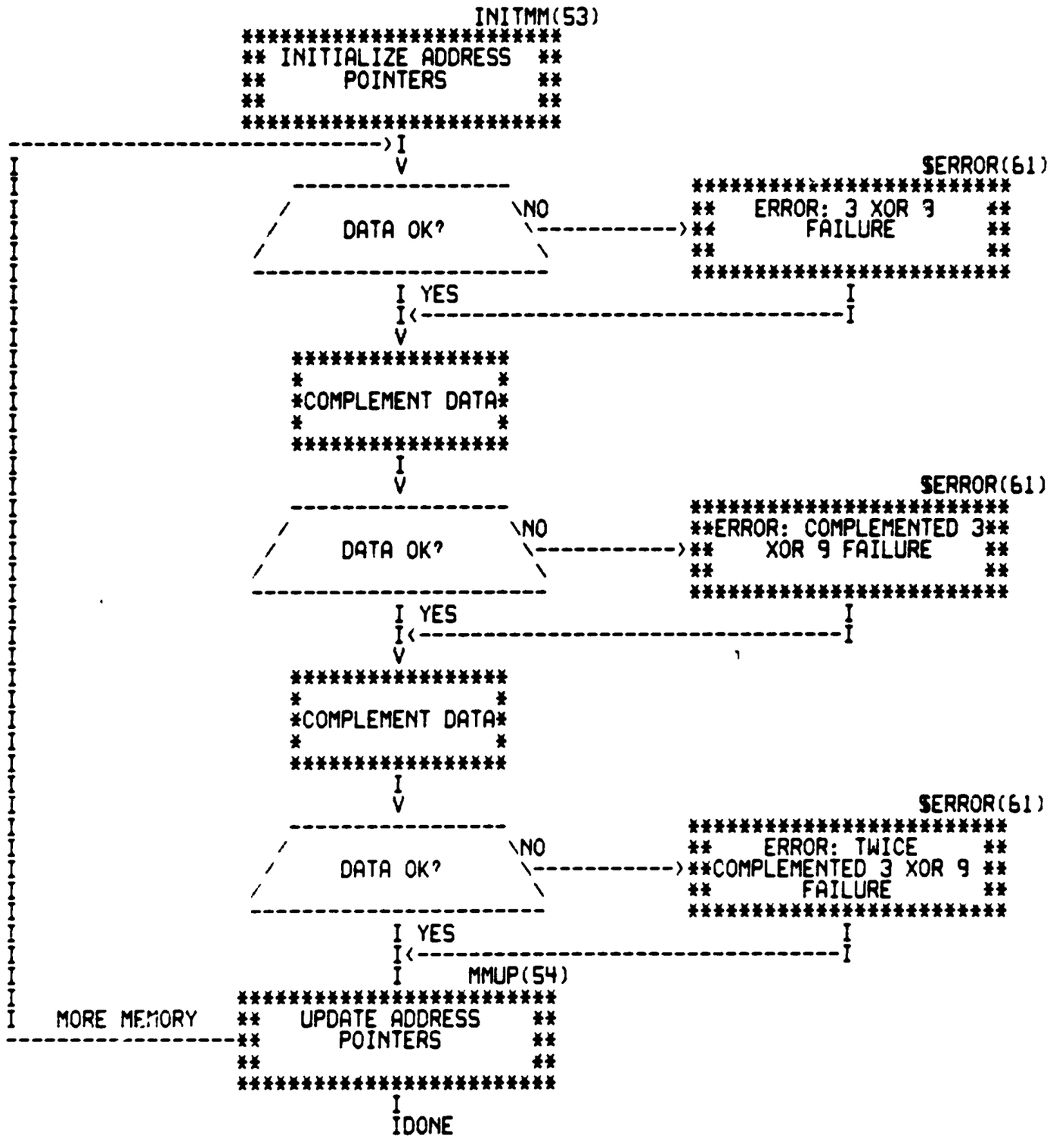

TST14
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(55)
** WRITE 256. WORD **
** BLOCKS WITH **
** 0,0,0,0,-1,-1,-1,-1 **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/ WRITTEN WITH \
/ 0,0,0,0,-1,-1,-1,-1 \
I YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(61)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

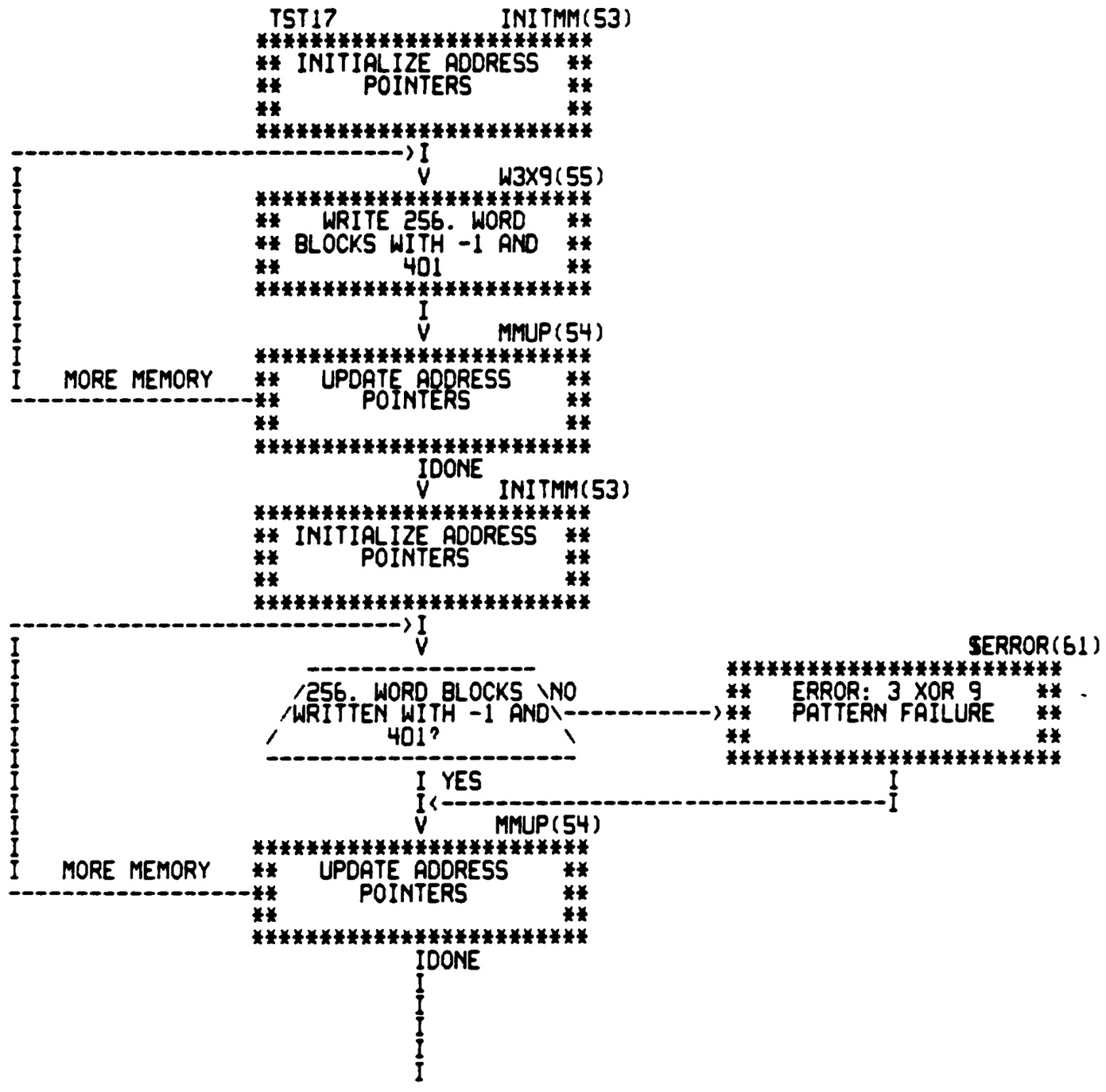

INITMM(53)
INITIALIZE ADDRESS POINTERS
DATA OK?
NO
$ERROR(61)
ERROR: 3 XOR 9 FAILURE
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
ERROR: COMPLEMENTED 3 XOR 9 FAILURE
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
ERROR: TWICE COMPLEMENTED 3 XOR 9 FAILURE
YES
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

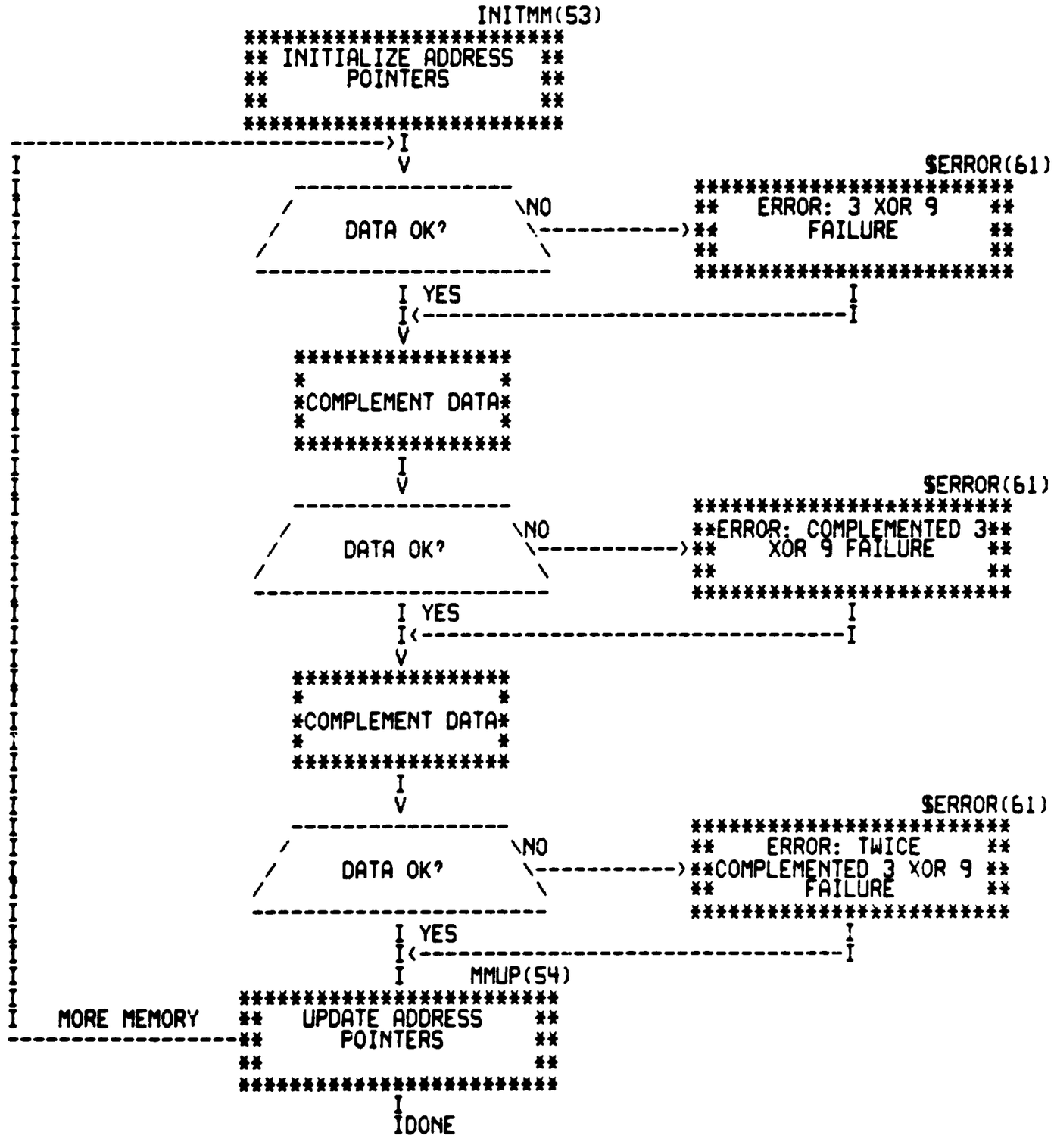

TST15
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
IDONE
W3X9(55)
** WRITE 256. WORD **
** BLOCKS WITH **
** -1,-1,-1,-1,0,0,0,0 **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
256. WORD BLOCKS
WRITTEN WITH
-1,-1,-1,-1,0,0,0,0
NO
YES
$ERROR(51)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

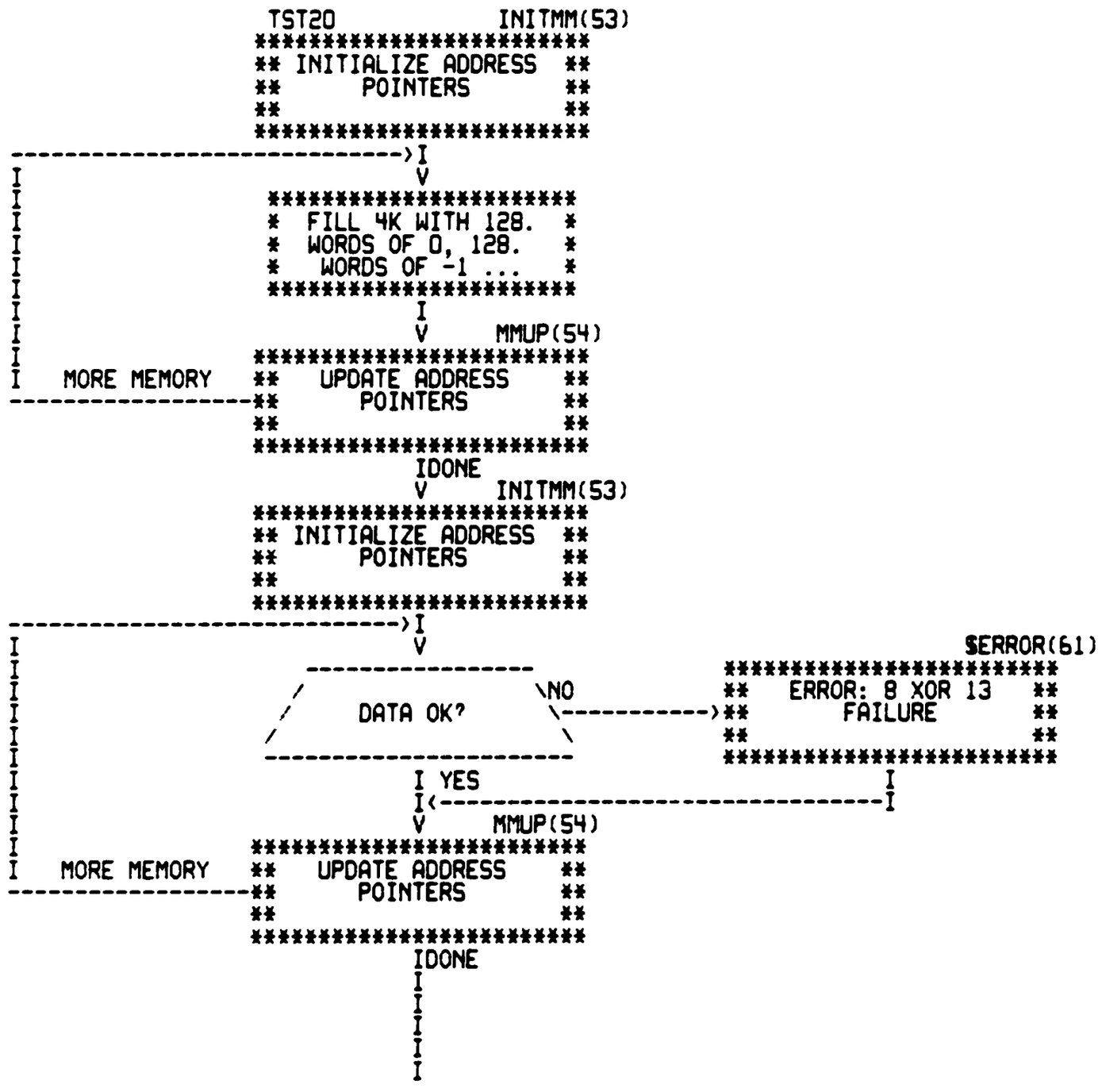

INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(61)
** ERROR: 3 XOR 9 **
** FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
YES
MMUP(54)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
DONE


TST16
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(55)
** WRITE 256. WORD **
** BLOCKS WITH 401 AND **
** -1 **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
256. WORD BLOCKS \NO
WRITTEN WITH 401
AND -1?
YES
$ERROR(61)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

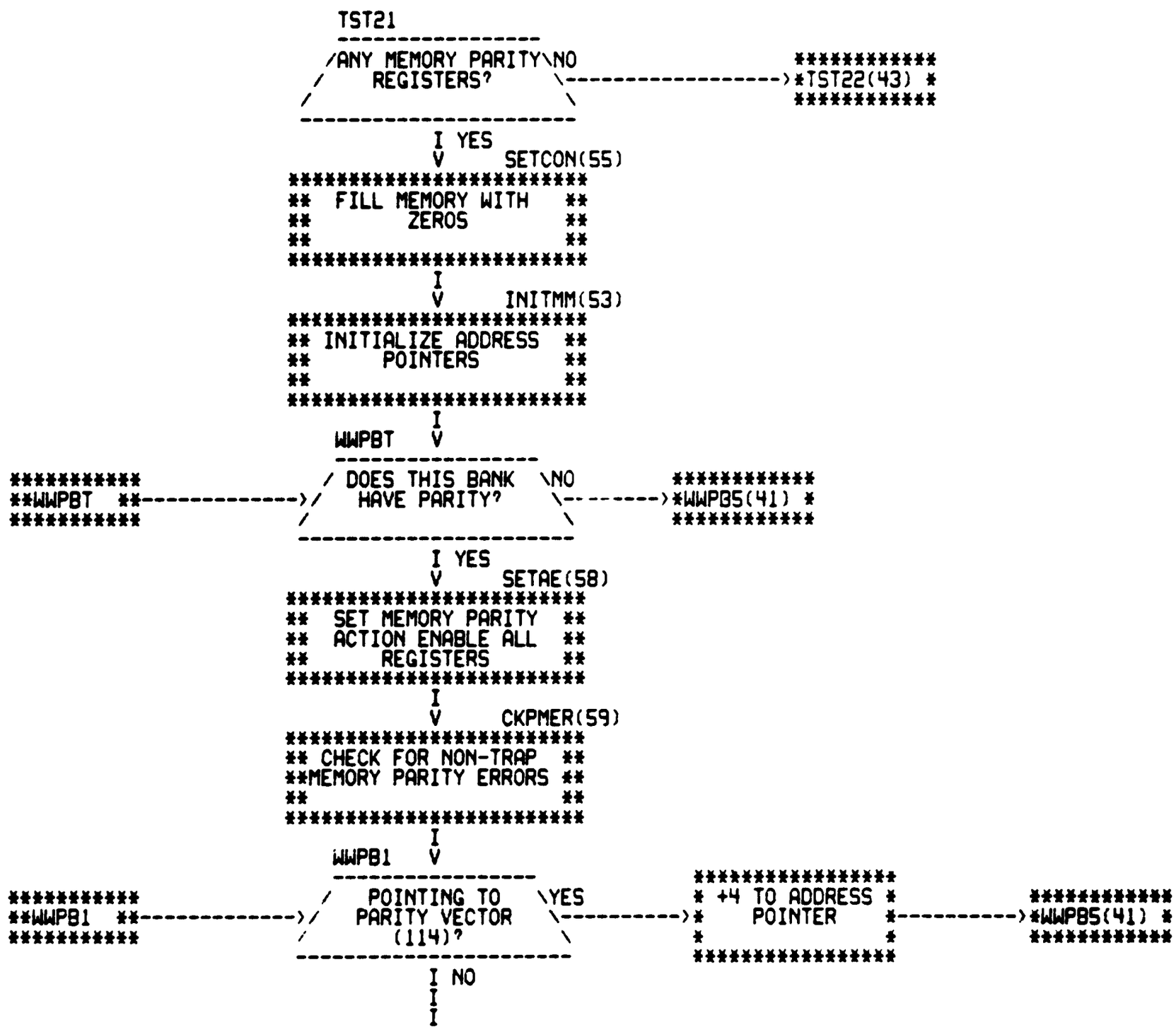

INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(61)
** ERROR: 3 XOR 9 **
** FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(61)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
YES
MMUP(54)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
DONE

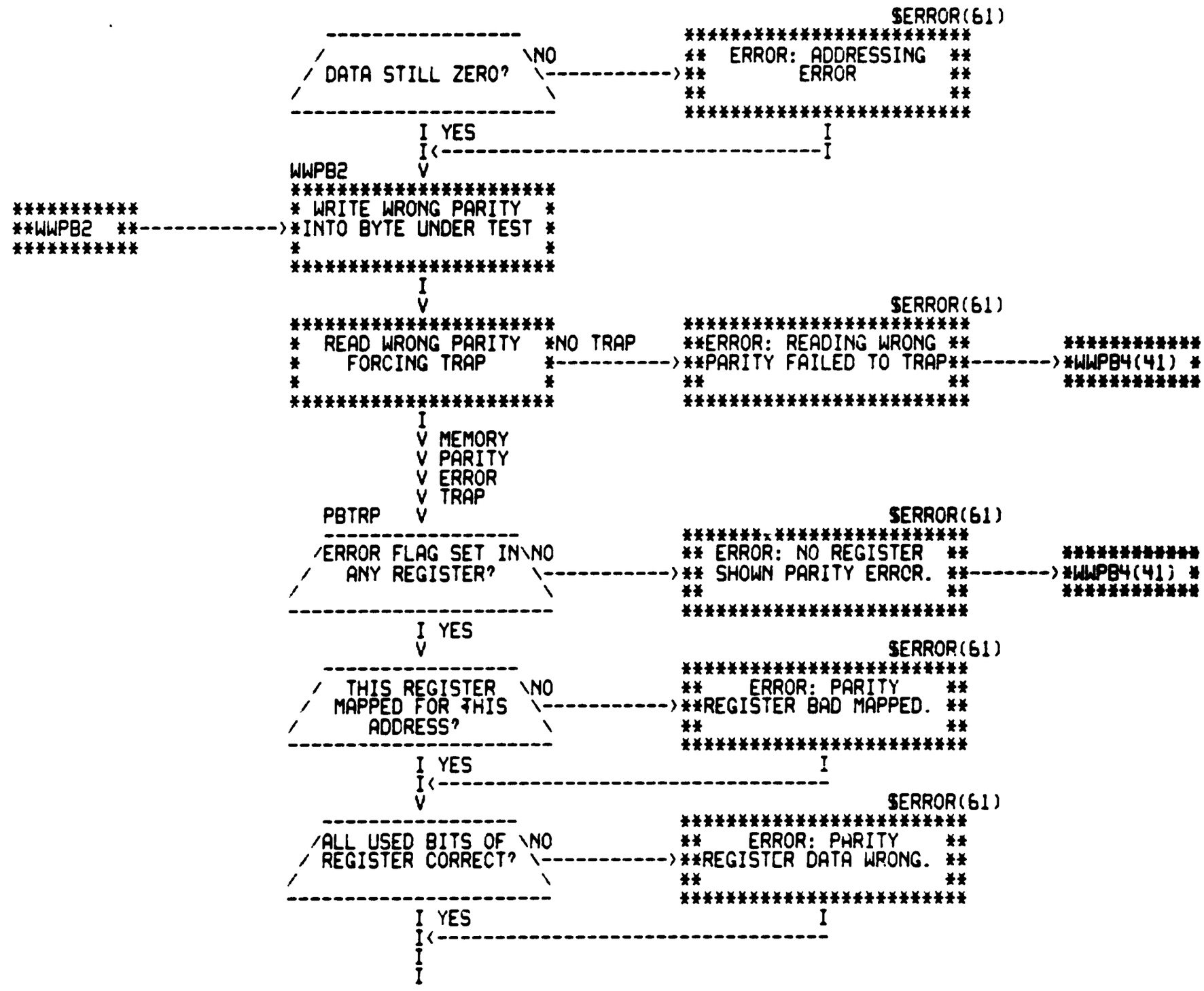

TST17
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(55)
** WRITE 256. WORD **
** BLOCKS WITH -1 AND **
** 401 **
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/WRITTEN WITH -1 AND\
/ 401? \
I YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(61)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

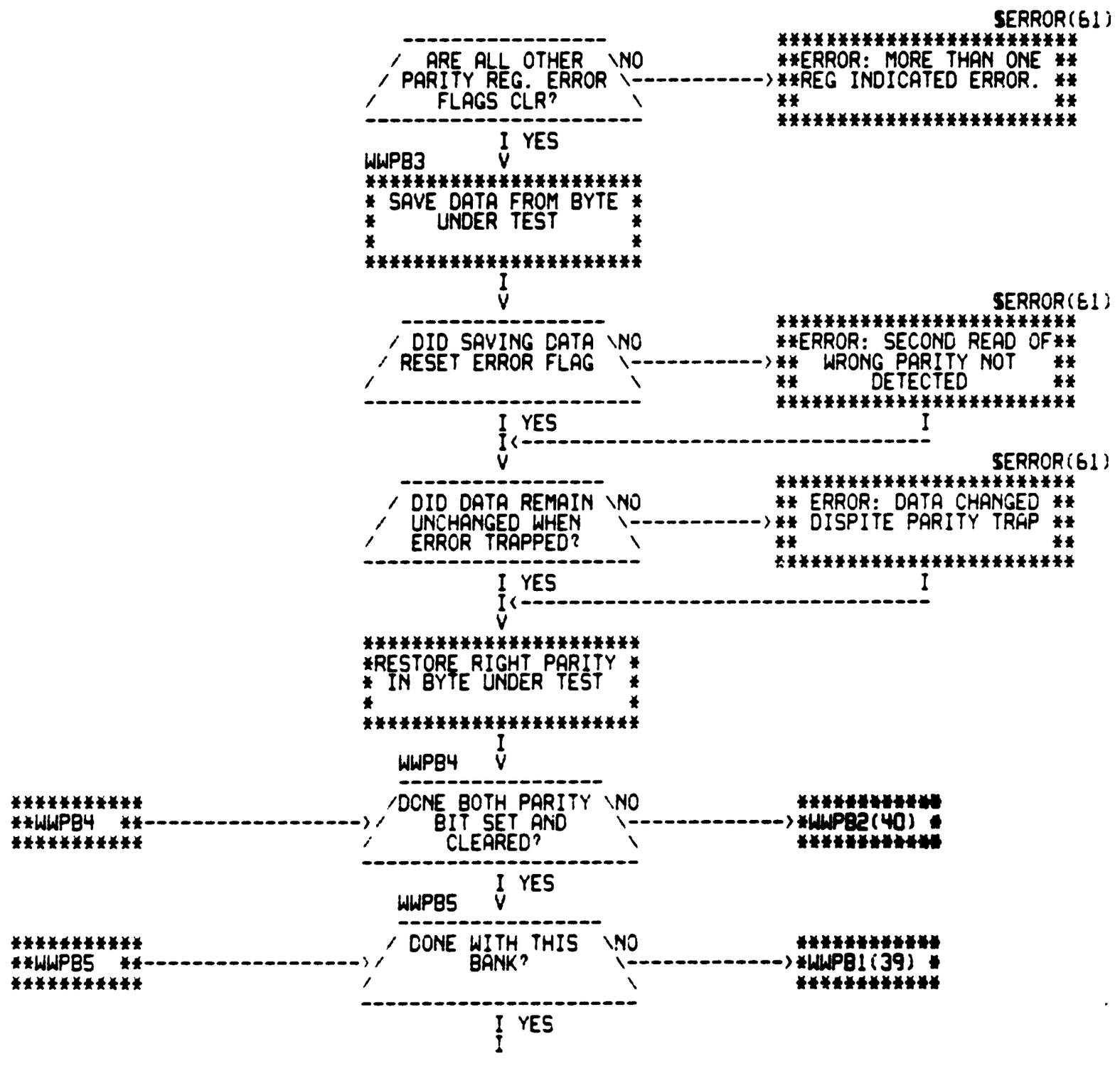

INITMM(53)
INITIALIZE ADDRESS POINTERS
DATA OK?
NO
YES
$ERROR(61)
ERROR: 3 XOR 9 FAILURE
COMPLEMENT DATA
DATA OK?
NO
YES
$ERROR(61)
ERROR: COMPLEMENTED 3 XOR 9 FAILURE
COMPLEMENT DATA
DATA OK?
NO
YES
$ERROR(61)
ERROR: TWICE COMPLEMENTED 3 XOR 9 FAILURE
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

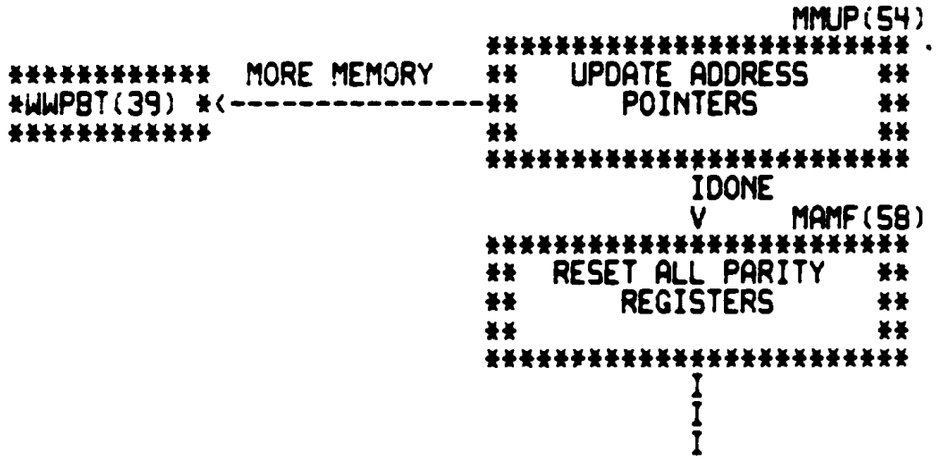

TST20
INITMM(53)
INITIALIZE ADDRESS POINTERS
FILL 4K WITH 128. WORDS OF 0, 128. WORDS OF -1 ...
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITMM(53)
INITIALIZE ADDRESS POINTERS
DATA OK?
NO
YES
$ERROR(61)
ERROR: 8 XOR 13 FAILURE
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

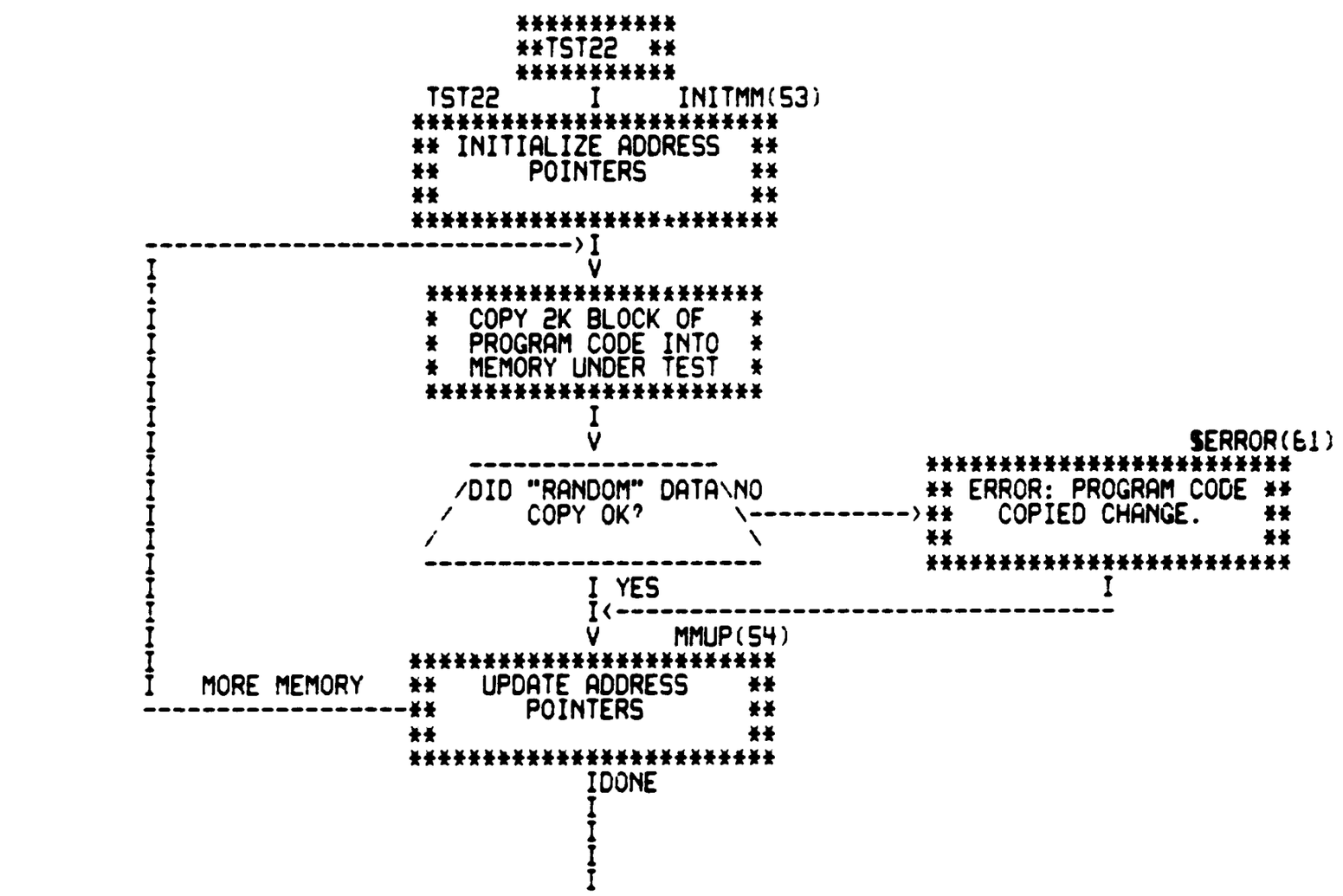

INITMM(53)
INITIALIZE ADDRESS POINTERS
COMPLEMENT DATA IN ALL MEMORY
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITMM(53)
INITIALIZE ADDRESS POINTERS
COMPLEMENTED DATA OK?
NO
SERROR(61)
ERROR: COMPLEMENTED 8 XOR 13 FAILURE
YES
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

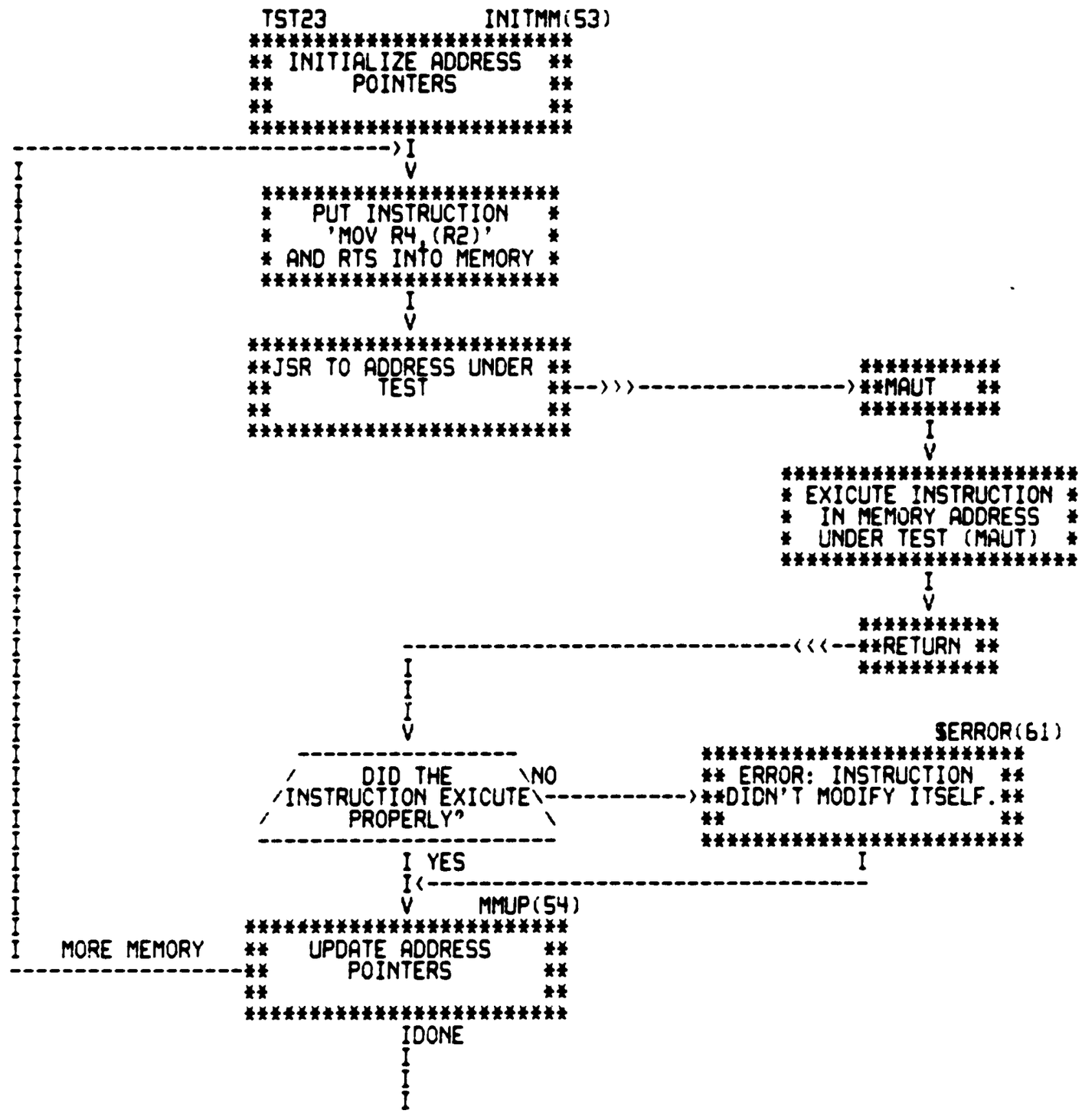

TST21
ANY MEMORY PARITY REGISTERS?
NO
TST22(43)
YES
SETCON(55)
FILL MEMORY WITH ZEROS
INITMM(53)
INITIALIZE ADDRESS POINTERS
WWPBT
DOES THIS BANK HAVE PARITY?
NO
WWPB5(41)
YES
SETAE(58)
SET MEMORY PARITY ACTION ENABLE ALL REGISTERS
CKPMER(59)
CHECK FOR NON-TRAP MEMORY PARITY ERRORS
WWPB1
POINTING TO PARITY VECTOR (114)?
YES
+4 TO ADDRESS POINTER
WWPB5(41)
NO

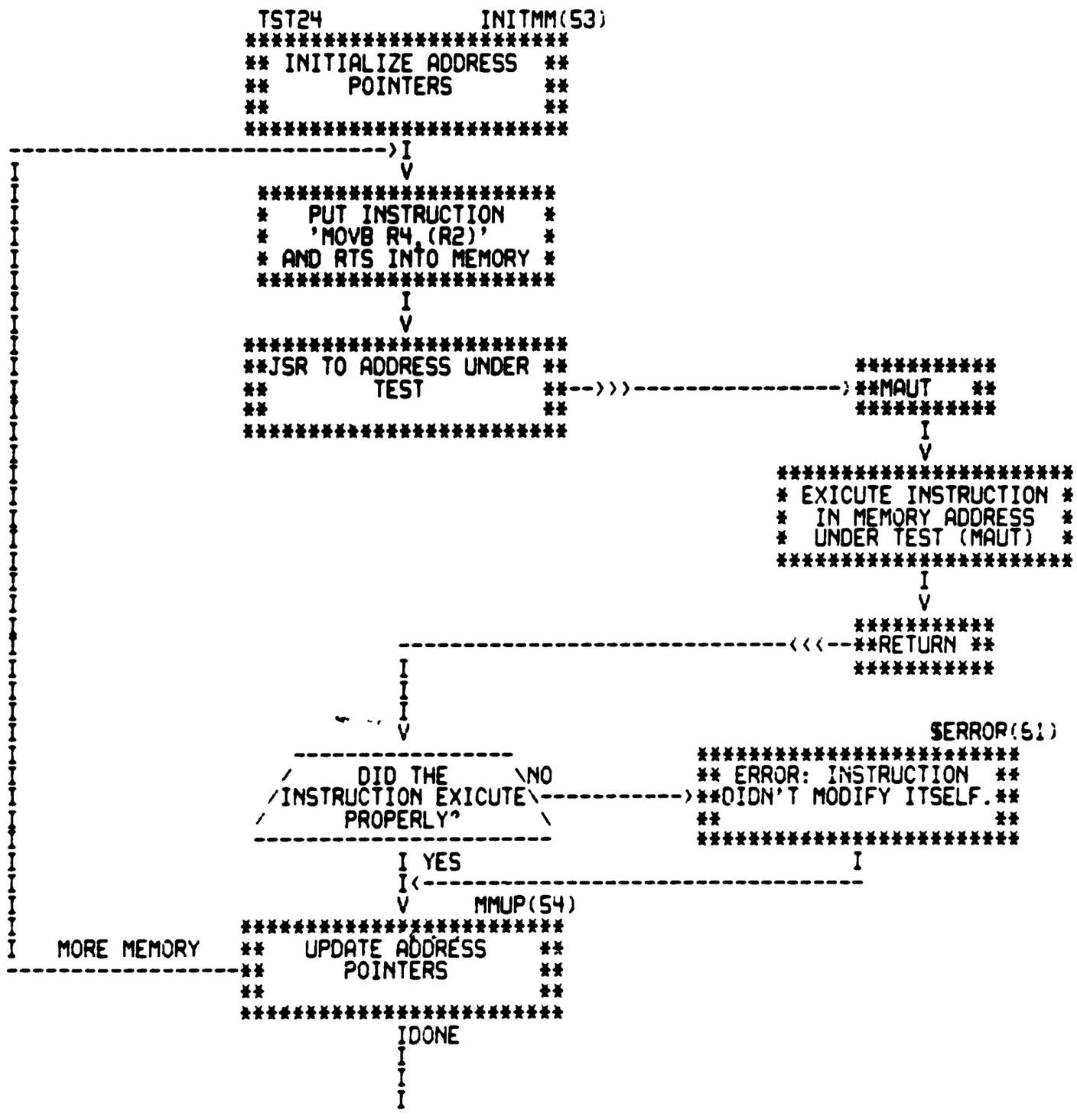

$ERROR(61)
** ERROR: ADDRESSING **
** ERROR **
DATA STILL ZERO?
NO
YES
WWPB2
**WWPB2 **
* WRITE WRONG PARITY *
*INTO BYTE UNDER TEST *
READ WRONG PARITY
FORCING TRAP
NO TRAP
**ERROR: READING WRONG **
PARITY FAILED TO TRAP
*WWPB4(41) *
MEMORY
PARITY
ERROR
TRAP
PBTRP
ERROR FLAG SET IN ANY REGISTER?
** ERROR: NO REGISTER **
** SHOWN PARITY ERROR. **
THIS REGISTER MAPPED FOR THIS ADDRESS?
**REGISTER BAD MAPPED. **
ALL USED BITS OF REGISTER CORRECT?
**REGISTER DATA WRONG. **

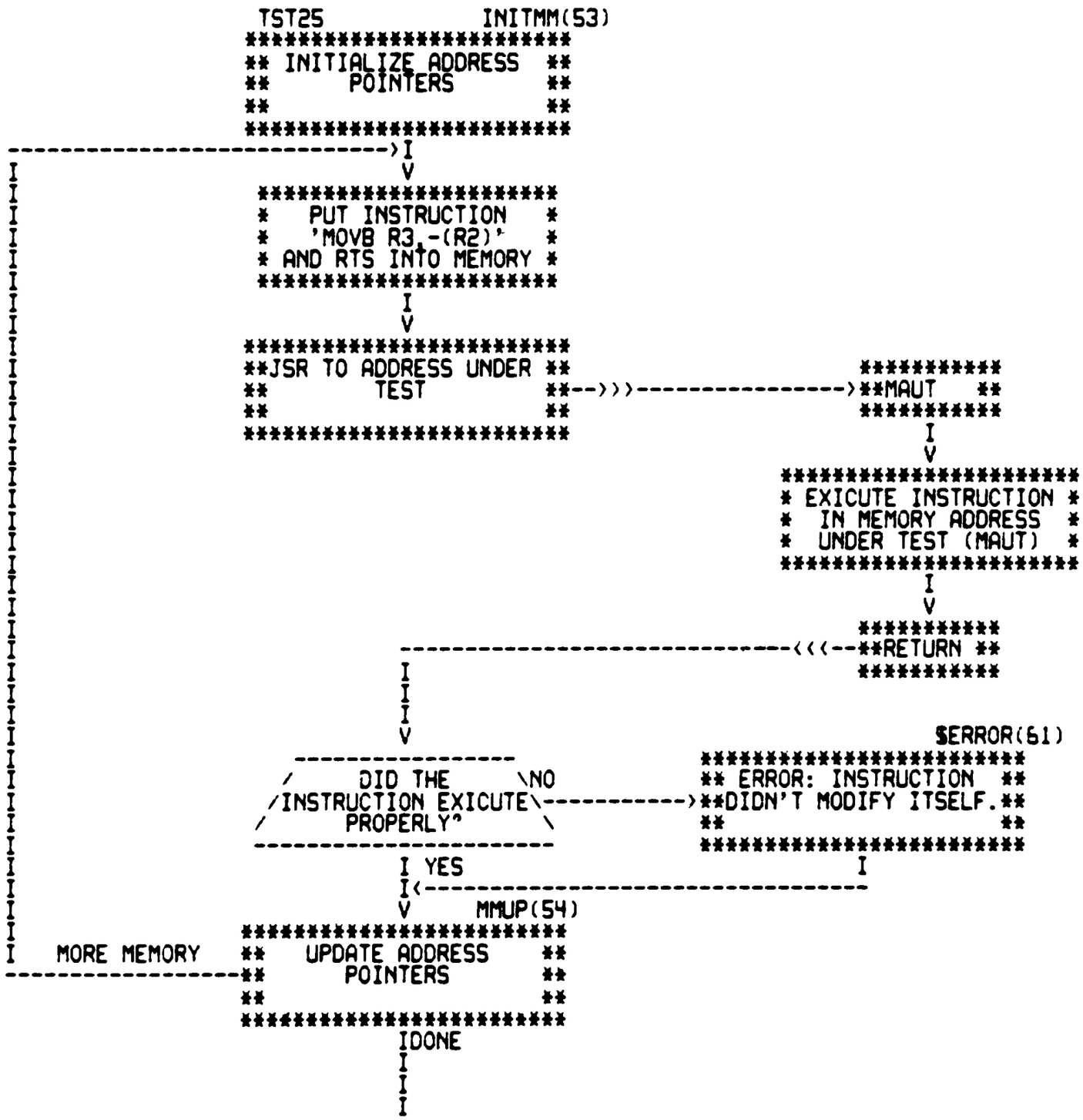

$ERROR(61)
ARE ALL OTHER PARITY REG. ERROR FLAGS CLR?
NO
ERROR: MORE THAN ONE REG INDICATED ERROR.
YES
WWPB3
SAVE DATA FROM BYTE UNDER TEST
$ERROR(61)
DID SAVING DATA RESET ERROR FLAG
NO
ERROR: SECOND READ OF WRONG PARITY NOT DETECTED
YES
$ERROR(61)
DID DATA REMAIN UNCHANGED WHEN ERROR TRAPPED?
NO
** ERROR: DATA CHANGED DISPITE PARITY TRAP **
YES
RESTORE RIGHT PARITY IN BYTE UNDER TEST
WWPB4
DONE BOTH PARITY BIT SET AND CLEARED?
NO
WWPB4
*WWPB2(40) *
YES
WWPB5
DONE WITH THIS BANK?
NO
WWPB5
*WWPB1(39) *
YES

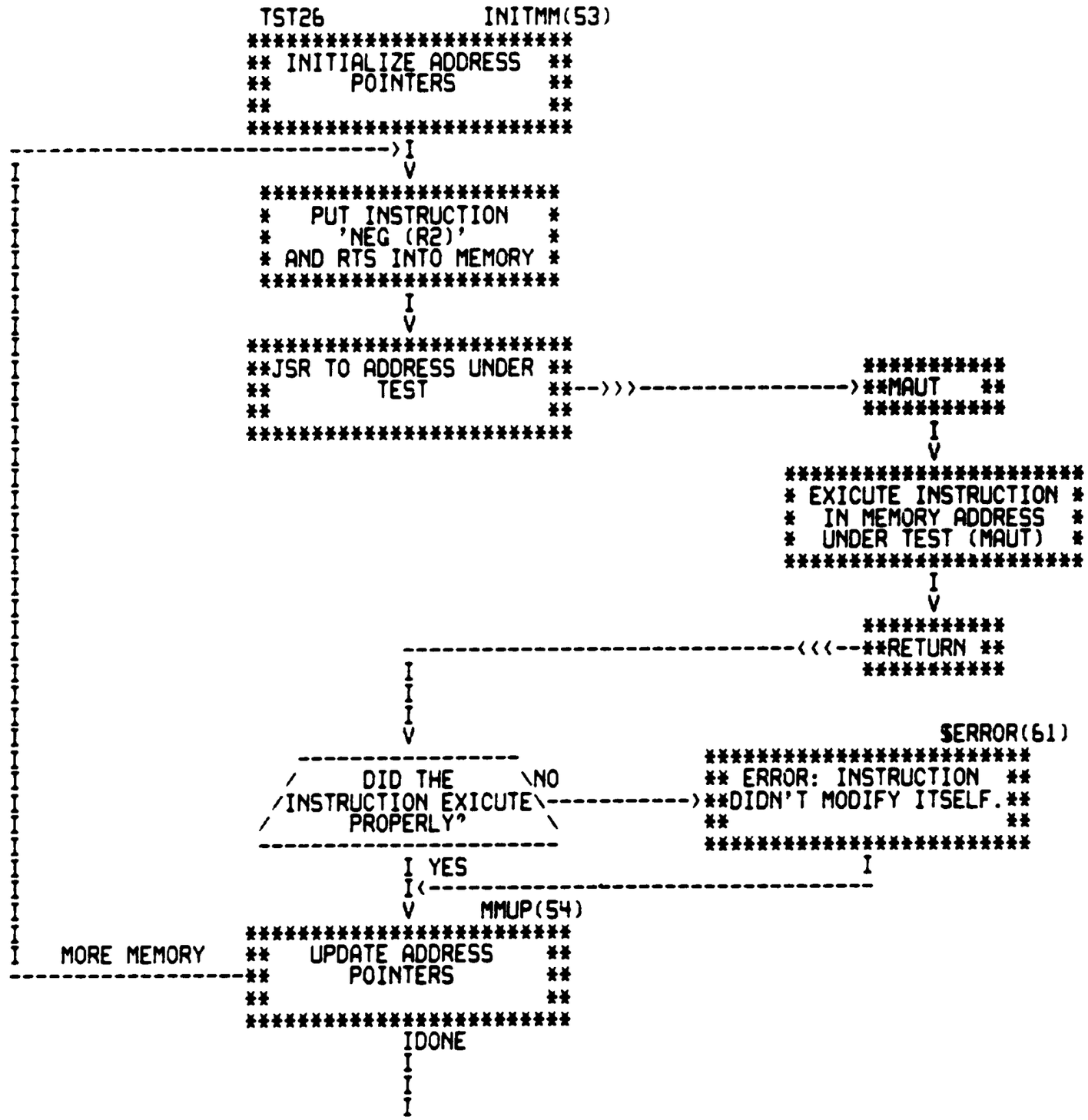

MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
WWPBT(39)
IDONE
MAMF(58)
RESET ALL PARITY REGISTERS

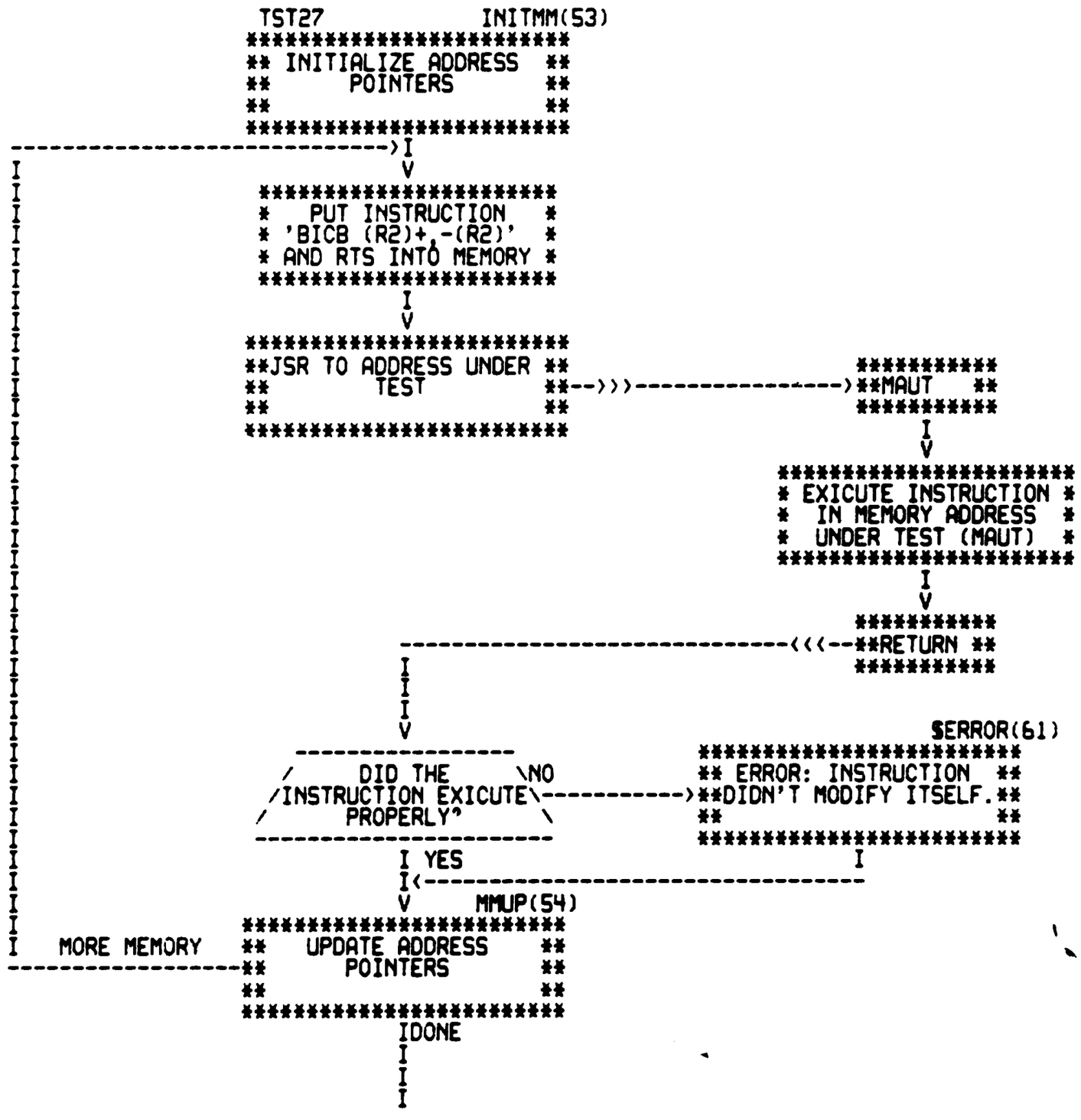

TST22
INITMM(53)
INITIALIZE ADDRESS POINTERS
COPY 2K BLOCK OF PROGRAM CODE INTO MEMORY UNDER TEST
DID "RANDOM" DATA COPY OK?
NO
$ERROR(61)
ERROR: PROGRAM CODE COPIED CHANGE.
YES
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
DONE

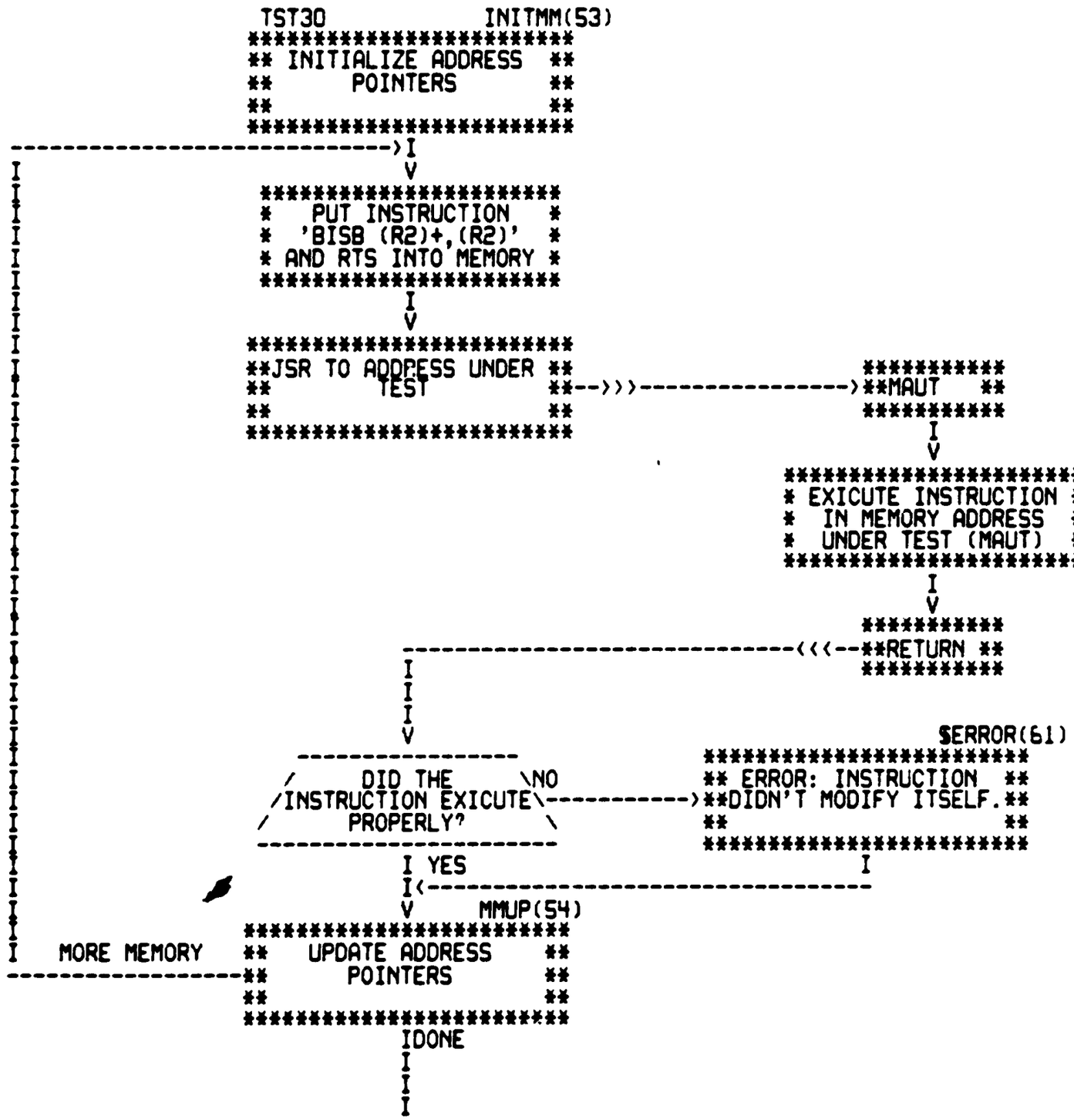

TST23
INITMM(53)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'MOV R4,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
$ERROR(61)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

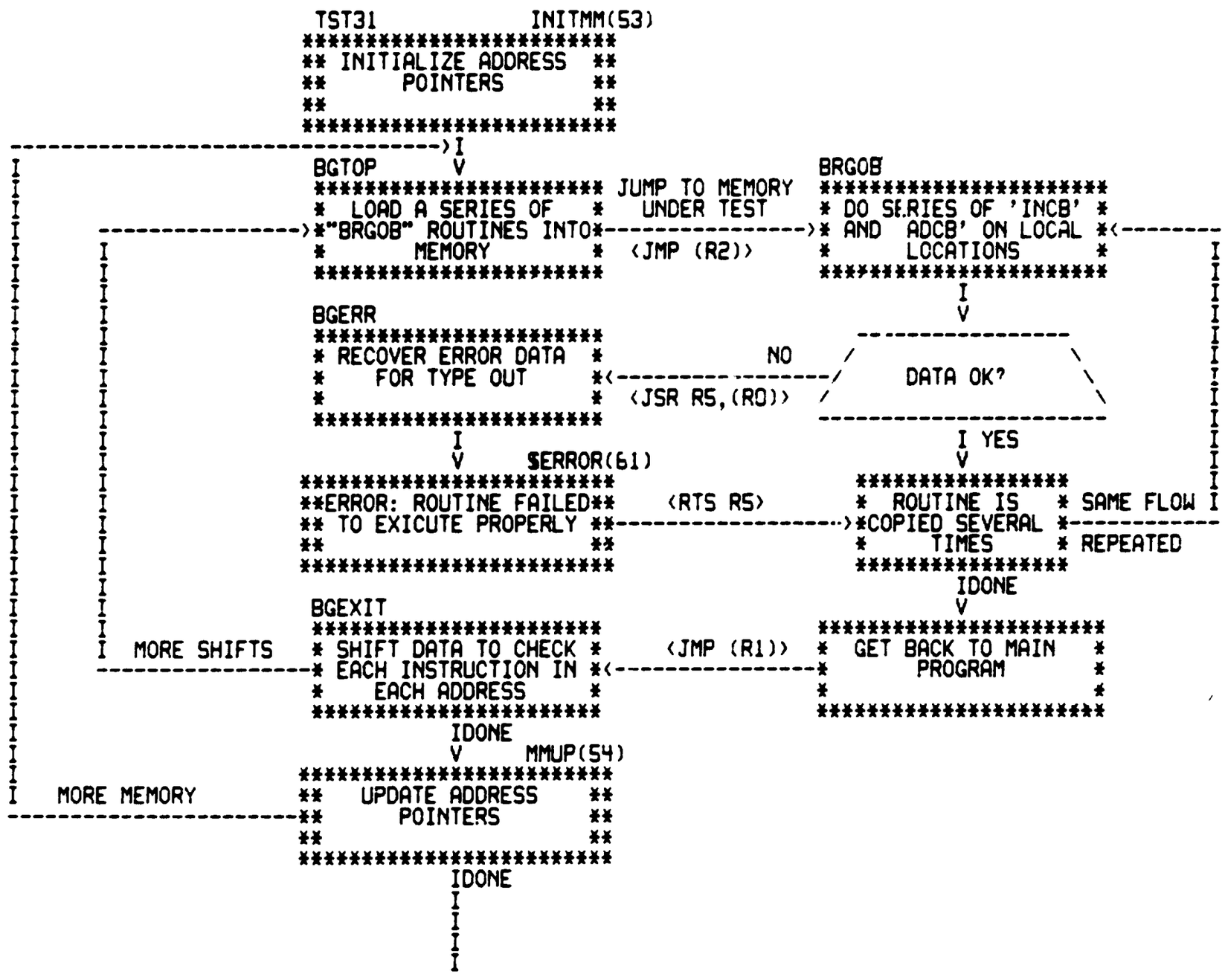

TST24
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION 'MOVB R4,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
**MAUT **
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
**RETURN **
$ERROR(51)
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
** ERROR: INSTRUCTION **
DIDN'T MODIFY ITSELF.
YES
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE


TST25
INITMM(53)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'MOVB R3,-(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
$ERROR(61)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

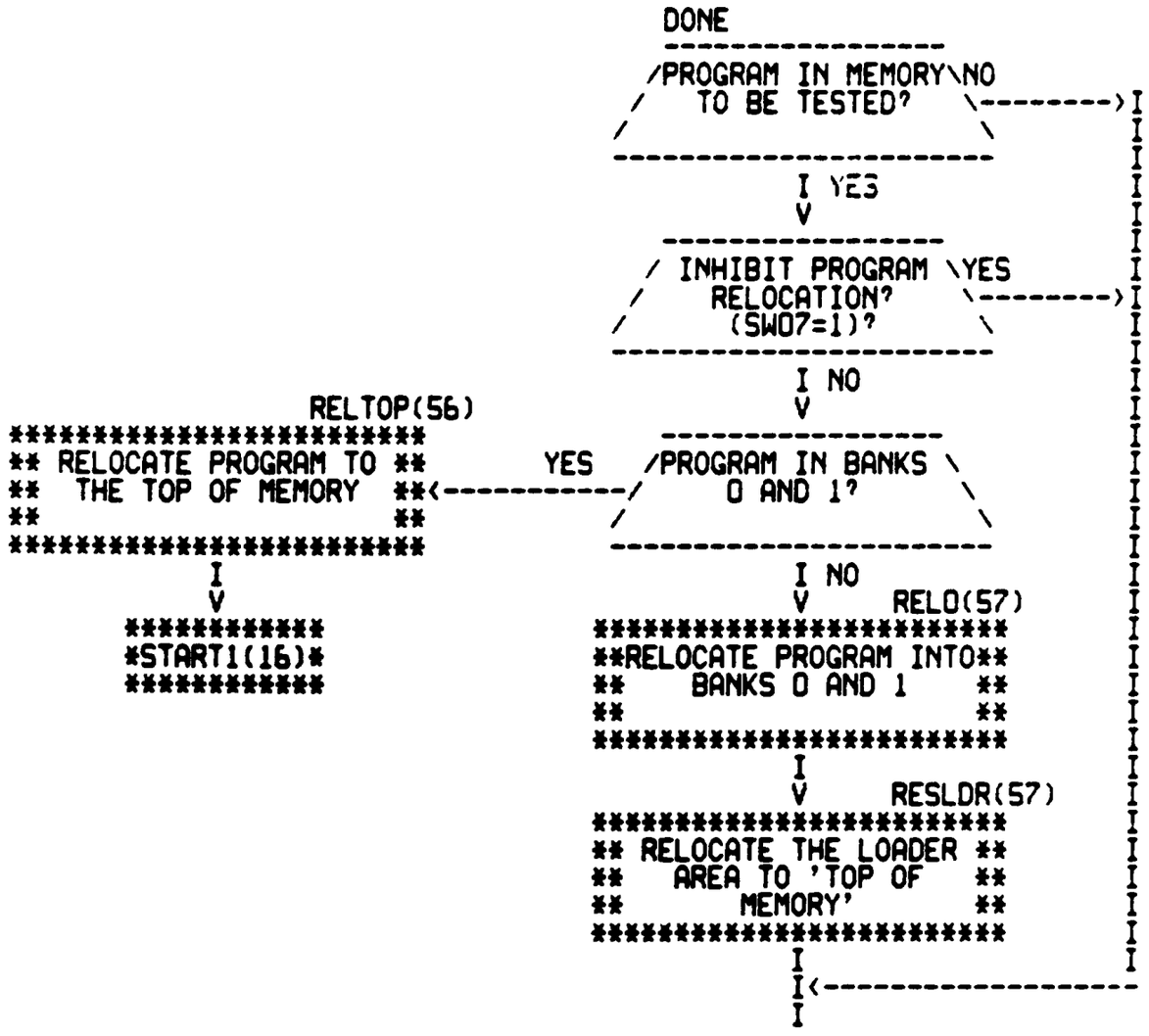

TST26
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION 'NEG (R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
**MAUT **
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
**RETURN **
$ERROR(61)
** ERROR: INSTRUCTION **
DIDN'T MODIFY ITSELF.
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
MMUP(54)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE


TST27
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'BICB (R2)+,-(R2)'
AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
$ERROR(61)
ERROR: INSTRUCTION
DIDN'T MODIFY ITSELF.
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

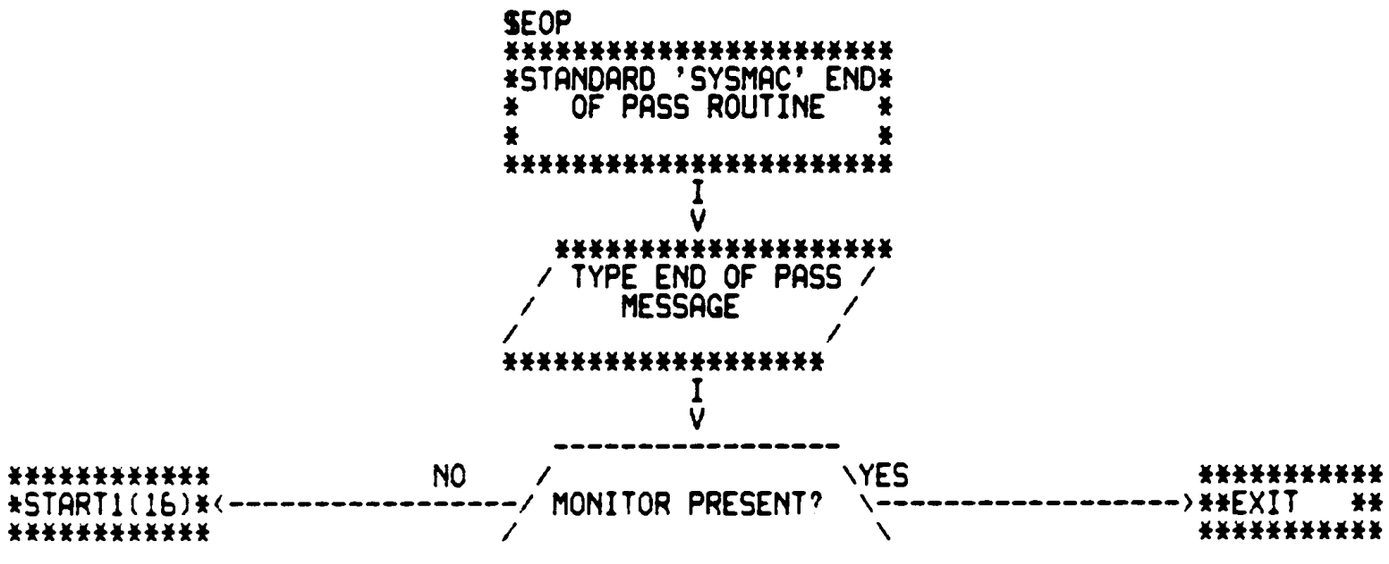

TST30
INITMM(53)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'BISB (R2)+,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
$MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
$RETURN
$ERROR(61)
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(54)
UPDATE ADDRESS POINTERS
MORE MEMORY
$DONE

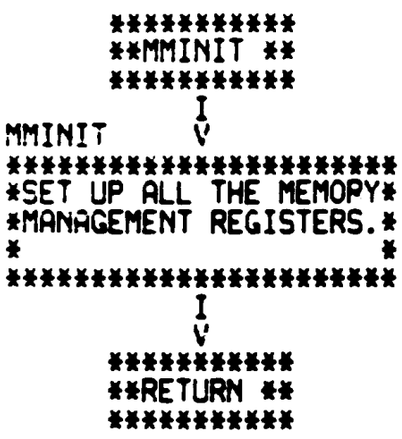

TST31
INITMM(53)
** INITIALIZE ADDRESS **
** POINTERS **
BGTOP
LOAD A SERIES OF "BRGOB" ROUTINES INTO MEMORY
JUMP TO MEMORY UNDER TEST
(JMP (R2))
BRGOB
DO SERIES OF 'INCB' AND ADCB' ON LOCAL LOCATIONS
BGERR
RECOVER ERROR DATA FOR TYPE OUT
(JSR R5,(R0))
NO
DATA OK?
YES
SERROR(61)
ERROR: ROUTINE FAILED
** TO EXICUTE PROPERLY **
(RTS R5)
ROUTINE IS COPIED SEVERAL TIMES
SAME FLOW REPEATED
IDONE
BGEXIT
MORE SHIFTS
SHIFT DATA TO CHECK EACH INSTRUCTION IN EACH ADDRESS
(JMP (R1))
GET BACK TO MAIN PROGRAM
IDONE
MMUP(54)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

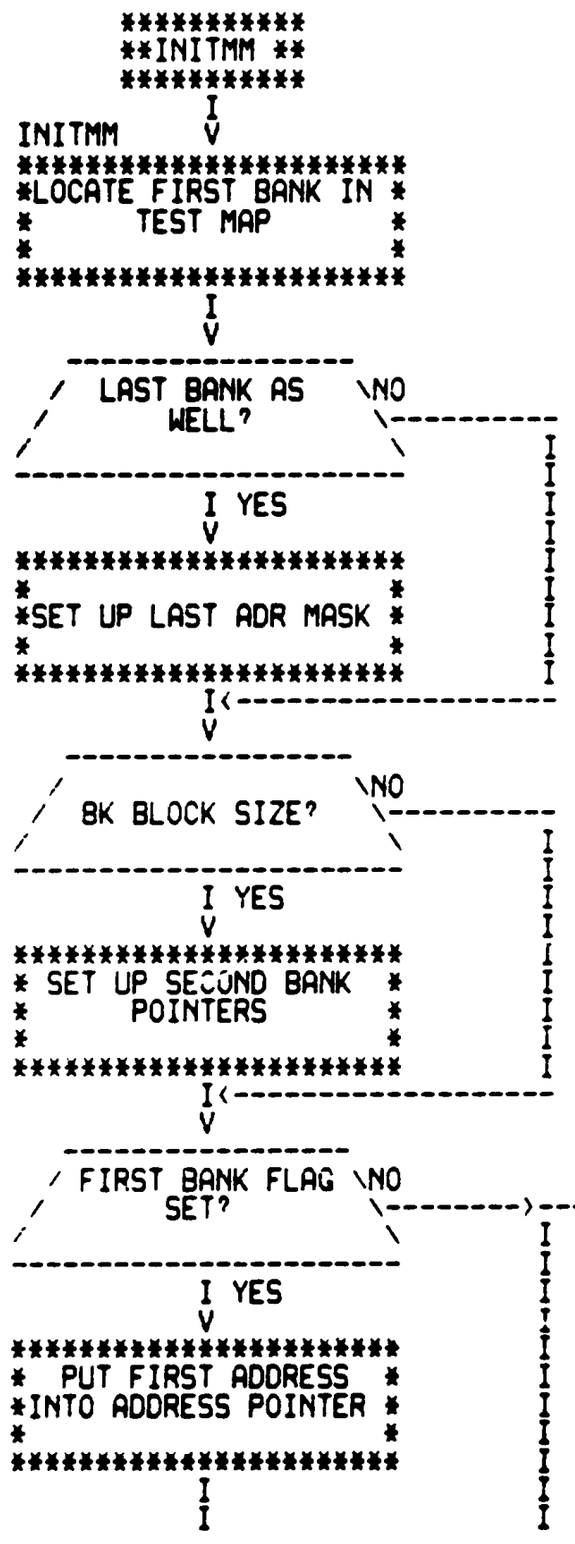
DONE

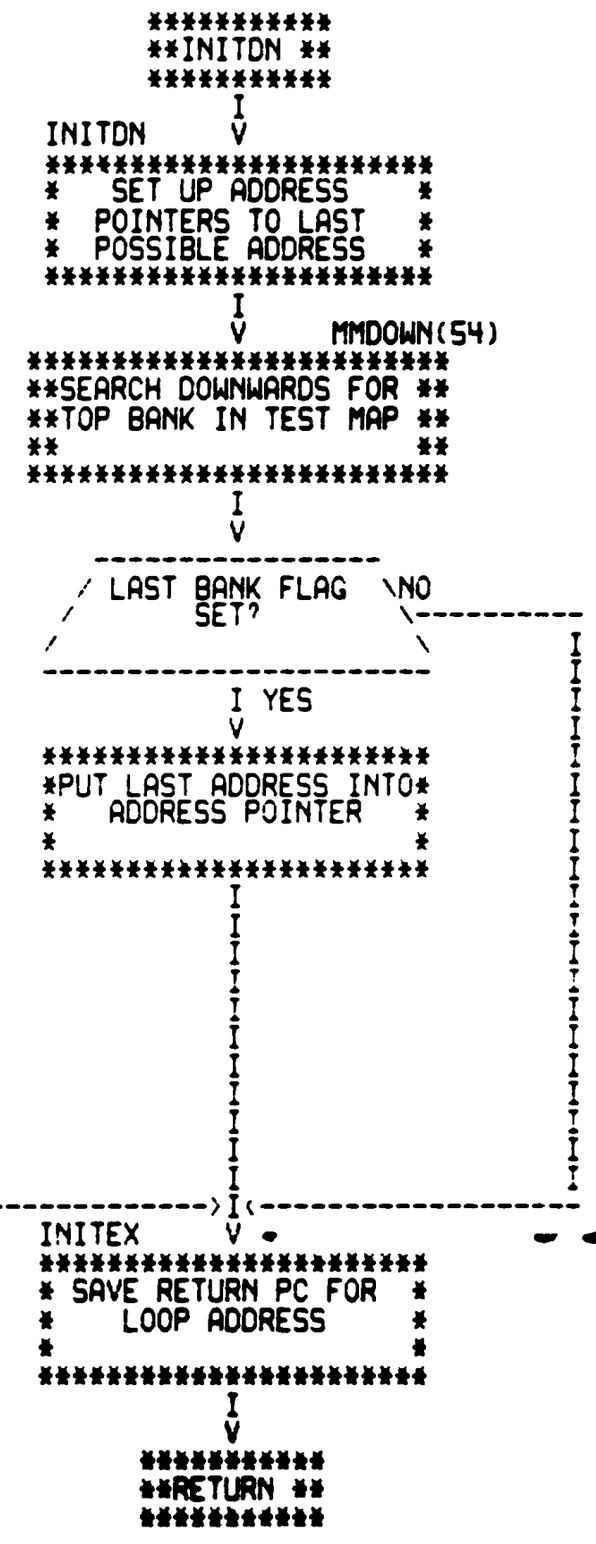

DONE
PROGRAM IN MEMORY TO BE TESTED?
NO
YES
INHIBIT PROGRAM RELOCATION? (SW07=1)?
YES
NO
PROGRAM IN BANKS 0 AND 1?
YES
NO
RELTOP(56)
** RELOCATE PROGRAM TO THE TOP OF MEMORY **
RELO(57)
RELOCATE PROGRAM INTO BANKS 0 AND 1
START1(16)
RESLDR(57)
** RELOCATE THE LOADER AREA TO 'TOP OF MEMORY' **

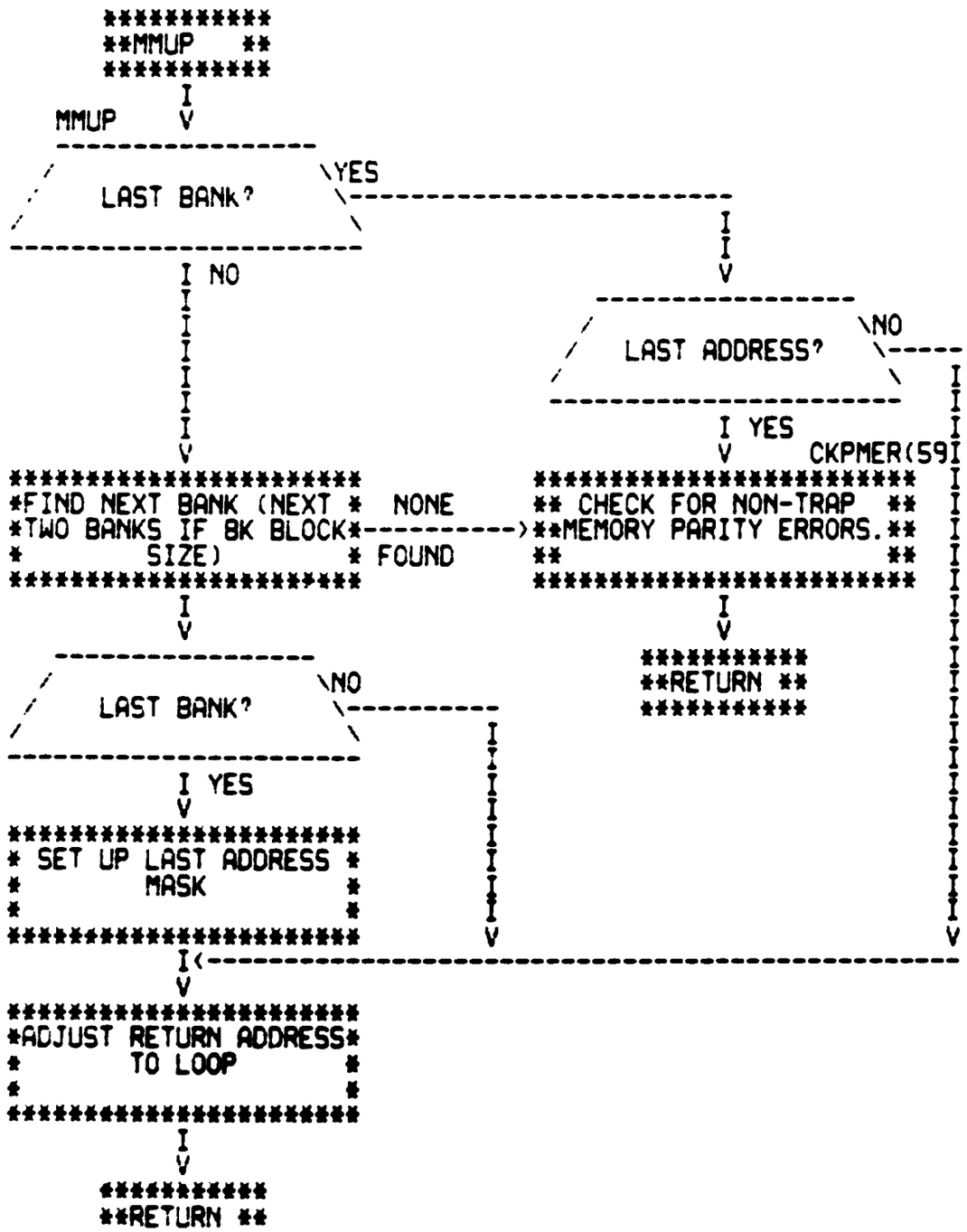
END OF PASS

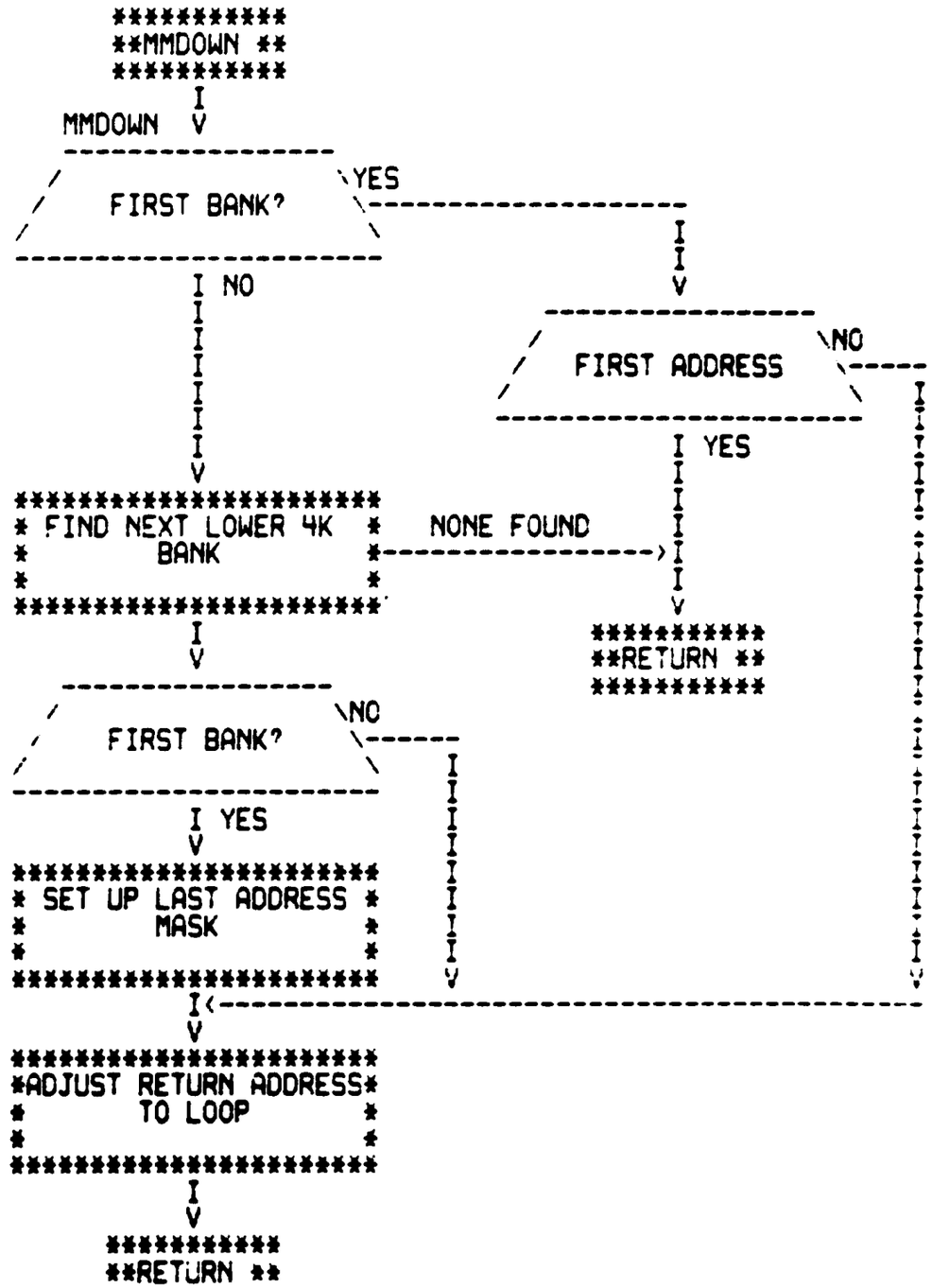

$EOP
STANDARD 'SYSMAC' END
* OF PASS ROUTINE *
TYPE END OF PASS MESSAGE
NO
MONITOR PRESENT?
YES
START1(16)
**EXIT **

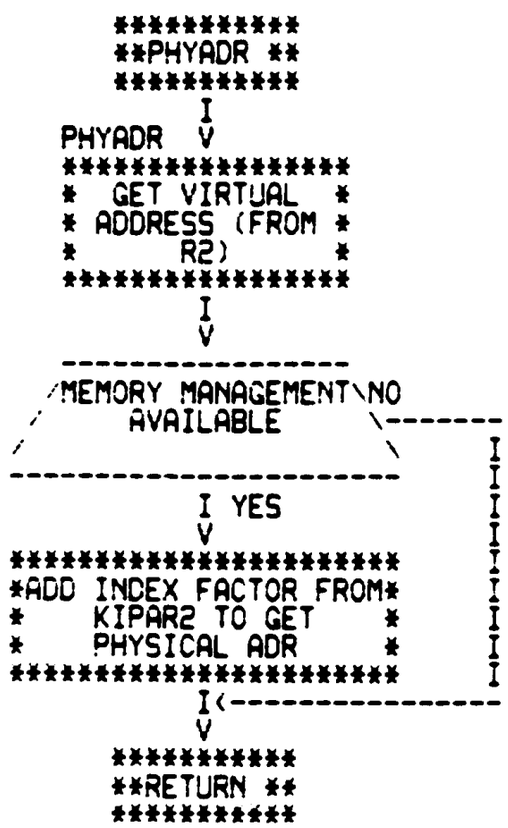

**MMINIT **
MMINIT
SET UP ALL THE MEMORY
MANAGEMENT REGISTERS.
**RETURN **

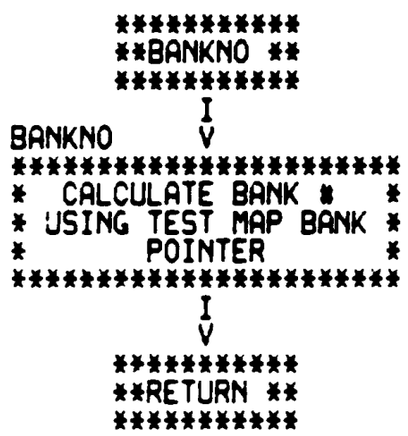

**INITMM **
INITMM
*LOCATE FIRST BANK IN *
* TEST MAP *
LAST BANK AS WELL?
NO
I YES
*SET UP LAST ADR MASK *
8K BLOCK SIZE?
NO
I YES
* SET UP SECOND BANK *
* POINTERS *
FIRST BANK FLAG SET?
NO
I YES
* PUT FIRST ADDRESS *
*INTO ADDRESS POINTER *

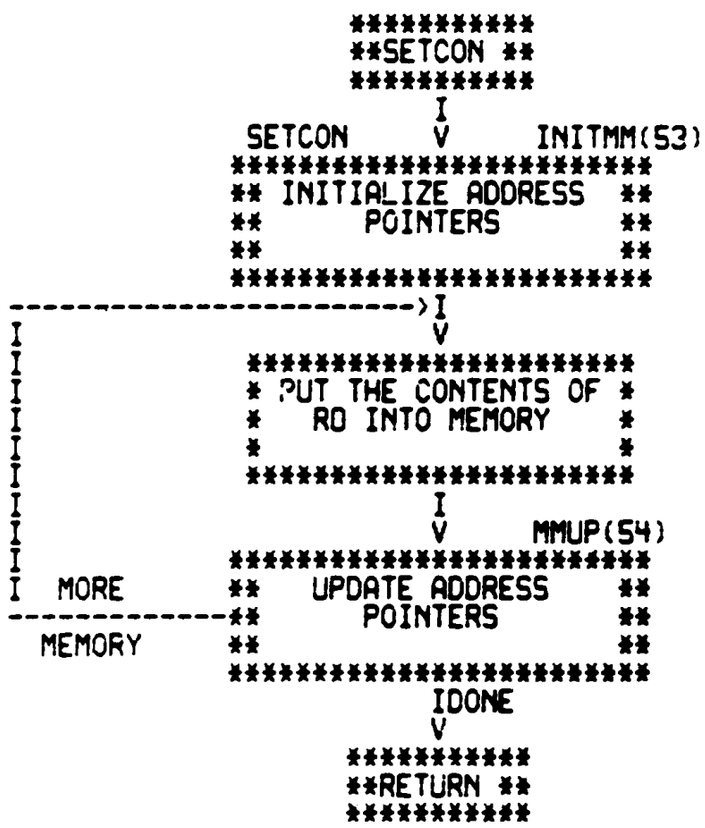

**INITDN **
INITDN
* SET UP ADDRESS *
* POINTERS TO LAST *
* POSSIBLE ADDRESS *
MMDOWN(54)
**SEARCH DOWNWARDS FOR **
**TOP BANK IN TEST MAP **
LAST BANK FLAG SET?
NO
I YES
PUT LAST ADDRESS INTO
* ADDRESS POINTER *
INITEX
* SAVE RETURN PC FOR *
* LOOP ADDRESS *
**RETURN **

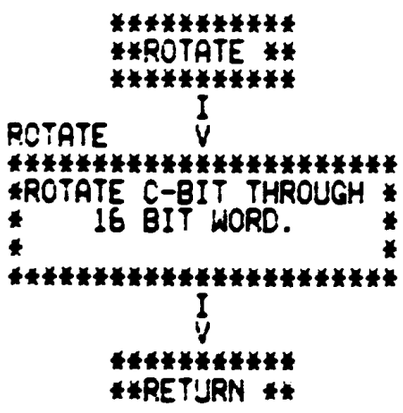

**MMUP **
MMUP
LAST BANK?
YES
NO
LAST ADDRESS?
NO
YES
CKPMER(59
*FIND NEXT BANK (NEXT * NONE
TWO BANKS IF 8K BLOCK
* SIZE) * FOUND
** CHECK FOR NON-TRAP **
MEMORY PARITY ERRORS.
**RETURN **
LAST BANK?
NO
YES
* SET UP LAST ADDRESS *
* MASK *
ADJUST RETURN ADDRESS
* TO LOOP *
**RETURN **

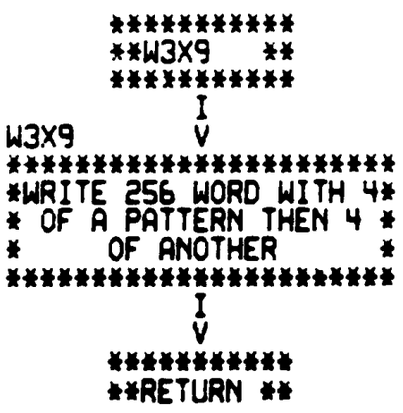

**MMDOWN **
MMDOWN
FIRST BANK?
YES
NO
FIRST ADDRESS
NO
YES
* FIND NEXT LOWER 4K *
* BANK *
NONE FOUND
**RETURN **
FIRST BANK?
NO
YES
* SET UP LAST ADDRESS *
* MASK *
ADJUST RETURN ADDRESS
* TO LOOP *
**RETURN **


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

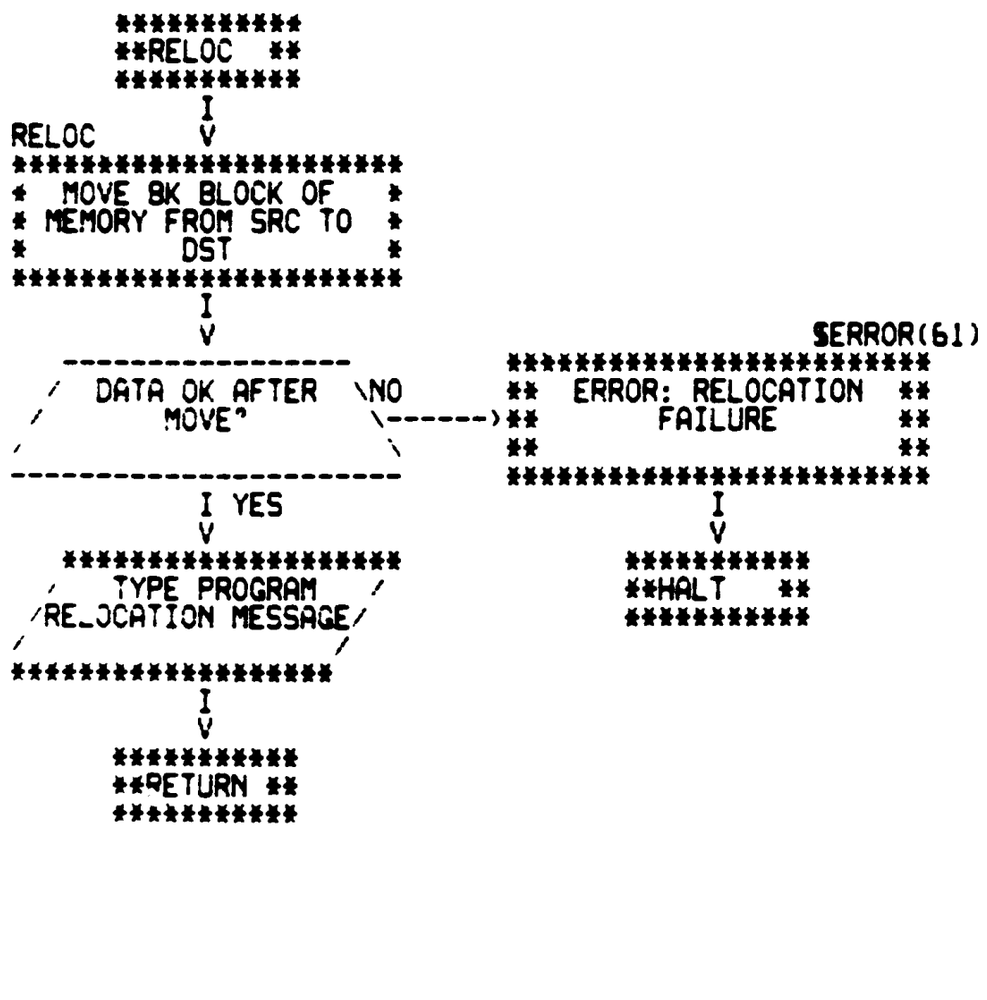

**BANKNO **
BANKNO
CALCULATE BANK # USING TEST MAP BANK POINTER
**RETURN **

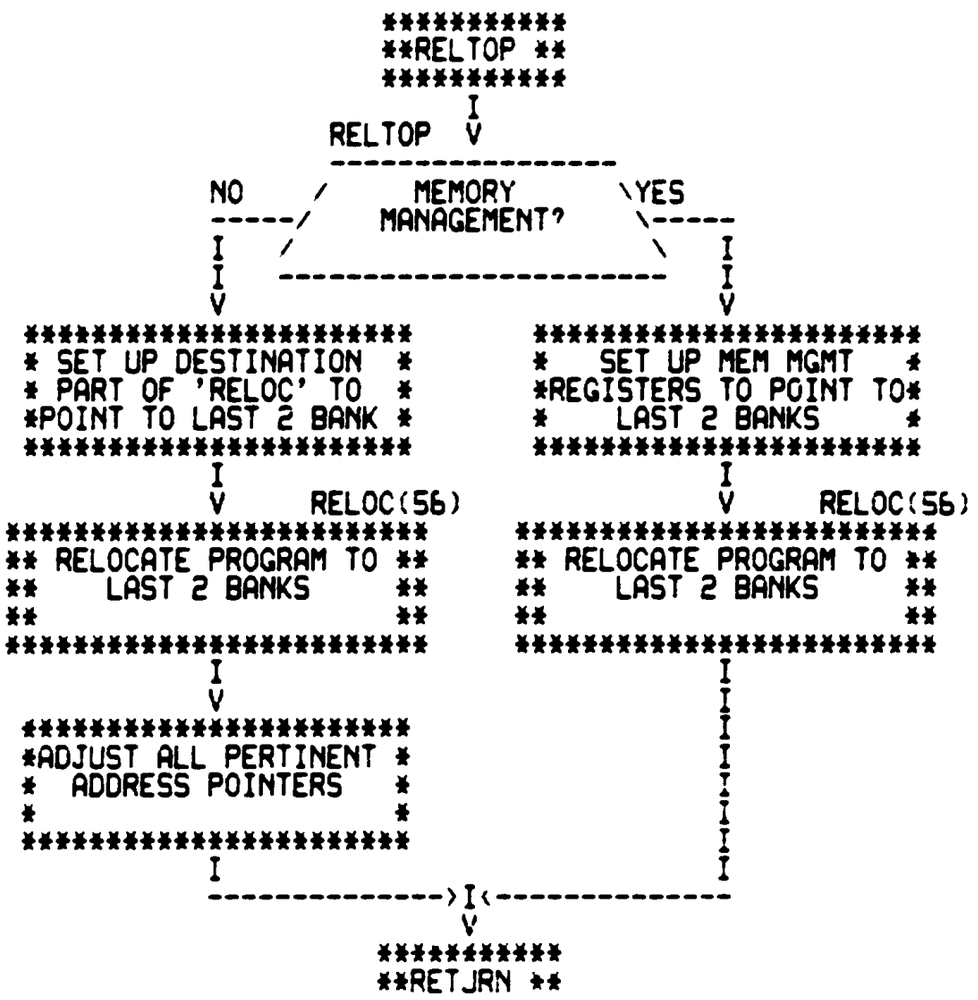

**SETCON **
SETCON
INITMM(53)
** INITIALIZE ADDRESS POINTERS **
PUT THE CONTENTS OF R0 INTO MEMORY
MMUP(54)
MORE MEMORY
** UPDATE ADDRESS POINTERS **
IDONE
**RETURN **

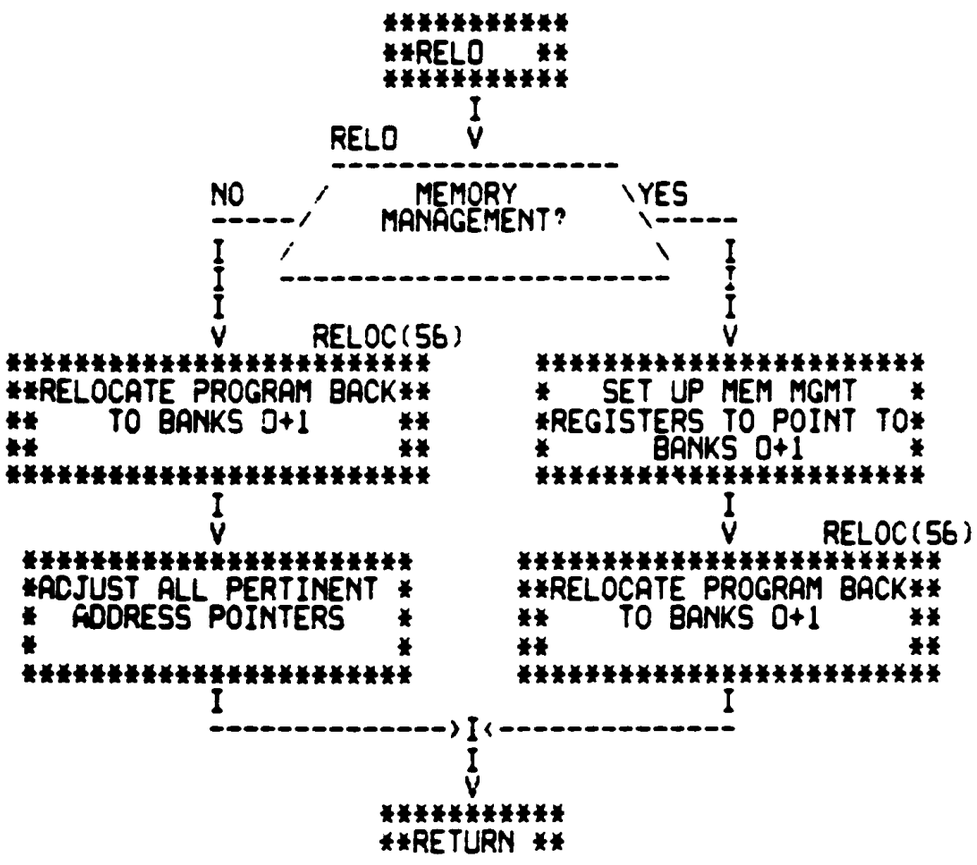

**ROTATE **
ROTATE
ROTATE C-BIT THROUGH 16 BIT WORD.
**RETURN **

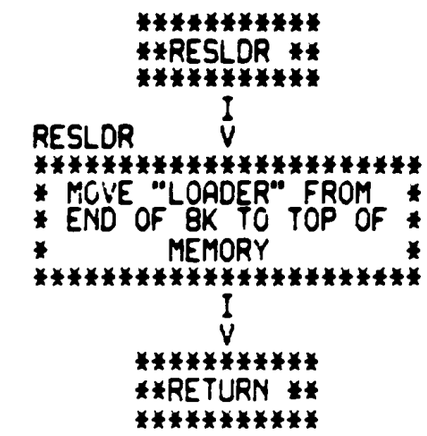

**W3X9 **
W3X9
WRITE 256 WORD WITH 4 OF A PATTERN THEN 4 OF ANOTHER
**RETURN **

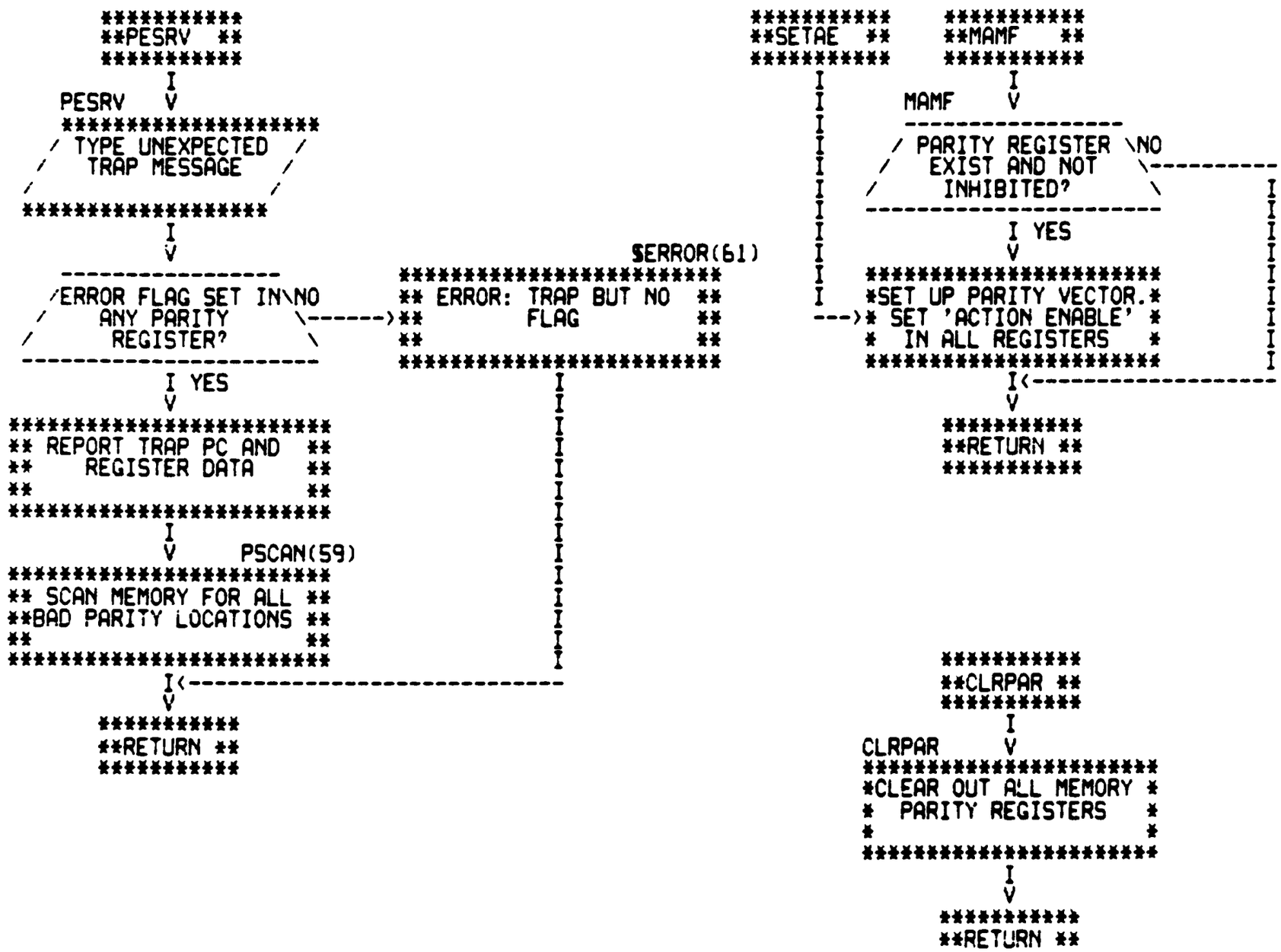
RELOCATION SUBROUTINES

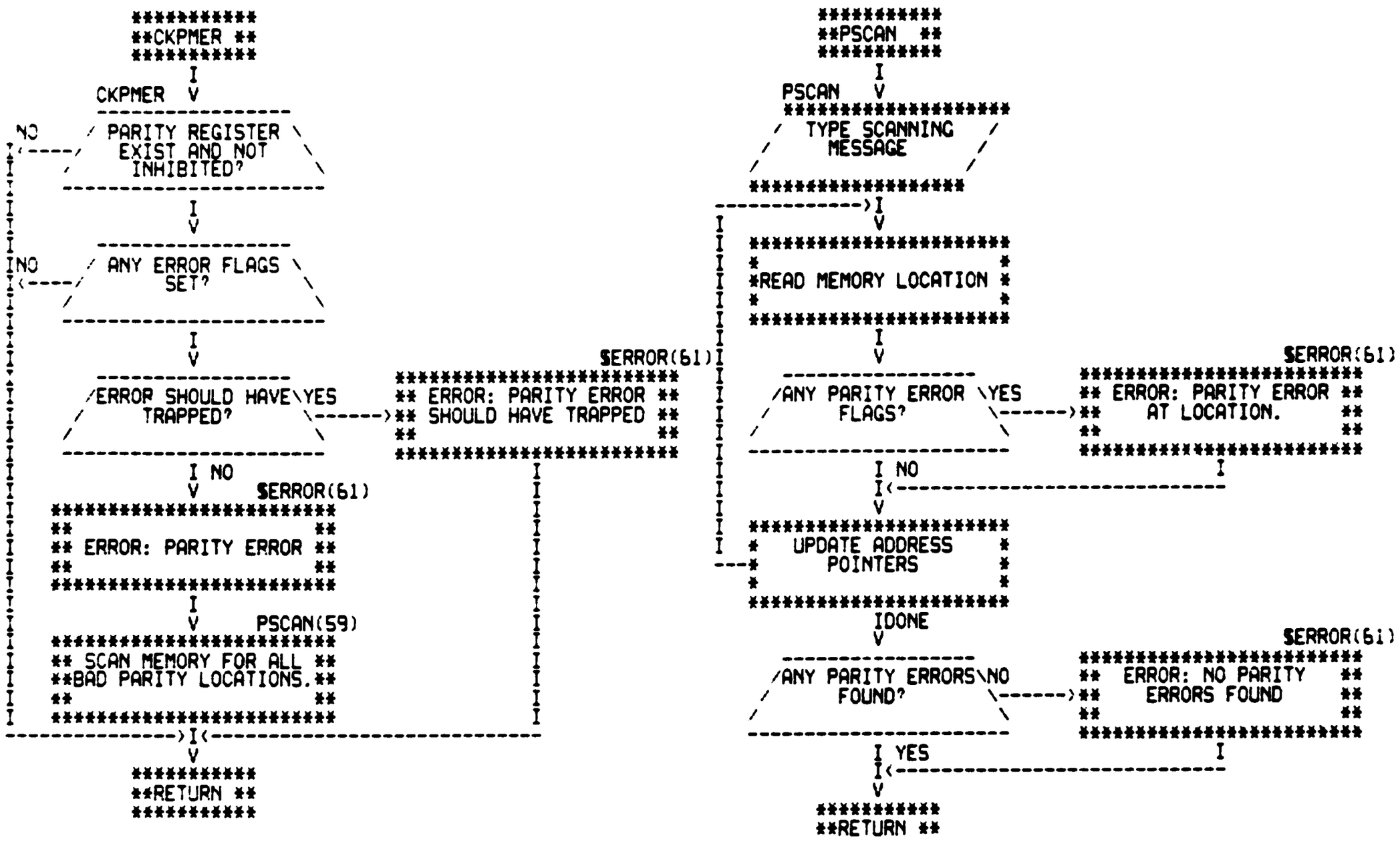


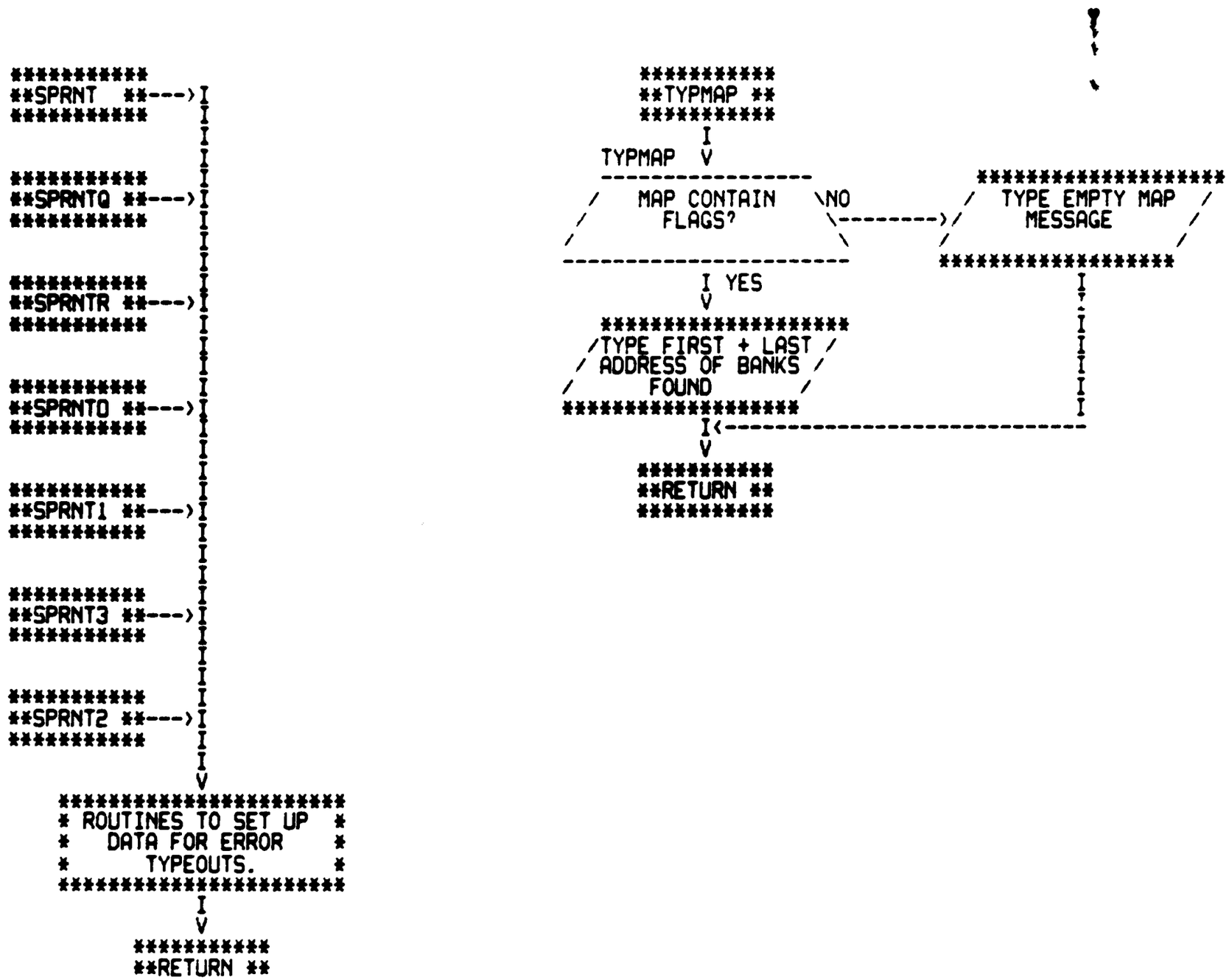


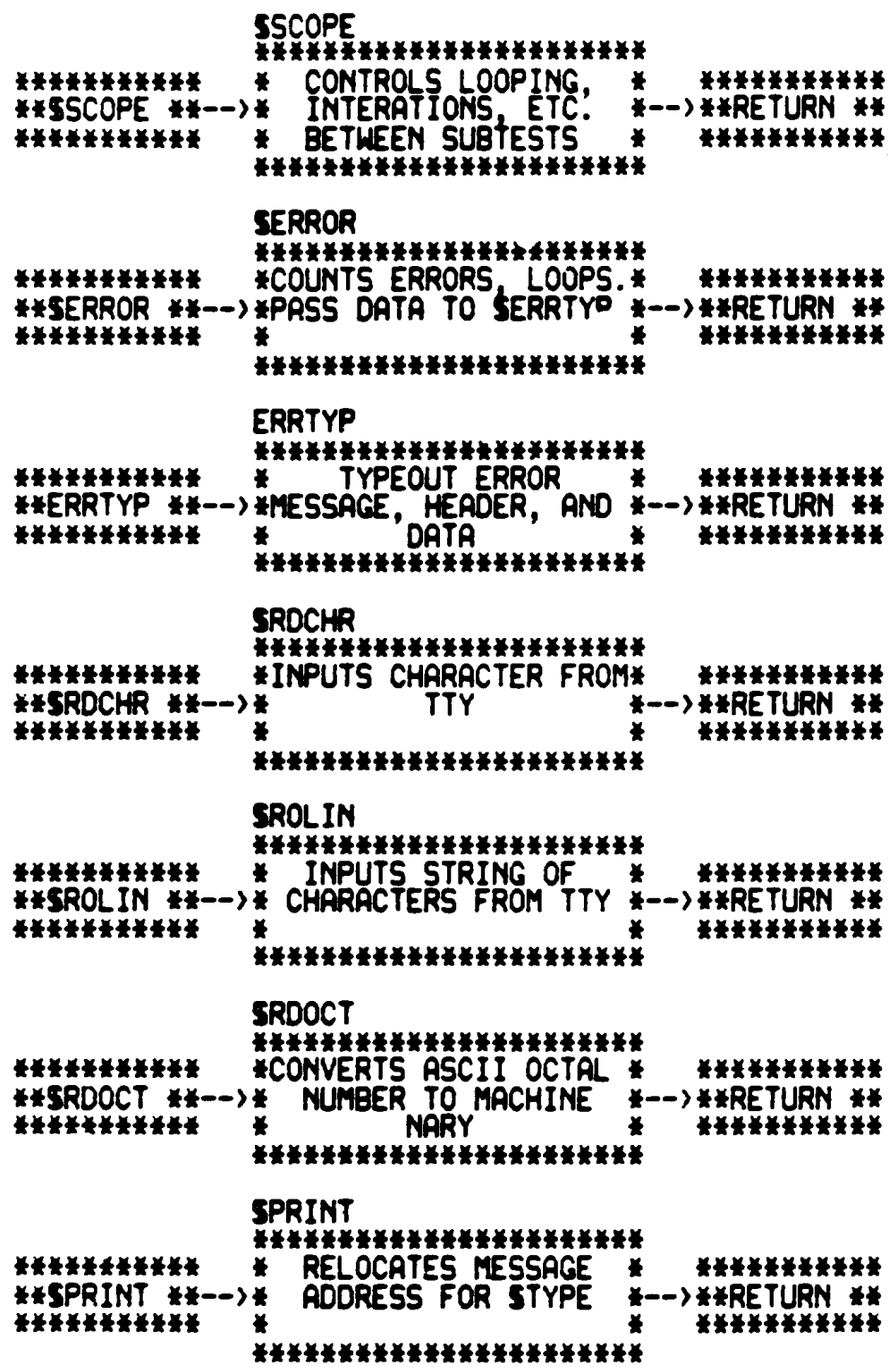

**RELO **
RELO
NO
MEMORY MANAGEMENT?
YES
RELOC(56)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
*ADJUST ALL PERTINENT *
* ADDRESS POINTERS *
* SET UP MEM MGMT *
REGISTERS TO POINT TO
* BANKS 0+1 *
RELOC(56)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
**RETURN **

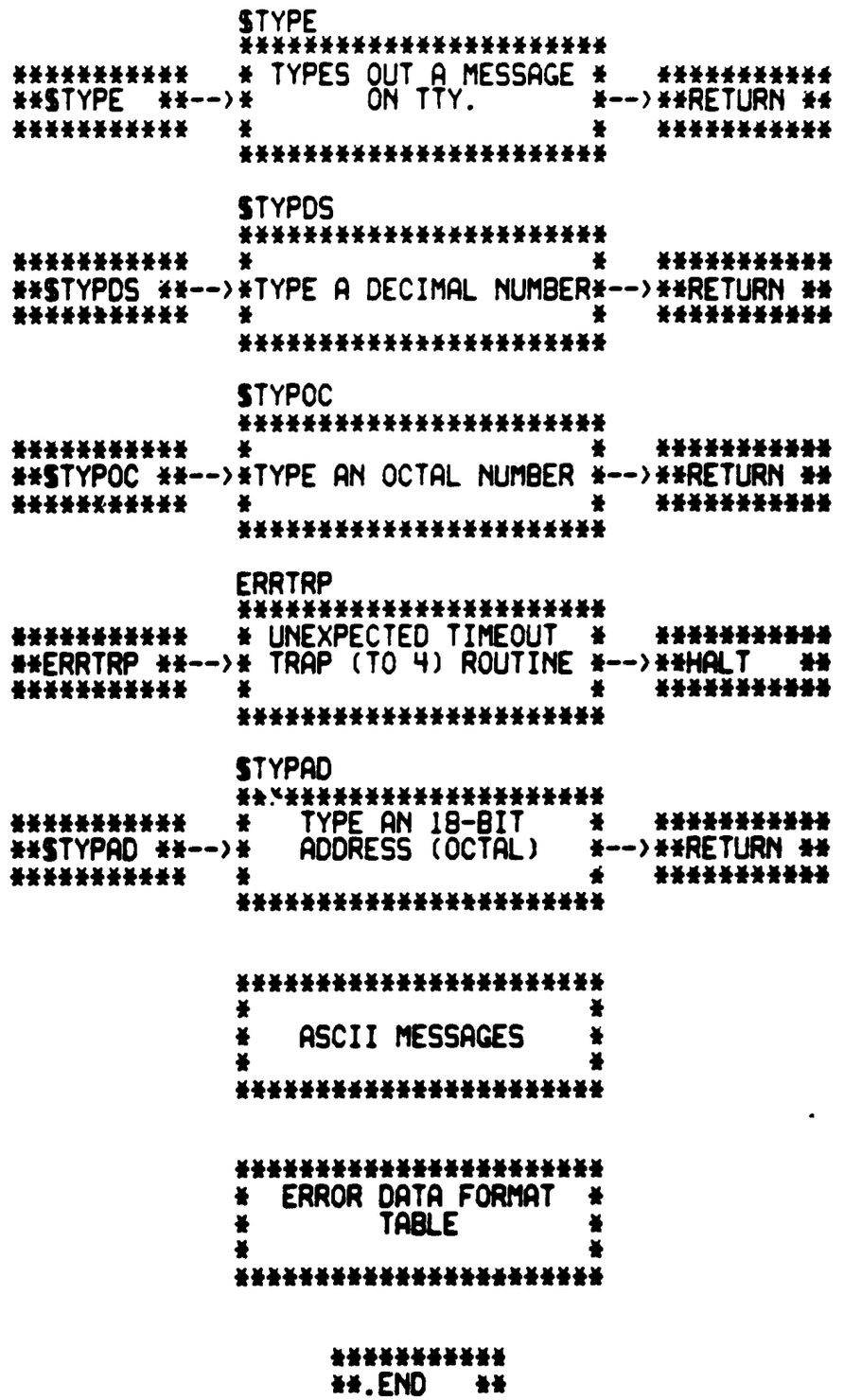

**RESLDR **
RESLDR
* MOVE "LOADER" FROM *
* END OF 8K TO TOP OF *
* MEMORY *
**RETURN **


**PESRV **
PESRV
TYPE UNEXPECTED TRAP MESSAGE
ERROR FLAG SET IN ANY PARITY REGISTER?
NO
YES
** REPORT TRAP PC AND REGISTER DATA **
PSCAN(59)
** SCAN MEMORY FOR ALL **BAD PARITY LOCATIONS **
**RETURN **
$ERROR(61)
** ERROR: TRAP BUT NO FLAG **
**SETAE **
**MAMF **
MAMF
PARITY REGISTER EXIST AND NOT INHIBITED?
NO
YES
SET UP PARITY VECTOR. SET 'ACTION ENABLE' IN ALL REGISTERS
**RETURN **
**CLRPAR **
CLRPAR
*CLEAR OUT ALL MEMORY * PARITY REGISTERS
**RETURN **


**CKPMER **
CKPMER
PARITY REGISTER EXIST AND NOT INHIBITED?
NO
ANY ERROR FLAGS SET?
NO
ERROR SHOULD HAVE TRAPPED?
YES
$ERROR(61)
** ERROR: PARITY ERROR **
** SHOULD HAVE TRAPPED **
$ERROR(61)
** ERROR: PARITY ERROR **
PSCAN(59)
** SCAN MEMORY FOR ALL **
BAD PARITY LOCATIONS.
**RETURN **
**PSCAN **
PSCAN
TYPE SCANNING MESSAGE
*READ MEMORY LOCATION *
ANY PARITY ERROR FLAGS?
YES
$ERROR(61)
** ERROR: PARITY ERROR **
** AT LOCATION. **
NO
*UPDATE ADDRESS POINTERS *
IDONE
ANY PARITY ERRORS FOUND?
NO
$ERROR(61)
** ERROR: NO PARITY ERRORS FOUND **
YES
**RETURN **


**SPRNT **
**SPRNTQ **
**SPRNTR **
**SPRNTO **
**SPRNT1 **
**SPRNT3 **
**SPRNT2 **
* ROUTINES TO SET UP *
* DATA FOR ERROR *
* TYPEOUTS. *
**RETURN **
**TYPMAP **
TYPMAP
MAP CONTAIN FLAGS?
NO
TYPE EMPTY MAP MESSAGE
YES
TYPE FIRST + LAST ADDRESS OF BANKS FOUND
**RETURN **


$SCOPE
CONTROLS LOOPING, INTERATIONS, ETC. BETWEEN SUBTESTS
**$SCOPE **
**RETURN **
$ERROR
COUNTS ERRORS, LOOPS.
PASS DATA TO $ERRTYP
**$ERROR **
**RETURN **
ERRTYP
TYPEOUT ERROR MESSAGE, HEADER, AND DATA
**ERRTYP **
**RETURN **
$RDCHR
INPUTS CHARACTER FROM TTY
**$RDCHR **
**RETURN **
$ROLIN
INPUTS STRING OF CHARACTERS FROM TTY
**$ROLIN **
**RETURN **
$RDOCT
CONVERTS ASCII OCTAL NUMBER TO MACHINE NARY
**$RDOCT **
**RETURN **
$PRINT
RELOCATES MESSAGE ADDRESS FOR $TYPE
**$PRINT **
**RETURN **


$TYPE
TYPES OUT A MESSAGE ON TTY.
**$TYPE **
**RETURN **
$TYPDS
TYPE A DECIMAL NUMBER
**$TYPDS **
**RETURN **
$TYPOC
TYPE AN OCTAL NUMBER
**$TYPOC **
**RETURN **
ERRTRP
UNEXPECTED TIMEOUT TRAP (TO 4) ROUTINE
**ERRTRP **
**HALT **
$TYPAD
TYPE AN 18-BIT ADDRESS (OCTAL)
**$TYPAD **
**RETURN **
ASCII MESSAGES
ERROR DATA FORMAT TABLE
**.END **

FLOW CHART CROSS REFERENCE LIST

```
BANKNO      55*   55
BGERR       50
BGEXIT      50
BGTOP       50
BRGCB       50
CKPMER      24    39    54    59*   59
CLRPAR      58*   58
CTRLS       14
DONE        51
ERRTRP      61    61*
ERRTYP      61    61*
EXIT        52
GMPR        08    12    12
GMPRA       12
GMPRB       12
GMPRC       12
HALT        03    04    04    08    11    56    61
INITDN      17    18    19    21    21    53*   53
INITEX      53
INITMM      17    18    19    20    20    22    23    24    25    26    27    27    28    28    29    29    30    31
            31    32    33    33    34    35    35    36    37    37    38    38    39    43    44    45    46    47
            48    49    50    53*   53    55
MAMF        42    58*   58
MANUAL      12    15*   15
MANUL1      15
MANUL2      16
MAPRB       13
MAUT        44    45    46    47    48    49
MMDOWN      17    18    19    21    21    53    54*   54
MMINIT      09    53*   53
MMUP        17    18    19    20    20    22    23    24    25    26    27    27    28    28    29    29    30    31
            31    32    33    33    34    35    35    36    37    37    38    38    42    43    44    45    46    47
            48    49    50    54*   54    55
PBTRP       40
PESRV       58*   58
PHYADR      55*   55
PSCAN       58    59    59*   59
RELO        02    51    57*   57
RELOC       56*   56    56    56    57    57
RELTOP      51    56*   56
RESCHK      14
RESLDR      03    51    57*   57
RESTI       02
RESTAR      02*   02    06
RESTOR      02*
RETURN      04    44    45    46    47    48    49    53    53    54    54    54    54    55    55    55    55    55
            56    56    57    57    58    58    58    59    59    60    60    61    61    61    61    61    61    61
            61    61    61    61
ROTATE      25    26    55*   55
SELECT      06*
SETAE       39    58*
```

FLOW CHART CROSS REFERENCE LIST

	26	39	55*	55													
SPRNT	60*																
SPRNT0	60*																
SPRNT1	60*																
SPRNT2	60*																
SPRNT3	60*																
SPRNTQ	60*																
SPRNTR	60*																
START	06*																
START1	03	16*	16	51	52												
STARTA	06																
TIMOUT	09*																
TMAP	13																
TST1	17																
TST11	25																
TST12	26																
TST13	27																
TST14	29																
TST15	31																
TST16	33																
TST17	35																
TST2	18																
TST20	37																
TST21	39																
TST22	39	43*	43														
TST23	44																
TST24	45																
TST25	46																
TST26	47																
TST27	48																
TST3	19																
TST30	49																
TST31	50																
TST4	20																
TST5	21																
TST6	22																
TST7	23																
TYPMAP	13	60*	60														
W3X9	29	31	33	35	55*	55											
WWPB1	39	39*	41														
WWPB2	40	40*	41														
WWPB3	41																
WWPB4	40	40	41	41*													
WWPB5	39	39	41	41*													
WWPBT	39	39*	42														
$EOP	52																
$ERROR	14	14	14	17	18	19	20	21	23	24	25	26	27	28	29	30	30	30
	31	32	32	32	33	34	34	34	35	36	36	36	37	38	40	40	40	40
	40	41	41	41	43	44	45	46	47	48	49	50	56	58	59	59	59	59
	61	61*																
$ILLUP	04*																	
$PRINT	04	09	10	10	15	61	61*											

$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#		

```
                        .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.
                        :*

                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          --------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            INHIBIT KT11 (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 EXRERCISING 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
                        .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
```

```
                        PC=     %7              ;;PROGRAM COUNTER

                        ;*PRIORITY LEVEL DEFINITIONS
000000                  PR0=    0               ;;PRIORITY LEVEL 0
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7

                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06=   100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
                        .EQUIV  SW02,SW2
                        .EQUIV  SW01,SW1
                        .EQUIV  SW00,SW0

                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
```

```
                        BIT00=  1
                        .EQUIV  BIT09,BIT9
                        .EQUIV  BIT08,BIT8
                        .EQUIV  BIT07,BIT7
                        .EQUIV  BIT06,BIT6
                        .EQUIV  BIT05,BIT5
                        .EQUIV  BIT04,BIT4
                        .EQUIV  BIT03,BIT3
                        .EQUIV  BIT02,BIT2
                        .EQUIV  BIT01,BIT1
                        .EQUIV  BIT00,BIT0

                        ;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014                  TBITVEC=14              ;;"T" BIT
000014                  TRTVEC= 14              ;;TRACE TRAP
000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024                  PWRVEC= 24              ;;POWER FAIL
000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
000034                  TRAPVEC=34              ;;"TRAP" TRAP
000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
000064                  TPVEC=  64              ;;TTY PRINTER VECTOR
000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR


                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                        ;*KT11 VECTOR ADDRESS

000250                  MMVEC=  250

                        ;*KT11 STATUS REGISTER ADDRESSES

177572                  SR0=    177572
177574                  SR1=    177574
177576                  SR2=    177576
172516                  SR3=    172516

                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

172300                  KIPDR0= 172300
172302                  KIPDR1= 172302
172304                  KIPDR2= 172304
172306                  KIPDR3= 172306
172310                  KIPDR4= 172310
172312                  KIPDR5= 172312
172314                  KIPDR6= 172314
172316                  KIPDR7= 172316

                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

172340                  KIPAR0= 172340
172342                  KIPAR1= 172342
172344                  KIPAR2= 172344
```

```
                        172346          KIPAR3= 172346
                        172350          KIPAR4= 172350
                        172352          KIPAR5= 172352
                        172354          KIPAR6= 172354
                        172356          KIPAR7= 172356

                        000000          UP = 0                  ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
                        000006          RW = 6                  ;CODE FOR READ/WRITE IN MEM MGMT PDR'S

                                        ;*  PARITY MEMORY DEFINITIONS.
                        000004          WWP=4                   ;WRITE WRONG PARITY
                        000001          AE=1                    ;PARITY ACTION ENABLE
                        000114          PARVEC=114              ;PARITY TRAP VECTOR

                                        ;*  MISCELLANEOUS ASSIGNMENTS
                        017777          MASK4K= 17777           ;MASK FOR 4K ADDRESS BANK BOUNDRY.

                                        ;* CACHE REGISTER DEFINITIONS.
                        177746          IMPCHE= 177746

                                        .SBTTL  TRAP CATCHER

                        000000                  .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                                .=174
000174  000000                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  002666                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000167  002464                  JMP     SELECT          ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
                                                                ;SELECT VARIOUS PARAMETERS.
000210  000167  000064                  JMP     RESTAR          ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
000214  000167  000064                  JMP     RESTOR          ;RESTORE LOADERS TO END OF MEMORY AND HALT.
000220  000167  003406                  JMP     TIMOUT          ;TYPE OUT MEMORY MAP, BYTE BY BYTE.

                        000004          .=ERRVEC
000004  025200                          .WORD   ERRTRP
000006  000000                          .WORD   0

                                .SBTTL  ACT11 HOOKS

                                ;;**************************************************************
                                ;HOOKS REQUIRED BY ACT11
                        000010          $SVPC=.                 ;SAVE PC
                        000046          .=46
000046  014760                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SCOP
                        000052          .=52
000052  040000                          .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
                        000010          .=$SVPC                 ;; RESTORE PC
```

```
                        .=300
                ;;*****************************************************************
                ;*  THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
                ;*   THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
                ;*   THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
                ;*   MANAGEMENT IS DISABLED BY "CONSOLE START".
                ;;*****************************************************************
000300  005005       RESTAR: CLR     R5              ;CLEAR FLAG TO INDICATE RESTART.
000302  000401               BR      REST1           ;GO RESTORE PROGRAM BEFORE RESTARTING.
000304  010705       RESTOR: MOV     PC,     R5      ;PUT DATA INTO FLAG FOR RESTORE.
000306  012706  001100 REST1:  MOV     #STACK, SP      ;SET UP THE STACK POINTER.
000312  005767  001206         TST     MEMMAP          ;CHECK IF THE MEMORY HAS BEEN MAPPED.
000316  001002               BNE     REST2           ;BR IF MEMORY MAPPED.
000320  000167  002356         JMP     START           ;GO START
000324  005767  000256 REST2:  TST     MMAVA           ;CHECK IF MEM MGMT AVAILABLE.
000330  001470               BEQ     10$             ;BR IF NO MEM MGMT.
000332  032737  000001  177572 BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ACTIVE.
000340  001034               BNE     2$              ;BR IF MEM MGMT ALREADY SET UP.
000342  012700  172300         MOV     #KIPDR0,R0      ;POINT TO FIRST MEM MGMT DDATA REG.
000346  012701  000010         MOV     #8.,    R1      ;SET UP COUNTER.
000352  012720  077406 1$:     MOV     #077406,(R0)+   ;MAP FIRST 28K 1-FOR-1.
000356  005301               DEC     R1              ;COUNT REGESTERS.
000360  001374               BNE     1$              ;BR IF MORE REG.
000362  012700  172340         MOV     #KIPAR0,R0      ;POINT TO FIRST MEM MGMT ADDRESS REG.
000366  005020               CLR     (R0)+           ;PAR0 MAPPED INTO BANK0.
000370  012720  000200         MOV     #200,   (R0)+   ;PAR1 MAPPED INTO BANK1.
000374  012720  000400         MOV     #400,   (R0)+   ;PAR2 MAPPED INTO BANK2.
000400  012720  000600         MOV     #600,   (R0)+   ;PAR3 MAPPED INTO BANK3.
000404  012720  001000         MOV     #1000,  (R0)+   ;PAR4 MAPPED INTO BANK4.
000410  012720  001200         MOV     #1200,  (R0)+   ;PAR5 MAPPED INTO BANK5.
000414  012720  001400         MOV     #1400,  (R0)+   ;PAR6 MAPPED INTO BANK6.
000420  012720  007600         MOV     #7600,  (R0)+   ;PAR7 MAPPED INTO BANK37.
000424  012737  000001  177572 MOV     #BIT0,  @#SR0   ;ENABLE MEM MGMT.
000432  005000         2$:     CLR     R0              ;INIT TEMP PAR REG.
000434  016701  000142         MOV     PRGMAP, R1      ;GET THE PROGRAM MAP...LO 64K.
000440  016702  000140         MOV     PRGMAP+2,R2     ;...HI 64K.
000444  006202         3$:     ASR     R2              ;SHIFT THE MAP POINTER...HI
000446  006001               ROR     R1              ;...LO.
000450  103404               BCS     4$              ;BR WHEN FIRST BANK FOUND.
000452  062700  000200         ADD     #200,   R0      ;UPDATE TMP PAR TO NEXT BANK.
000456  100372               BPL     3$              ;BR IF MORE.
000460  000000               HALT                    ;FATAL ERROR!!!  MAP EMPTY?
000462  010037  172340 4$:     MOV     R0,     @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
000466  000137  000472         JMP     @#5$            ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
000472  062700  000200 5$:     ADD     #200,   R0      ;KEEP UPDATING TEMP PAR REG.
000476  006202               ASR     R2              ;SHIFT POINTER...HI
000500  006001               ROR     R1              ;...LO
000502  103373               BCC     5$              ;BR IF TOP BANK NOT YET FOUND.
000504  010037  172342         MOV     R0,     @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
000510  000410               BR      20$             ;BR TO RELOCATE SECTION.
000512  016700  000062 10$:     MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
000516  062700  001100         ADD     #STACK, R0      ;SET UP STACK POINTER.
000522  010006               MOV     R0,     SP      ;SET STACK TO RELOCATE PROGRAM.
000524  062700  177432         ADD     #20$-STACK,R0   ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
000530  000110               JMP     (R0)            ;GO TO "20$" (RELOCATED).
000532  022767  000003  000042 20$:    CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.
```

```
  276  000540  001402                        BEQ     21$             ;BR IF IN BANKS 0 AND ..
  277  000542  004767  017030                JSR     PC,     RELO    ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
  278  000546  005705                21$:    TST     R5              ;CHECK RESTART/RESTORE FLAG.
  279  000550  001006                        BNE     22$             ;BR IF RESTORE.
  280  000552  005067  000412                CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
  281  000556  105067  000320                CLRB    $TSTNM
  282  000562  000167  005220                JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
  283  000566  004767  017212        22$:    JSR     PC,     RESLDR  ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
  284  000572  000000                        HALT                    ;HALT AFTER RESTORING THE LOADERS.
  285  000574  000167  002102                JMP     STARTA          ;CONTINUE WILL RESTART THE PROGRAM.
  286                                ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
  287                                ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
  288                                ;*  CIRCUMSTANCES.
  289  000600  000000                RELOCF: .WORD   0               ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
  290  000602  000000  000000        PRGMAP: .WORD   0,0             ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
  291  000606  000000                MMAVA:  .WORD   0               ;MEMORY MANAGEMENT AVAILABLE FLAG.
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*****************************************************************
                                ;POWER DOWN ROUTINE
000610  012737  000756  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
000616  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
000624  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
000626  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
000630  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
000632  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
000634  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
000636  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
000640  017746  000274                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
000644  010667  000112                MOV     SP,$SAVR6       ;;SAVE SP
000650  012737  000662  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
000656  000000                        HALT
000660  000776                        BR      .-2             ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
000662  012737  000756  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
000670  016706  000066                MOV     $SAVR6,SP       ;;GET SP
000674  005067  000062                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
000700  005267  000056          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
000704  001375                        BNE     1$              ;;OF  WORD
000706  012677  000226                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
000712  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
000714  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
000716  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
000720  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
000722  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
000724  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
000726  012737  000610  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
000734  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
000742  004567  022630                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000746  025725                $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
000750  012716                        MOV     (PC)+,(SP)      ;;RESTART AT RESTART
000752  000300                $PWRAD: .WORD   RESTART         ;;RESTART ADDRESS
000754  000002                        RTI
000756  000000                $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
000760  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
000762  000000                $SAVR6: 0                       ;;PUT THE SP HERE
```

```
                                .SBTTL  COMMON TAGS

                                ;;**********************************************************************
                                ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                ;*USED IN THE PROGRAM.

                    001100              .=1100
001100                          $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                          .WORD   0
001102     000      $TSTNM: .BYTE   0   ;;CONTAINS THE TEST NUMBER
001103     000      $ERFLG: .BYTE   0   ;;CONTAINS ERROR FLAG
001104  000000      $ICNT:  .WORD   0   ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000      $LPADR: .WORD   0   ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000      $LPERR: .WORD   0   ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000      $ERTTL: .WORD   0   ;;CONTAINS TOTAL ERRORS DETECTED
001114     000      $ITEMB: .BYTE   0   ;;CONTAINS ITEM CONTROL BYTE
001115     001      $ERMAX: .BYTE   1   ;;CONTAINS MAX. ERRORS PER TEST
001116  000000      $ERRPC: .WORD   0   ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000      $GDADR: .WORD   0   ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000      $BDADR: .WORD   0   ;;CONTAINS ADDRESS OF 'BAD' DATA
001124  000000      $GDDAT: .WORD   0   ;;CONTAINS 'GOOD' DATA
001126  000000      $BDDAT: .WORD   0   ;;CONTAINS 'BAD' DATA
001130  000000              .WORD   0   ;;RESERVED--NOT TO BE USED
001132  000000              .WORD   0
001134     000      $AUTOB: .BYTE   0   ;;AUTOMATIC MODE INDICATOR
001135     000      $INTAG: .BYTE   0   ;;INTERRUPT MODE INDICATOR
001136  000000              .WORD   0
001140  177570      SWR:    .WORD   DSWR        ;;ADDRESS OF SWITCH REGISTER
001142  177570      DISPLAY: .WORD  DDISP       ;;ADDRESS OF DISPLAY REGISTER
001144  177560      $TKS:   177560              ;;TTY KBD STATUS
001146  177562      $TKB:   177562              ;;TTY KBD BUFFER
001150  177564      $TPS:   177564              ;;TTY PRINTER STATUS REG. ADDRESS
001152  177566      $TPB:   177566              ;;TTY PRINTER BUFFER REG. ADDRESS
001154     000      $NULL:  .BYTE   0           ;;CONTAINS NULL CHARACTER FOR FILLS
001155     002      $FILLS: .BYTE   2           ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156     012      $FILLC: .BYTE   12          ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157     000      $TPFLG: .BYTE   0           ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
001160  000000      $TMP0:  .WORD   0           ;;USER DEFINED
001162  000000      $TMP1:  .WORD   0           ;;USER DEFINED
001164  000000      $TMP2:  .WORD   0           ;;USER DEFINED
001166  000000      $TMP3:  .WORD   0           ;;USER DEFINED
001170  000000      $TIMES: 0                   ;;MAX. NUMBER OF ITERATIONS
001172  000000      $ESCAPE:0                   ;;ESCAPE ON ERROR ADDRESS
001174  177607  000377  $BELL:  .ASCIZ  <207><377><377>  ;;CODE FOR BELL
001200     077      $QUES:  .ASCII  /?/         ;;QUESTION MARK
001201     015      $CRLF:  .ASCII  <15>        ;;CARRIAGE RETURN
001202  000012      $LF:    .ASCIZ  <12>        ;;LINE FEED
                                ;;**********************************************************************
                                .SBTTL  APT MAILBOX-ETABLE

                                ;;**********************************************************************
                                .EVEN
001204                          $MAIL:                      ;;APT MAILBOX
001204  000000      $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001206  000000      $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001210  000000      $TESTN: .WORD   ATESTN   ;;TEST NUMBER
```

```
001212  000000          $PASS:   .WORD   APASS    ;;PASS COUNT
001214  000000          $DEVCT:  .WORD   ADEVCT   ;;DEVICE COUNT
001216  000000          $UNIT:   .WORD   AUNIT    ;;I/O UNIT NUMBER
001220  000000          $MSGAD:  .WORD   AMSGAD   ;;MESSAGE ADDRESS
001222  000000          $MSGLG:  .WORD   AMSGLG   ;;MESSAGE LENGTH
001224                  $ETABLE:                  ;;APT ENVIRONMENT TABLE
001224     000          $ENV:    .BYTE   AENV     ;;ENVIRONMENT BYTE
001225     000          $ENVM:   .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001226  000000          $SWREG:  .WORD   ASWREG   ;;APT SWITCH REGISTER
001230  000000          $USWR:   .WORD   AUSWR    ;;USER SWITCHES
001232  000000          $CPUOP:  .WORD   ACPUOP   ;;CPU TYPE,OPTIONS
                        ;*                        BITS 15-11=CPU TYPE
                        ;*                                11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
                        ;*                                11/70=06,PDQ=07,Q=10
                        ;*                        BIT 10=REAL TIME CLOCK
                        ;*                        BIT  9=FLOATING POINT PROCESSOR
                        ;*                        BIT  8=MEMORY MANAGEMENT
001234     000          $MAMS1:  .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
001235     000          $MTYP1:  .BYTE   AMTYP1   ;;MEM. TYPE,BLK#1
                        ;*                        MEM.TYPE BYTE   --  (HIGH BYTE)
                        ;*                                900 NSEC CORE=001
                        ;*                                300 NSEC BIPOLAR=002
                        ;*                                500 NSEC MOS=003
001236  000000          $MADR1:  .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                        ;*                          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001240     000          $MAMS2:  .BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
001241     000          $MTYP2:  .BYTE   AMTYP2   ;;MEM. TYPE,BLK#2
001242  000000          $MADR2:  .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
001244     000          $MAMS3:  .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
001245     000          $MTYP3:  .BYTE   AMTYP3   ;;MEM. TYPE,BLK#3
001246  000000          $MADR3:  .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
001250     000          $MAMS4:  .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
001251     000          $MTYP4:  .BYTE   AMTYP4   ;;MEM. TYPE,BLK#4
001252  000000          $MADR4:  .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
001254  000000          $VECT1:  .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001256  000000          $VECT2:  .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001260  000000          $BASE:   .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001262  000000          $DEVM:   .WORD   ADEVM    ;;DEVICE MAP
001264  000000          $CDW1:   .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
001266  000000          $CDW2:   .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
001270  000000          $DDW0:   .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
001272  000000          $DDW1:   .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
001274  000000          $DDW2:   .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
001276  000000          $DDW3:   .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
001300  000000          $DDW4:   .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
001302  000000          $DDW5:   .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
001304  000000          $DDW6:   .WORD   ADDW6    ;;DEVICE DESCRIPTOR WORD#6
001306  000000          $DDW7:   .WORD   ADDW7    ;;DEVICE DESCRIPTOR WORD#7
001310  000000          $DDW8:   .WORD   ADDW8    ;;DEVICE DESCRIPTOR WORD#8
001312  000000          $DDW9:   .WORD   ADDW9    ;;DEVICE DESCRIPTOR WORD#9
001314  000000          $DDW10:  .WORD   ADDW10   ;;DEVICE DESCRIPTOR WORD#10
001316  000000          $DDW11:  .WORD   ADDW11   ;;DEVICE DESCRIPTOR WORD#11
001320  000000          $DDW12:  .WORD   ADDW12   ;;DEVICE DESCRIPTOR WORD#12
001322  000000          $DDW13:  .WORD   ADDW13   ;;DEVICE DESCRIPTOR WORD#13
001324  000000          $DDW14:  .WORD   ADDW14   ;;DEVICE DESCRIPTOR WORD#14
001326  000000          $DDW15:  .WORD   ADDW15   ;;DEVICE DESCRIPTOR WORD#15
```

```
001330                          $ETEND:
                                .MEXIT
                                .SBTTL  APT PARAMETER BLOCK

                                ;;*****************************************************************
                                ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                ;;*****************************************************************
          001330                        .$X=.   ;;SAVE CURRENT LOCATION
          000024                        .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000024    000200                        200     ;;FOR APT START UP
          000044                        .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044    001330                        $APTHDR ;;POINT TO APT HEADER BLOCK
          001330                        .=.$X   ;;RESET LOCATION COUNTER
                                ;;*****************************************************************
                                ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                ;INTERFACE SPEC.

001330                          $APTHD:
001330  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001332  001204                  $MBADR: .WORD   $MAIL   ;;ADDRESS OF APT MAILBOX (BITS 0-15)
001334  004540                  $TSTM:  .WORD   2400.   ;;RUN TIM OF LONGEST TEST
001336  000170                  $PASTM: .WORD   120.    ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
001340  000360                  $UNITM: .WORD   240.    ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
001342  000052                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
                                .SBTTL  APT STATISTICS TABLE

                                ;;*****************************************************************
001344                          $ASTAT:
001344  177777  000000                .WORD   -1,0
001350  177777  000000                .WORD   -1,0
001354  177777  000000                .WORD   -1,0
001360  177777  000000                .WORD   -1,0
001364  177777  000000                .WORD   -1,0
001370  177777  000000                .WORD   -1,0
001374  177777  000000                .WORD   -1,0
001400  177777  000000                .WORD   -1,0
001404  177777  000000                .WORD   -1,0
001410  177777  000000                .WORD   -1,0
001414  177777  000000                .WORD   -1,0
001420  177777  000000                .WORD   -1,0
001424  177777  000000                .WORD   -1,0
001430  177777  000000                .WORD   -1,0
001434  177777  000000                .WORD   -1,0
001440  177777  000000                .WORD   -1,0
001444  177777  000000                .WORD   -1,0
001450  177777  000000                .WORD   -1,0
001454  177777  000000                .WORD   -1,0
001460  177777  000000                .WORD   -1,0
001464  177777  000000                .WORD   -1,0
001470  177777  000000                .WORD   -1,0
001474  177777  000000                .WORD   -1,0
001500  177777  000000                .WORD   -1,0
001504  177777  000000                .WORD   -1,0
001510  177777                  $ASTEND:        -1
001512  001344                  $APTR:  $ASTAT
```

```
                                ;;*****************************************************************
                                ;*THE FOLLOWING TAGS ARE USER DEFINED
                                ;;*****************************************************************
001514  000000                  $VERPC: .WORD   0       ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
001516  070032                  RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
                                                        ;NOTE:  FOR MS1' MEMORY WITH PARITY, CHANGE TO 077772.
001520  000000                  LMAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
001522  000000                  LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE
001524                          MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
001524  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001526  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001530                          TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
001530  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001532  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001534                          SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
                                                        ; TEST EACH BANK ONCE.
001534  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001536  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001540                          PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
001540  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001542  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001544                          BITPT:                  ;POINTER TO CURRENT 4K BANK OF MEMORY
001544  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001546  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001550                          TMPPT:                  ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
001550  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001552  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001554  000000                  MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
                                                        ;SET UP BY "INITMM" AND "INITDN" ROUTINEES.
                                                        ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
001556  000                     SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
001557  000                     FLAG8K: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP".
001560  000                     OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
        001562                          .EVEN
001562  000000                  FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
                                                        ;FIRST ADDRESS IS USER SELECTABLE.
001564  000000                  TMPFAD: .WORD   0       ;ADJUSTED FIRST ADDRESS.
001566  000000                  FADMSK: .WORD   0       ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
                                                        ; TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
001570  000000  000000          FADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
001574  000000                  LSTADR: .WORD   0       ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
                                                        ;LAST ADDRESS IS USER SELECTABLE.
001576  000000                  TMPLAD: .WORD   0       ;ADJUSTED LAST ADDRESS.
001600  000000                  LADMSK: .WORD   0       ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
                                                        ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
001602  000000  000000          LADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
001606  000000                  BLKMSK: .WORD   0       ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
001610  000000                  .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.

                                ;;*****************************************************************
                                ;*  RELATIVE ADDRESSING TABLE.
                                ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
                                ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMEMT TAGS.
                                ;;*****************************************************************
001612                          RADTAB:
001612  001100                  .STACK: STACK           ;STACK POINTER INITIAL ADDRESS.
```

```
                                .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
                                .MPRO:  MPRO            ;MEMORY PARITY REGISTER TABLE ADDRESS.
                                .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
                                .BRGOB: BRGOB           ;"BRANCH GOBBLE" ROUTINE ADDRESS.
                                .BGERR: BGERR           ;"BRANCH GOBBLE" ERROR ROUTINE ADDRESS.
                                .BGEXI: BGEXIT          ;"BRANCH GOBBLE" EXIT ROUTINE ADDRESS.
                                .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
                                .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
                                .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
                                .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
                                .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
                                .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
                                ;;**************************************************************
                                ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
                                ;;**************************************************************
                                DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0

                                DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0

                                DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0

                                DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0

                                DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0

                                DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0

                                DT22:   $TMP0,$TMP1,$TMP2,$TMP3,$GDADR,$GDDAT,$BDDAT,0

                                DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0

                                DT24:   $VERPC,$ERRPC,$BDADR,0

                                DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0

                                DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0

                                DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0

                                DT31:   $TMP3,0
                                        .WORD   -1      ;TABLE TERMINATOR.

                                .SBTTL  MEMORY PARITY PATTERNS TABLE
                                ;;**************************************************************
                                ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
                                ;;**************************************************************

                                MPPATS: 125325                  ;EVEN,ODD
                                        152652                  ;ODD,EVEN
                                        052452                  ;EVEN,ODD
                                        025125                  ;ODD,EVEN
                                        102070                  ;EVEN,EVEN
```

```
614  002100  072527                  072527                  ;ODD,ODD
615  002102  177777                  177777                  ;EVEN,EVEN
616  002104  107030                  107030                  ;ODD,ODD
617  002106  152525                  152525                  ;ODD,EVEN
618  002110  000000                  0                       ;EXTRA PATTERN HOLDER FOR
619                                                          ;FUTURE USE
620  002112  000000          MPEND:  0                       ;TABLE TERMINATOR
621
622
623                          .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
624                          ;////////////////////////////////////////////////////////////////
625                          ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
626                          ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
627                          ;*
628                          ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
629                          ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
630                          ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
631                          ;*  THE DEVICE ADDRESS,  EACH BIT REPRESENTING A 4K BLOCK.  I.E.
632                          ;*  FIRST WORD BIT0 = 0 - 4K,  BIT1 = 4 - 8K,...  BIT15 = 60 - 64K
633                          ;*  SECOND WORD BIT0 = 64 - 68K,... BIT14 = 120 - 124K.
634                          ;////////////////////////////////////////////////////////////////
635  002114  172101          MPR0:   172100 +1               ;PARITY STATUS REGISTER
636  002116  000000                  0                       ;CONTROL MAP (LOW 64K)
637  002120  000000                  0                       ;CONTROL MAP (HIGH 64K)
638  002122  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
639  002124  172103          MPR1:   172102 +1               ;PARITY STATUS REGISTER
640  002126  000000                  0                       ;CONTROL MAP (LOW 64K)
641  002130  000000                  0                       ;CONTROL MAP (HIGH 64K)
642  002132  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
643  002134  172105          MPR2:   172104 +1               ;PARITY STATUS REGISTER
644  002136  000000                  0                       ;CONTROL MAP (LOW 64K)
645  002140  000000                  0                       ;CONTROL MAP (HIGH 64K)
646  002142  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
647  002144  172107          MPR3:   172106 +1               ;PARITY STATUS REGISTER
648  002146  000000                  0                       ;CONTROL MAP (LOW 64K)
649  002150  000000                  0                       ;CONTROL MAP (HIGH 64K)
650  002152  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
651  002154  172111          MPR4:   172110 +1               ;PARITY STAUS REGISTER
652  002156  000000                  0                       ;CONTROL MAP (LOW 64K)
653  002160  000000                  0                       ;CONTROL MAP (HIGH 64K)
654  002162  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
655  002164  172113          MPR5:   172112 +1               ;PARITY STATUS REGISTER
656  002166  000000                  0                       ;CONTROL MAP (LOW 64K)
657  002170  000000                  0                       ;CONTROL MAP (HIGH 64K)
658  002172  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
659  002174  172115          MPR6:   172114 +1               ;PARITY STATUS REGISTER
660  002176  000000                  0                       ;CONTROL MAP (LOW 64K)
661  002200  000000                  0                       ;CONTROL MAP (HIGH 64K)
662  002202  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
663  002204  172117          MPR7:   172116 +1               ;PARITY STATUS REGISTER
664  002206  000000                  0                       ;CONTROL MAP (LOW 64K)
665  002210  000000                  0                       ;CONTROL MAP (HIGH 64K)
666  002212  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
667  002214  172121          MPR8:   172120 +1               ;PARITY STATUS REGISTER
668  002216  000000                  0                       ;CONTROL MAP (LOW 64K)
669  002220  000000                  0                       ;CONTROL MAP (HIGH 64K)
```

```
  670  002222  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  671  002224  172123          MPR9:     172122 +1               ;PARITY STATUS REGISTER
  672  002226  000000                    0                       ;CONTROL MAP (LOW 64K)
  673  002230  000000                    0                       ;CONTROL MAP (HIGH 64K)
  674  002232  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  675  002234  172125          MPR10:    172124 +1               ;PARITY STATUS REGISTER
  676  002236  000000                    0                       ;CONTROL MAP (LOW 64K)
  677  002240  000000                    0                       ;CONTROL MAP (HIGH 64K)
  678  002242  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  679  002244  172127          MPR11:    172126 +1               ;PARITY STATUS REGISTER
  680  002246  000000                    0                       ;CONTROL MAP (LOW 64K)
  681  002250  000000                    0                       ;CONTROL MAP (HIGH 64K)
  682  002252  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  683  002254  172131          MPR12:    172130 +1               ;PARITY STATUS REGISTER
  684  002256  000000                    0                       ;CONTROL MAP (LOW 64K)
  685  002260  000000                    0                       ;CONTROL MAP (HIGH 64K)
  686  002262  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  687  002264  172133          MPR13:    172132 +1               ;PARITY STATUS REGISTER
  688  002266  000000                    0                       ;CONTROL MAP (LOW 64K)
  689  002270  000000                    0                       ;CONTROL MAP (HIGH 64K)
  690  002272  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  691  002274  172135          MPR14:    172134 +1               ;PARITY STATUS REGISTER
  692  002276  000000                    0                       ;CONTROL MAP (LOW 64K)
  693  002300  000000                    0                       ;CONTROL MAP (HIGH 64K)
  694  002302  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  695  002304  172137          MPR15:    172136 +1               ;PARITY STATUS REGISTER
  696  002306  000000                    0                       ;CONTROL MAP (LOW 64K)
  697  002310  000000                    0                       ;CONTROL MAP (HIGH 64K)
  698  002312  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
  699                          ;THIS IS THE END OF THE TABLE !
  700  002314  000021          MPRX:     .BLKW   17.             ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
  701                                                            ; (THE EXTRA WORD IS FOR A TERMINATOR.)
  702
```

```
                                .SBTTL  ERROR POINTER TABLE

                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT


002356                          $ERRTB:
                                ;*  ITEM 1
002356  027071                      DM1             ;PARITY REGISTER DATA ERROR.
002360  030473                      DH1             ;PC,REG,S/B,WAS
002362  001644                      DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
002364  031103                      DF1             ;16,18,16,16
                                ;*  ITEM 2
002366  027125                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002370  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002372  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002374  031107                      DF2             ;16,18,18,16,16
                                ;*  ITEM 3
002376  027125                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002400  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002402  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002404  031114                      DF3             ;16,18,18,8,8
                                ;*  ITEM 4
002406  027161                      DM4             ;CONSTANT DATA ERROR(TST6-10).
002410  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002412  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT.$BDDAT
002414  031107                      DF2             ;16,18,18,16,16
                                ;*  ITEM 5
002416  027217                      DM5             ;ROTATING BIT ERROR(TST11-12).
002420  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002422  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002424  031107                      DF2             ;16,18,18,16,16
                                ;*  ITEM 6
002426  027255                      DM6             ;1 XOR 8 PATTERN ERROR (TST13).
002430  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002432  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT.$BDDAT
002434  031107                      DF2             ;16,18,18,16.16
                                ;*  ITEM 7
002436  027313                      DM7             ;3 XOR 9 PATTERN ERROR(TST14-17).
002440  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002442  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT.$BDDAT
002444  031107                      DF2             ;16,18,18,16,16
                                ;*  ITEM 10
002446  027354                      DM10            ;8 XOR 13 PATTERN ERROR(TST20).
002450  030512                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002452  001656                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002454  031107                      DF2             ;16,18.18.16,16
                                ;*  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,$GDDAT
  772  002504  031114                    DF3     ;16,18,18,8
  773                          ;*  ITEM 14
  774  002506  027623                    DM14    ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST21).
  775  002510  030560                    DH14    ;V/PC,P/PC,REG,MA
  776  002512  001704                    DT14    ;$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,$BDDAT
  798  002554  031107                    DF2     ;16,18,18,16,16
  799                          ;*  ITEM 21
  800  002556  030131                    DM21    ;INSTRUCTION EXICUTION ERROR(TST23-30).
  801  002560  030634                    DH21    ;V/PC,P/PC,IUT,MA,S/B,WAS
  802  002562  001734                    DT21    ;$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
```

```
  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,$BDADR
  818  002614  031121                    DF14          ;16,18,18
  819                            ;*  ITEM 25
  820  002616  030357                    DM25          ;TRAPPED TO 114.
  821  002620  031011                    DH25          ;V/PC,P/PC,TRP/PC,REG,WAS
  822  002622  002020                    DT25          ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
  823  002624  031121                    DF14          ;16,18,18,18,16
  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,$TMP0,$TMP1
  828  002634  031107                    DF2           ;16,18,18,16
  829                            ;*  ITEM 27
  830  002636  030417                    DM27          ;(ACTION ENABLE WASN'T SET).
  831  002640  031042                    DH26          ;V/PC,P/PC,REG,WAS
  832  002642  002034                    DT26          ;$VERPC,$ERRPC,$TMP0,$BDDAT
  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          ;$TMP0,$TMP1,$GDADR,$BDDAT
  838  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
  843  002664  031144                    DF30          ;18
```

```
844                                     .SBTTL START:   SETUP AND MAP MEMORY

                        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
                        ;*      THIS IS THE NORMAL (SA = 200) BEGINNING OF THE PROGRAM.
                        ;*      NOTE:   THIS CODE IS NOT POSITION INDEPENDENT.
                        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

851  002666  105067  176664          START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
852  002672  000403                          BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
853  002674  112767  177777  176654 SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
854  002702                         STARTA:
855                                  .SBTTL  INITIALIZE THE COMMON TAGS
856                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
857  002702  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
858  002706  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
859  002710  022706  001140                  CMP     #SWR,R6 ;;DONE?
860  002714  001374                          BNE     .-6             ;;LOOP BACK IF NO
861  002716  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
862                                  ;;INITIALIZE A FEW VECTORS
863  002722  012737  000610  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
864  002730  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
865  002736  016767  011752  011742          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
866                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
867                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
868  002744  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
869  002750  012737  003004  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
870  002756  012767  177570  176154          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
871  002764  012767  177570  176150          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
872  002772  022777  177777  176140          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
873  003000  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
874                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
875  003002  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
876  003004  012716  003012          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
877  003010  000002                          RTI
878  003012  012767  000176  176120  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
879  003020  012767  000174  176114          MOV     #DISPREG,DISPLAY
880  003026  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
881
882  003032  005067  176154                  CLR     $PASS           ;;CLEAR PASS COUNT
883  003036  132767  000200  176161          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
884  003044  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
885  003046  012767  001226  176064          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
886  003054                          67$:
887  003054  005067  176442                  CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
888  003060  005077  176056                  CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
889                                  .SBTTL  TYPE PROGRAM NAME
890                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
891  003064  005227  177777                  INC     #-1             ;;FIRST TIME?
892  003070  001023                          BNE     68$             ;;BRANCH IF NO
893  003072  022737  014760  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
894  003100  001417                          BEQ     68$             ;;BRANCH IF YES
895  003102  004567  020470                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
896  003106  003112                          .WORD   69$             ;ADDRESS OF MESSAGE TO BE TYPED
897  003110  000413                          BR      68$             ;;GET OVER THE ASCIZ
898                                  ;;69$:  .ASCIZ  <CRLF>'MAINDEC-11-DZQMC-C'<CRLF>
899  003140                          68$:
```

```
 900  003140  010700                        MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
 901  003142  022700  003142                CMP     #.,     R0      ;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$:  MOV     #3,     PRGMAP  ;INITIALIZE PROGRAM MAP....LO 64K.
 905  003162  005067  175416                CLR     PRGMAP+2        ;...HI 64K.
 906  003166  005067  175406                CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
 907
 908                                  ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
 909  003172  005767  176322                TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
 910  003176  001024                        BNE     14$             ;BRANCH IF ALREADY SAVED
 911  003200  012700  040000                MOV     #40000, R0      ;GET END OF 8K
 912  003204  010001                        MOV     R0,     R1      ;GET END OF 8K
 913  003206  012737  003220  000004        MOV     #12$, @#ERRVEC  ;SET UP TIMEOUT VECTOR
 914  003214  011020                 11$:   MOV     (R0),   (R0)+   ;SEARCH FOR END OF MEMORY
 915  003216  000776                        BR      11$             ;KEEP SEARCHING
 916  003220  022626                 12$:   CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
 917  003222  012737  025200  000004        MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
 918  003230  010046                        MOV     R0,     -(SP)   ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
 919  003232  012702  002734                MOV     #1500., R2      ;SET UP WORD COUNTER
 920  003236  014041                 13$:   MOV     -(R0),  -(R1)   ;SAVE THE LOADERS
 921  003240  005302                        DEC     R2              ;COUNT THE WORDS
 922  003242  001375                        BNE     13$             ;BRANCH IF MORE WORDS
 923  003244  012667  176250                MOV     (SP)+,  LMAD    ;SAVE LAST MEMORY ADDRESS
 924
 925                                  ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
 926  003250  005067  175332          14$:  CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG
 927  003254  032777  010000  175656        BIT     #SW12,  @SWR    ;CHECK FOR INHIBIT KT11 SWITCH
 928  003262  001014                        BNE     IMPCK           ;BRANCH IF SET
 929  003264  012737  003314  000004        MOV     #IMPCK,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
 930  003272  005037  177572                CLR     @#SR0           ;CLEAR MEM MGMT STATUS REG
 931  003276  004767  011512                JSR     PC,     MMINIT  ;MEM MGMT INITIALIZATION ROUTINE.
 932  003302  005267  175300                INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
 933  003306  004567  020264                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 934  003312  025440                        .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
 935                                                                 ;"KT11 AVAILABLE"
 936
 937                                  ;*      CHECK IF 11/60 CACHE PRESENT,IF SO TURN IT OFF!!!
 938  003314  012706  001100          IMPCK:  MOV     #STACK, SP
 939  003320  012737  003334  000004        MOV     #MAPMEM,@#ERRVEC
 940  003326  052767  000014  174412        BIS     #14,    IMPCHE
 941
 942                                  ;;*****************************************************************
 943                                  ;*  ROUTINE TO MAP ALL OF MEMORY.
 944                                  ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
 945                                  ;*      R0 = MEMMAP POINTER...LO 64K.
 946                                  ;*      R1 = MEMMAP POINTER...HI 64K.
 947                                  ;*      R2 = ADDRESS POINTER
 948                                  ;*      R3 = BANK POINTER...LO 64K.
 949                                  ;*      R4 = BANK POINTER...HI 64K.
 950                                  ;*      R5 = SCRATCH REGISTER.
 951                                  ;;*****************************************************************
 952  003334  012706  001100          MAPMEM: MOV     #STACK, SP      ;RESET THE STACK
 953  003340  012700  001524                  MOV     #MEMMAP,R0      ;SET UP MEMORY MAP POINTER...LO 64K.
 954  003344  012701  001526                  MOV     #MEMMAP+2,R1    ;...HI 64K.
 955  003350  005010                          CLR     (R0)            ;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 962  003374  025505                          .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
 963                                                                  ;"MEMORY MAP:"
 964  003376  012737  003512  000004          MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
 965  003404  011222                  1$:     MOV     (R2),   (R2)+   ;READ+WRITE ALL MEMORY
 966  003406  032702  017777                  BIT     #MASK4K,R2      ;CHECK FOR 4K BOUNDRY
 967  003412  001374                          BNE     1$              ;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                                  ;*  TIMEOUT TRAPS TO HERE
 990  003512  022626                  2$:     CMP     (SP)+,  (SP)+   ;RESTORE THE STACK POINTER
 991  003514  052702  017777                  BIS     #MASK4K,R2      ;LAST ADDRESS OF 4K BANK
 992  003520  005202                          INC     R2              ;FIRST ADDRESS OF NEXT BANK.
 993  003522  005767  175060          3$:     TST     MMAVA           ;CHECK FOR MEM MGMT
 994  003526  001411                          BEQ     4$              ;BRANCH IF NO MEM MGMT
 995  003530  062737  000200  172344          ADD     #200, @#KIPAR2  ;UPDATE THIRD PAR
 996  003536  012702  040000                  MOV     #40000, R2      ;POINT TO START OF THIRD PAR
 997  003542  006303                          ASL     R3              ;UPDATE LO BANK POINTER.
 998  003544  006104                          ROL     R4              ;UPDATE HI BANK POINTER
 999  003546  100316                          BPL     1$              ;BRANCH IF MORE MEMORY TO MAP.
1000  003550  000402                          BR      5$              ;EXIT WHEN DONE.
1001
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   $CRLF           ;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
```

```
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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1015  003626  025610                    .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
1016                                                            ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1017  003630  000000            6$:     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    TIMOUT: 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$,    @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
1028  003660  005037  177572            CLR     @#SR0           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1029  003664  004767  011124            JSR     PC,     MMINIT  ;INIT ALL MEM MGMT REGISTERS.
1030  003670  005267  174712            INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
1031  003674                    1$:
1032  003674  004567  017676            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1033  003700  025523                    .WORD   BYTMES          ;ADDRESS OF MESSAGE TO BE TYPED
1034                                                            ;"BYTE MEMORY MAP:"
1035  003702  005000                    CLR     R0              ;SET UP TYPE OUT FLAG.
1036  003704  005002                    CLR     R2              ;SET ADDRESS POINTER TO ZERO.
1037  003706  012737  003752  000004    MOV     #20$,   @#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,     $PRINT  ;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,     $PRINT  ;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,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1061  003776  005202                    INC     R2              ;RESET ADDRESS POINTER.
1062  004000  005100            29$:    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 BOUNDRY.
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  004066  005302                        DEC     R2              ;BACK ADDRESS POINTER UP ONE BYTE.
1081  004070  010246                        MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
1082  004072  004767  021140                JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1083  004076  000000                32$:    HALT                    ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1084                                                                ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1085  004100  000654                        BR      TIMOUT          ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1086
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     #MPR0,  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     #ERRTRP,@#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 TES"ING
1120
```

```
1121                                    .SBTTL  MAP PARITY MEMORY
1122                                    ;;*****************************************************************
1123                                    ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
1124                                    ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
1125                                    ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
1126                                    ;       - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
1127                                    ;       - PARITY GENERATE OR DETECT LOGIC FAILED
1128                                    ;       - PARITY ERROR BIT FAILED TO SET
1129                                    ;       - PARITY BITS IN MEMORY LOCATION FAILED
1130                                    ;       - I.E. BIT STUCK AT GOOD PARITY VALUE
1131                                    ;;*****************************************************************
1132
1133  004220  004767  014474           MPAMEM: JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
1134  004224  012767  000001  175312           MOV     #1,     BITPT   ;INITIALIZE 4K POINTER
1135  004232  005067  175310                   CLR     BITPT+2         ;CLEAR HI 64K POINTER
1136  004236  012702  014000                   MOV     #14000, R2      ;SET ADR POINTER TO 14000.
1137  004242  005767  174340                   TST     MMAVA           ;CHECK FOR MEM MGMT
1138  004246  001404                           BEQ     MAPRB           ;BRANCH IF NO MEM MGMT
1139  004250  012702  054000                   MOV     #54000, R2      ;SET ADR POINTER TO PAR2
1140  004254  004767  010534                   JSR     PC,     MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.
1141
1142                                    ;;*****************************************************************
1143                                    ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
1144                                    ;*  THEN WRITE TEST LOCATION VIA DATO & READ TEST LOCATION VIA DATI
1145                                    ;*  THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
1146                                    ;;*****************************************************************
1147
1148  004260  005067  175254           MAPRB:  CLR     PMEMAP          ;CLEAR THE PARITY MEMORY MAP
1149  004264  005067  175252                   CLR     PMEMAP+2
1150  004270  012703  002114           1$:     MOV     #MPRO,  R3      ;INITIALIZE TABLE ADDRESS
1151  004274  032713  000001           2$:     BIT     #1,     (R3)    ;IS THIS REGISTER PRESENT?
1152  004300  001052                           BNE     3$              ;NO - GET THE NEXT ONE
1153  004302  012773  000004  000000           MOV     #WWP,   @(R3)   ;YES - SET WRITE WRONG PARITY
1154                                                                   ;AND CLEAR REST OF REGISTER
1155  004310  011212                           MOV     (R2),   (R2)    ;WRITE WRONG PARITY
1156  004312  005712                           TST     (R2)            ;READ WRONG PARITY
1157  004314  042773  000004  000000           BIC     #WWP,   @(R3)   ;CLEAR WRITE WRONG PARITY
1158  004322  005773  000000                   TST     @(R3)           ;OTHERWISE, CHECK TO SEE IF THIS
1159                                                                   ;CONTROL REGISTER GOT A PARITY
1160                                                                   ;ERROR
1161  004326  100014                           BPL     6$              ;BRANCH IF IT DIDN'T AND CHECK
1162  004330  032773  007740  000000           BIT     #7740,  @(R3)   ;IS IT A CORE PAR. REG.
1163  004336  001404                           BEQ     5$              ;BRANCH IF NOT.
1164  004340  012763  070032  000006           MOV     #70032, 6(R3)   ;IF IT IS SET UP MASK
1165  004346  000413                           BR      7$              ;AND BRANCH TO SET BITS.
1166  004350  012763  077772  000006  5$:      MOV     #77772, 6(R3)   ;IF MOS SET UP MASK
1167  004356  000407                           BR      7$              ;AND BRANCH TO SET BIT.
1168  004360  032773  007740  000000  6$:      BIT     #7740,  @(R3)   ;IF ANY BITS ARE SET
1169  004366  001417                           BEQ     3$              ;THEN CSR IS MS11-K.
1170  004370  012763  070000  000006           MOV     #70000, 6(R3)   ;IF MS11-K SET MASK.
1171  004376  056763  175142  000002  7$:      BIS     BITPT,  2(R3)   ;SET FLAG IN MAP FOR THIS PARITY REGISTER
1172  004404  056763  175136  000004           BIS     BITPT+2,4(R3)
1173  004412  056767  175126  175120           BIS     BITPT,  PMEMAP  ;SET FLAG IN PARITY MAP
1174  004420  056767  175122  175114           BIS     BITPT+2,PMEMAP+2
1175  004426  062703  000010           3$:     ADD     #10,    R3      ;STEP UP TO NEXT REGISTER
1176  004432  020327  002314                   CMP     R3,     #MPRX   ;ARE WE DONE WITH TABLE?
```

```
1177  004436  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,     CLRPAR    ;INITIALIZE ALL PARITY REGISTERS PRESENT
1202  004540  004567  017032                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1203  004544  025546                        .WORD   MTMAP             ;ADDRESS OF MESSAGE TO BE TYPED
1204                                                                    ;"PARITY MEMORY MAP:"
1205  004546  012703  002114                MOV     #MPR0,  R3        ;INITIALIZE TABLE POINTER
1206  004552  032713  000001           1$:  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,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1223  004634  026242                        .WORD   MX5               ;ADDRESS OF MESSAGE TO BE TYPED
1224                                                                    ;"MS11-K CSR"
1225  004636                           5$:
1226  004636  004567  016734                JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1227  004642  026155                        .WORD   MX1               ;ADDRESS OF MESSAGE TO BE TYPED
1228                                                                    ;"REGISTER AT"
1229  004644  011346                        MOV     (R3),-(SP)        ;;SAVE (R3) FOR TYPEOUT
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,     $TYPOC  ;GO TO THE SUBROUTINE
1234  004656  004567  016714                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1235  004662  026172                          .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
1236                                                                  ;"CONTROLS"
1237  004664  010300                          MOV     R3,     R0      ;SET UP R0 FOR TYPMAP ROUTINE.
1238  004666  005720                          TST     (R0)+           ;UPDATE POINTER TO MAP.
1239  004670  004767  014146                  JSR     PC,     TYPMAP  ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1240  004674  062703  000010          2$:     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,     $PRINT  ;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,     $PRINT  ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1256  004766  026260                          .WORD   MX6
1257  004770  005737  002314                  TST     @#MPRX          ;ARE THERE ANY PARITY REGISTERS TO TEST?
1258  004774  001002                          BNE     CTRLS           ;IF SO TEST THE BITS IN THE REGISTERS.
1259  004776  000167  000266                  JMP     MANUAL          ;IF NO JUMP OVER REGISTER TESTS.
1260
1261                                          .SBTTL  TEST PARITY REGISTERS
1262                                          ;;*****************************************************************
1263                                          ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1264                                          ;*  CAN BE SET AND CLEARED.
1265                                          ;*  THIS IS A ONCE ONLY TEST.
1266                                          ;;*****************************************************************
1267
1268  005002  012703  002114          CTRLS.  MOV     #MPR0,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1269  005006  011302                  1$:     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,     SPRNT   ;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,     (R2)    ;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     R0,     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,     SPRNTO  ;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     R0              ;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     #MPR0,  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,    @(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     #MPR0,  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  RESE
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.,   R0      ;SET COUNTER TO CLEAR 12 WORDS.
1340  005274  012701  001562              MOV     #FSTADR,R1      ;STARTING AT FSTADR.
1341  005300  005021      1$:     CLR     (R1)+           ;CLEAR THE LOCATIONS.
1342  005302  005300                      DEC     R0              ;COUNT.
1343  005304  001375                      BNE     1$              ;BR IF MORE.
1344  005306  105767  174244              TSTB    SELFLG          ;CHECK FOR SELECT PARAMETERS STARTUP.
```

```
1345  005312  001005                        BNE     MANUL1          ;BR IF PARAMETERS TO BE SELECTED.
1346  005314  016767  173644  174252        MOV     $TMP2,  LSTADR  ;SET UP VIRTUAL LAST ADDRESS.
1347  005322  000167  000402                JMP     MANUL2          ;SKIP PARAMETER SELECTION SECTION.
```

```
                                        .SBTTL  USER PARAMETER SELECTION SECTION
                                        ;;*****************************************************************
                                        ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                        ;;*****************************************************************
005326  012700  000001          MANUL1: MOV     #BIT0,  R0      ;SET UP BANK POINTER.
005332  005001                          CLR     R1              ;...HI 64K.
005334  005002                          CLR     R2              ;CLEAR ADDRESS POINTER.
005336  005003                          CLR     R3              ;...HI ADDRESS BITS.
005340  004567  016232                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005344  026372                          .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 **SYSMAC**.
005346  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005352  004767  016046                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005356  042716  000001                  BIC     #BIT0,  (SP)    ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
005362  005067  174146                  CLR     SAVTST          ;INIT TEST MAP...LO 64K.
005366  005067  174144                  CLR     SAVTST+2        ;...HI 64K.
005372  062702  020000          1$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
005376  005503                          ADC     R3
005400  020367  016170                  CMP     R3,     $HIOCT  ;CHECK HI ADDRESS BITS.
005404  103403                          BLO     2$              ;BR IF NOT HI ENOUGH YET.
005406  101006                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005410  020216                          CMP     R2,     (SP)    ;CHECK THE LO ADDRESS BITS.
005412  101004                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005414  006300                  2$:     ASL     R0              ;UPDATE POINTER...LO 64K.
005416  006101                          ROL     R1              ;...HI 64K.
005420  100364                          BPL     1$              ;BR BACK TO CHECK NEXT BANK.
005422  000507                          BR      17$             ;BR IF OVERFLOW.
005424  030067  174074          3$:     BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS.
005430  001003                          BNE     4$              ;BR IF BANK EXISTS.
005432  030167  174070                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K.
005436  001501                          BEQ     17$             ;BR IF ADDRESS IN UN-MAPPED BANK.
005440  016704  016130          4$:     MOV     $HIOCT, R4      ;SAVE FIRST ADR HI BITS.
005444                          10$:
005444  004567  016126                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005450  026457                          .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 **SYSMAC**.
005452  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005456  004767  015742                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005462  005716                          TST     (SP)            ;CHECK IF ADR 0 SELECTED (DEFAULT).
005464  001010                          BNE     11$             ;BR IF NOT 0 (DEFAULT)
005466  005767  016102                  TST     $HIOCT          ;CHECK HI BITS.
005472  001005                          BNE     11$             ;BR IF NOT 0 (DEFAULT).
005474  016716  173464                  MOV     $TMP2,  (SP)    ;SET UP DEFAULT LAST ADR.
005500  016767  173462  016066          MOV     $TMP3,  $HIOCT
005506  012667  174062          11$:    MOV     (SP)+,  LSTADR  ;GET THE DATA.
005512  020467  016056                  CMP     R4,     $HIOCT  ;CHECK FOR LAST ADR BELOW FIRST ADR.
005516  101352                          BHI     10$             ;BB IF LAST BELOW FIRST.
005520  103403                          BLO     12$             ;BR IF LAST ABOVE FIRST.
005522  021667  174046                  CMP     (SP),   LSTADR  ;CHECK FOR LAST BELOW FIRST.
005526  101346                          BHI     10$             ;BR IF LAST BELOW FIRST.
005530  032716  017777          12$:    BIT     #MASK4K,(SP)    ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005534  001404                          BEQ     13$             ;BR IF ON BOUNDRY.
```

```
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 **SYSMAC**.
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
```

```
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;*  THE REST OF THE PROGRAM IS POSITION INDEPENDENT CODE, SO THAT IT CAN EXICUTE PROPERLY WHEN THE PROGRAM HAS BEEN RELO
        ;*  THIS IS DONE SO THAT THE FIRST TWO BANKS OF MEMORY CAN BE EXERCISED IN EXACTLY THE SAME MANNER AS THE REST OF MEMORY
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

  1465  006006  016706  173600          START1: MOV     .STACK, SP      ;SET STACK POINTER
  1466  006012  012767  006006  173066          MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
  1467  006020  066767  172554  173060          ADD     RELOCF, $LPADR
  1468  006026  004767  012134                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
  1469  006032  005767  172550                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
  1470  006036  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
  1471  006040  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
  1472  006046  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
  1473  006050  004767  006740                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
```

```
                                .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.
                                ;;*****************************************************************
006054                          TST1:
006054  004567  013200                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006060  000001                          .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
                                                                ; REQUIRED FOR THIS TEST.
006062  000167  006472                  JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
                                ;*  UPWARDS WORD ADDRESSING.
006066  004467  007050                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006072  004767  010472          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006076  010012                  2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
006100  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006102  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006104  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006106  004767  012704          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006112  004767  014132                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006116  000002                          .WORD   2               ;ERROR TYPE CODE.
006120                          65$:
006120  062700  000002                  ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
006124  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006126  001363                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006130  004767  007564                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                ;*  DOWNWARDS WORD ADDRESSING.
006134  004467  007440                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006140  004767  010424          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006144  162700  000002          4$:     SUB     #2,     R0      ;DEC DATA BY 2
006150  014201                          MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
006152  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006154  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006156  004767  012610          66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
006162  004767  014062                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006166  000002                          .WORD   2               ;ERROR TYPE CODE.
006170                          67$:
006170  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006172  001364                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006174  004767  010210                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
1521                                    ;;*****************************************************************
1522                                    ;*TEST 2        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1523                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1524                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1525                                    ;*      R2 = VIRTUAL ADDRESS
1526                                    ;*      R3 = NOT USED
1527                                    ;*      R4 = NOT USED
1528                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1529                                    ;;*****************************************************************
1530  006200                            TST2:
1531  006200  004567  013054                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1532  006204  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1533                                    ;*  UPWARDS BYTE ADDRESSING.
1534  006206  004467  006730                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1535  006212  004767  010352            1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1536  006216  110022                    2$:    MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
1537  006220  005200                            INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
1538  006222  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1539  006224  001374                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1540  006226  004767  007466                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1541
1542                                    ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1543                                    ;*  DOWNWARDS BYTE ADDRESSING.
1544  006232  004467  007342                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1545  006236  004767  010326            3$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1546  006242  005300                    4$:    DEC     R0              ;DEC DATA BY 1
1547  006244  114201                            MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
1548  006246  120001                            CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
1549  006250  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1550  006252  004767  012514            64$:   JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
1551  006256  004767  013766                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1552  006262  000003                            .WORD   3               ;ERROR TYPE CODE.
1553  006264                            65$:
1554  006264  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1555  006266  001365                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1556  006270  004767  010114                   JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
1557
1558                                    ;;*****************************************************************
1559                                    ;*TEST 3        WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
1560                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1561                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1562                                    ;*      R2 = VIRTUAL ADDRESS
1563                                    ;*      R3 = NOT USED
1564                                    ;*      R4 = NOT USED
1565                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1566                                    ;;*****************************************************************
1567  006274                            TST3:
1568  006274  004567  012760                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1569  006300  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1570                                    ;*  DOWNWARDS WORD ADDRESSING.
1571  006302  004467  007272                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1572  006306  004767  010256            1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1573  006312  005100                            COM     R0              ;COMPLEMENT THE ADR
1574  006314  062700  000002            2$:    ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
1575  006320  010042                            MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
1576  006322  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```

```
                                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
                                        JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
                                ;*  UPWARDS WORD ADDRESSING.
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
                                        COM     R0              ;COMPLEMENT IT
                                4$:
                                        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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                        .WORD   2               ;ERROR TYPE CODE.
                                65$:
                                        SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
                                        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 3$.

                                ;;*****************************************************************
                                ;*TEST 4         WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
                                ;*      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.
                                ;;*****************************************************************
                                T$T4:
                                        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,     BANKNO  ;GET THE BANK NUMBER INTO R0
                                2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
                                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
                                        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
                                ;*  UPWARDS BYTE ADDRESSING.
                                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
                                4$:     MOVB    (R2)+,  R1      ;READ THE 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,     SPRNT1  ;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,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
```

```
                                ;;*****************************************************************
                                ;*TEST 5        WRITE 1'S COMPLEMENT OF BANK #.
                                ;*      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.
                                ;;*****************************************************************
006470                          T5$:
006470  004567  012564                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006474  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                ;*  DOWNWARDS BYTE ADDRESSING.
006476  004467  007076                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006502  004767  010136          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006506  005100                          COM     R0              ;1'S COMPLEMENT OF BANK #
006510  110042                  2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
006512  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006514  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006516  004767  007666                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN CAN BE READ.
                                ;*  DOWNWARDS BYTE ADDRESSING.
006522  004467  007052                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006526  004767  010112          3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
006532  005100                          COM     R0              ;SET 1'S COMPLEMENT OF BANK #
006534  114201                  4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
006536  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006540  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006542  004767  012224          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
006546  004767  013476                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006552  000003                          .WORD   3               ;ERROR TYPE CODE.
006554                          65$:
006554  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006556  001366                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006560  004767  007624                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
                                .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 BOUNDRY BIT MASK.
                                ;;**********************************************************
006564                          TST6:
006564  004567  012470                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006570  000000                         .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006572  016700  173012         TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
006576  004467  006340                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006602  010022                 1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
006604  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006606  001375                         BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006610  004767  007104                 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.
                                ;;**********************************************************
006614                          TST7:
006614  004567  012440                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006620  000000                         .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006622  016700  172762                 MOV     .CONST, R0      ;GET USER CONSTANT
006626  004467  006310                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006632                          1$:
006632  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006634  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006636  001405                         BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006640  004767  012152         64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006644  004767  013400                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006650  000004                         .WORD   4               ;ERROR TYPE CODE.
006652                          65$:
006652  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006654  001366                         BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006656  004767  007036                 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.
006662  032777  000400  172250         BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
006670  001416                         BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
006672  017746  172242                 MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
006676  042716  177740                 BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
006702  022726  000006                 CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
006706  001007                         BNE     TST10           ;BRANCH IF NOT TEST 6
006710  162767  000001  172164         SUB     #1,     $TSTNM  ;RESET TEST NUM
006716  162767  000030  172162         SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
```

```
1725  006724  000722                        BR      TST6A           ;GO TO TEST 6
1726
1727                                  ;;*****************************************************************
1728                                  ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
1729                                          ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
1730                                  ;;*****************************************************************
1731  006726                          TST10:
1732  006726  004567  012326                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1733  006732  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1734  006734  016704  172672                MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
1735  006740  004767  011322        1$:     JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
1736  006744  012400                        MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
1737  006746  001420                        BEQ     TST11           ;;BR IF END OF TABLE.
1738  006750  004467  006166                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1739  006754  010012          2$:     MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
1740  006756  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1741  006760  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1742  006762  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1743  006764  004767  012026        64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1744  006770  004767  013254                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1745  006774  000004                        .WORD   4               ;ERROR TYPE CODE.
1746  006776                          65$:
1747  006776  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1748  007000  001365                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1749  007002  004767  006712                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
1750  007006  000754                        BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
                                ;;*****************************************************************
                                ;*TEST 11       ROTATE A "0" BIT THROUGH A FIELD OF ONES.
                                ;;*****************************************************************
007010                          TST11:
007010  004567  012244                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007014  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
007016  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007022  004767  007656                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
007026  004467  006110                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007032  000241                  1$:     CLC                     ;CLEAR CARRY BIT IN PSW
007034  004767  007664                  JSR     PC,     ROTATE
007040  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
007044  103402                          BCS     63$             ;BRANCH IF 'C' BIT WAS SET
007046  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007050  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007052  004767  011740          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007056  004767  013166                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007062  000005                          .WORD   5               ;ERROR TYPE CODE.
007064                          64$:
007064  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007066  001361                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007070  004767  006624                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*TEST 12       ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
                                ;;*****************************************************************
007074                          TST12:
007074  004567  012160                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007100  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
007102  005000                          CLR     R0              ;SET CHECK WORD
007104  004767  007574                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
007110  004467  006026                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007114  000261                  1$:     SEC                     ;SET 'C' BIT IN PSW
007116  004767  007602                  JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
007122  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
007126  103002                          BCC     63$             ;BRANCH IF 'C' IS CLEAR
007130  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007132  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007134  004767  011656          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007140  004767  013104                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007144  000005                          .WORD   5               ;ERROR TYPE CODE.
007146                          64$:
007146  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007150  001361                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007152  004767  006542                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                        ;;*****************************************************************
                                        ;*TEST 13       1 XOR 8 TEST PATTERN
                                        ;;*****************************************************************
007156                                  TST13:
007156  004567  012076                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007162  000377                                  .WORD   377             ;MINIMUM BLOCK SIZE OF 128. WORDS
                                                                        ; REQUIRED FOR THIS TEST.
007164  000167  000434                          JMP     TST14           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
007170  012700  177777                          MOV     #-1,    R0      ;SET UP CHECK WORD.
007174  005003                                  CLR     R3              ;SET UP COM DATA REG
007176  004467  005740                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007202  005100                          1$:     COM     R0
007204  005103                                  COM     R3
007206  012704  000010                          MOV     #8.,R4          ;SET 128. WORD COUNTER
007212  010022                          2$:     MOV     R0,(R2)+        ;WRITE 128. WORDS
007214  010322                                  MOV     R3,(R2)+
007216  010022                                  MOV     R0,(R2)+
007220  010322                                  MOV     R3,(R2)+

007222  010022                                  MOV     R0,(R2)+
007224  010322                                  MOV     R3,(R2)+
007226  010022                                  MOV     R0,(R2)+
007230  010322                                  MOV     R3,(R2)+

007232  010022                                  MOV     R0,(R2)+
007234  010322                                  MOV     R3,(R2)+
007236  010022                                  MOV     R0,(R2)+
007240  010322                                  MOV     R3,(R2)+

007242  010022                                  MOV     R0,(R2)+
007244  010322                                  MOV     R3,(R2)+
007246  010022                                  MOV     R0,(R2)+
007250  010322                                  MOV     R3,(R2)+

007252  005304                                  DEC     R4              ;DECREMENT 128. WORD COUNTER
007254  001356                                  BNE     2$
007256  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007260  001350                                  BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007262  004767  006432                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;;*****************************************************************
                                        ;*  CHECK 1 XOR 8 TEST PATTERN WRITTEN ABOVE.
                                        ;;*****************************************************************
007266  005000                                  CLR     R0              ;CLEAR TEST WORD
007270  004467  005646                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007274  012704  000040                  11$:    MOV     #32.,   R4      ;SET 128. WORD COUNTER
007300                                  12$:
007300  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007302  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007304  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007306  004767  011504                  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007312  004767  012732                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007316  000006                                  .WORD   6               ;ERROR TYPE CODE.
007320                                  65$:
007320  005100                                  COM     R0
```

```
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,     $PRNT2  ;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,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1864  007356  004767  012566                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,     $PRNT2  ;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.
```

```
1908  007504  004767  011306          72$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1909  007510  004767  012534                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1910  007514  000006                          .WORD   6               ;ERROR TYPE CODE.
1911  007516                          73$:
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,     $PRNT2  ;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,     $PRNT2  ;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,     $PRNT2  ;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$.
```

```
                                ;;*****************************************************************
                                ;*TEST 14        3 XOR 9 TEST PATTERN.
                                ;;*****************************************************************
007624                          TST14:
007624  004567  011430                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007630  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007632  000167  000312                  JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007636  005000                  .3X9:   CLR     R0              ;SET UP TEST DATA
007640  012703  177777                  MOV     #-1,    R3      ;SET COM DATA REG
007644  004467  005272                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007650  004767  007116          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007654  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007656  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007660  004767  006034                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
                                ;;*****************************************************************
007664  005000                          CLR     R0              ;SET CHECK WORD
007666  004467  005250                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007672  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007676                          12$:
007676  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007700  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007702  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007704  004767  011106          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007710  004767  012334                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007714  000007                          .WORD   7               ;ERROR TYPE CODE.
007716                          65$:
007716  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007720  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007722  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007724  004767  011066          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007730  004767  012314                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007734  000007                          .WORD   7               ;ERROR TYPE CODE.
007736                          67$:
007736  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007740  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007742  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007744  004767  011046          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007750  004767  012[illegible]          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007754  000007                          .WORD   7               ;ERROR TYPE CODE.
007756                          69$:
007756  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007760  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007762  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007764  004767  011026          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007770  004767  012254                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007774  000007                          .WORD   7               ;ERPOR TYPE CODE.
007776                          71$:
007776  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010000  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
010002  001335                          BNE     12$
010004  005100                          COM     R0              ;COMPLEMENT CHECK WORD
```

```
 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,     SPRNT2  ;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,     SPRNT2  ;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,     SPRNT2  ;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  005550            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
2045                                    ;;*****************************************************************
2046                                    ;*TEST 15       COMPLEMENT 3 XOR 9 TEST PATTERN
2047                                    ;;*****************************************************************
2048  010150                            TST15:
2049  010150  004567  011104                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2050  010154  000777                           .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
2051                                                                    ; REQUIRED FOR THIS TEST.
2052  010156  000167  000316                   JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2053                                                                    ; AVAILABLE FOR TEST.
2054  010162  012700  177777                   MOV     #-1,    R0      ;SET UP TEST DATA
2055  010166  005003                           CLR     R3              ;SET COM DATA REG
2056  010170  004467  004746                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2057  010174  004767  006572           1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2058  010200  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2059  010202  001374                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2060  010204  004767  005510                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2061
2062
2063                                    ;;*****************************************************************
2064                                    ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
2065                                    ;;*****************************************************************
2066  010210  012700  177777                   MOV     #-1,    R0      ;SET CHECK WORD
2067  010214  004467  004722                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2068  010220  012704  000100           11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
2069  010224                            12$:
2070  010224  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2071  010226  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2072  010230  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2073  010232  004767  010560           64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2074  010236  004767  012006                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2075  010242  000007                           .WORD   7               ;ERROR TYPE CODE.
2076  010244                            65$:
2077  010244  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2078  010246  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2079  010250  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2080  010252  004767  010540           66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2081  010256  004767  011766                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2082  010262  000007                           .WORD   7               ;ERROR TYPE CODE.
2083  010264                            67$:
2084  010264  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2085  010266  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2086  010270  001405                           BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2087  010272  004767  010520           68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2088  010276  004767  011746                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2089  010302  000007                           .WORD   7               ;ERROR TYPE CODE.
2090  010304                            69$:
2091  010304  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2092  010306  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2093  010310  001405                           BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2094  010312  004767  010500           70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2095  010316  004767  011726                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2096  010322  000007                           .WORD   7               ;ERROR TYPE CODE.
2097  010324                            71$:
2098  010324  005100                           COM     R0              ;COMPLEMENT CHECK WORD
2099  010326  005304                           DEC     R4              ;DECREMENT 256. WORD COUNTER
2100  010330  001335                           BNE     12$
```

```
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$.
2105
2106                                  ;;*****************************************************************
2107                                  ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
2108                                  ;;*****************************************************************
2109  010344  012700  177777                MOV     #-1,    R0      ;SET UP CHECK WORD.
2110  010350  004467  004566                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2111  010354  012704  000100        21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
2112  010360  012703  000004        22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
2113  010364                        23$:
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,     SPRNT2  ;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,     SPRNT2  ;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,     SPRNT2  ;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$.
```

```
                                ;;*****************************************************************
                                ;*TEST 16       MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
                                ;;*****************************************************************
010500                          TST16:
010500  004567  010554                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010504  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
010506  000167  000610                  JMP     TST17           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
010512  012700  000401                  MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN
010516  012703  177777                  MOV     #-1,    R3      ;SET COM DATA REG
010522  004467  004414                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010526  004767  006240          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
010532  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010534  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
010536  004767  005156                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
010542  012700  000401                  MOV     #401,   R0      ;RESET PARITY "ALL ZEROS" PATTERN.
010546  012703  177777                  MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
010552  004467  004364                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010556  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
010562                          12$:
010562  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010564  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010566  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010570  004767  010222          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010574  004767  011450                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010600  000007                          .WORD   7               ;ERROR TYPE CODE.
010602                          65$:
010602  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010604  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010606  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010610  004767  010202          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010614  004767  011430                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010620  000007                          .WORD   7               ;ERROR TYPE CODE.
010622                          67$:
010622  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010624  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010626  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010630  004767  010162          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010634  004767  011410                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010640  000007                          .WORD   7               ;ERROR TYPE CODE.
010642                          69$:
010642  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010644  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010646  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010650  004767  010142          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010654  004767  011370                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010660  000007                          .WORD   7               ;ERROR TYPE CODE.
010662                          71$:
010662  010046                          MOV     R0,     -(SP)   ;SAVE R0
010664  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
010666  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
```

```
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$.
2212
2213                                  ;;*****************************************************************
2214                                  ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
2215                                  ;;*****************************************************************
2216  010712  012700  000401                MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN.
2217  010716  012703  177777                MOV     #-1,    R3      ;SET UP ALL ONES PATTERN.
2218  010722  004467  004214                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2219  010726  012704  000100        21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
2220  010732                        22$:
2221  010732  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2222  010734  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2223  010736  001405                        BEQ     73$             ;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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2226  010750  000007                        .WORD   7               ;ERROR TYPE CODE.
2227  010752                        73$:
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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2235  010774  000007                        .WORD   7               ;ERROR TYPE CODE.
2236  010776                        75$:
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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2244  011020  000007                        .WORD   7               ;ERROR TYPE CODE.
2245  011022                        77$:
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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2251  011040  000007                        .WORD   7               ;ERROR TYPE CODE.
2252  011042                        79$:
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)
```

```
2260  011064  000007                        .WORD   7               ;ERROR TYPE CODE.
2261  011066                        81$:
2262  011066  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2263  011070  005142                        COM     -(R2)           ;RESTORE DATA
2264  011072  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2265  011074  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2266  011076  001405                        BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2267  011100  004767  007712        82$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2268  011104  004767  011140                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2269  011110  000007                        .WORD   7               ;ERROR TYPE CODE.
2270  011112                        83$:
2271  011112  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2272  011114  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2273  011116  001405                        BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2274  011120  004767  007672        84$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2275  011124  004767  011120                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2276  011130  000007                        .WORD   7               ;ERROR TYPE CODE.
2277  011132                        85$:
2278  011132  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2279  011134  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2280  011136  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2281  011140  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2282  011142  001405                        BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2283  011144  004767  007646        86$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2284  011150  004767  011074                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2285  011154  000007                        .WORD   7               ;ERROR TYPE CODE.
2286  011156                        87$:
2287  011156  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2288  011160  005142                        COM     -(R2)           ;RESTORE DATA
2289  011162  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2290  011164  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2291  011166  001405                        BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2292  011170  004767  007622        88$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2293  011174  004767  011050                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2294  011200  000007                        .WORD   7               ;ERROR TYPE CODE.
2295  011202                        89$:
2296  011202  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2297  011204  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2298  011206  001405                        BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2299  011210  004767  007602        90$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2300  011214  004767  011030                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2301  011220  000007                        .WORD   7               ;ERROR TYPE CODE.
2302  011222                        91$:
2303  011222  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2304  011224  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2305  011226  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2306  011230  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2307  011232  001405                        BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2308  011234  004767  007556        92$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2309  011240  004767  011004                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2310  011244  000007                        .WORD   7               ;ERROR TYPE CODE.
2311  011246                        93$:
2312  011246  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2313  011250  005142                        COM     -(R2)           ;RESTORE DATA
2314  011252  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2315  011254  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
2316  011256  001405                        BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2317  011260  004767  007532        94$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2318  011264  004767  010760                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2319  011270  000007                        .WORD   7               ;ERROR TYPE CODE.
2320  011272                        95$:
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                TST17:
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,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2362  011416  004767  010626                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2363  011422  000007                        .WORD   7               ;ERROR TYPE CODE.
2364  011424                        65$:
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,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2369  011436  004767  010606                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2370  011442  000007                        .WORD   7               ;ERROR TYPE CODE.
2371  011444                        67$:
```

```
2372  011444  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2373  011446  020001                  CMP     R0,     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,     SPRNT2  ;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     R0,     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,     SPRNT2  ;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     R0,     -(SP)   ;SAVE R0
2387  011506  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
2388  011510  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2389  011512  005304                  DEC     R4              ;COUNT 256. WORDS
2390  011514  001333                  BNE     12$             ;BRANCH IF MORE
2391  011516  010046                  MOV     R0,     -(SP)   ;SAVE R0
2392  011520  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
2393  011522  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 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$.
2397
2398                          ;;**************************************************************
2399                          ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
2400                          ;;**************************************************************
2401  011534  012700  177777          MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
2402  011540  012703  000401          MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
2403  011544  004467  003372          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2404  011550  012704  000100  21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
2405  011554                  22$:
2406  011554  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2407  011556  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2408  011560  001405                  BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2409  011562  004767  007230  72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2410  011566  004767  010456          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2411  011572  000007                  .WORD   7               ;ERROR TYPE CODE.
2412  011574                  73$:
2413  011574  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2414  011576  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
2415  011600  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2416  011602  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2417  011604  001405                  BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2418  011606  004767  007204  74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2419  011612  004767  010432          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2420  011616  000007                  .WORD   7               ;ERROR TYPE CODE.
2421  011620                  75$:
2422  011620  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2423  011622  005142                  COM     -(R2)           ;RESTORE DATA
2424  011624  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2425  011626  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2426  011630  001405                  BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2427  011632  004767  007160  76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
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     R0,     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     R0              ;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     R0,     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     R0              ;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     R0,     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     R0,     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     R0              ;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     R0,     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     R0              ;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     R0,     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     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2483  012030  001405                      BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

```
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     R0              ;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     R0,     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     R0              ;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     R0,     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     R0,     -(SP)   ;SAVE R0
2507  012116  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
2508  012120  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2509  012122  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
2510  012124  001213                          BNE     22$             ;BRANCH IF MORE.
2511  012126  010046                          MOV     R0,     -(SP)   ;SAVE R0
2512  012130  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
2513  012132  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 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$.
```

```
2517                                    ;;*****************************************************************
2518                                    ;*TEST 20       8 XOR 13 TEST PATTERN
2519                                    ;;*****************************************************************
2520  012144                            TST20:
2521  012144  004567  007110                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2522  012150  017777                           .WORD   17777           ;MINIMUM BLOCK SIZE OF 4096. WORDS
2523                                                                    ; REQUIRED FOR THIS TEST.
2524  012152  000167  000320                   JMP     TST21           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2525                                                                    ; AVAILABLE FOR TEST.
2526  012156  005000                    .8X13:  CLR     R0
2527  012160  004467  002756                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2528  012164  012704  000040            1$:     MOV     #32.,   R4      ;SET UP 4K LOOP COUNTER
2529  012170  005100                           COM     R0
2530  012172  012703  000200            2$:     MOV     #128.,  R3      ;EACH SMALL LOOP WRITES 129 WORDS
2531  012176  005100                           COM     R0
2532  012200  010022                    3$:     MOV     R0,     (R2)+   ;WRITE INTO MEMORY ADDRESSES
2533  012202  005303                           DEC     R3              ;DECREMENT WORD COUNT
2534  012204  001375                           BNE     3$
2535  012206  005304                           DEC     R4              ;DECREMENT 128. WORD COUNT
2536  012210  001370                           BNE     2$
2537  012212  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2538  012214  001363                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2539  012216  004767  003476                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2540
2541                                    ;;*****************************************************************
2542                                    ;*  CHECK 8 XOR 13 TEST PATTERN WRITTEN ABOVE
2543                                    ;;*****************************************************************
2544  012222  005000                           CLR     R0              ;SET UP CHECK WORD.
2545  012224  004467  002712                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2546  012230  012704  000040            11$:    MOV     #32.,   R4      ;SET 4K WORD COUNTER
2547  012234  005100                           COM     R0              ;COMPLEMENT TEST WORD
2548  012236  012703  000100            12$:    MOV     #64.,   R3      ;SET 128 WORD COUNTER
2549  012242  005100                           COM     R0              ;COMPLEMENT TEST WORD
2550  012244                            13$:
2551  012244  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2552  012246  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2553  012250  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2554  012252  004767  006540            64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2555  012256  004767  007766                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2556  012262  000010                           .WORD   10              ;ERROR TYPE CODE.
2557  012264                            65$:
2558  012264  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2559  012266  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2560  012270  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2561  012272  004767  006520            66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2562  012276  004767  007746                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2563  012302  000010                           .WORD   10              ;ERROR TYPE CODE.
2564  012304                            67$:
2565  012304  005303                           DEC     R3              ;DECREMENT 128 WORD COUNTER
2566  012306  001356                           BNE     13$             ;BR IF MORE.
2567  012310  005304                           DEC     R4              ;DECREMENT 4096. WORD COUNTER
2568  012312  001351                           BNE     12$             ;BR IF MORE.
2569  012314  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2570  012316  001344                           BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
2571  012320  004767  003374                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2572
```

```
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,     SPRNT2  ;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,     SPRNT2  ;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$.
```

```
                                ;;*****************************************************************
                                ;*TEST 21       WORSE CASE NOISE PARITY BYTE TESTING
                                ;*  CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
                                ;*      1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
                                ;*      2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
                                ;*      3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
                                ;*      4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
                                ;;*****************************************************************
012476                          TST21:
012476  004567  006556                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012502  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
012504  005767  167604        WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
012510  001404                        BEQ     1$              ;BR IF NO PARITY MEMORY.
012512  032777  000100  166420        BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
012520  001402                        BEQ     2$              ;BR IF NOT SET.
012522  000167  000614        1$:     JMP     TST22           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
012526  005000                2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
012530  004767  004150                JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
012534  004467  002402                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012540  036767  167000  166772  WWPBYT: BIT     BITPT,  PMEMAP  ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
012546  001010                        BNE     2$              ;BR IF PARITY MEM.
012550  036767  166772  166764        BIT     BITPT+2,PMEMAP+2 ;....HI 64K.
012556  001004                        BNE     2$              ;BR IF PARITY MEM.
012560  050502                        BIS     R5,     R2      ;POINT TO END OF BLOCK.
012562  005202                        INC     R2              ;FIRST ADR OF NEXT BLOCK.
012564  000167  000532                JMP     WWPB5           ;BR TO FIND NEXT BLOCK.
012570  004767  005436        2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
012574  004767  005466                JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
012600  020227  000114        WWPB1:  CMP     R2,     #114    ;CKECK IF POINTING TO PARITY ERROR VECTOR.
012604  001004                        BNE     3$              ;BR IF NOT AT VECTOR.
012606  062702  000004                ADD     #4,     R2      ;SKIP PARITY VECTOR.
012612  000167  000504                JMP     WWPB5           ;CHECK FOR BLOCK END.
012616  111201                3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
012620  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012622  004767  006114        64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
012626  004767  007416                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012632  000011                        .WORD   11              ;ERROR TYPE CODE.
012634                          65$:
012634  105067  166720                CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
012640  112700  000252                MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
012644  110012                WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
012646  016703  166746                MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
012652  052733  000005        10$:    BIS     #WWP+AE,@(R3)+  ;SET WRITE WRONG PARITY.
012656  005713                        TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
012660  001374                        BNE     10$             ;BR IF MORE REGS IN TABLE.
                                ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
012662  110012                        MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
012664  016703  166730                MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
012670  042733  000004        11$:    BIC     #WWP,   @(R3)+  ;CLEAR WRITE WRONG PARITY.
012674  005713                        TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
012676  001374                        BNE     11$             ;BR IF MORE PARITY REGISTERS.
012700  016737  166724  000114        MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
                                ;*  DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
012706  105412                        NEGB    (R2)            ;DATIP (DATOB AND COM PARITY BIT.)
                                ;*  SHOULD HAVE TRAPPED TO PBTRP.
012710  016737  166720  000114        MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
```

```
2676  012716  004767  006050          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
2677  012722  004767  007322                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2678  012726  000012                          .WORD   12              ;ERROR TYPE CODE.
2679  012730  000562                          BR      WWPB4           ;SKIP TRAP SERVICE.
2680
2681                                  ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
2682  012732  016737  166676  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2683  012740  022626                          CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
2684  012742  016703  166650                  MOV     .MPR0,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
2685  012746  032713  000001          21$:    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$:    ADD     #10,    R3      ;MOVE POINTER TO NEXT REG.
2690  012766  020367  166626                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
2691  012772  103765                          BLO     21$             ;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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2702  013040  000014                          .WORD   14              ;ERROR TYPE CODE.
2703  013042                          24$:
2704  013042  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2705  013044  010200                          MOV     R2,     R0      ;GET THE ADDRESS POINTER.
2706  013046  042700  003777                  BIC     #3777,  R0      ;CLEAR LOW ADDRESS BITS.
2707  013052  000300                          SWAB    R0              ;SHIFT 6 PLACES RIGHT.
2708  013054  006300                          ASL     R0
2709  013056  006300                          ASL     R0
2710  013060  005767  165522                  TST     MMAVA           ;CHECK FOR MEM MGMT.
2711  013064  001404                          BEQ     25$             ;BR IF NO MEM MGMT.
2712  013066  042700  177600                  BIC     #177600,R0      ;CLEAR BANK BITS
2713  013072  063700  172344                  ADD     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
2714  013076  052700  100001          25$:    BIS     #BIT15+BIT0,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
2715  013102  016367  000006  166406          MOV     6(R3),  RESRVD  ;GET APPROPIATE MASK.
2716  013110  046700  166402                  BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2717  013114  046701  166376                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2718                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2719  013120  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2720  013122  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2721  013124  004767  005636          66$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2722  013130  004767  007114                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2723  013134  000015                          .WORD   15              ;ERROR TYPE CODE.
2724  013136                          67$:
2725  013136  005073  000000                  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     @(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,     SPRNTP  ;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     @(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,     SPRNTP  ;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,     @(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,     SPRNTO  ;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 WWPBYT
2775  013336  004767  004624          JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
2776                                    ;;*****************************************************************
2777                                    ;*TEST 22       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
2778                                    ;;*****************************************************************
2779  013342                            TST22:
2780  013342  004567  005712                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2781  013346  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2782  013350  010703                    RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
2783  013352  042703  007777                   BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
2784  013356  004467  001560                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2785  013362  010246                    1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
2786  013364  010346                            MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
2787  013366  012322                    2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
2788  013370  032703  007777                   BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
2789  013374  001002                           BNE     3$              ;BRANCH IF MORE
2790  013376  162703  010000                   SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA"
2791  013402  030502                    3$:     BIT     R5,     R2      ;CHECK FOR END OF BLOCK
2792  013404  001370                           BNE     2$              ;BRANCH IF MORE.
2793  013406  012603                           MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
2794  013410  012602                           MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
2795  013412  012300                    4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
2796  013414  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2797  013416  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2798  013420  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2799  013422  004767  005370            64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2800  013426  004767  006616                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2801  013432  000020                           .WORD   20              ;ERROR TYPE CODE.
2802  013434                            65$:
2803  013434  032703  007777                   BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
2804  013440  001002                           BNE     5$              ;BR IF MORE.
2805  013442  162703  010000                   SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
2806  013446                            5$:
2807  013446  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2808  013450  001360                           BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
2809  013452  004767  002242                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND _OOP TO 1$.
```

```
2810                                    .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
2811                                    ;;*****************************************************************
2812                                    ;*TEST 23       EXECUTE DATI, DATO THRU MEMORY.
2813                                    ;*  EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
2814                                    ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
2815                                    ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2816                                    ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
2817                                    ;*
2818                                    ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
2819                                    ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
2820                                    ;*
2821                                    ;*   1ST PASS /  40000           010412          000205
2822                                    ;*  THRU TEST /  40002           000205          000205
2823                                    ;*
2824                                    ;*   2ND PASS /  40002           010412          000205
2825                                    ;*  THRU TEST /  40004           000205          000205
2826                                    ;*
2827                                    ;*      ETC., ETC., ETC.
2828                                    ;*
2829                                    ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2830                                    ;*  R1 = DATA READ FROM MEMORY (WAS).
2831                                    ;*  R2 = ADDRESS OF IUT/DATA.
2832                                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
2833                                    ;*  R4 = RTS R5 (CODE 205).
2834                                    ;*  R5 = BLOCK BOUNDRY BIT MASK.
2835                                    ;*****************************************************************
2836  013456                            TST23:
2837  013456  004567  005576                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2838  013462  000003                            .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
2839                                                                    ;  REQUIRED FOR THIS TEST.
2840  013464  000167  000056                   JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2841                                                                    ;  AVAILABLE FOR TEST.
2842  013470  012703  010412           DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
2843  013474  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
2844  013500  010400                            MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
2845  013502  004467  001434                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2846  013506  010322                    1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2847  013510  010412                    2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2848  013512  004542                            JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2849  013514  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2850  013516  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2851  013520  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2852  013522  004767  005264           64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
2853  013526  004767  006516                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2854  013532  000021                            .WORD   21              ;ERROR TYPE CODE.
2855  013534                            65$:
2856  013534  010322                            MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2857  013536  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2858  013540  001363                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2859  013542  004767  002152                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;;*****************************************************************
                        ;*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 BOUNDRY BIT MASK.
                        ;*****************************************************************
013546                  TST24:
013546  004567  005506          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013552  000003          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                ; REQUIRED FOR THIS TEST.
013554  000167  000060          JMP     TST25           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                ; AVAILABLE FOR TEST.
013560  012703  110412  DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
013564  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013570  012700  110605          MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
013574  004467  001342          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013600  010322  1$:    MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013602  010412  2$:    MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013604  004542          JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013606  012201          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013610  020001          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013612  001405          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013614  004767  005172  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013620  004767  006424          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013624  000021          .WORD   21              ;ERROR TYPE CODE.
013626                  65$:
013626  010322          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013630  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013632  001363          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013634  004767  002060          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2909                                    ;;*****************************************************************
2910                                    ;*TEST 25       EXECUTE DATI, DATOB (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 BOUNDRY BIT MASK.
2933                                    ;*****************************************************************
2934  013640                            TST25:
2935  013640  004567  005414                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2936  013644  000003                           .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
2937                                                                    ; REQUIRED FOR THIS TEST.
2938  013646  000167  000064                   JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2939                                                                    ; AVAILABLE FOR TEST.
2940  013652  012703  110342           DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
2941  013656  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
2942  013662  012700  161342                   MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
2943  013666  004467  001250                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2944  013672  010322                    1$:    MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2945  013674  010412                    2$:    MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2946  013676  004562  177776                   JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
2947  013702  005302                           DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
2948  013704  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2949  013706  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2950  013710  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2951  013712  004767  005074           64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
2952  013716  004767  006326                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2953  013722  000021                           .WORD   21              ;ERROR TYPE CODE.
2954  013724                            65$:
2955  013724  010322                           MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2956  013726  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2957  013730  001361                           BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2958  013732  004767  001762                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;;*****************************************************************
                        ;*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       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 BOUNDRY BIT MASK.
                        ;*****************************************************************
013736                  TST26:
013736  004567  005316          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013742  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
013744  000167  000060          JMP     TST27           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
013750  012703  005412  DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
013754  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013760  012700  172366          MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
013764  004467  001152          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013770  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013772  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013774  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013776  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
014000  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
014002  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
014004  004767  005002  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
014010  004767  006234          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
014014  000021                  .WORD   21              ;ERROR TYPE CODE.
014016                  65$:
014016  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
014020  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
014022  001363                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
014024  004767  001670          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*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        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 BOUNDRY BIT MASK.
                                ;;*****************************************************************
3033  014030                    TST27:
3034  014030  004567  005224           JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3035  014034  000003                   .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
3036                                                            ; REQUIRED FOR THIS TEST.
3037  014036  000167  000060           JMP     TST30           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
3038                                                            ; AVAILABLE FOR TEST.
3039  014042  012703  142242   DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
3040  014046  012704  000205           MOV     #205,   R4      ;GET 'RTS R5'
3041  014052  012700  142000           MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
3042  014056  004467  001060           JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3043  014062  010322           1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
3044  014064  010412           2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
3045  014066  004542                   JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
3046  014070  012201                   MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
3047  014072  020001                   CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3048  014074  001405                   BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3049  014076  004767  004710   64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
3050  014102  004767  006142           JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3051  014106  000021                   .WORD   21              ;ERROR TYPE CODE.
3052  014110                   65$:
3053  014110  010322                   MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
3054  014112  030502                   BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3055  014114  001363                   BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
3056  014116  004767  001576           JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
3057                                    ;;*****************************************************************
3058                                    ;*TEST 30       EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
3059                                    ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
3060                                    ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
3061                                    ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
3062                                    ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
3063                                    ;*
3064                                    ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
3065                                    ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
3066                                    ;*
3067                                    ;*   1ST PASS /  40000           152212          157212
3068                                    ;*  THRU TEST /  40002           000205          000205
3069                                    ;*
3070                                    ;*   2ND PASS /  40002           152212          157212
3071                                    ;*  THRU TEST /  40004           000205          000205
3072                                    ;*
3073                                    ;*      ETC., ETC., ETC.
3074                                    ;*
3075                                    ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
3076                                    ;*  R1 = DATA READ FROM MEMORY (WAS).
3077                                    ;*  R2 = ADDRESS OF IUT/DATA.
3078                                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
3079                                    ;*  R4 = RTS R5 (CODE 205).
3080                                    ;*  R5 = BLOCK BOUNDRY BIT MASK.
3081                                    ;*****************************************************************
3082  014122                            TST30:
3083  014122  004567  005132                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3084  014126  000003                            .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
3085                                                                    ; REQUIRED FOR THIS TEST.
3086  014130  000167  000062                   JMP     TST31           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
3087                                                                    ; AVAILABLE FOR TEST.
3088  014134  012703  152212           DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
3089  014140  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
3090  014144  012700  157212                   MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
3091  014150  004467  000766                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3092  014154  010322                    1$:    MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
3093  014156  010412                    2$:    MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
3094  014160  004542                            JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
3095  014162  005302                            DEC     R2              ;RESET R2 TO POINT TO IUT.
3096  014164  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
3097  014166  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3098  014170  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3099  014172  004767  004614           64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
3100  014176  004767  006046                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3101  014202  000021                            .WORD   21              ;ERROR TYPE CODE.
3102  014204                            65$:
3103  014204  010322                            MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
3104  014206  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3105  014210  001362                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
3106  014212  004767  001502                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
3107                                    ;;*****************************************************************
3108                                    ;*TEST 31       "BRANCH GOBBLE" TEST
3109                                    ;* THIS TEST LOADS THE ROUTINE FOUND AT "BRGOB" BELOW INTO THE BEGINNING
3110                                    ;*  OF MEMORY UNDER TEST.  IF THERE IS ROOM, IT LOADS THE ROUTINE AGAIN
3111                                    ;*  STARTING AT THE LAST INSTRUCTION OF THE FIRST COPY, THUS OVERLAYING
3112                                    ;*  THE EXIT INSTRUCTION (JMP (R1)) OF THE FIRST COPY.  THIS CONTINUES
3113                                    ;*  UNTIL THE CURRENT BANK(S) ARE FULL.  THE CODE IS THEN EXECUTED.
3114                                    ;*  THIS ENTIRE PROCESS IS THEN REPEATED STARTING AT THE SECOND ADDRESS
3115                                    ;*  OF THE BANK UNDER TEST, THEN THE THIRD, ETC., UNTIL EACH INSTRUCTION
3116                                    ;*  OF THE ROUTINE HAS BEEN LOADED INTO AND EXECUTED OUT OF EVERY LOCATION.
3117                                    ;*****************************************************************
3118  014215                            TST31:
3119  014216  004567  005036                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3120  014222  000077                            .WORD   77              ;MINIMUM BLOCK SIZE OF 32. WORDS
3121                                                                    ; REQUIRED FOR THIS TEST.
3122  014224  000167  000330                    JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
3123                                                                    ; AVAILABLE FOR TEST.
3124  014230                            BRGOBL:
3125  014230  004467  000706                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3126  014234  012746  000024            BGTOP:  MOV     #BGLEN-1, -(SP) ;SAVE ROUTINE LENGTH
3127  014240  010246                    BGLOOP: MOV     R2,     -(SP)   ;SAVE 1ST BLOCK ADDRESS.
3128  014242  016700  165354            5$:     MOV     .BRGOB, R0      ;GET FIRST LOC OF "BRANCH GOBBLE" ROUTINE.
3129  014246  012701  000025                    MOV     #BGLEN, R1      ;SET UP LENGTH COUNTER
3130  014252  012022                    6$:     MOV     (R0)+,  (R2)+   ;MOV ROUTINE INTO MEMORY
3131  014254  030502                            BIT     R5,     R2      ;CHECK FOR END OF BLOCK.
3132  014256  001404                            BEQ     7$              ;BRANCH WHEN DONE.
3133  014260  00053C                            DEC     R1              ;COUNT WORDS IN ROUTINE.
3134  014262  001373                            BNE     6$              ;BR IF MORE.
3135  014264  005742                            TST     -(R2)           ;BACK UP ONE TO OVERLAY RETURN.
3136  014266  000765                            BR      5$              ;BR TO START ANOTHER ROUTINE.
3137  014270  005301                    7$:     DEC     R1              ;CHECK FOR EXACT FINISH.
3138  014272  001407                            BEQ     9$              ;BR IF ALL OK.
3139  014274  162701  000025                    SUB     #BGLEN, R1      ;SET COUNTER FOR BACKING UP.
3140  014300  005042                    8$:     CLR     -(R2)           ;PUT HALTS IN EXTRA MEMORY.
3141  014302  005201                            INC     R1              ;COUNT NUMBER OF WORDS TO BACK UP..
3142  014304  001375                            BNE     8$              ;BR IF MORE.
3143  014306  016722  000064                    MOV     BGEND,  (R2)+   ;INSERT RETURN INSTRUCTION.
3144  014312  011602                    9$:     MOV     (SP),   R2      ;RESET POINTER TO BEGINNING OF BLOCK.
3145  014314  016700  165304                    MOV     .BGERR, R0      ;POINT TO ERROR ROUTINE.
3146  014320  016701  165302                    MOV     .BGEXI, R1      ;POINT TO EXIT ROUTINE.
3147  014324  000112                            JMP     (R2)            ;GO TO BRANCH GOBBLE ROUTINES IN MEMORY.
3148
3149                                    ;*  "BRANCH GOBBLE" ROUTINE WHICH IS RELOCATED INTO MEMORY UNDER TEST
3150                                    ;*    AND THEN EXDIEICUTED THERE.
3151  014326  010703                    BRGOB:  MOV     PC,     R3      ;GET CURRENT ROUTINE LOCATION.
3152  014330  062703  000024            BG1:    ADD     #BG3-BG1, R3    ;POINT TO DATA WORD "BG3"
3153  014334  010304                            MOV     R3,     R4      ;COPY
3154  014336  005204                            INC     R4              ;POINT TO DATA BYTE "BG4"
3155  014340  005013                            CLR     (R3)            ;CLR DATA WORD "BG3" & "BG4"
3156  014342  000261                    BG2:    SEC                     ;SET CARRY...TO BE ADDED TO "BG3"
3157  014344  105513                            ADCB    (R3)            ;ADD THE CARRY... AS IF INC INST
3158  014346  100403                            BMI     BG5             ;BRANCH WHEN BIT 7 IS SET
3159  014350  105214                            INCB    (R4)
3160  014352  000773                            BR      BG2             ;BR BACK AND CONTINUE TO COUNT.
3161  014354     000                    BG3:    .BYTE   0               ;DATA BYTE POINTED TO BY R3
3162  014355     000                    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     R5,     (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     R5,     (R0)    ;ERROR!!!       CC NOT = 2
3171  014376  000111                  BGEND:  JMP     (R1)            ;EXIT CODE...ONLY AT END OF BLOCK.
3172          000025                  BGLEN=  (.-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     R5,     $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     R5,     $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     R5              ;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.
```

```
3206                                        .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
3207  014552                                DONE:
3208  014552  004567  004502                      JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
3209  014556  000000                              .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3210  014560  005067  164404          TST32:      CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
3211  014564  105067  164312                      CLRB    $TSTNM          ;RESET TEST NUMBER.
3212  014570  036767  164006  164736  1$:         BIT     PRGMAP, SAVTST  ;CHECK IF PROGRAM IS IN TEST AREA.
3213  014576  001004                              BNE     2$              ;BR IF IT PROG IN MEM TO BE TESTED.
3214  014600  036767  164000  164730              BIT     PRGMAP+2,SAVTST+2 ;CHECK HI 64K
3215  014606  001426                              BEQ     $EOP            ;BR IF PROG NOT IN MEM TO BE TESTED.
3216  014610  032777  000200  164322  2$:         BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
3217  014616  001022                              BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
3218  014620  022767  000003  163754              CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IN FIRST 8K.
3219  014626  001007                              BNE     4$              ;BR IF NOT IN FIRST 8K.
3220  014630  005737  000042                      TST     @#42            ;CHECK FOR A MONITOR.
3221  014634  001006                              BNE     5$              ;BR IF A MONITOR.
3222  014636  004767  002332                      JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
3223  014642  000167  177140          3$:         JMP     START1          ;LOOP BACK AND RUN ALL TESTS AGAIN.
3224
3225  014646  004767  002724          4$:         JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
3226  014652  004767  003126          5$:         JSR     PC,     RESLDR  ;RESTORE LOADERS.
3227  014656  004567  006714                      JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3228  014662  001201                              .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

```
3229                                    ;;*****************************************************************
3230
3231                                    .SBTTL  END OF PASS ROUTINE
3232
3233                                    ;*INCREMENT THE PASS NUMBER ($PASS)
3234                                    ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
3235                                    ;*IF THERES A MONITOR GO TO IT
3236                                    ;*IF THERE ISN'T JUMP TO START1
3237
3238  014664                            $EOP:
3239  014664  000240                           NOP
3240  014666  005067  164276                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
3241  014672  005267  164314                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
3242  014676  042767  100000  164306          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
3243  014704  005327                          DEC     (PC)+           ;;LOOP?
3244  014706  000001                    $EOPCT: .WORD   1
3245  014710  003027                          BGT     $DOAGN          ;;YES
3246  014712  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
3247  014714  000001                    $ENDCT: .WORD   1
3248  014716  014706                          $EOPCT
3249  014720  004567  006652                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3250  014724  014774                          .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
3251  014726  016746  164260                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
3252                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
3253                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3254  014732  013746  177776                   MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
3255  014736  004767  007554                   JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
3256  014742  004567  006630                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3257  014746  015011                          .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
3258  014750                            $GET42:
3259
3260  014750  016700  163066                   MOV     42,     R0              ;;GET MONITOR ADDRESS
3261  014754  001405                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
3262  014756  000005                          RESET                   ;;CLEAR THE WORLD
3263  014760  004710                    $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
3264  014762  000240                          NOP                     ;;SAVE ROOM
3265  014764  000240                          NOP                     ;;FOR
3266  014766  000240                          NOP                     ;;ACT11
3267  014770                            $DOAGN:
3268  014770  000167  171012                   JMP     START1          ;;RETURN*****
3269  014774  005015  047105  020104    $ENDMG: .ASCIZ  <15><12>/END PASS #/
3270  015002  040520  051523  021440
3271  015010     000
3272  015011     377     377     000    $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
3273                                    .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
3274                                    .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
3275                                    ;;*****************************************************************
3276                                    ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
3277                                    ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
3278                                    ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
3279                                    ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
3280                                    ;*  PARS 4, 5, AND 6 ARE UNUSED.
3281                                    ;;*****************************************************************
3282  015014                            MMINIT:
3283  015014  012737  077406  172300           MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
3284  015022  012737  077406  172302           MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
```

```
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
3290  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,     @#SR0   ;ENABLE MEMORY MANAGEMENT
3300  015140  000207                          RTS     PC              ;RETURN
3301
3302
3303                                  ;;*****************************************************************
3304                                  ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
3305                                  ;;*****************************************************************
3306  015142  012767  000001  164374  INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
3307  015150  005067  164372                  CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
3308  015154  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
3309  015156  016705  164424                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
3310  015162  005767  163420                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
3311  015166  001514                          BEQ     10$             ;BRANCH IF NO MEM MGMT
3312  015170  005037  172344                  CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
3313  015174  012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
3314  015200  036767  164340  164322  1$:     BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
3315  015206  001015                          BNE     2$              ;BRANCH IF MATCH
3316  015210  036767  164332  164314          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3317  015216  001011                          BNE     2$              ;BRANCH IF MATCH
3318  015220  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3319  015226  006367  164312                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
3320  015232  006167  164310                  ROL     BITPT+2         ;...HI POINTER.
3321  015236  100360                          BPL     1$              ;BR IF MORE.
3322  015240  000000                          HALT                    ;FATAL ERROR!!!  NO 4K BANK FOUND?
3323  015242  036767  164276  164332  2$:     BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
3324  015250  001004                          BNE     3$              ;BR IF LAST BANK.
3325  015252  036767  164270  164324          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
3326  015260  001405                          BEQ     4$              ;BR IF NOT LAST BANK.
3327  015262  016705  164312          3$:     MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
3328  015266  042767  020000  164302          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
3329  015274  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3330  015302  016767  164236  164240          MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
3331  015310  016767  164232  164234          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
3332  015316  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
3333  015322  001505                          BEQ     21$             ;BRANCH IF NOT 8K.
3334  015324  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
3335  015332  006367  164212                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
3336  015336  006167  164210                  ROL     TMPPT+2         ;...HI POINTER.
3337  015342  100473                          BMI     20$             ;BR IF NO MORE.
3338  015344  036767  164200  164156          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3339  015352  001004                          BNE     6$              ;BRANCH IF A MATCH.
3340  015354  036767  164172  164150          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
```

```
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          7$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
3347  015410  052767  020000  164160          BIS     #20000, TMPLAD  ;MAKE SURE LAST ADDRESS IS IN BANK 3.
3348  015416  000447                          BR      21$             ;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          MOVB    #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$:    MOV     TMPFAD, R2      ;RESET ADDRESS POINTER TO FIRST ADR.
3378  015576  000445                          BR      INITEX
3379
3380  015600  016705  164002          INITDN: MOV     BLKMSK, R5      ;RESET R5 TO CURRENT BLOCK MASK.
3381  015604  005002                          CLR     R2              ;INIT ADDRESS POINTER.
3382  015606  005767  162774                  TST     MMAVA           ;CHECK FOR MEM MGMT
3383  015612  001411                          BEQ     31$             ;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$,   MMORE   ;SET "MMDOWN" EXIT ADDRESS.
3391  015652  066767  162722  163674          ADD     RELOCF, MMORE   ;ADD OFFSET
3392  015660  004767  000524                  JSR     PC,     MMDOWN  ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3393  015664  000000                          HALT                    ;FATAL ERROR!!!  NO MEM INDICATED IN MEM MAP ABOVE 8K!
3394  015666  036767  163652  163706  33$:    BIT     BITPT,  LADMAP  ;CHECK FOR NON BOUNDRY LAST ADDR.
3395  015674  001004                          BNE     34$             ;BR IF LAST BANK FLAG FOUND.
3396  015676  036767  163644  163700          BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
```

```
3397  015704  001402                          BEQ     INITEX          ;BR IF NO LAD FLG FOUND.
3398  015706  016702  163662          34$:    MOV     LSTADR, R2      ;SET UP R2.
3399  015712  010467  163636          INITEX: MOV     R4,     MMORE   ;PUT RETURN PC INTO "MMORE"
3400  015716  000204                          RTS     R4              ;RETURN
3401
3402                                  ;;**************************************************************
3403                                  ;*  COMMON UPWARDS ADDRESSING ROUTINE
3404                                  ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3405                                  ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
3406                                  ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3407                                  ;;**************************************************************
3408  015720  036767  163624  163654  MMUP:   BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3409  015726  001122                          BNE     10$             ;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     MMAVA           ;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$:     ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3417  015764  006367  163554                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
3418  015770  006167  163552                  ROL     BITPT+2         ;...HI POINTER.
3419  015774  100577                          BMI     32$             ;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          3$:     MOV     LADMSK, R5      ;RESET MASK.
3429  016042  042767  020000  163526          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3430  016050  016767  163470  163472  4$:     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     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3435  016100  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
3436  016106  006367  163436                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
3437  016112  006167  163434                  ROL     TMPPT+2         ;...HI POINTER.
3438  016116  100513                          BMI     30$             ;BR IF NO MORE.
3439  016120  036767  163424  163402  6$:     BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3440  016126  001003                          BNE     7$              ;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          8$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADDRESS.
3448  016164  052767  020000  163404          BIS     #20000, TMPLAD  ;SET VIRTUAL ADR TO BANK 3.
3449  016172  000467                          BR      31$
3450
3451  016174  026702  163376          10$:    CMP     TMPLAD, R2      ;CHECK IF LAST ADR REACHED.
3452  016200  001064                          BNE     31$             ;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                  CLRB    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          MOVB    #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 TOFIND 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.
```

```
 3509  016460  006067  163060                 ROR     BITPT             ;...LO MAP.
 3510  016464  103440                         BCS     10$               ;BR IF NO MORE.
 3511  016466  036767  163052  163034         BIT     BITPT,  TSTMAP    ;CHECK FOR BANK EXISTING
 3512  016474  001004                         BNE     4$                ;BR IF BANK TO BE TESTED.
 3513  016476  036767  163044  163026         BIT     BITPT+2,TSTMAP+2  ;CHECK FOR BANK IN HI MAP.
 3514  016504  001760                         BEQ     3$                ;BR IF NOT THERE.
 3515  016506  012702  060000          4$:    MOV     #60000, R2        ;SET ADR POINTER TO TOP OF BANK
 3516  016512  000411                         BR      7$                ;GO TO COMMON EXIT
 3517  016514  162702  020000          5$:    SUB     #20000, R2        ;BACK POINTER DOWN ONE BANK
 3518  016520  006267  163020          6$:    ASR     BITPT             ;MOVE POINTER TO NEXT LOWER BANK
 3519  016524  103420                         BCS     10$               ;BRANCH TO EXIT IF NO MORE  MEM
 3520  016526  036767  163012  162774         BIT     BITPT,  TSTMAP    ;CHECK IF BANK EXISTS
 3521  016534  001767                         BEQ     5$                ;BRANCH IF BANK DOESN'T EXIST
 3522  016536  036767  163002  163024  7$:    BIT     BITPT,  FADMAP    ;CHECK IF FIRST BANK FLAG.
 3523  016544  001004                         BNE     8$                ;BR IF FIRST BANK.
 3524  016546  036767  162774  163016         BIT     BITPT+2,FADMAP+2  ;CHECK IF FIRST BANK FLAG.
 3525  016554  001402                         BEQ     9$                ;BR IF NO FLAG FOUND.
 3526  016556  016705  163004          8$:    MOV     FADMSK, R5        ;SET UP R5 TO FIND FIRST ADDRESS.
 3527  016562  016716  162766          9$:    MOV     MMORE,  (SP)      ;RESET RETURN ADDRESS
 3528  016566  000207                 10$:    RTS     PC                ;RETURN
```

```
                                .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
                                ;*  BITS 16 AND 17 ARE IN $TMP0.
                                ;;*****************************************************************
016570  010200                  PHYADR: MOV     R2,     R0      ;VITRUAL INTO R0
016572  005067  162362                  CLR     $TMP0           ;CLEAR TEMP SAVE OF HIGH BITS
016576  005767  162004                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
016602  001417                          BEQ     1$              ;BRANCH IF NO MEM MGMT
016604  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016606  013701  172344                  MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
016612  006301                          ASL     R1              ;SHIFT IT 6 TIMES
016614  006301                          ASL     R1
016616  006301                          ASL     R1
016620  006301                          ASL     R1
016622  006301                          ASL     R1
016624  006167  162330                  ROL     $TMP0           ;SAVE EXTRA BITS
016630  006301                          ASL     R1
016632  006167  162322                  ROL     $TMP0
016636  060100                          ADD     R1,     R0      ;ADD SHIFTED PAR TO VIRTUAL
016640  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016642  000207                  1$:     RTS     PC              ;RETURN
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
                                ;;*****************************************************************
016644  005000                  BANKNO: CLR     R0              ;INIT R0
016646  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016650  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016652  016701  162666                  MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
016656  016702  162664                  MOV     BITPT+2,R2      ;...HI 64K.
016662  006202                  1$:     ASR     R2              ;SHIFT POINTER...HI
016664  006001                          ROR     R1              ;...LO
016666  103403                          BCS     2$              ;BR WHEN POINTER FOUND.
016670  105200                          INCB    R0              ;COUNT BANKS.
016672  100373                          BPL     1$              ;BR IF NOT OVERFLOW.
016674  000000                          HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
016676                          2$:
016676  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016700  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016702  000207                          RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
                                ;;*****************************************************************
016704                          SETCON:
016704  004467  176232                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
016710  010022                  2$:     MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
016712  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
016714  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
016716  004767  176776                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
016722  000207                          RTS     PC              ;RETURN
```

```
                                ;;*****************************************************************
                                ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
                                ;;*****************************************************************
016724  106112                ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
016726  106112                        ROLB    (R2)            ;(R2)=177775 OR 000002
016730  106112                        ROLB    (R2)            ;(R2)=177773 OR 000004
016732  106112                        ROLB    (R2)            ;(R2)=177767 OR 000010
016734  106112                        ROLB    (R2)            ;(R2)=177757 OR 000020
016736  106112                        ROLB    (R2)            ;(R2)=177737 OR 000040
016740  106112                        ROLB    (R2)            ;(R2)=177677 OR 000100
016742  106112                        ROLB    (R2)            ;(R2)=177777 OR 000000
016744  106122                        ROLB    (R2)+           ;(R2)=177577 OR 000200
016746  106112                        ROLB    (R2)            ;(R2)=177377 OR 000400
016750  106112                        ROLB    (R2)            ;(R2)=176777 OR 001000
016752  106112                        ROLB    (R2)            ;(R2)=175777 OR 002000
016754  106112                        ROLB    (R2)            ;(R2)=173777 OR 004000
016756  106112                        ROLB    (R2)            ;(R2)=167777 OR 010000
016760  106112                        ROLB    (R2)            ;(R2)=157777 OR 020000
016762  106112                        ROLB    (R2)            ;(R2)=137777 OR 040000
016764  106112                        ROLB    (R2)            ;(R2)=077777 OR 100000
016766  106122                        ROLB    (R2)+           ;(R2)=177777 OR 000000
016770  000207                        RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
                                ;;*****************************************************************
016772  012704  000020        W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS

016776  010022                2$:     MOV     R0,(R2)+
017000  010022                        MOV     R0,(R2)+
017002  010022                        MOV     R0,(R2)+
017004  010022                        MOV     R0,(R2)+

017006  010322                        MOV     R3,(R2)+
017010  010322                        MOV     R3,(R2)+
017012  010322                        MOV     R3,(R2)+
017014  010322                        MOV     R3,(R2)+

017016  010022                        MOV     R0,(R2)+
017020  010022                        MOV     R0,(R2)+
017022  010022                        MOV     R0,(R2)+
017024  010022                        MOV     R0,(R2)+

017026  010322                        MOV     R3,(R2)+
017030  010322                        MOV     R3,(R2)+
017032  010322                        MOV     R3,(R2)+
017034  010322                        MOV     R3,(R2)+

017036  005304                        DEC     R4
017040  001356                        BNE     2$
017042  010046                        MOV     R0,     -(SP)   ;SAVE R0
017044  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
017046  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
017050  000207                        RTS     PC              ;RETURN
```

```
3635                                    .SBTTL  RELOCATION SUBROUTINES.
3636                                    ;;*****************************************************************
3637                                    ;* ROUTINE TO RELOCATE PROGRAM CODE
3638                                    ;;*****************************************************************
3639  017052                            RELOC:
3640  017052  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3641  017054  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3642  017056  010446                           MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3643  017060  012502                    4$:    MOV     (R5)+, R2       ;GET FIRST LOCATION.
3644  017062  012503                           MOV     (R5)+, R3       ;GET FIRST LOCATION OF DESTINATION.
3645  017064  012704  020000                   MOV     #20000, R4      ;SET UP 8K COUNTER.
3646  017070  012223                    1$:    MOV     (R2)+, (R3)+    ;MOV THE DATA.
3647  017072  005304                           DEC     R4              ;COUNT THE WORDS.
3648  017074  001375                           BNE     1$              ;BR IF MORE.
3649  017076  012704  02⌴⌴00                   MOV     #20000, R4      ;RESET THE COUNTER.
3650  017102  024243                    2$:    CMP     -(R2), -(R3)    ;CHECK THE DATA JUST MOVED.
3651  017104  001417                           BEQ     3$              ;BR IF DATA OK.
3652  017106  011267  162012                   MOV     (R2), $GDDAT    ;GET SOURCE DATA.
3653  017112  011367  162010                   MOV     (R3), $BDDAT    ;GET DESTINATION DATA.
3654  017116  010267  161776                   MOV     R2, $GDADR      ;GET SOURCE ADDRESS.
3655  017122  010367  161774                   MOV     R3, $BDADR      ;GET DESTINATION ADDRESS.
3656  017126  004767  003116                   JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
3657  017132  000023                           .WORD   23              ;ERROR TYPE CODE.
3658  017134  000000                           HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
3659  017136  162705  000004                   SUB     #4, R5          ;ADJUST RETURN POINTER.
3660  017142  000746                           BR      4$              ;GO BACK AND TRY AGAIN.
3661  017144  005304                    3$:    DEC     R4              ;COUNT WORDS.
3662  017146  001355                           BNE     2$              ;BR IF MORE.
3663  017150  004567  004422                   JSR     R5, $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
3664  017154  026623                           .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
3665                                                                    ;"PROGRAM RELOCATED TO "
3666  017156  010346                           MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
3667  017160  004767  006052                   JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
3668  017164  012604                           MOV     (SP)+,R4        ;;POP STACK INTO R4
3669  017166  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
3670  017170  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
3671  017172  000205                           RTS     R5              ;RETURN
3672                                    ;;*****************************************************************
3673                                    ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
3674                                    ;;*****************************************************************
3675  017174  022767  000003  161400   RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
3676  017202  001401                           BEQ     1$              ;BR IF OK
3677  017204  000000                           HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
3678  017206                            1$:
3679  017206  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3680  017210  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3681  017212  005767  161370                   TST     MMAVA
3682  017216  001465                           BEQ     10$
3683  017220  012737  007600  172346           MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
3684  017226  005000                           CLR     R0              ;INIT BANK POINTER...LO 64K
3685  017230  012701  100000                   MOV     #BIT15, R1      ;...HI 64K.
3686  017234  162737  000200  172346   2$:     SUB     #200.   @#KIPAR3        ;BACK DOWN ONE BANK.
3687  017242  006001                           ROR     R1              ;MOVE POINTER...HI 64K.
3688  017244  006000                           ROR     R0              ;..LO 64K.
3689  017246  103500                           BCS     90$
3690  017250  030167  162252                   BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
3691  017254  001003                        BNE     3$              ;BR IF AVAILABLE
3692  017256  030067  162242                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
3693  017262  001764                        BEQ     2$              ;BR IF NO BANK FOUND.
3694  017264  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3695  017272  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3696  017274  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3697  017276  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3698  017304  006001                        ROR     R1              ;SHIFT POINTER.
3699  017306  006000                        ROR     R0              ;...LO 64K.
3700  017310  103457                        BCS     90$             ;BR IF OVERFLOW.
3701  017312  030167  162210          5$:   BIT     R1,     MEMMAP+2 ;      CHECK IF BANK EXISTS...HI 64K.
3702  017316  001003                        BNE     6$              ;BR IF BANK EXISTS.
3703  017320  030067  162200                BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
3704  017324  001764                        BEQ     4$              ;BR IF BANK DOESN'T EXIST.
3705  017326  052601                  6$:   BIS     (SP)+,  R1      ;GET SECOND BANK POINTER.
3706  017330  052600                        BIS     (SP)+,  R0      ;...LO 64K.
3707  017332  030067  161244                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
3708  017336  001044                        BNE     90$             ;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     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3713  017356  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
3714                            ;*  PROGRAM SHOULD NOW BE EXICUTING 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$:  MOV     #BIT8,  R0      ;SET BANK POINTER TO TOP OF MEM.
3719  017376  005001                        CLR     R1              ;SET ADDRESS POINTER TO TOP.
3720  017400  162701  020000          11$:  SUB     #20000, R1      ;BACK DOWN ONE BANK.
3721  017404  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
3722  017406  103420                        BCS     90$             ;BR IF OVERFLOW.
3723  017410  030067  162110                BIT     R0,     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     R0              ;MOV POINTER DOWN ONE BANK.
3727  017424  103411                        BCS     90$             ;BR IF OVERFLOW.
3728  017426  030067  162072                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3729  017432  001762                        BEQ     11$             ;BR TO START OVER IF NO LOWER BANK.
3730  017434  010046                        MOV     R0,     -(SP)   ;SAVE THE POINTER.
3731  017436  006300                        ASL     R0              ;RESET POINTER TO HI BANK.
3732  017440  052600                        BIS     (SP)+,  R0      ;SET BIT FOR LO BANK.
3733  017442  030067  161134                BIT     R0,     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,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3742  017472  060107                        ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
3743                            ;*  PROGRAM NOW EXICUTING 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,     RELOCF  ;SET THE RELOCATION FACTOR.
3746  017502  060137  000004                ADD     R1,     @#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$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
3760  017564  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3761  017566  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3762  017570  066716  161004                  ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
3763  017574  000207                          RTS     PC              ;RETURN
3764
3765                                  ;;*****************************************************************
3766                                  ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3767                                  ;;*****************************************************************
3768  017576  032767  000003  160776  RELO:   BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3769  017604  001401                          BEQ     1$              ;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 8K 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                                  ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3781  017652  000444                          BR      30$             ;BR TO COMMON EXIT.
3782
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$:    .WORD   0               ;SOURCE STARTING ADDRESS.
3787  017672  000000                          .WORD   0               ;DESTINATION STARTING ADDRESS.
3788  017674  161607                          SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
3789                                  ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3790  017676  161606                          SUB     (SP),   SP      ;RESET THE STACK POINTER.
3791  017700  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3792  017702  012700  001612                  MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3793  017706  166620  000002          21$:    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 8K OF
3815                                ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3816                                ;;*****************************************************************
3817  020004  016700  161510        RESLDR: MOV     LMAD,   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     @#SR0           ;DISABLE MEM MGMT.
3823  020026  012701  040000        2$:     MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
3824  020032  012702  002734                MOV     #1500., R2      ;GET COUNTER.
3825  020036  014140                3$:     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     @#SR0           ;ENABLE MEM MGMT.
3831  020056  000207                4$:     RTS     PC              ;RETURN.
```

```
                                        .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.
                                        ;;*****************************************************************
020060  011667  161036        PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
020064  004567  003506                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020070  026562                        .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"UNEXPECTED MEMORY PARITY TRAP."
020072  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
020074  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
020076  016703  161516                MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
020102  005733                1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
020104  100415                        BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
020106  005713                        TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
020110  001374                        BNE     1$              ;BR IF MORE REGISTERS.
020112  004767  000132                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;***ERROR***    NO REGISTER INDICATED ERROR
020116  000024                        .WORD   24              ;ERROR TYPE CODE.
020120  000417                        BR      4$              ;EXIT
020122  005713                2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
020124  001415                        BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
020126  005733                        TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
020130  100374                        BPL     2$              ;BR IF NO ERROR FLAG.
020132  004567  003440                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
020136  026653                        .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"MORE THAN ONE ERROR FOUND."
020140                          3$:
020140  004767  000610          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
020144  004767  002100                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
020150  000025                        .WORD   25              ;ERROR TYPE CODE.
020152  004767  000216                JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
020156  000761                        BR      2$              ;GO LOOK FOR MORE ERRORS.
020160                          4$:
020160  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
020162  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
020164  000002                        RTI                     ;RETURN.

                                        ;;*****************************************************************
                                        ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
                                        ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                        ;;*****************************************************************
020166  005767  162122          MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
020172  001434                        BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
020174  032777  000100  160736        BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
020202  001030                        BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
020204  005767  160370                TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
020210  001410                        BEQ     SETAE           ;BR IF PROG IN BANK 0.
020212  032777  000040  160720        BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
020220  001004                        BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
020222  026727  161334  001000        CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
020230  103415                        BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
020232  016737  161376  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
020240  005037  000116                CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
020244  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
```

```
3888  020246  016703  161346          MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3889  020252  052733  000001  MAMF1:  BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
3890  020256  005713                  TST     (R3)            ;CHECK FOR END OF TABLE.
3891  020260  001374                  BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
3892  020262  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3893  020264  000207          MAMF2:  RTS     PC              ;RETURN.
3894
3895                          ;;*****************************************************************
3896                          ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3897                          ;;*****************************************************************
3898  020266  005767  162022  CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3899  020272  001437                  BEQ     4$              ;BR IF NO PARITY REGISTERS.
3900  020274  032777  000100  160636  BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3901  020302  001033                  BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
3902  020304  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3903  020306  016703  161306          MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
3904  020312  005733          1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3905  020314  100023                  BPL     3$              ;BR IF NO ERROR
3906  020316  032773  000001  177776  BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3907  020324  001010                  BNE     2$              ;BR IF NO ACTION ENABLE.
3908  020326  004767  000422  64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3909  020332  004767  001712          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3910  020336  000026                  .WORD   26              ;ERROR TYPE CODE.
3911  020340  000411                  BR      3$
3912  020342  004767  000026          JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3913  020346                  2$:
3914  020346  004767  000402  65$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3915  020352  004767  001672          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3916  020356  000027                  .WORD   27              ;ERROR TYPE CODE.
3917  020360  004767  000010          JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3918  020364  005713          3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3919  020366  001351                  BNE     1$              ;BR IF MORE.
3920  020370  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3921  020372  000207          4$:     RTS     PC              ;RETURN.
3922
3923                          ;;*****************************************************************
3924                          ;*  THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY,
3925                          ;*  TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
3926                          ;*  LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
3927                          ;;*****************************************************************
3928  020374                  PSCAN:
3929  020374  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3930  020376  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3931  020400  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3932  020402  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3933  020404  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3934  020406  013746  000114          MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
3935  020412  013746  000116          MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
3936  020416  004567  003154          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3937  020422  026717                  .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
3938                                                          ;"SCANNING MEMORY FOR BAD PARITY."
3939  020424  012700  000001          MOV     #BIT0,  R0      ;SET BIT POINTER TO FIRST BANK.
3940  020430  005001                  CLR     R1              ;CLR HI 64K POINTER.
3941  020432  005002                  CLR     R2              ;INIT ADDRESS POINTER.
3942  020434  005004                  CLR     R4              ;INIT ERROR DETECTED FLAG.
3943  020436  004767  000256          JSR     PC,     CLRPAR  ;CLEAR THE PARITY REGISTERS.
```

```
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$:
3956  020512  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3957  020514  111201          3$:     MOVB    (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          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3967  020544  004767  001500                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3968  020550  000030                          .WORD   30              ;ERROR TYPE CODE.
3969  020552  111212                          MOVB    (R2),   (R2)    ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
3970  020554  005053                          CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
3971  020556  105712                          TSTB    (R2)            ;CHECK IF THE PARITY ERROR WAS CLEARED.
3972  020560  005733                          TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
3973  020562  100005                          BPL     6$              ;BR IF IT IS OK.
3974  020564  004567  003306                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3975  020570  026761                          .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
3976                                                                  ;"PARITY ERROR WILL NOT CLEAR."
3977  020572  005073  177776                  CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
3978  020576  005713          6$:     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              ;UPDATE 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   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
4001  020674                        21$:
4002  020674  012637  000116                MOV     (SP)+,@#116     ;;POP STACK INTO @#116
4003  020700  012637  000114                MOV     (SP)+,@#114     ;;POP STACK INTO @#114
4004  020704  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
4005  020706  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
4006  020710  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
4007  020712  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4008  020714  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4009  020716  000207                        RTS     PC              ;RETURN.
4010
4011                                ;;*****************************************************************
4012                                ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
4013                                ;;*****************************************************************
4014  020720                        CLRPAR:
4015  020720  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4016  020722  016703  163672                MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
4017  020726  005713                1$:     TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
4018  020730  001402                        BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
4019  020732  005033                        CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
4020  020734  000774                        BR      1$              ;BR FOR MORE
4021  020736                        2$:
4022  020736  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
4023  020740  000207                        RTS     PC              ;RETURN.
4024
4025                                .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
4026                                ;;*****************************************************************
4027                                ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
4028                                ;*  FOR ERROR PRINTOUT BY .SERROR & .SERRTYP ROUTINES FROM **SYSMAC**.
4029                                ;;*****************************************************************
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),  $TMPO   ;GET THE PARITY REGISTER ADDRESS.
4035  020760  013367  160176                MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
4036  020764  000402                        BR      SPRNTO
4037
4038  020766  011367  160166        SPRNTP: MOV     (R3),   $TMPO   ;GET THE PARITY REGISTER ADDRESS.
4039  020772  010267  160122        SPRNTO: 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,     $TMPO   ;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
4052
4053                                ;;*****************************************************************
4054                                ;*  SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
4055                                ;*  R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
```

```
                        ;;*****************************************************************
021042  005710          TYPMAP: TST     (R0             ;CHECK IF ANY MEMORY IN MAP...LO 64K.
021044  001007                  BNE     1$              ;BR IF MEMORY IN MAP.
021046  005760  000002          TST     2(R0)           ;...HI 64K.
021052  001004                  BNE     1$              ;BR IF MEMORY IN MAP.
021054  004567  002516          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021060  026347                  .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
                                                        ;"NO MEMORY FOUND."
021062  000475                  BR      6$              ;EXIT
021064                  1$:
021064  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
021066  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
021070  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
021072  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
021074  012701  000001          MOV     #BIT0,  R1      ;SET UP BANK POINTER...LO 64K.
021100  005002                  CLR     R2              ;...HI 64K.
021102  012703  177777          MOV     #-1,    R3      ;SET UP ADDRESS POINTER TO -1.
021106  010304                  MOV     R3,     R4      ;HI BITS OF ADDRESS AS WILL.
021110  030110          2$:     BIT     R1,     (R0)    ;CHECK THE MAP FOR THIS BANK.
021112  001014                  BNE     3$              ;BR IF THIS BANK PRESENT.
021114  030260  000002          BIT     R2,     2(R0)   ;CHECK HI 64K MAP.
021120  001011                  BNE     3$              ;BR IF THIS BANK PRESENT.
021122  105703                  TSTB    R3              ;CHECK FOR PREVIOUS PRINTOUT.
021124  001042                  BNE     5$              ;BR IF ALREADY TYPED "TO"
021126  162703  000001          SUB     #1,     R3      ;BACK UP TO LAST ADR OF PREVIOUS BANK.
021132  005604                  SBC     R4              ;...HI ADDRESS BITS.
021134  004567  002436          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021140  025603                  .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
021142  000410                  BR      4$              ;GO TO TYPE THE ADDRESS.
021144  105703          3$:     TSTB    R3              ;CHECK FOR PREVIOUS TYPEOUT.
021146  001431                  BEQ     5$              ;BR IF ALREADY TYPE "FROM".
021150  062703  000001          ADD     #1,     R3      ;POINT TO FIRST ADDRESS OF THIS BANK.
021154  005504                  ADC     R4              ;...HI BITS OF ADDRESS.
021156  004567  002414          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021162  025573                  .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
021164                  4$:
021164  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
021166  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
021170  006303                  ASL     R3              ;BIT 15 INTO C-BIT
021172  006104                  ROL     R4              ;BIT 15 INTO R4.
021174  006003                  ROR     R3              ;RESTORE BITS 14-0.
021176  010446                  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**.
021200  013746  177776          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021204  004767  003540          JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
021210     003                  .BYTE   3               ;;TYPE 3 DIGIT(S)
021211     000                  .BYTE   0               ;;SUPPRESS LEADING ZEROS
021212  010346                  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**.
021214  013746  177776          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021220  004767  003524          JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
021224     005                  .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     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
4161  021352  122667  157524                CMPB    (SP)+,$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          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
4184  021502  116767  157374  157500          MOVB    $TSTNM,$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 TES
4189  021532  016777  157344  157402  $OVER:  MOV     $TSTNM,@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 8K BOUNDRY 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    $TSTNM, #23     ;CHECK IF IN SECTION 3.
4198  021576  103002                          BHIS    3$              ;BR IF IN SECTION 3.
4199  021600  006267  160002          2$:     ASR     BLKMSK          ;RESET BOUNDRY TO 4K.
4200  021604  016767  157752  157752  3$:     MOV     FSTADR, TMPFAD  ;GET FIRST ADDRESS.
4201  021612  005767  156762                  TST     RELOCF          ;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4212  021670  027020                          .WORD   NOMTST          ;ADDRESS OF MESSAGE TO BE TYPED
4213                                                                  ;"NO MEMORY TESTED"
4214  021672  016716  157744                  MOV     .TST32, (SP)    ;ADJUST RETURN ADR FOR ABORT.
4215  021676  000207                          RTS     PC              ;ABORT.
4216  021700  016767  157670  157670  4$:     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$:    TST     R4                 ;CHECK FOR A MINIMUM BLOCK SIZE.
4228  021770  001503                        BEQ     20$                ;BR IF NO MIN BLOCK SIZE.
4229  021772  030467  157566                BIT     R4,     TMPFAD     ;CHECK IF FIRST ADR ON BLOCK BOUNDRY.
4230  021776  001416                        BEQ     11$                ;BR IF FIRST ADR ON BLOCK BOUNDRY.
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 BOUNDRY.
4234  022016  001006                        BNE     11$                ;BR IF NOT ON 4K BOUNDRY.
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$:    BIT     R4,     TMPLAD     ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
4238  022040  001414                        BEQ     12$                ;BR IF ON BLOCK BOUNDRY.
4239  022042  040467  157530                BIC     R4,     TMPLAD     ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
4240  022046  032767  017777  157522        BIT     #MASK4K,TMPLAD     ;CHECK IF ADJUSTED TO 4K BOUNDRY.
4241  022054  001006                        BNE     12$                ;BR IF NOT ON 4K BOUNDRY.
4242  022056  046767  157520  157444        BIC     LADMAP, TSTMAP     ;SKIP TESTING LAST BANK.
4243  022064  046767  157514  157440        BIC     LADMAP+2,TSTMAP+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$:  CMP     TMPLAD, TMPFAD     ;CHECK IF ANY MEMORY LEFT.
4249  022120  101406                        BLOS    15$                ;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                MOVB    $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,     $TYPOS     ;GO TO THE SUBROUTINE
4264  022162     003                        .BYTE   3                  ;TYPE 3 DIGITS.
4265  022163     001                        .BYTE   1                  ;TYPE LEADING ZEROS.
4266  022164  000427                        BR      ENDINS             ;RETURN TO SKIP TEST.
4267  022166  062716  000004        16$:    ADD     #4,     (SP)       ;SKIP THE SKIP ON RETURN.
4268  022172  062767  000004  156706        ADD     #4,     $LPADR     ;ADJUST THE LOOP ADR PAST THE SKIP.
4269  022200  012767  017777  157360  20$:  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     TMPLAD, 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  022250                          $ERROR:
4296  022250  062716  000002                 ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
4297  022254  105267  156623          7$:    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     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4300  022270  032777  002000  156642        BIT     #BIT10,@SWR     ;;BELL ON ERROR?
4301  022276  001403                 BEQ     1$                      ;;NO - SKIP
4302  022300  004567  001272                 JSR     R5,     $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,$ERRPC
4307  022324  117767  156566  156562        MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
4308  022332  032777  020000  156600        BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
4309  022340  001005                 BNE     20$                     ;;SKIP TYPEOUTS
4310  022342  004767  000106                 JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
4311  022346  004567  001224                 JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4312  022352  001201                 .WORD   $CRLF                   ;ADDRESS OF MESSAGE TO BE TYPED
4313  022354                          20$:
4314  022354  122767  000001  156642        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     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
4332  022446  001001                 BNE     6$                      ;;BRANCH IF NO
4333  022450  000000                 HALT                            ;;YES
4334  022452                          6$:
4335  022452  000207                 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  022454                            $ERRTYP:
4345  022454  004567  001116                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4346  022460  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4347  022462  010046                          MOV     R0,-(SP)                ;SAVE R0
4348  022464  005000                          CLR     R0                      ;PICKUP THE ITEM INDEX
4349  022466  156700  156422                BISB    $ITEMB,R0
4350  022472  001007                          BNE     1$                      ;IF ITEM NUMBER IS ZERO, JUST
4351                                                                          ;TYPE THE PC OF THE ERROR
4352  022474  016746  156416                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  022500  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4357  022504  004767  002264                JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
4358  022510  000513                        BR      10$                     ;GET OUT
4359  022512  016767  156400  156774  1$:   MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
4360  022520  166767  156054  156766        SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
4361  022526  005300                        DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
4362  022530  006300                        ASL     R0                      ;       WORK FOR THE ERROR TABLE
4363  022532  006300                        ASL     R0
4364  022534  006300                        ASL     R0
4365  022536  066700  157074                ADD     .ERRTB, R0      ;FORM TABLE POINTER
4366  022542  012067  000006                MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
4367  022546  001406                        BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
4368  022550  004567  001022                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4369  022554  000000                2$:     .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
4370  022556  004567  001014                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4371  022562  001201                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4372  022564  012067  000006        3$:     MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
4373  022570  001406                        BEQ     5$                      ;SKIP TYPEOUT IF 0
4374  022572  004567  001000                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4375  022576  000000                4$:     .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
4376  022600  004567  000772                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4377  022604  001201                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4378  022606  010146                5$:     MOV     R1,-(SP)                ;SAVE R1
4379  022610  012001                        MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
4380  022612  001451                        BEQ     9$                      ;BR IF NO DATA TO BE TYPED
4381  022614  066701  155760                ADD     RELOCF, R1      ;ADJUST POINTER
4382  022620  012000                        MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
4383  022622  066700  155752                ADD     RELOCF, R0      ;ADJUST POINTER.
4384  022626  105720                6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
4385  022630  001006                        BNE     7$                      ;BR IF NOT 16-BIT OCTAL
4386  022632  013146                        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  022634  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4390  022640  004767  002130                JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
4391  022644  000436                        BR      8$
```

```
4392  022646  100406                  7$:     BMI     17$                 ;BRANCH IF NOT DECIMAL
4393  022650  013146                          MOV     @(R1)+,-(SP)        ;;SAVE @(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     @#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     @(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$:
4405  022704  005046                          CLR     -(SP)               ;CLEAR THE WORD ON THE STACK.
4406  022706  113116                          MOVB    @(R1)+, (SP)        ;PUT THE DATA ON THE STACK.
4407                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4408                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4409  022710  013746  177776                  MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
4410  022714  004767  002030                  JSR     PC,     $TYPOS      ;GO TO THE SUBROUTINE
4411  022720     003                          .BYTE   3                   ;TYPE 3 DIGITS.
4412  022721     001                          .BYTE   1                   ;TYPE LEADING ZEROS.
4413  022722  005711                  8$:     TST     (R1)                        ;IS THERE ANOTHER NUMBER?
4414  022724  001404                          BEQ     9$                          ;BR IF NO
4415  022726  004567  000644                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
4416  022732  022752                          .WORD   11$                 ;ADDRESS OF MESSAGE TO BE TYPED
4417  022734  000734                          BR      6$                          ;LOOP
4418
4419  022736  012601                  9$:     MOV     (SP)+,R1                    ;RESTORE R1
4420  022740  012600                  10$:    MOV     (SP)+,R0                    ;RESTORE R0
4421  022742  004567  000630                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
4422  022746  001201                          .WORD   $CRLF               ;ADDRESS OF MESSAGE TO BE TYPED
4423  022750  000207                          RTS     PC                          ;RETURN
4424  022752  000011                  11$:    .ASCIZ  /       /           ;TAB CHARACTER.
4425                                          .EVEN
4426                                  .SBTTL  TTY INPUT ROUTINE
4427
4428                                  ;;**********************************************************************
4429                                  .ENABL  LSB
4430
4431                                  .DSABL  LSB
4432
4433
4434                                  ;;**********************************************************************
4435                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4436                                  ;*CALL:
4437                                  ;*      RDCHR                       ;;INPUT A SINGLE CHARACTER FROM THE TTY
4438                                  ;*      RETURN HERE                 ;;CHARACTER IS ON THE STACK
4439                                  ;*                                  ;;WITH PARITY BIT STRIPPED OFF
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    @$TKS               ;;WAIT FOR
4445  022770  100375                          BPL     1$                  ;;A CHARACTER
4446  022772  117766  156150  000004          MOVB    @$TKB,4(SP)         ;;READ THE TTY
4447  023000  042766  177600  000004          BIC     #^C<177>,4(SP)      ;;GET RID OF JUNK IF ANY
```

```
 4448  023006  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
 4449  023014  001013                          BNE     3$              ;;BRANCH IF NO
 4450  023016  105777  156122          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
 4451  023022  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
 4452  023024  117746  156116                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
 4453  023030  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 4454  023034  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
 4455  023040  001366                          BNE     2$              ;;IF NOT DISCARD IT
 4456  023042  000750                          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  042766  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                                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
 4467                                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
 4468                                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
 4469
 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     @#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
4508  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     R5,     $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 CHACTER
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  /↑U/<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 $HIOCT
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
```

```
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,     $RDLIN  ;GO TO THE SUBROUTINE
4565  023452  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
4566  023454  010067  000102                MOV     R0,5$           ;;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
4577  023510  006301                        ASL     R1              ;;*4
4578  023512  006102                        ROL     R2
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     RELOCF, 1$      ;MAKE IT PHYSICAL.
4607                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4608                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4609  023610  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4610  023614  004767  000004                JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
4611  023620  000000                1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
4612  023622  000205                        RTS     R5              ;RETURN.
4613
4614                                  .SBTTL  TYPE ROUTINE
4615
```

```
                                ;;*****************************************************************
                                ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                ;*
                                ;*CALL:
                                ;*1) USING A TRAP INSTRUCTION
                                ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                ;*OR
                                ;*      TYPE
                                ;*      MESADR
                                ;*

023624  105767  155327          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
023630  100002                          BPL     1$              ;;BR IF YES
023632  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
023634  000430                          BR      3$              ;;LEAVE
023636  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
023640  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
023644  122767  000001  155352          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
023652  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
023654  132767  000100  155343          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
023662  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
023664  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
023670  004767  000222                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
023674  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
023676  132767  000040  155321  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
023704  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
023706  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
023710  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
023712  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
023714  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
023716  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
023722  000002                          RTI                     ;;RETURN
023724  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
023730  001431                          BEQ     8$
023732  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
023736  001007                          BNE     5$
023740  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
023742  004567  177630                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023746  001201                          $CRLF
023750  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
023754  000754                          BR      2$              ;;GET NEXT CHARACTER
023756  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
023762  126726  155170          5$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
023766  001347                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
023770  016746  155160                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
023774  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
024000  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
024002  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
024006  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
024012  000770                          BR      7$              ;;LOOP
```

```
                                        ;HORIZONTAL TAB PROCESSOR
                                
024014  112716  000040          8$:   MOVB    #' ,(SP)          ;;REPLACE TAB WITH SPACE
024020  004767  000014          9$:   JSR     PC,$TYPEC         ;;TYPE A SPACE
024024  132767  000007  000052        BITB    #7,$CHARCNT       ;;BRANCH IF NOT AT
024032  001372                        BNE     9$                ;;TAB STOP
024034  005726                        TST     (SP)+             ;;POP SPACE OFF STACK
024036  000723                        BR      2$                ;;GET NEXT CHARACTER
024040  105777  155104        $TYPEC: TSTB    @$TPS             ;;WAIT UNTIL PRINTER IS READY
024044  100375                        BPL     $TYPEC
024046  116677  000002  155076        MOVB    2(SP),@$TPB       ;;LOAD CHAR TO BE TYPED INTO DATA REG.
024054  122766  000015  000002        CMPB    #CR,2(SP)         ;;IS CHARACTER A CARRIAGE RETURN?
024062  001003                        BNE     1$                ;;BRANCH IF NO
024064  105067  000014                CLRB    $CHARCNT          ;;YES--CLEAR CHARACTER COUNT
024070  000406                        BR      $TYPEX            ;;EXIT
024072  122766  000012  000002  1$:   CMPB    #LF,2(SP)         ;;IS CHARACTER A LINE FEED?
024100  001402                        BEQ     $TYPEX            ;;BRANCH IF YES
024102  105227                        INCB    (PC)+             ;;COUNT THE CHARACTER
024104  000000          $CHARCNT:.WORD  0                 ;;CHARACTER COUNT STORAGE
024106  000207          $TYPEX: RTS     PC

                                        .SBTTL  APT COMMUNICATIONS ROUTINE

                                        ;;**************************************************************
024110  112767  000001  000376  $ATY1:  MOVB    #1,$FFLG          ;;TO REPORT FATAL ERROR
024116  112767  000001  000366  $ATY3:  MOVB    #1,$MFLG          ;;TO TYPE A MESSAGE
024124  000403                        BR      $ATYC
024126  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG          ;;TO ONLY REPORT FATAL ERROR
024134          $ATYC:
024134  010046                        MOV     R0,-(SP)          ;;PUSH R0 ON STACK
024136  010146                        MOV     R1,-(SP)          ;;PUSH R1 ON STACK
024140  105767  000346                TSTB    $MFLG             ;;SHOULD TYPE A MESSAGE?
024144  001450                        BEQ     5$                ;;IF NOT:  BR
024146  122767  000001  155050        CMPB    #APTENV,$ENV      ;;OPERATING UNDER APT?
024154  001031                        BNE     3$                ;;IF NOT:  BR
024156  132767  000100  155041        BITB    #APTSPOOL,$ENVM   ;;SHOULD SPOOL MESSAGES?
024164  001425                        BEQ     3$                ;;IF NOT:  BR
024166  017600  000004                MOV     @4(SP),R0         ;;GET MESSAGE ADDR.
024172  062766  000002  000004        ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
024200  005767  155000          1$:   TST     $MSGTYPE          ;;SEE IF DONE W/ LAST XMISSION?
024204  001375                        BNE     1$                ;;IF NOT:  WAIT
024206  010067  155006                MOV     R0,$MSGAD         ;;PUT ADDR IN MAILBOX
024212  105720                  2$:   TSTB    (R0)+             ;;FIND END OF MESSAGE
024214  001376                        BNE     2$
024216  166700  154776                SUB     $MSGAD,R0         ;;SUB START OF MESSAGE
024222  006200                        ASR     R0                ;;GET MESSAGE LNGTH IN WORDS
024224  010067  154772                MOV     R0,$MSGLGT        ;;PUT LENGTH IN MAILBOX
024230  012767  000004  154746        MOV     #4,$MSGTYPE       ;;TELL APT TO TAKE MSG.
024236  000413                        BR      5$
024240  017667  000004  000016  3$:   MOV     @4(SP),4$         ;;PUT MSG ADDR IN JSR LINKAGE
024246  062766  000002  000004        ADD     #2,4(SP)                  ;;BUMP RETURN ADDRESS
024254  016746  153516                MOV     177776,-(SP)      ;;PUSH 177776 ON STACK
024260  004767  177340                JSR     PC,$TYPE          ;;CALL TYPE MACRO
024264  000000          4$:   .WORD   0
024266          5$:
024266  105767  000221                TSTB    $LF_G             ;;SHOULD LOG AN ERROR?
```

```
4728  024272  001422                          BEQ     10$             ;;IF NOT:  BR
4729  024274  017600  000004                  MOV     @4(SP),R0               ;;GET ERROR #
4730  024300  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4731  024306  012701  001344                  MOV     #$ASTAT,R1      ;;POINT TO TABLE START
4732  024312  005711                  6$:     TST     (R1)            ;;END OF TABLE?
4733  024314  100404                          BMI     8$              ;;IF SO:  BR
4734  024316  020021                          CMP     R0,(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  155160          8$:     CMP     $APTR,R1        ;;TABLE FULL?
4739  024332  001402                          BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
4740  024334  010021                          MOV     R0,(R1)+        ;;SET UP NEW ENTRY
4741  024336  005211                  9$:     INC     (R1)            ;;BUMP ERROR COUNT
4742  024340  105767  000150          10$:    *TSTB   $FFLG           ;;SHOULD REPORT FATAL ERROR?
4743  024344  001416                          BEQ     12$             ;;IF NOT:  BR
4744  024346  005767  154552                  TST     $ENV            ;;RUNNING UNDER APT?
4745  024352  001413                          BEQ     12$             ;;IF NOT:  BR
4746  024354  005767  154624          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
4747  024360  001375                          BNE     11$             ;;IF NOT:  WAIT
4748  024362  017667  000004  154616          MOV     @4(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)+,R0        ;;POP STACK INTO R0
4756  024422  000207                          RTS     PC              ;;RETURN
4757  024424                          $ATY6:
4758  024424  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4759  024426  016700  155060                  MOV     $APTR,R0
4760  024432  162700  001344                  SUB     #$ASTAT,R0      ;;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     R0,$MSGLG       ;;SET MESSAGE LENGTH
4764  024450  012767  001344  154542          MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
4765  024456  012767  000002  154520          MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
4766  024464  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4767  024466  000207                          RTS     PC              ;;RETURN
4768  024470                          $ATY7:
4769  024470  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4770  024472  012701  001344                  MOV     #$ASTAT,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)+,R0        ;;POP STACK INTO R0
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
```

```
                000100                  APTSPOOL=100
                000040                  APTCSUP=040
                                        ;;*****************************************************************
                                        .SBTTL  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

024516                                  $TYPDS:
024516  010046                                 MOV     R0,-(SP)        ;;PUSH R0 ON STACK
024520  010146                                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
024522  010246                                 MOV     R2,-(SP)        ;;PUSH R2 ON STACK
024524  010346                                 MOV     R3,-(SP)        ;;PUSH R3 ON STACK
024526  010546                                 MOV     R5,-(SP)        ;;PUSH R5 ON STACK
024530  012746  020200                        MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024534  016605  000020                        MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024540  100004                                BPL     1$              ;;BR IF INPUT IS POS.
024542  005405                                NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024544  112766  000055  000001                MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024552  016700  154022                1$:     MOV     RELOCF,R0       ;GET RELOCATION FACTOR.
024556  012703  024740                        MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024562  060003                                ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
024564  112723  000040                        MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024570  005002                        2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024572  016001  024730                        MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024576  160105                        3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024600  002402                                BLT     4$              ;;BR IF DONE
024602  005202                                INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024604  000774                                BR      3$
024606  060105                        4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024610  005702                                TST     R2              ;;CHECK IF BCD DIGIT=0
024612  001002                                BNE     5$              ;;FALL THROUGH IF 0
024614  105716                                TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024616  100407                                BMI     7$              ;;BR IF YES
024620  105316                        5$:     ASLB    (SP)            ;;MSD?
024622  103003                                BCC     6$              ;;BR IF NO
024624  116663  000001  177777                MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024632  052702  000060                6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024636  052702  000040                7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024642  110223                                MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024644  005720                                TST     (R0)+           ;;JUST INCREMENTING
024646  020067  154766                        CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
024652  103746                                BLO     2$              ;GO DO THE NEXT DIGIT
024654  101002                                BHI     8$              ;;GO TO EXIT
024656  010502                                MOV     R5,R2           ;;GET THE LSD
024660  000764                                BR      6$              ;;GO CHANGE TO ASCII
024662  105726                        8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024664  100003                                BPL     9$              ;;BR IF NO
024666  116663  177777  177776                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 $TYPOC
4872                                  ;*CALL:
4873                                  ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
4874                                  ;*      TYPON                   ;;CALL FOR TYPEOUT
4975                                  ;*
4876                                  ;*$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          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
4882  024754  116667  000001  000213          MOVB    1(SP),$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
```

```
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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4917  025134  025172                     .WORD   8$              ,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$:     MOV     (SP)+,R5        ;;RESTORE R5
4924  025154  012604                     MOV     (SP)+,R4        ;;RESTORE R4
4925  025156  012603                     MOV     (SP)+,R3        ;;RESTORE R3
4926  025160  016666  000002  000004     MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
4927  025166  012616                     MOV     (SP)+,(SP)
4928  025170  000002                     RTI                     ;;RETURN
4929  025172     000            8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
4930  025173     000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
4931  025174     000            $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
4932  025175     000            $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
4933  025176  000000            $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
4934                                    ;ERROR TRAP SERVICE ROUTINE
4935  025200  005727            ERRTRP: TST     (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
4936  025202  000000            1$:     .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 TYPOUT.
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$
4946  025234  000002                    RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
4947
4948                                    .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
4949                                    ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4950  025236                            $TYPAD:
4951  025236  010046                            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,     @#SR0   ;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     #KIPAR0, R1     ;SET TO APPROPRIATE PAR
4969  025316  011101                        MOV     (R1),   R1      ;GET CONTENTS OF PAR
4970  025320  012700  000006                MOV     #6.     R0      ;SET UP COUNTER
4971  025324  006301                4$:     ASL     R1              ;SHIFT PAR
4972  025326  006103                        ROL     R3              ;SAVE OVERFLOW BITS
4973  025330  077003                        SOB     R0,     4$      ;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$:     ASL     R2              ;FIRST DIGIT TO R3
4978  025344  006103                        ROL     R3
4979  025346  012700  000006                MOV     #6,R0           ;DIGIT COUNT
4980  025352  000404                        BR      3$              ;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$:     MOV     #3,R1           ;DIGIT SHIFT COUNT
4986  025370  062703  000060                ADD     #60,    R3      ;MAKE IT AN ASCII DIGIT
4987  025374  110367  000036                MOVB    R3,     8$      ;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     R0              ;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)+,R0        ;;POP STACK INTO R0
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
```

```
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  MTMAP:  .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  NOPES:  .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  PROREL: .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:'
```

```
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
      026644  042105  052040  020117
      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  DM5:    .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).'
```

```
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
```

```
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  DM25:   .ASCIZ  'TRAPPED TO 114.'
5294  030364  042105  052040  020117
5295  030372  030461  027064     000
5296  030377     106  044501  042514  DM26:   .ASCIZ  'FAILED TO TRAP.'
5297  030404  020104  047524  052040
5298  030412  040522  027120     000
5299  030417     050  041501  044524  DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
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  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
5305  030460  050120  042105  052040
5306  030466  020117  020064     000
5307
5308                                  ;;*****************************************************************
5309                                  ;DATA COLUMN HEADINGS
5310                                  ;;*****************************************************************
5311
5312  030473     120  004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
5313  030500  004507  027523  004502
5314  030506  040527  000123
5315  030512  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
5316  030520  050057  004503  040515
5317  030526  051411  041057  053411
5318  030534  051501     000
5319  030537     126  050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
5320  030544  027520  041520  046411
5321  030552  004501  027523  000102
5322  030560  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
5323  030566  050057  004503  042522
5324  030574  004507  040515     000
5325  030601     126  050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
5326  030606  027520  041520  046411
5327  030614  052501  004524  042522
5328  030622  004507  027523  004502
5329  030630  040527  000123
5330  030634  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
5331  030642  050057  004503  052511
5332  030650  004524  040515  051411
5333  030656  041057  053411  051501
5334  030664     000
5335  030665     126  050057  004503  DH22:   .ASCIZ  'V/PC   P/PC    PS S/B  PS WAS  MA      S/B     WAS'
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  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
5342  030736  050057  004503  051123
5343  030744  020103  040515  042011
```

```
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
5361
5362                                   ;;*****************************************************************
5363                                   ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
5364                                   ;;*****************************************************************
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,-1,0,0
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
5380                                           .EVEN
5381
5382          032110                   . = 32110                    ;THE LOADERS ARE SAVE HERE TO END OF 9K
5383
5384          000001                           .END
```

```
ABASE = 000000            385     426
ACDW1 = 000000            385     428
ACDW2 = 000000            385     429
ACPUOP= 000000            385     400
ADDW0 = 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
ADEVCT= 000000            385     391
ADEVM = 000000            385     427
AE    = 000001            180#   2662    2748    3889
AENV  = 000000            385     396
AENVM = 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#
APTSPO= 000100           4639    4707    4784#
ASWREG= 000000            385     398
ATESTN= 000000            385     3[illegible]
AUNIT = 000000            385     39[illegible]
AUSWR = 000000            385     399
AVECT1= 000000            385     424
AVECT2= 000000            385     425
BADADR  026502           1429    5114#
BANKNO  016644           1612    1621    1647    1657    3555#
BGEND   014376           3143    3169    3171#
```

```
BGERR   014400            562    3174*
BGEXIT  014512            563    3194*
BGLEN = 000025           3126    3129    3139    3172*
BGLOOP  014240           3127*   3201
BGTOP   014234           3126*
BG1     014330           3152*
BG2     014342           3156*   3160
BG3     014354           3152    3161*   3177
BG4     014355           3162*
BG5     014356           3158    3163*
BG6     014362           3163    3165*
BG7     014374           3167    3168    3170*   3183
BITPT   001544            523*   1134*   1135*   1171    1172    1173    1174    1182*   1183*   1185    1187    1190    1193*
                         2639    2641    2696    2698    3306*   3307*   3314    3316    3319*   3320*   3323    3325    3330
                         3331    3350    3353*   3356    3357    3371    3372    3373    3375    3384*   3385*   3389*   3394
                         3396    3417*   3418*   3420    3422    3424    3426    3430    3431    3461*   3463    3465    3468
                         3475    3477    3482    3483    3498    3500    3508*   3509*   3511    3513    3518*   3520    3522
                         3524    3558    3559
BIT0  = 000001            123*    236     252    1039    1206    1352    1363    1453    1471    2685    2714    2730    3306
                         3906    3939    4070    4208
BIT00 = 000001            113*    123
BIT01 = 000002            112*    122
BIT02 = 000004            111*    121
BIT03 = 000010            110*    120
BIT04 = 000020            109*    119
BIT05 = 000040            106*    118
BIT06 = 000100            107*    117
BIT07 = 000200            106*    116
BIT08 = 000400            105*    115    4157
BIT09 = 001000            104*    114    4167    4324
BIT1  = 000002            122*
BIT10 = 002000            103*   4300
BIT11 = 004000            102*   4174
BIT12 = 010000            101*
BIT13 = 020000            100*   3332    3359    3432    3469    4308
BIT14 = 040000             99*    218    4143
BIT15 = 100000             98*   2714    3384    3685
BIT2  = 000004            121*
BIT3  = 000010            120*
BIT4  = 000020            119*
BIT5  = 000040            118*
BIT6  = 000100            117*
BIT7  = 000200            116*
BIT8  = 000400            115*   3389    3718
BIT9  = 001000            114*
BLKMSK  001606            548*   2576    2577*   2587*   3309    3380    3412    4194*   4199*
BPTVEC= 000014            130*
BRGOB   014326            561    3151*   3172
BRGOBL  014230           3124*
BYTMES  025523           1033    5019*
CKPMER  020266           1735    2647    3489    3898*
CLRPAR  020720           1133    1201    3943    4014*
CONST   026537           1441    5119*
CR    = 000015             38*   4683    4693
CRLF  = 000200             39*    899    4654    4693
CTRLS   005002           1247    1258    1268*
```

```
DDISP = 177570          45*    362    871
DF1     031103         722   5365*
DF14    031121         777    782    787    813    818    823   5371*
DF2     031107         727    737    742    747    752    757    798    828    933   5367*
DF21    031127         803   5373*
DF22    031135         808   5375*
DF3     031114         732    762    767    772    793   5369*
DF30    031144         838    843   5378*
DH1     030473         720   5312*
DH12    030537         765    770   5319*
DH14    030560         775    785   5322*
DH15    030601         780   5325*
DH2     030512         725    730    735    740    745    750    755    760    791    796   5315*
DH21    030634         801   5330*
DH22    030665         806   5335*
DH23    030730         811   5341*
DH24    030770         816   5347*
DH25    031011         821   5350*
DH26    031042         826    831   5355*
DH30    031064         836   5358*
DIDBH   013652        2940*
DIDBL   013560        2891*
DIDO    013470        2842*
DIPDO   013750        2990*
DISPLA  001142         362*   871*   979*   888*  3752*  3805*  4189*  4299*
DISPRE  000174         196*   879
DM1     027071         719    779   5166*
DM10    027354         754   5200*
DM11    027413         759   5206*
DM12    027457         764   5213*
DM13    027533         769   5221*
DM14    027623         774   5231*
DM16    027722         784   5242*
DM17    030001         789   5250*
DM2     027125         724    729   5171*
DM20    030077         795   5261*
DM21    030131         800   5266*
DM22    030200         805   5273*
DM23    030236         810   5278*
DM24    030303         815   5285*
DM25    030357         820   5293*
DM26    030377         825   5296*
DM27    030417         830   5299*
DM31    030453         840   5304*
DM4     027161         734   5176*
DM5     027217         739   5182*
DM6     027255         744   5188*
DM7     027313         749   5194*
DONE    014552        3207*
DPDBH   014134        3088*
DPDBL   014042        3039*
DSWR  = 177570          44*    361    870
DT1     001644         573*   721
DT12    001672         577*   766    771
DT14    001704         579*   776    786
DT15    001716         581*   781
```

```
DT2     001656          575*    726     731     736     741     746     751     756     761     792     797
DT21    001734          584*    802
DT22    001752          587*    807
DT23    001772          590*    812
DT24    002010          593*    817
DT25    002020          595*    822
DT26    002034          597*    827     832
DT30    002046          599*    837
DT31    002060          601*    842
EMTVEC= 000030          133*
ENDINS  022244         4193    4266    4279*
ERRTRP  025200          207     917    1004    1112    4935*
ERRVEC= 000004          126*    206     868     869*    880*    913*    917*    929*    939*    964*   1034*   1027*   1037*
                       1097*   1112*   3746*   3799*   4148    4149*   4151*   4154*
FADMAP  001570          541*   1404*   1405*   3373    3375    3498    3500    3522    3524   4208*   4235    4236    4244
                       4246
FADMES  026372         1357    5100*
FADMSK  001566          539*   3526    4269*   4271*
FILL2   027066         4999    5160*
FLAG8K  001557          533*   3365*   3455*   3458*   3476*
FROM    025573         1046    4090    5027*
FSTADR  001562          536*   1340    1438*   3883    4200
GMPR    004102         1013    1093*
GMPRA   004130         1098*   1110
GMPRB   004144         1097    1104*
GMPRC   004152         1102    1106*
GMPRD   004212         1095    1118*
GNS     = ****** U      195     898
HT      = 000011         36*   4652    4693
IMPCHE= 177746          187*    940*
IMPCK   003314          928     929     938*
INITDN  015600         1508    1544    1571    1646    1656    3380*
INITEX  015712         3376    3378    3397    3399*
INITMM  015142         1491    1534    1582    1611    1620    1688    1702    1738    1759    1782    1807    1841    1986
                       1902    1954    1964    2007    2056    2067    2110    2159    2170    2218    2344    2355    2403
                       2527    2545    2578    2593    2638    2784    2845    2894    2943    2993    3042    3091    3125
                       3306*   3575
INSERT  021544         4191*
INSUFF  025610         1015    5030*
IOTVEC= 000020          131*
KIPAR0= 172340          166*    243     262*    3291*   3712*   3778*   4968
KIPAR1= 172342          167*    268*   3292*   3713*   3779*
KIPAR2= 172344          168*    975     995*   1072*   1181*   2713    3293*   3312*   3318*   3329    3386*   3416*   3434
                       3507*   3539    3694*   3697*   3712    3773*   3948    3949*   3988*   3993*
KIPAR3= 172346          169*   1069    1072    1073*   3294*   3329*   3334*   3434*   3435*   3683*   3686*   3694    3713
                       3774*
KIPAR4= 172350          170*   3295*
KIPAR5= 172352          171*   3296*
KIPAR6= 172354          172*   3297*
KIPAR7= 172356          173*   3298*
KIPDR0= 172300          155*    238    3283*
KIPDR1= 172302          156*   3284*
KIPDR2= 172304          157*   3285*
KIPDR3= 172306          158*   3286*
KIPDR4= 172310          159*   3287*
KIPDR5= 172312          160*   3288*
```

```
KIPDR6= 172314           161*  3289*
KIPDR7= 172316           162*  3290*
LADMAP  001602           547*  1432*  1433*  1456*  1457*  3323  3325  3342  3344  3357  3366  3394  3396
                        3408   3410   3424   3426   3443   3445  3465  3475  3477  4242  4243  4244  4246
LADMES  026457          1384   5110*
LADMSK  001600           545*  3327   3346   3368   3428   3447  3467  3479  4274*  4276*
LDDISP  001522           509*   887*
LF    = 000012            37*  4687   4693
LMAD    001520           508*   909    923*  3817
LSTADR  001574           542*  1346*  1396*  1400   1411   1450*  1451*  1452*  1453*  1454  3398  4209  4216
MAMF    020166          1468   2775   3875*
MAMF1   020252          3889*  3891
MAMF2   020264          3876   3878   3884   3893*
MANUAL  005270          1119   1259   1326   1339*  1431
MANUL1  005326          1345   1352*
MANUL2  005730          1347   1448*
MAPMEM  003334           939    952*
MAPRB   004260          1138   1148*
MASK4K= 017777           184*   966    991   1065   1402   1454  2577  3370  3481  3981  4233  4240  4269
                        4274
MEMMAP  001524           231    510*   953    954   1185   1187  1190  1377  1379  1418  1420  1423  1425
                        3690   3692   3701   3703   3723   3728  3951  3953
MEMMES  025505           962   5016*
MMAMES  025440           934   5009*
MMAVA   000606           234    291*   926*   932*   972    993  1024*  1030*  1067  1137  1179  1434  1448
                        1469   2710   3310   3382   3413   3505  3536  3681  3771  3820  3828  3946  3986
                        4957
MMDOWN  016410          1520   1556   1578   1652   1668   3392  3498*
MMINIT  015014           931   1029   1140   1473   3282*
MMORE   001554           529*  3390*  3391*  3399*  3484   3527
MMUP    015720          1504   1540   1596   1616   1631   1692  1713  1749  1772  1795  1835  1881  1896
                        1942   1958   2001   2044   2060   2104  2147  2163  2211  2331  2348  2396  2516
                        2539   2571   2586   2619   2774   2809  2859  2908  2958  3007  3056  3106  3205
                        3408*  3579
MMVEC = 000250           144*
MPAMEM  004220          1114   1133*
MPEND   002112           620*
MPPATS  002066           565    609*
MPRX    002314           560    700*  1093   1109   1113   1176  1241  1250  1257  1273  1317  1325  2631
                        3487   3875   3898
MPR0    002114           559    635*  1096   1150   1205   1268  1311  1320
MPR1    002124           639*
MPR10   002234           675*
MPR11   002244           679*
MPR12   002254           683*
MPR13   002264           687*
MPR14   002274           691*
MPR15   002304           695*  1246   1248   1249   1251   1254*
MPR2    002134           643*
MPR3    002144           647*
MPR4    002154           651*
MPR5    002164           655*
MPR6    002174           659*
MPR7    002204           663*
MPR8    002214           667*
MPR9    002224           671*
```

```
MTMAP   025546          1203    5023#
MTOE    026653          3858    5134#
MTR     025671          1116    5039#
MX1     026155          1227    5072#
MX2     026172          1235    5075#
MX3     026205          1211    5077#
MX4     026224          1217    5080#
MX5     026242          1223    5083#
MX6     026260          1256    5086#
NOMEM   026247          4062    5096#
NOMTST  027020          4212    5153#
NOPES   025772          4000    5051#
OEFLG   001560           534#   2658*   2767*
PARVEC= 000114           181#   2671*   2675*   2682*   3748*   3801*   3885*   3886*
PBTRP   012732           564    2682#
PESRV   020060           566    3838#
PEWNC   026761          3975    5147#
PHYADR  016570          1492    1509    1535    1545    1572    1583    3534#
PIRQ  = 177772            43#
PIRQVE= 000240           137#
PMEMAP  001540           520#   1148*   1149*   1173*   1174*   1248*   1249*   2639    2641
PRELOC  026623          3664    5129#
PRGMAP  000602           254     255     275     290#    904*    905*    3212    3214    3218    3675    3707    3715*   3733
                        3759*   3768    3908*   3809*   4219    4220
PROREL  026044          5058#
PR0   = 000000            60#
PR1   = 000040            61#
PR2   = 000100            62#
PR3   = 000140            63#
PR4   = 000200            64#
PR5   = 000240            65#
PR6   = 000300            66#
PR7   = 000340            67#
PS    = 177776            40#     41
PSCAN   020374          3864    3912    3917    3928#
PSW   = 177776            41#   1232    1361    1388    1445    3174    3254    4101    4109    4262    4356    4399    4396
                        4409    4477    4563    4609
PWRMSG  025725           327    5044#
PWRVEC= 000024           132#    296*    297*    306*    312*    324*    325*    863*    864*   3747*   3800*
RADTAB  001612           556#   3753    3792
RANTST  013350          2782#
RELOC   017052          3639*   3709    3738    3775    3785
RELOCF  000600           270     289#    906*   1467    3391    3741*   3745*   3754    3762    3783    3807*   3879    4201
                        4360    4381    4383    4606    4810
RELTOP  017174          3222    3675#
RELO    017576           277    3225    3768#
RESCHK  005154          1274    1311#
RESLDR  020004           283    3226    3817#
RESRVD  001516           506#    558    1275*   1279    1284    1289    1297    2715*   2716    2717    2746
RESTAR  000300           202     227#    329     903
RESTOR  000304           203     229#
REST1   000306           228     230#
REST2   000324           232     234#
RESVEC= 000010           127#
ROTATE  016724          1761    1784    3584#
RW    = 000006           176#   3283    3284    3285    3286    3290
```

```
SAVTST  001534          516*  1008*  1009*  1364*  1365*  1406*  1407*  3212  3214  4217  4218
SCANM   026717         3937   5141*
SELECT  002674          200    853*
SELFLG  001556          532*   851*   853*  1344
SETAE   020232         2646   3880   3682  3885*
SETCON  016704         1758   1781   2637  3574*
SKPMES  027044         4256   5157*
SPRNT   020742         1281   1299   1332  2654  4030*
SPRNTA  021030         4040   4044   4049*
SPRNTB  021034         4032   4050*
SPRNTP  020766         2700   2721   2734  2751  4038*
SPRNTQ  020754         3861   3908   3914  3966  4034*
SPRNTO  020772         1292   1514   1550  1662  2676  2692  2760  4036  4039*
SPRNT1  021000         1625   4042*
SPRNT2  021016         1497   1589   1707  1743  1766  1789  1847  1855  1863  1871  1908  1916  1924
                       1932   1970   1977  1984  1991  2014  2023  2032  2073  2080  2087  2094  2117
                       2126   2135   2176  2183  2190  2197  2224  2233  2242  2249  2258  2267  2274
                       2283   2292   2299  2308  2317  2361  2368  2375  2382  2409  2418  2427  2434
                       2443   2452   2459  2468  2477  2484  2493  2502  2554  2561  2602  2609  2799
                       4047*
SPRNT3  021012         2852   2901   2951  3000  3049  3099  4046*
SR0    = 177572         148*   236    252*  930*  1028*  1471  3299*  3822*  3830*  4959
SR1    = 177574         149*
SR2    = 177576         150*
SR3    = 172516         151*
STACK  = 001100          31*   230    271   273   557   861   939   952  1023
START   002666         199    851*
STARTA  002702         233    285    852   854*
START1  006006         282   1455   1465*  1466  3223  3268
STKLMT= 177774          42*
SWR     001140         304    317*   361*   859   870*   872   878*   885*   927  1025  1094  1717  1719
                      2633   3198   3216  3749  3751*  3802  3804*  3877  3881  3900  4143  4157  4159
                      4167   4174   4203  4300  4308  4321  4324
SWREG   000176         197*   878
SW0    = 000001         95*
SW00   = 000001         85*    95
SW01   = 000002         84*    94
SW02   = 000004         83*    93
SW03   = 000010         82*    92
SW04   = 000020         81*    91
SW05   = 000040         80*    90   4203
SW06   = 000100         79*    89   1094  2633
SW07   = 000200         78*    88   3216
SW08   = 000400         77*    87   1717
SW09   = 001000         76*    86
SW1    = 000002         94*
SW10   = 002000         75*
SW11   = 004000         74*
SW12   = 010000         73*   927  1025
SW13   = 020000         72*
SW14   = 040000         71*
SW15   = 100000         70*
SW2    = 000004         93*
SW3    = 000010         92*
SW4    = 000020         91*
SW5    = 000040         90*  3891
```

```
SW6    = 000100            89*   3877    3900
SW7    = 000200            88*
SW8    = 000400            97*
SW9    = 001000            86*
TBITVE= 000014            128*
TIMOUT  003632            204   1023*   1085
TKVEC  = 000060            135*
TMAP    004534           1184   1201*
TMPFAD  001564            538*  3377    3502    4200*   4205    4207*   4229    4231*   4232*   4233    4248    4270
TMPLAD  001576            544*  3328*   3347*   3429*   3448*   3451    4216*   4237    4239*   4240    4248    4275
TMPPT   001550            526*  3330*   3331*   3335*   3336*   3338    3340    3342    3344    3356*   3361*   3363    3366
                         3371*  3372*   3408    3410    3430*   3431*   3436*   3437*   3439    3441    3443    3445    3468*
                         3471*  3473    3482*   3483*
TO      025603           1056   1078    4083    5029*
TPVEC  = 000064            136*
TRAPVE= 000034            134*
TRTVEC= 000014            129*
TSTMAP  001530            513*  3314    3316    3338    3340    3350    3363    3420    3422    3439    3441    3463    3473
                         3511   3513    3520    4217*   4218*   4219*   4220*   4223    4225*   4226*   4235*   4236*   4242*
                         4243*  4250    4252
TST1    006054           1470   1472    1484*
TST10   006726           1718   1722    1731*
TST11   007010           1737   1754*
TST12   007074           1777*
TST13   007156           1799*
TST14   007624           1803   1946*
TST15   010150           1950   2048*
TST16   010500           2052   2151*
TST17   011322           2155   2336*
TST2    006200           1530*
TST20   012144           2340   2520*
TST21   012476           2524   2628*
TST22   013342           2635   2779*
TST23   013456           2836*
TST24   013546           2840   2885*
TST25   013640           2889   2934*
TST26   013736           2938   2984*
TST27   014030           2988   3033*
TST3    006274           1567*
TST30   014122           3037   3082*
TST31   014216           3086   3118*
TST32   014560            569   1488    3122    3210*
TST4    006400           1607*
TST5    006470           1642*
TST6    006564           1684*  1724
TST6A   006572           1687*  1725
TST7    006614           1698*  1724
TYPMAP  021042           1005   1239    4057*
UNEXPT  026562           3840   5123*
UP     = 000000            175*  3283    3284    3285    3286    3290
WWP    = 000004            179*  1153    1157    2662    2668
WWPBYT  012540           2639*
WWPB0   012504           2631*
WWPB1   012600           2648*  2773
WWPB2   012644           2660*  2769
WWPB3   013206           2729   2738*
```

```
WWPB4   013276          2679    2695    2764*
WWPBS   013322          2645    2651    2771*
W3X9    016772          1955    2057    2160    2345    3607*
$APTHD  001330           457     463*
$APTR   001512           500*   4738    4759
$ASTAT  001344           473*    500    4727    4731    4757    4760    4764    4770
$ASTEN  001510           499*
$ATYC   024134          4698    4700*
$ATY1   024110          4696*
$ATY3   024116          4642    4697*
$ATY4   024126          4317    4699*
$ATY6   024424          4757*
$ATY7   024470          4768*
$AUTOB  001134           358*
$BASE   001260           426*
$BDADR  001122           353*    590     593     595    3655*   3838*
$BDDAT  001126           355*    573     575     581     584     587     590     599    3182*   3653*   4050*
$BELL   001174           377*   4303    4336
$CDW1   001264           428*
$CDW2   001266           429*
$CHARC  024104          4659*   4669*   4676    4685*   4690*
$CMTAG  001100           341*    856     857     865
$CM3  = 000000           371*
$CM4  = 000004           371*    372*    373*    374*    375*
$CNTLG  023375          4536*
$CNTLU  023370          4504    4535*
$CPUOP  001232           400*
$CRLF   001201           379*   1007    1244    3228    4312    4336    4346    4371    4377    4422    4510    4535    4599
                        4658    4693
$DBLK   024740          4811    4847    4855*
$DDW0   001270           430*
$DDW1   001272           431*
$DDW10  001314           440*
$DDW11  001316           441*
$DDW12  001320           442*
$DDW13  001322           443*
$DDW14  001324           444*
$DDW15  001326           445*
$DDW2   001274           432*
$DDW3   001276           433*
$DDW4   001300           434*
$DDW5   001302           435*
$DDW6   001304           436*
$DDW7   001306           437*
$DDW8   001310           438*
$DDW9   001312           439*
$DEVCT  001214           391*
$DEVM   001262           427*
$DOAGN  014770          3245    3261    3267*
$DTBL   024730          4815    4851*
$ENDAD  014760           216     893    3263*   4331
$ENDCT  014714           865    3247*
$ENDMG  014774          3250    3269*
$ENULL  015011          3257    3272*
$ENV    001224           396*   4314    4637    4705    4744
$ENVM   001225           397*    883    4639    4644    4707
```

```
$EOP    014664          3215    3217    3238*
$EOPCT  014706           865*   3244*   3248
$ERFLG  001103           344*   4131    4163    4165    4171*   4281    4297*   4336
$ERMAX  001115           350*   4165    4188*   4281
$ERROR  022250          1282    1293    1300    1333    1498    1515    1551    1590    1626    1663    1708    1744    1767
                        1790    1848    1956    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    3189    3656    3849    3862
                        3909    3915    3967    4295*   4940
$ERRPC  001116           351*    573     575     577     579     581     584     590     593     595     597    4305*   4306*
                        4307    4336    4352    4359
$ERRTB  002356           567     717*
$ERRTY  022454          4310    4344*
$ERTTL  001112           348*   3963*   4304*   4336
$ESCAP  001172           376*   4187*   4327    4329    4336
$ETABL  001224           395*
$ETEND  001330           446*    469
$FATAL  001206           388*   4748*
$FFLG   024514          4696*   4699*   4742    4751*   4780*
$FILLC  001156           369*   4662    4693
$FILLS  001155           368*   4693
$GDADR  001120           352*    573     575     577     579     581     584     587     590     599    3181*   3654*   4030*
                        4039*   4042*   4043*   4047*   4048*
$GDDAT  001134           354*    573     575     577     581     584     587     590    3185*   3187*   3652*   4031*   4049*
$GET42  014750          3258*
$HD   = 000000            12
$HIBTS  001330           464*
$HICCT  023574          1368    1381    1392    1395*   1397    1408    4586*   4598*
$ICNT   001104           345*   4178*   4179    4181*   4280
$ILLUP  000756           296     312     331*
$INTAG  001135           359*
$ITEMB  001114           349*   4307*   4316    4336    4349
$LF     001202           380*   4336    4525    4535    4599    4693
$LFLG   024513          4727    4752*   4779*
$LPADR  001106           346*   1466*   1467*   1724*   4169*   4185*   4190    4268*   4280
$LPERR  001110           347*   4169    4186*   4280    4326
$MADR1  001236           413*
$MADR2  001242           417*
$MADR3  001246           420*
$MADR4  001252           423*
$MAIL   001204           386*    465     469     882    4184    4314    4637
$MAMS1  001234           407*
$MAMS2  001240           415*
$MAMS3  001244           418*
$MAMS4  001250           421*
$MBADR  001332           465*
$MFLG   024512          4697*   4703    4753*   4778*
$MNEW   023413          4539*
$MSGAD  001220           393*   4713*   4716    4764*
$MSGLG  001222           394*   4718*   4763*
$MSGTY  001204           387*   4711    4719*   4746    4750*   4761    4765*
$MSWR   023402          4537*
$MTYP1  001235           408*
```

```
$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
$QCNT   025174           4888*  4918*  4931*
$QMODE  025176           4883*  4887*  4892   4895*  4906*  4933*
$QVER   021532           4144   4162   4170   4180   4189*
$PASS   001212            390*   882*  3196   3241*  3242*  3251   3269   4176   4195   4221   4281
$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     18     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   3089
                         3124   3235   3240   3259   3268   3269   4132   4133   4134   4135   4136   4143   4155
                         4157   4158   4163   4164   4165   4172   4173   4174   4186   4139   4280   4287   4289
                         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
$TMP0   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*  4939*
$TN   = 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
```

```
                    1799    1805*   1943    1946    1952*   2045    2048    2054*   2148    2151    2157*   2333    2336
                    2342*   2517    2520    2526*   2620    2631*   2776    2782*   2811    2836    2842*   2860    2885
                    2891*   2909    2934    2940*   2959    2984    2990*   3008    3033    3039*   3057    3082    3088*
                    3107    3118    3124*
$TPB      001152     366*   4682*   4693
$TPFLG    001157     370*   4631    4693
$TPS      001150     365*   4680    4693
$TSTM     001334     466*
$TSTNM    001102     281*    343*   1723*   3211*   4131    4161    4183*   4184    4189    4197    4259    4281    4299
                    4336
$TTYIN    023360    4472    4473    4489    4512    4530    4534*
$TYPAD    025236    1049    1060    1082    3667    4402    4950*
$TYPDS    024516    3255    4397    4799*
$TYPE     023624    4610    4631*   4724
$TYPEC    024040    4661    4668    4675    4680*   4681
$TYPEX    024106    4686    4688    4691*
$TYPOC    024774    1233    4357    4390    4886*
$TYPON    025010    4885    4888*
$TYPOS    024750    4102    4110    4263    4410    4881*
$UNIT     001216     392*
$UNITM    001340     468*
$USWR     001230     399*
$VECT1    001254     424*
$VECT2    001256     425*
$VERPC    001514     505*     575     577     579     581     584     590     593     595     597    4359*   4360*
$XTSTR    021274    4146*
$OFILL    025175    4882*   4886*   4896    4932*
$40CAT=   ******  U 4143    4310
.         032110     191*    195*    206*    214     215*    217*    219*    220*    308     332     340*    381     453
                     454*    456*    458*    535*    700*    860     899*    901    3172    3269    3273    4280    4282
                    4336    4429    4534*   4535    4541    4599    4693    4781*   4855*   5382*
.BGERR    001624     562*   3145
.BGEXI    001626     563*   3146
.BRGOB    001622     561*   3128
.CONST    001610     549*   1447*   1687    1701
.EIGHT    001640     568*   4832
.ERRTB    001636     567*   4365
.MPPAT    001632     565*   1734
.MPRX     001620     560*   2661    2667    2690    2728    3844    3888    3903    3958    4016
.MPRO     001616     559*   2684
.PBTRP    001630     564*   2671
.PESRV    001634     566*   2675    2682    3885
.RESRV    001614     558*
.STACK    001612     557*   1465
.TST32    001642     569*   4214
.$ASTA=   ******  U 4697    4700
.$X       001330     453*    458
.3X9      007636    1952*
.3X13     012156    2526*
```

```
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
        2148  2333  2517  2620  2776  2811  2860  2909  2959  3008  3057  3107
POP        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
SCOPIN  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*   624   634
SPACE    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  1483  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  2589  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
```

```
        3581    3583    3604    3606    3636    3638    3672    3674    3765    3767    3812    3816    3833    3837    3871
        3874    3895    3897    3923    3927    4011    4013    4026    4029    4053    4056    4128    4283    4336    4428
        4434    4463    4543    4600    4604    4616    4695    4786    4858    5005    5008    5163    5165    5308    5310
        5362    5364
SWRSU    138#    866#
SWRDR3     1#    2201    2206    2321    2326    2386    2391    2506    2511    3631
SIREGS  1475#    1477    1523    1560    1600    1635    1677
TYPADR     1#    1048    1059    1081    3666    4401
TYPBIN   138#
TYPBN      1#
TYPBYT     1#    4256    4404
TYPDEC   138#    3251    4393
TYPDS      1#    3252    4394
TYPE       1#     326     895     933     961    1006    1014    1032    1045    1055    1077    1115    1202    1210    1216
        1222    1226    1234    1356    1383    1428    1440    3227    3249    3256    3663    3839    3857    3936    3974
        3999    4061    4082    4089    4211    4255    4302    4311    4345    4369    4370    4374    4376    4415    4421
        4485    4492    4498    4503    4509    4511    4514    4519    4524    4593    4595    4657    4846    4916    4988
        4998
TYPNAM     1#    138#    889
TYPNUM   138#
TYPOC      1#    1230    4354    4387
TYPOCS   138#    4097    4105
TYPOCT   138#    1229    4352    4386
TYPON      1#
TYPOS      1#    4099    4107    4260    4407
TYPTXT   138#
$CKWD      1#    1291    1331    1496    1513    1549    1588    1624    1661    1706    1742    1765    1788    1845    1854
        1862    1870    1907    1915    1923    1931    1969    1976    1983    1990    2013    2022    2031    2072    2079
        2086    2093    2116    2125    2134    2175    2182    2189    2196    2223    2232    2241    2248    2257    2266
        2273    2282    2291    2298    2307    2316    2360    2367    2374    2381    2408    2417    2426    2433    2442
        2451    2458    2467    2476    2483    2492    2501    2553    2560    2601    2608    2653    2720    2750    2759
        2798    2851    2900    2950    2999    3048    3098
$INDN      1#    1508    1544    1571    1646    1656
$INMM      1#    1491    1534    1582    1611    1620    1688    1702    1738    1759    1782    1807    1841    1985    1902
        1954    1964    2007    2056    2067    2110    2159    2170    2218    2344    2355    2403    2527    2545    2578
        2593    2638    2784    2845    2894    2943    2993    3042    3091    3124    3574
$MMDN      1#    1518    1554    1576    1650    1666
$MMUP      1#    1502    1538    1594    1614    1629    1690    1711    1747    1770    1793    1833    1879    1894    1940
        1956    1999    2042    2058    2102    2145    2161    2209    2329    2346    2394    2514    2537    2569    2584
        2617    2806    2857    2906    2956    3005    3054    3104    3577
$$CKWD     1#    1281    1292    1299    1332    1497    1514    1550    1589    1625    1662    1707    1743    1756    1789
        1847    1855    1863    1871    1908    1916    1924    1932    1970    1977    1984    1991    2014    2023    2032
        2073    2080    2087    2094    2117    2126    2135    2176    2183    2190    2197    2224    2233    2242    2249
        2258    2267    2274    2283    2292    2299    2308    2317    2361    2368    2375    2382    2409    2418    2427
        2434    2443    2452    2459    2468    2477    2484    2493    2502    2554    2561    2602    2609    2654    2676
        2692    2700    2721    2734    2751    2760    2799    2852    2901    2951    3000    3049    3099    3860    3909
        3913    3965
$$CMRE   334#
$$CMTM   334#     371     372     373     374
$$ESCA   138#
$$NEWT   138#    1475    1521    1558    1598    1633    1674    1694    1727    1751    1774    1796    1943    2045    2148
        2333    2517    2620    2776    2811    2860    2909    2959    3008    3057    3107
$$SETM   882#
$$SKIP   138#    1737
.BGHC   3107#    3109
.CII    2620#    2622
```

DZQMCC.P11 02-DEC-76 08:47 CROSS REFERENCE TABLE -- MACRO NAMES

```
.DIDBH   2909*   2911
.DIDBL   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   3082   3118   3207
.SETUP      1*    189
.SWRHI      1*     13
.SWRLO      1*     25*
.TM7     1694*   1696
.$ACT1      1*    210
.$APTB      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)
```