MS11K

0-124K MEMORY DIAGNOSTIC

MD-11-DZQMC-D

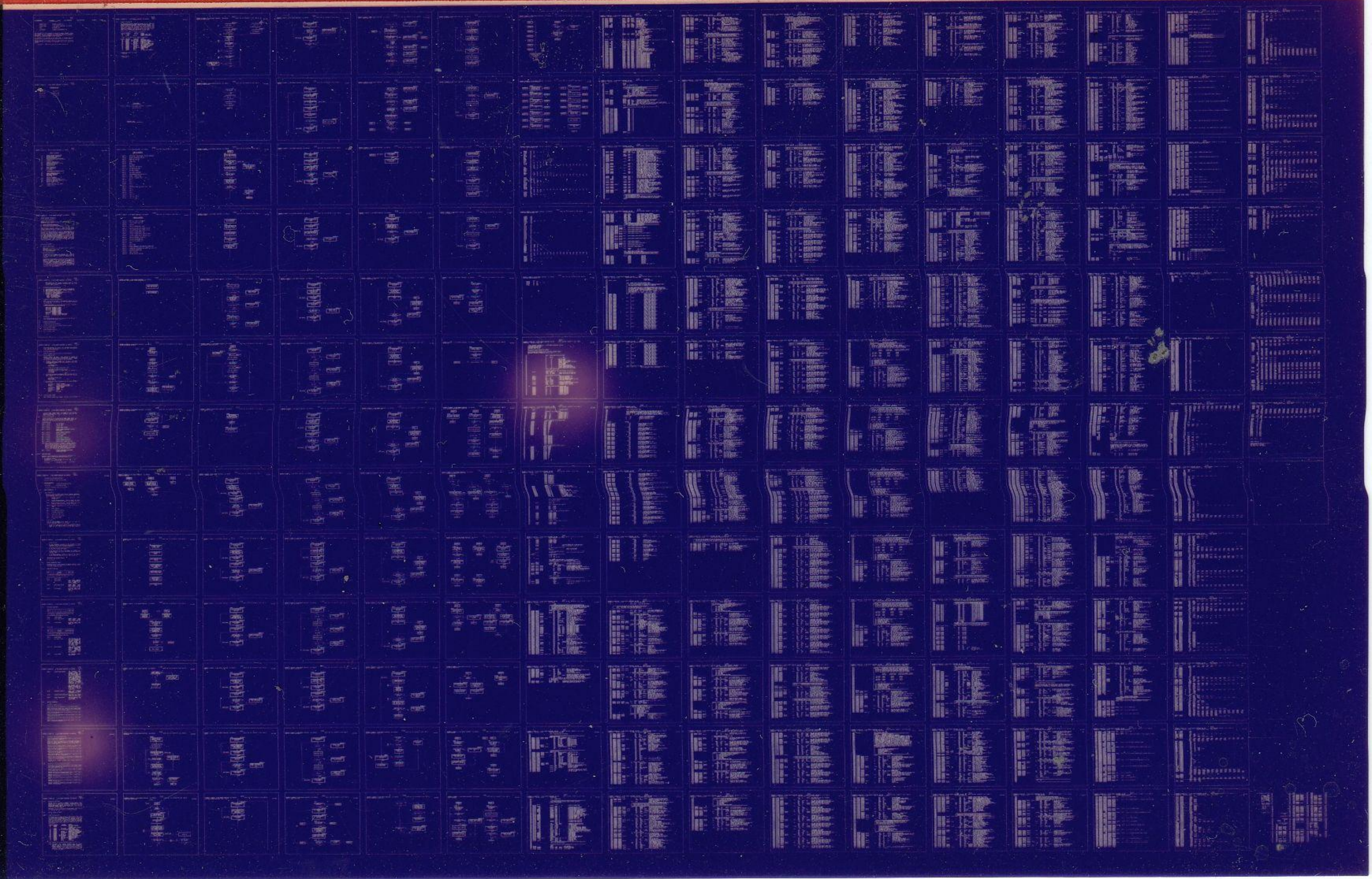
EP-DZQMC-D-DL-D

FICHE 1 OF 1

OCT 1977

digital

MADE IN USA



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

PRODUCT NAME: D-12AK MEMORY EXERCISER, 16K VERSION.

DATE: SEPTEMBER,1977

MAINTAINER: DIAGNOSTIC ENGINEERING, 32J

REVISION HISTORY

Revision A: May 1975

Revision B: October 1975

Revision C: October 1976

Revision D: June 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.

The intent of this program is to test as comprehensively as possible all memory systems manufactured by DEC without concentrating on any one system. Although the tests relate to general designs they may be complete for certain systems. E.G. Any core memory from the 8K MM11-L on up need not have any other addressing or worst case patterns run but in order to completely test the MS11-K MOS memory another diagnostic is required. This test is also not intended to be a 100% test of the memory. Other tests that do I/O may find memory problems that this test is unable to.

1.2 System Requirements

A. Hardware Requirements

PDP11 family processor with a minimum of 16K of memory.
optional...
Any 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 and APT.

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-11/34 - MAINDEC-11-DFKTG
PDP-11/40 - MAINDEC-11-DBQEA
OR MAINDEC-11-DCQKC
PDP-11/45 - MAINDEC-11-DCQKC
PDP-11/60 - MAINDEC-11-DQKDA
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 SWOS=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 or APT11 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: 65 seconds for first 16k + 15 seconds for each additional 16k.

Full Pass: 3 minutes 40 seconds for first 16k +

3 minutes for each additional 16k.

Iteration Inhibited: same as first pass

b. For Parity Memory
First Pass: 1 minute 40 seconds per 16k.

Full Pass: 8 minutes per 16K

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:

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.

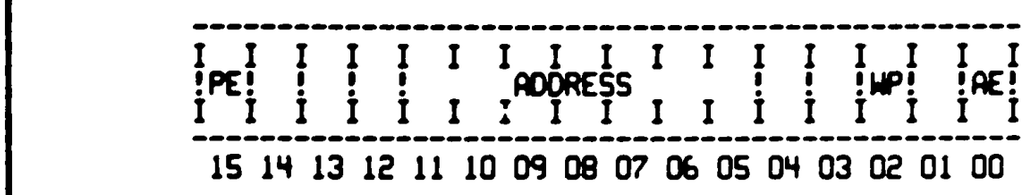
4.0 PERFORMANCE AND PROGRESS REPORTS

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:

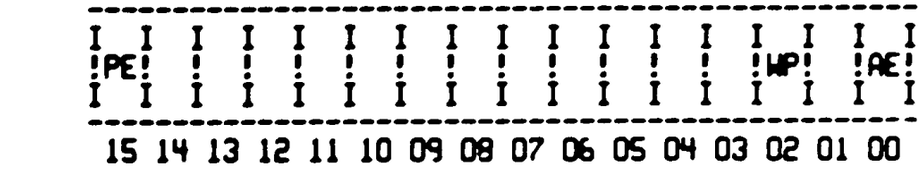
5.1 CORE PARITY REGISTER



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	
BIT 13	SET INHIBIT MODE	WHEN THIS BIT IS SET TO A 1, IT ENABLES THE INH MODE POINTER TO INHIBIT EITHER THE FIRST OR SECOND 16K FROM EVER GOING INTO THE DIAG. CHECK OR ECC DISABLE MODE.
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

BIT03	INHIBIT MODE POINTER	THE INHIBIT MODE POINTER WORKS IN CONJUNCTION WITH THE SET INHIBIT MODE BIT. WHEN BIT 13 IS SET TO A 1, A 16K PORTION OF MEMORY IS INHIBITTED FROM OPERATING IN THE ECC DISABLE MODE OR DIAGNOSTIC CHECK MODE. THE INHIBIT MODE POINTER INDICATES WHICH 16K IS BEING INHIBITED,,,BIT 3 =1 THE SECOND 16K OF MEMORY IS INHIBITTED. WHEN BIT 13 IS SET TO A 0, BIT 3 BECOMES INOPERATIVE.
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 13,14,15, AND 16 Write a pattern which complements when address BIT 3 XOR BIT 9 complements.

TEST 17 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 20 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 21 Executes an instruction which does a DATI and a DATO on the memory under test.

TEST 22 Executes an instruction which does a DATI and a DATOB on the low byte of memory under test.

TEST 23 Executes an instruction which does a DATI and a DATOB on the high byte.

TEST 24 Executes an instruction which does a DATIP and a DATO.

TEST 25 Executes an instruction which does a DATIP and a DATOB on the low byte.

TEST 26 EXECUTES AN INSTRUCTION WHICH DOES A DATIP and a DATOB on the high byte.

6.4 Section 4: Mos Tests

TEST 27 -Writes a pattern of 000377 through memory, then compliments it addressing downward, compliments the new pattern addressing upward, compliments the third pattern addressing upward and finally compliments this new AB patterns addressing downward.

TEST 30-31 Write a checkerboard through memory then stalls for 2 seconds and then verifies no data has changed.

6.5 Special Toggle In Tests

6.5.1 Toggle-in-program #1

The following is a toggle in memory address test. This test is useful when an 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     027037     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.

6.5.2 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
****

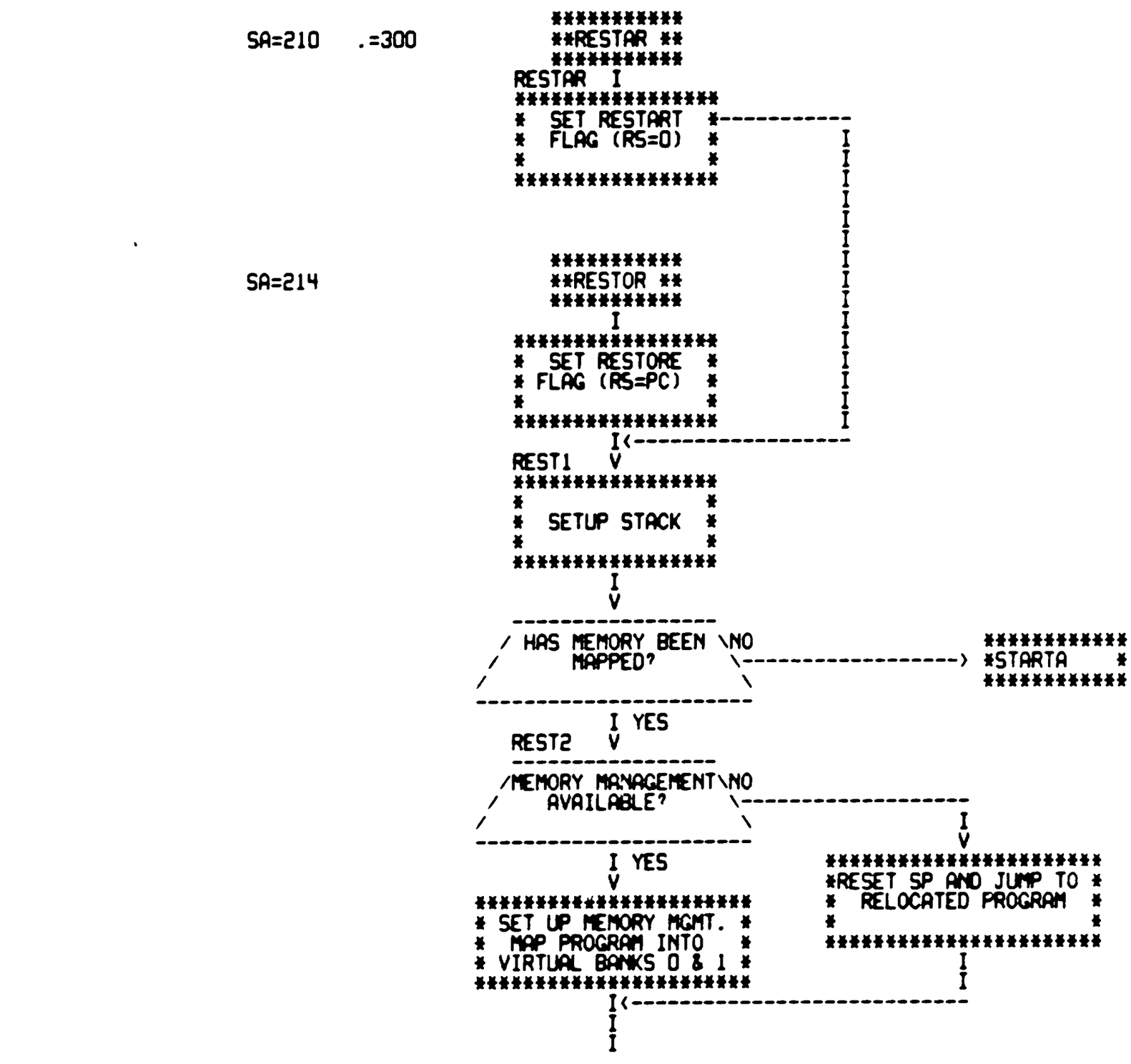
MAINDEC-11-DZQMC-D 0-124K MEMORY EXERCISER. (16K VERSION)

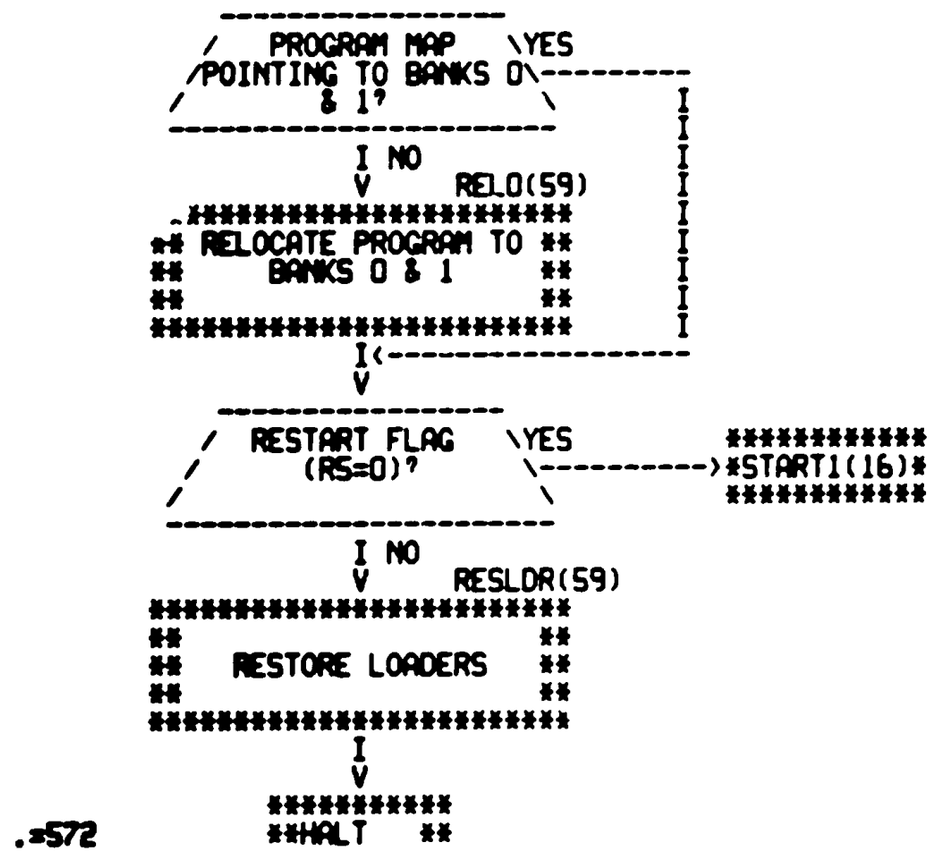
****

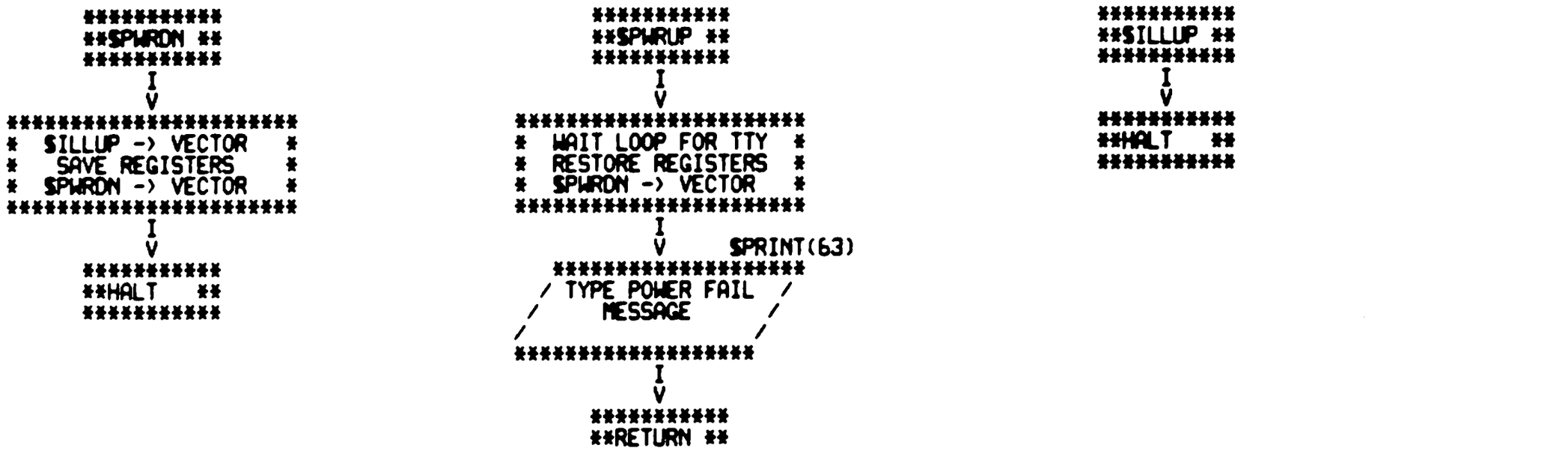
TABLE OF CONTENTS
****

TABLE OF CONTENTS
****

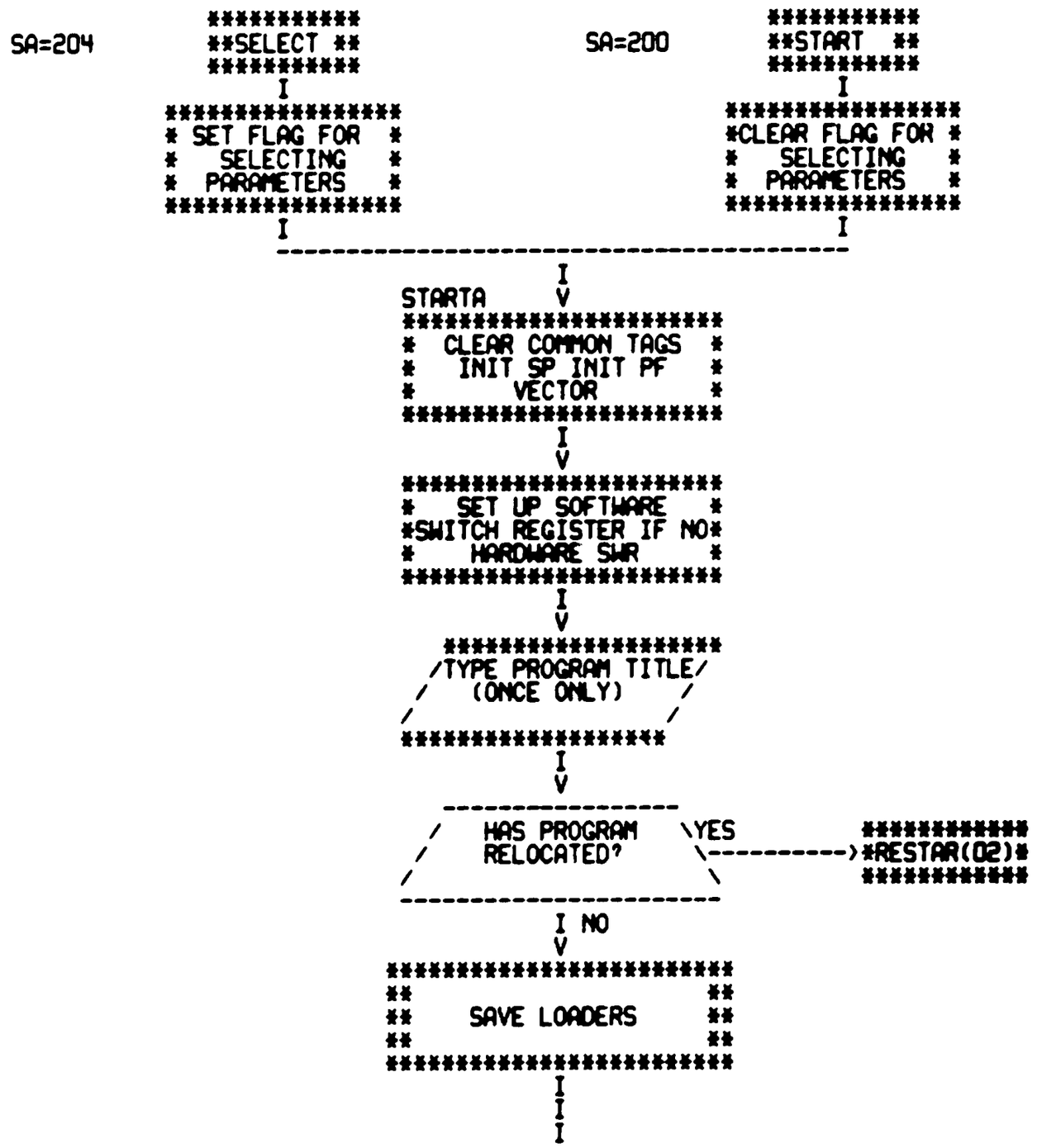
```
                        **************************
                        * SWITCH SETTINGS AND    *
                        *  BASIC DEFINITIONS     *
                        *                        *
                        **************************


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

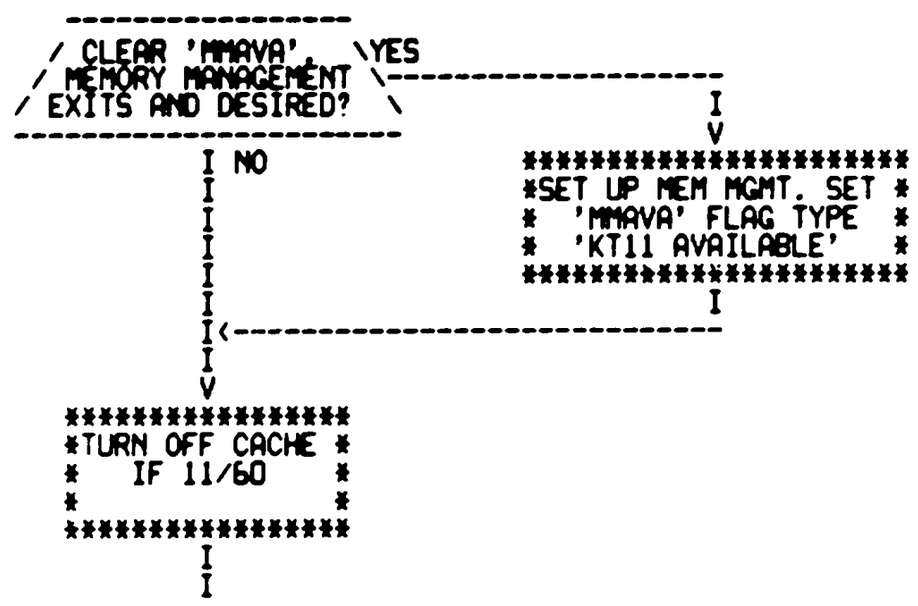

SA=210 .=300
****
**RESTAR **
****
RESTAR I
SET RESTART FLAG (RS=0)
SA=214
****
**RESTOR **
****
I
SET RESTORE FLAG (RS=PC)
I<
REST1 V
SETUP STACK
I
V
HAS MEMORY BEEN MAPPED?
NO
****
*START A *
****
I YES
REST2 V
MEMORY MANAGEMENT AVAILABLE?
NO
I
V
I YES
V
RESET SP AND JUMP TO RELOCATED PROGRAM
SET UP MEMORY MGMT. MAP PROGRAM INTO VIRTUAL BANKS 0 & 1
I<
I
I

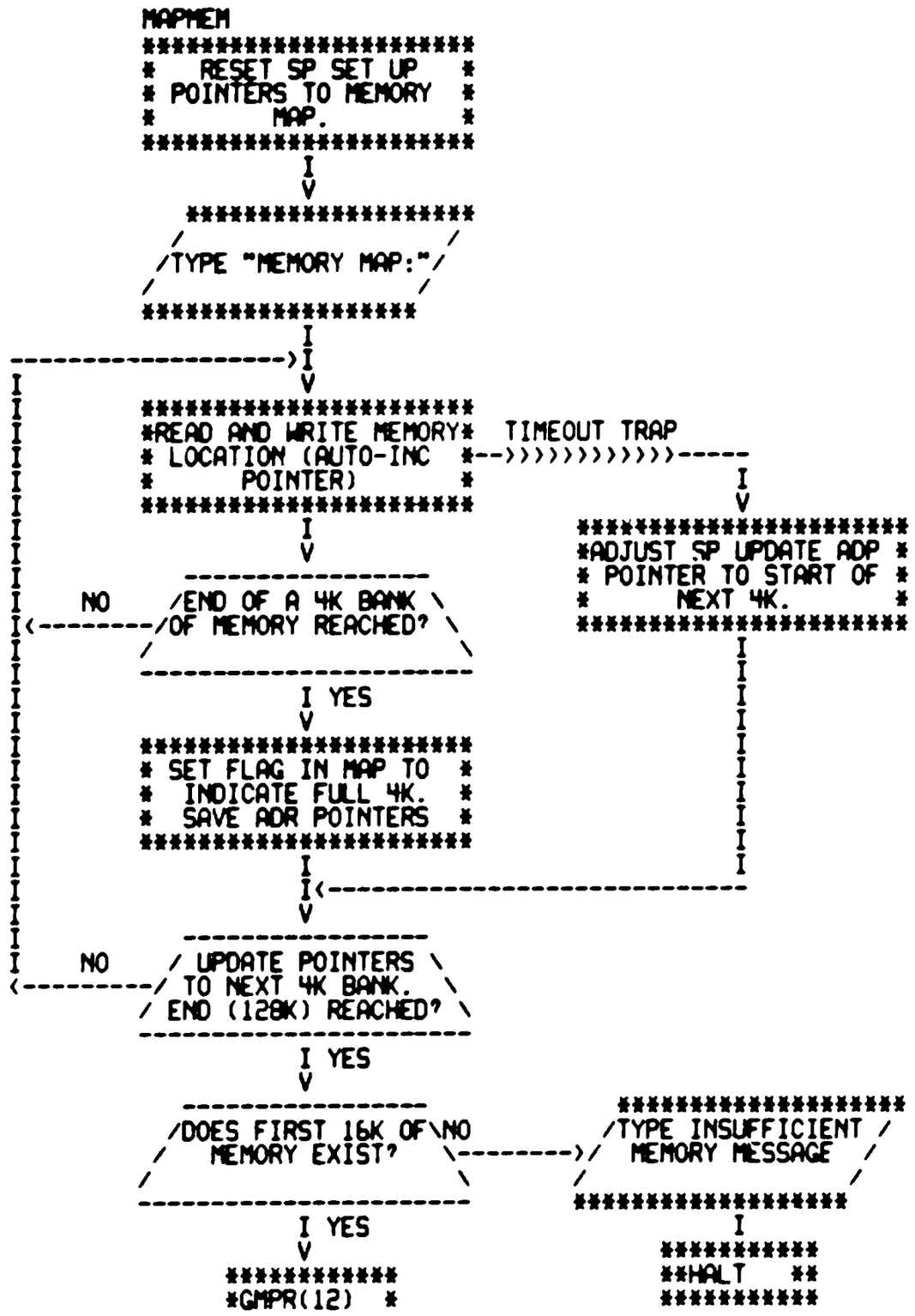

PROGRAM MAP POINTING TO BANKS 0 & 1?
YES
NO
RELO(59)
RELOCATE PROGRAM TO BANKS 0 & 1
RESTART FLAG (RS=0)?
YES
START1(16)
NO
RESLDR(59)
RESTORE LOADERS
.=572
HALT

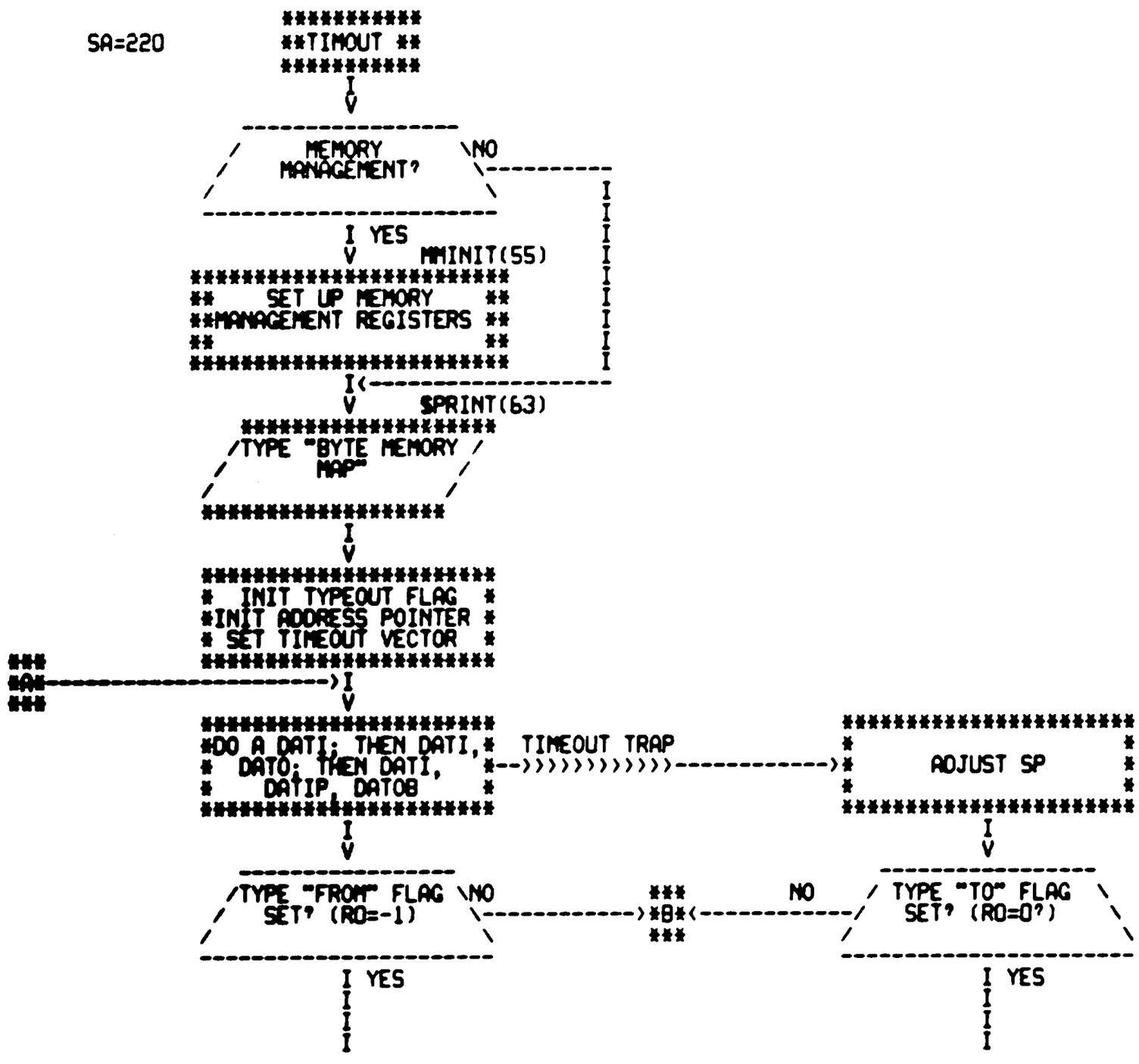

**$PWRDN **
$ILLUP -> VECTOR
SAVE REGISTERS
$PWRDN -> VECTOR
**HALT **
**$PWRUP **
WAIT LOOP FOR TTY
RESTORE REGISTERS
$PWRDN -> VECTOR
$PRINT(63)
TYPE POWER FAIL MESSAGE
**RETURN **
**$ILLUP **
**HALT **

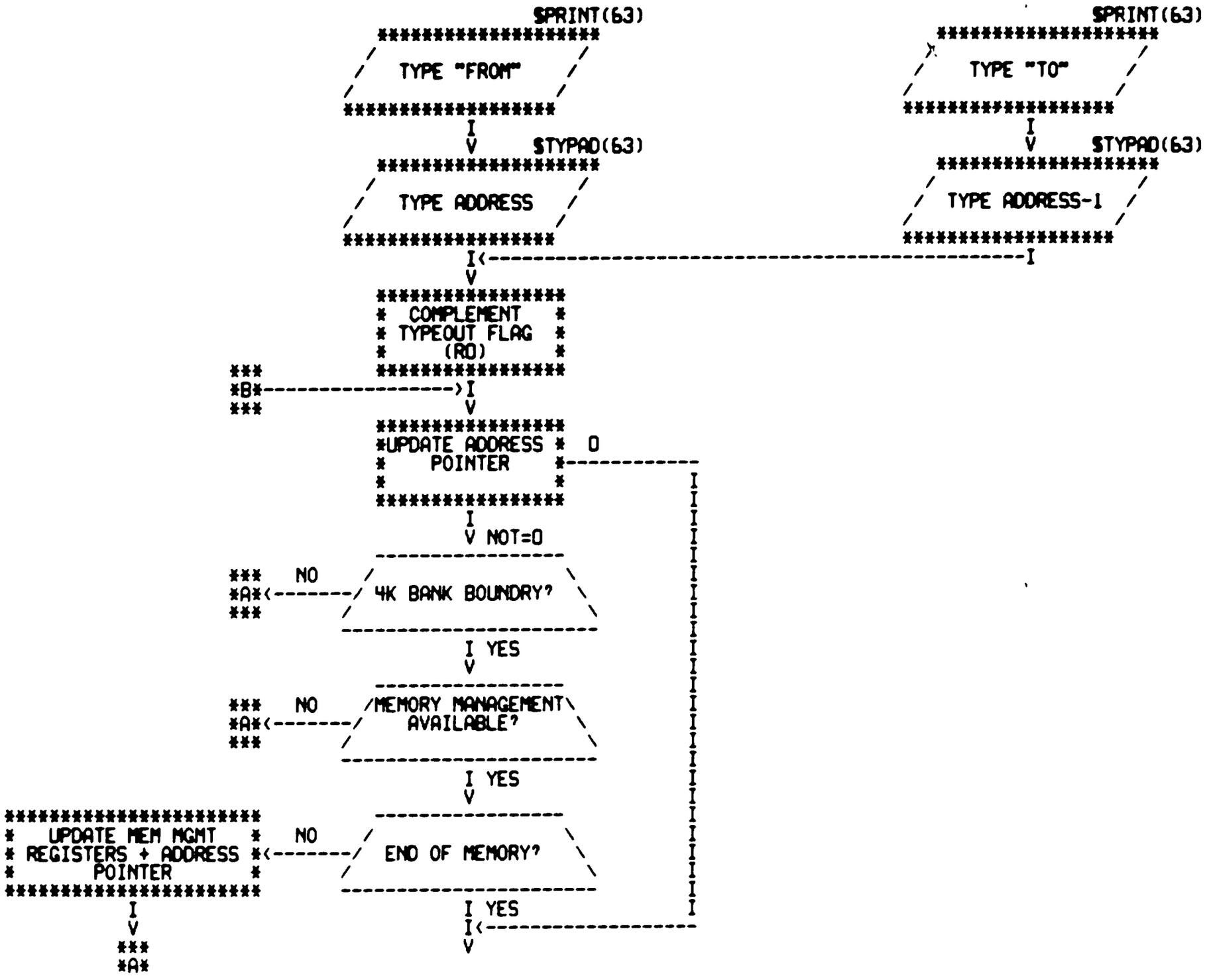
```
                        **************************
                        *  STANDARD 'SYSMAC'     *
                .=1100  *     COMMON TAGS        *
                        *                        *
                        **************************


                        **************************
                        *   APT MAILBOX AND      *
                        *       ETABLE           *
                        *                        *
                        **************************


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

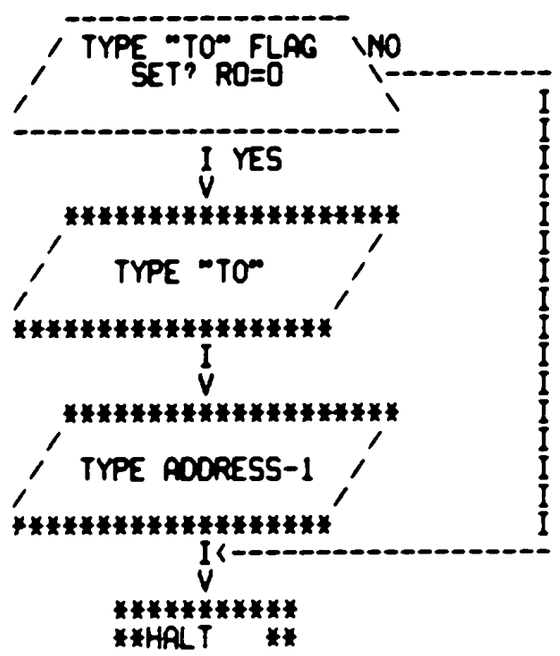

SA=204
**SELECT **
SET FLAG FOR SELECTING PARAMETERS
SA=200
**START **
CLEAR FLAG FOR SELECTING PARAMETERS
STARTA
CLEAR COMMON TAGS
INIT SP INIT PF VECTOR
SET UP SOFTWARE
SWITCH REGISTER IF NO HARDWARE SWR
TYPE PROGRAM TITLE
(ONCE ONLY)
HAS PROGRAM RELOCATED?
YES
RESTAR(02)
NO
SAVE LOADERS

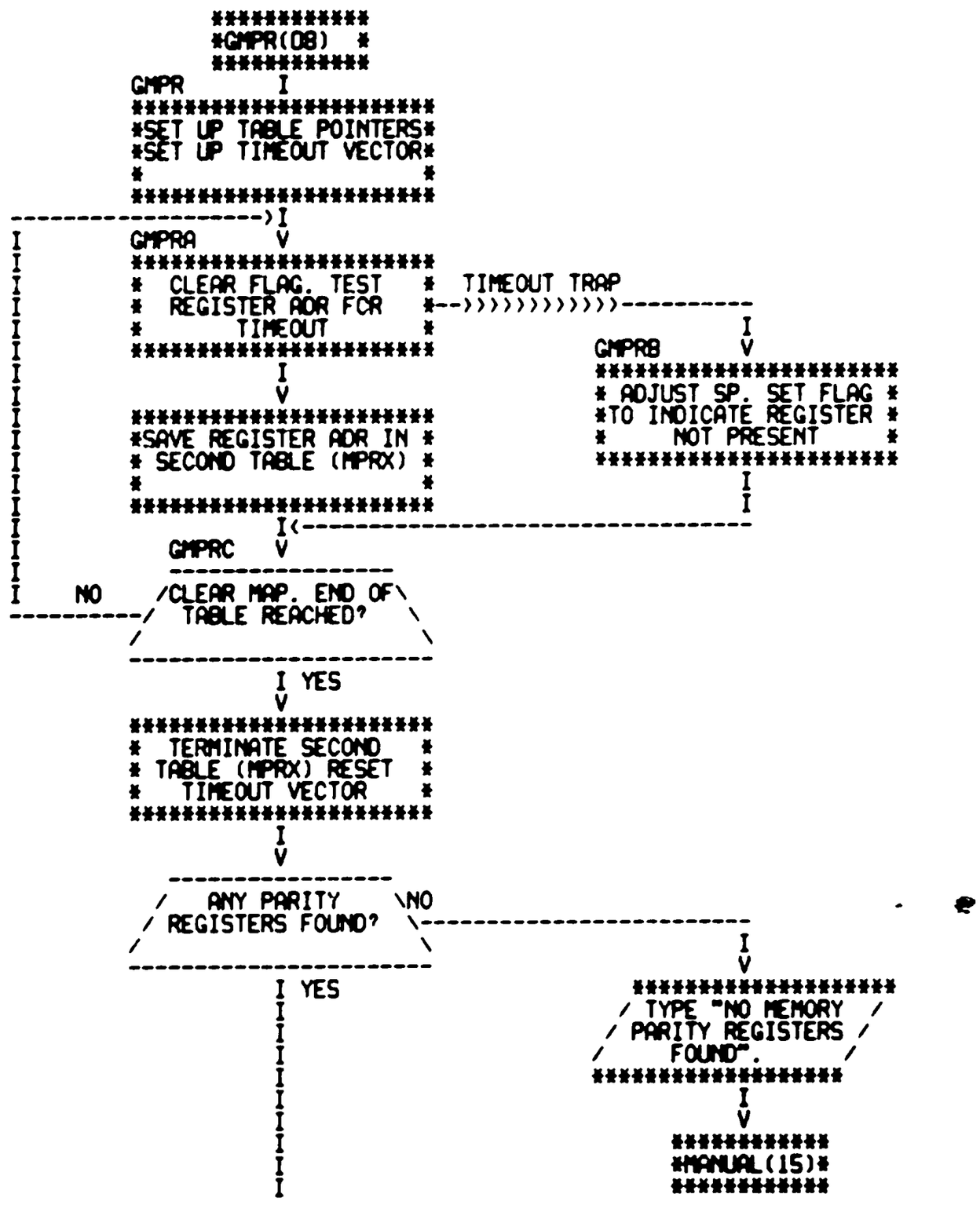

/ CLEAR 'MMAVA', \YES
/ MEMORY MANAGEMENT \
/ EXITS AND DESIRED? \
NO
*SET UP MEM MGMT. SET *
* 'MMAVA' FLAG TYPE *
* 'KT11 AVAILABLE' *
*TURN OFF CACHE *
* IF 11/60 *

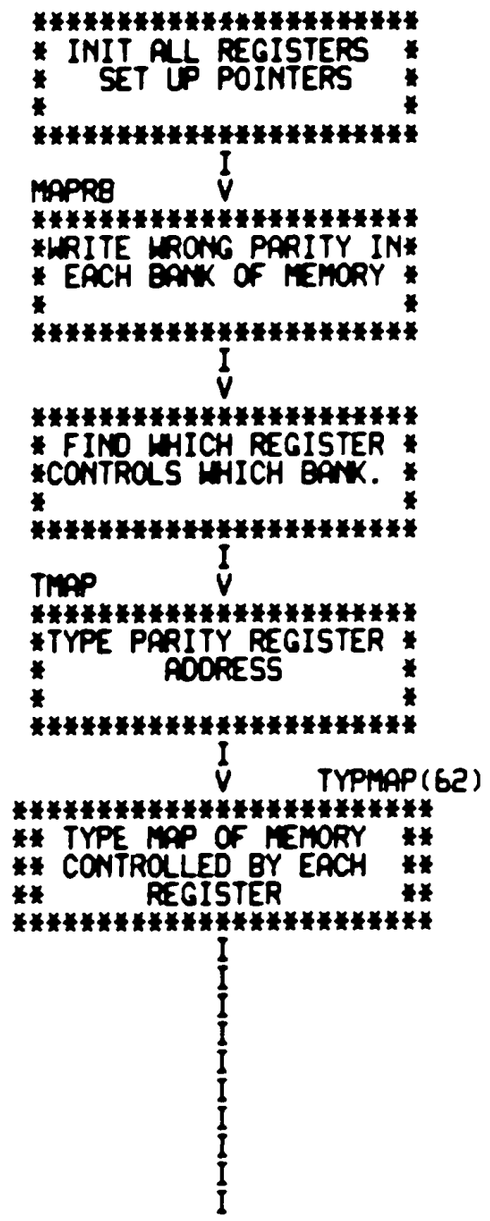

MAPMEM
RESET SP SET UP POINTERS TO MEMORY MAP.
TYPE "MEMORY MAP:"
READ AND WRITE MEMORY LOCATION (AUTO-INC POINTER)
TIMEOUT TRAP
ADJUST SP UPDATE ADP POINTER TO START OF NEXT 4K.
END OF A 4K BANK OF MEMORY REACHED?
NO
YES
SET FLAG IN MAP TO INDICATE FULL 4K. SAVE ADR POINTERS
UPDATE POINTERS TO NEXT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

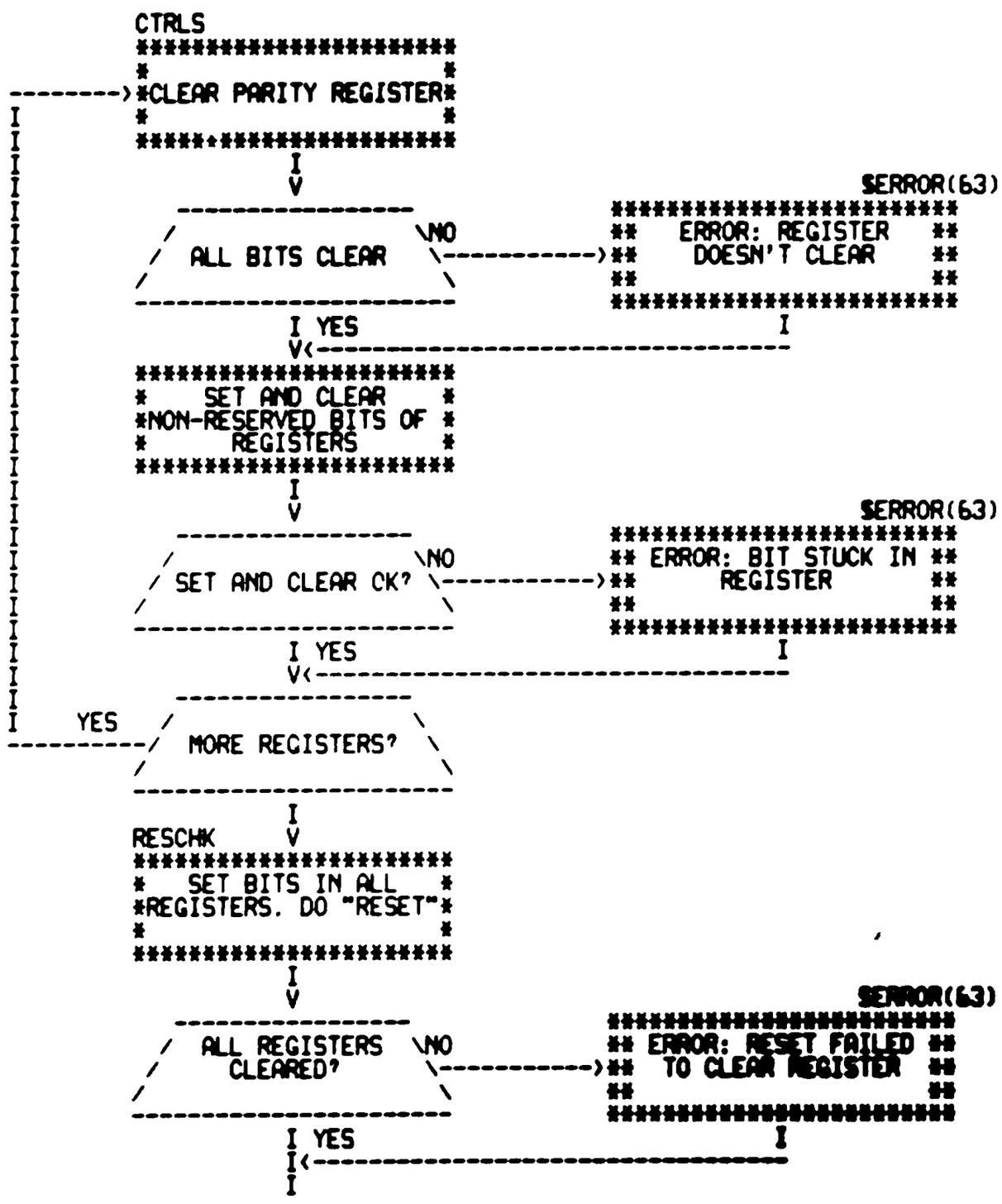

SA=220
**TIMOUT **
MEMORY MANAGEMENT?
NO
I YES
MMINIT(55)
** SET UP MEMORY **
**MANAGEMENT REGISTERS **
$PRINT(63)
TYPE "BYTE MEMORY MAP"
INIT TYPEOUT FLAG
INIT ADDRESS POINTER
SET TIMEOUT VECTOR
#A
DO A DATI; THEN DATI,
DATO; THEN DATI,
DATIP, DATOB
TIMEOUT TRAP
ADJUST SP
TYPE "FROM" FLAG SET? (R0=-1)
NO
TYPE "TO" FLAG SET? (R0=0?)
YES

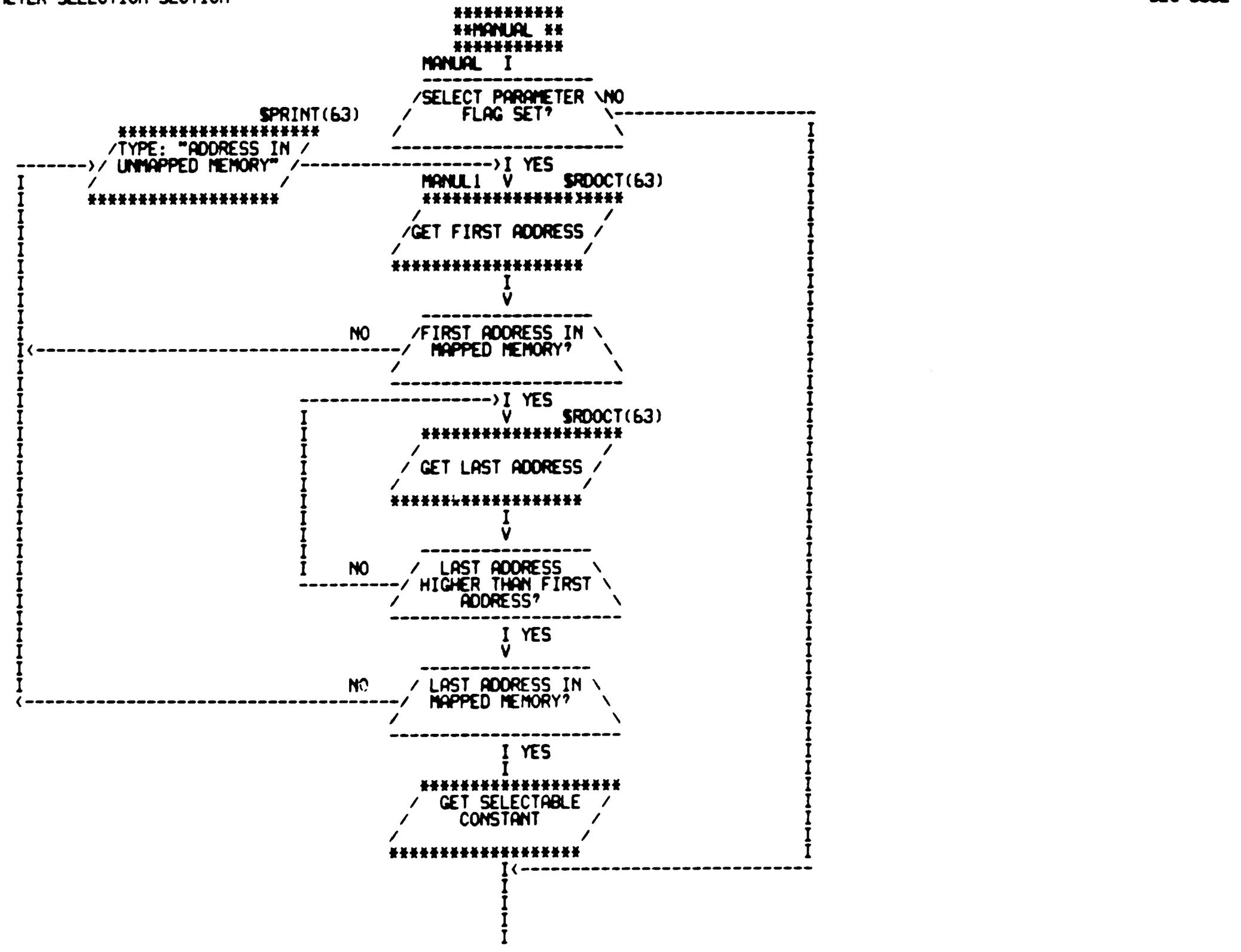

SPRINT(63)
TYPE "FROM"
STYPAD(63)
TYPE ADDRESS
SPRINT(63)
TYPE "TO"
STYPAD(63)
TYPE ADDRESS-1
COMPLEMENT TYPEOUT FLAG (R0)
B
UPDATE ADDRESS POINTER
0
NOT=0
NO
A
4K BANK BOUNDRY?
YES
NO
A
MEMORY MANAGEMENT AVAILABLE?
YES
UPDATE MEM MGMT REGISTERS + ADDRESS POINTER
NO
END OF MEMORY?
YES
A

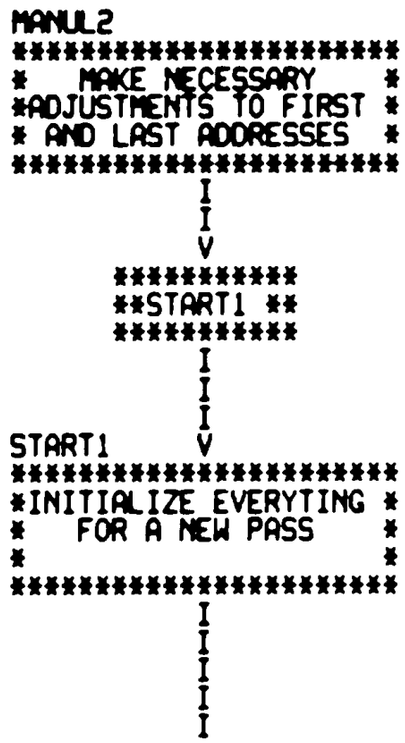

TYPE "TO" FLAG SET? R0=0
NO
YES
TYPE "TO"
TYPE ADDRESS-1
**HALT **

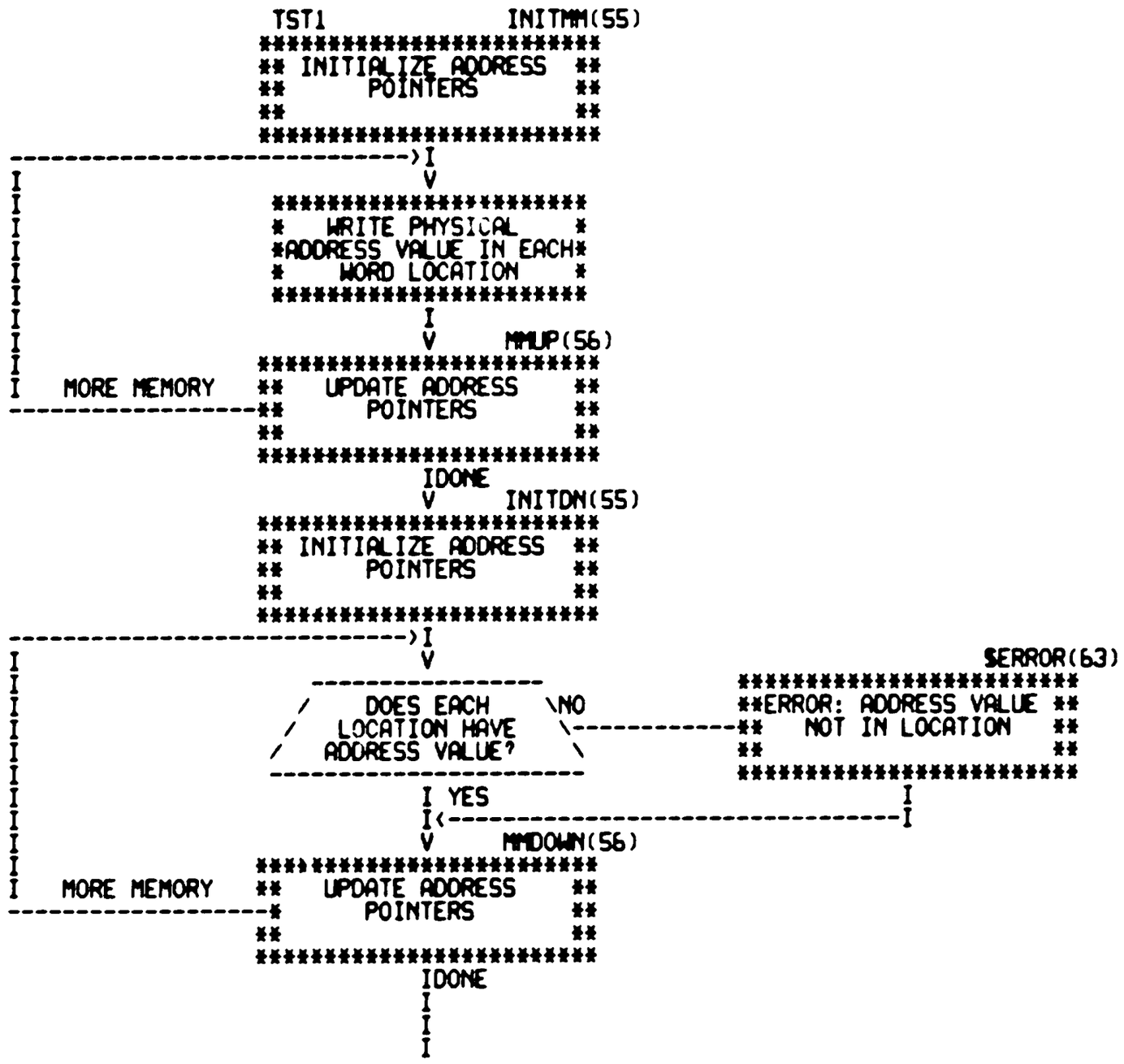

GMPR(08)
GMPR
SET UP TABLE POINTERS
SET UP TIMEOUT VECTOR
GMPRA
* CLEAR FLAG. TEST REGISTER ADR FOR TIMEOUT *
TIMEOUT TRAP
GMPRB
* ADJUST SP. SET FLAG TO INDICATE REGISTER NOT PRESENT *
SAVE REGISTER ADR IN SECOND TABLE (MPRX)
GMPRC
NO
CLEAR MAP. END OF TABLE REACHED?
YES
* TERMINATE SECOND TABLE (MPRX) RESET TIMEOUT VECTOR *
ANY PARITY REGISTERS FOUND?
NO
YES
TYPE "NO MEMORY PARITY REGISTERS FOUND".
MANUAL(15)

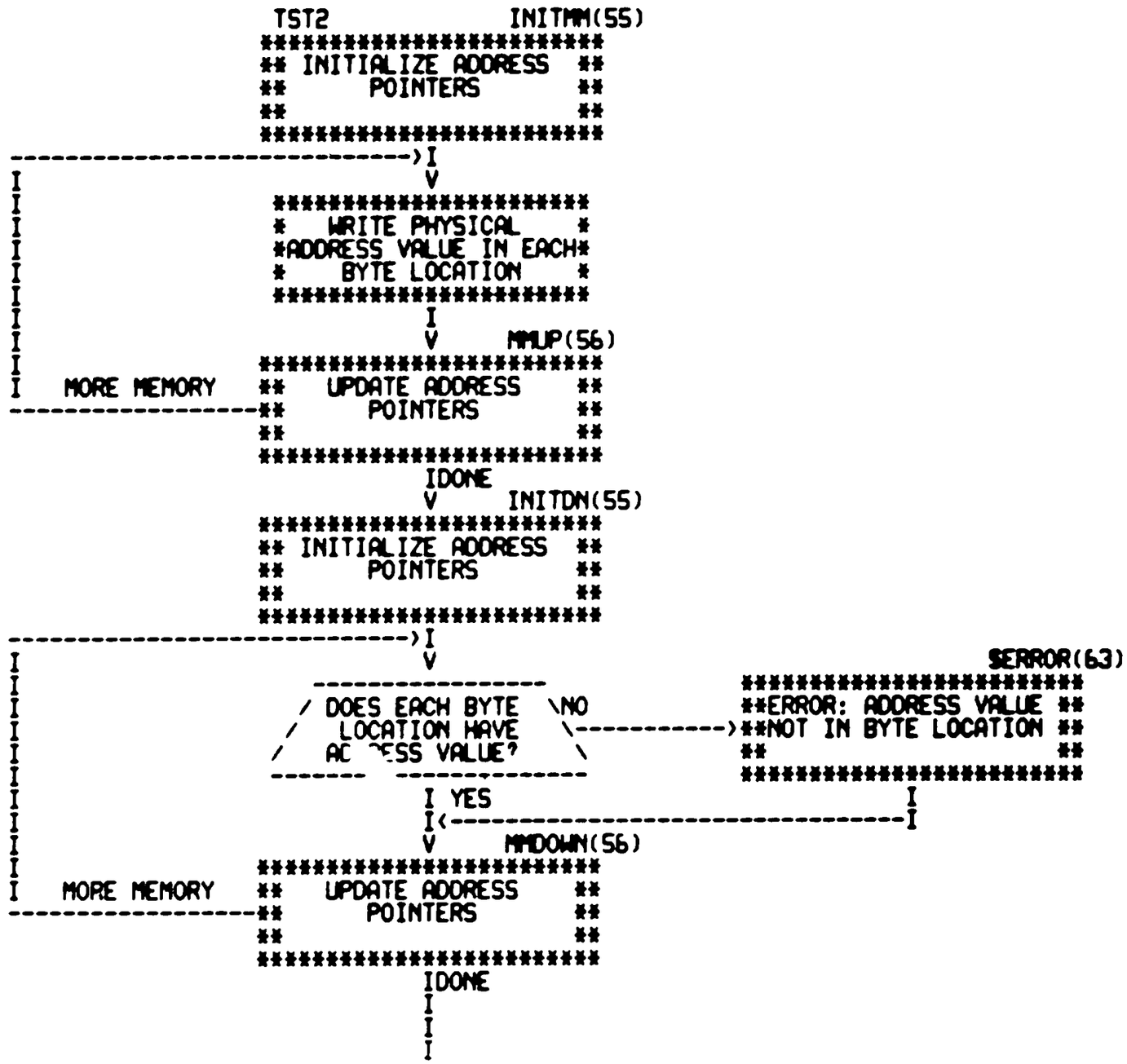

INIT ALL REGISTERS
SET UP POINTERS
MAPRB
WRITE WRONG PARITY IN
EACH BANK OF MEMORY
FIND WHICH REGISTER
CONTROLS WHICH BANK.
TMAP
TYPE PARITY REGISTER
ADDRESS
TYPMAP(62)
TYPE MAP OF MEMORY
CONTROLLED BY EACH
REGISTER

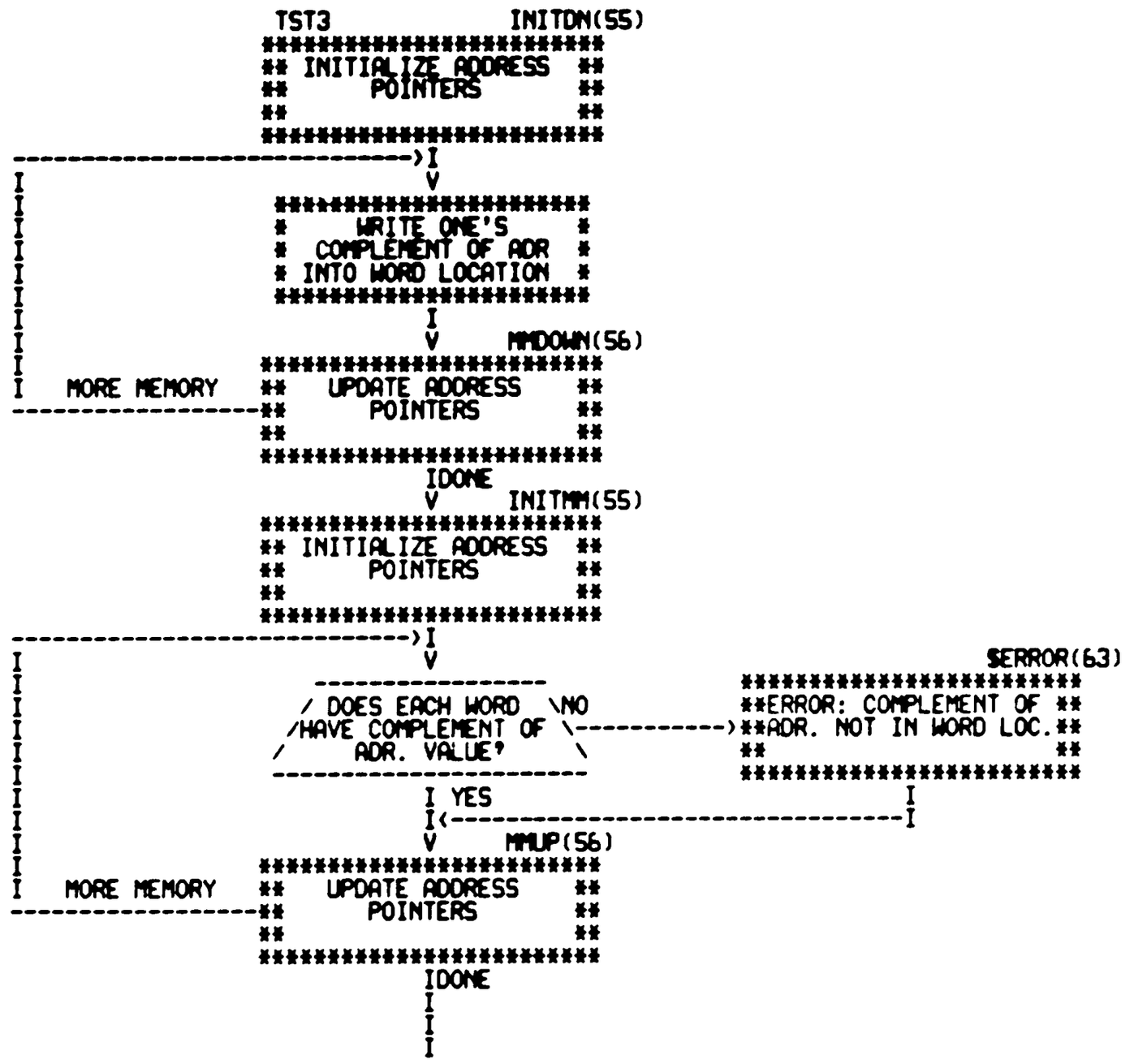

CTRLS
CLEAR PARITY REGISTER
ALL BITS CLEAR
NO
YES
SERROR(63)
** ERROR: REGISTER DOESN'T CLEAR **
SET AND CLEAR OF NON-RESERVED BITS OF REGISTERS
SET AND CLEAR CK?
NO
YES
SERROR(63)
** ERROR: BIT STUCK IN REGISTER **
MORE REGISTERS?
YES
RESCHK
SET BITS IN ALL REGISTERS. DO "RESET"
ALL REGISTERS CLEARED?
NO
YES
SERROR(63)
** ERROR: RESET FAILED TO CLEAR REGISTER **

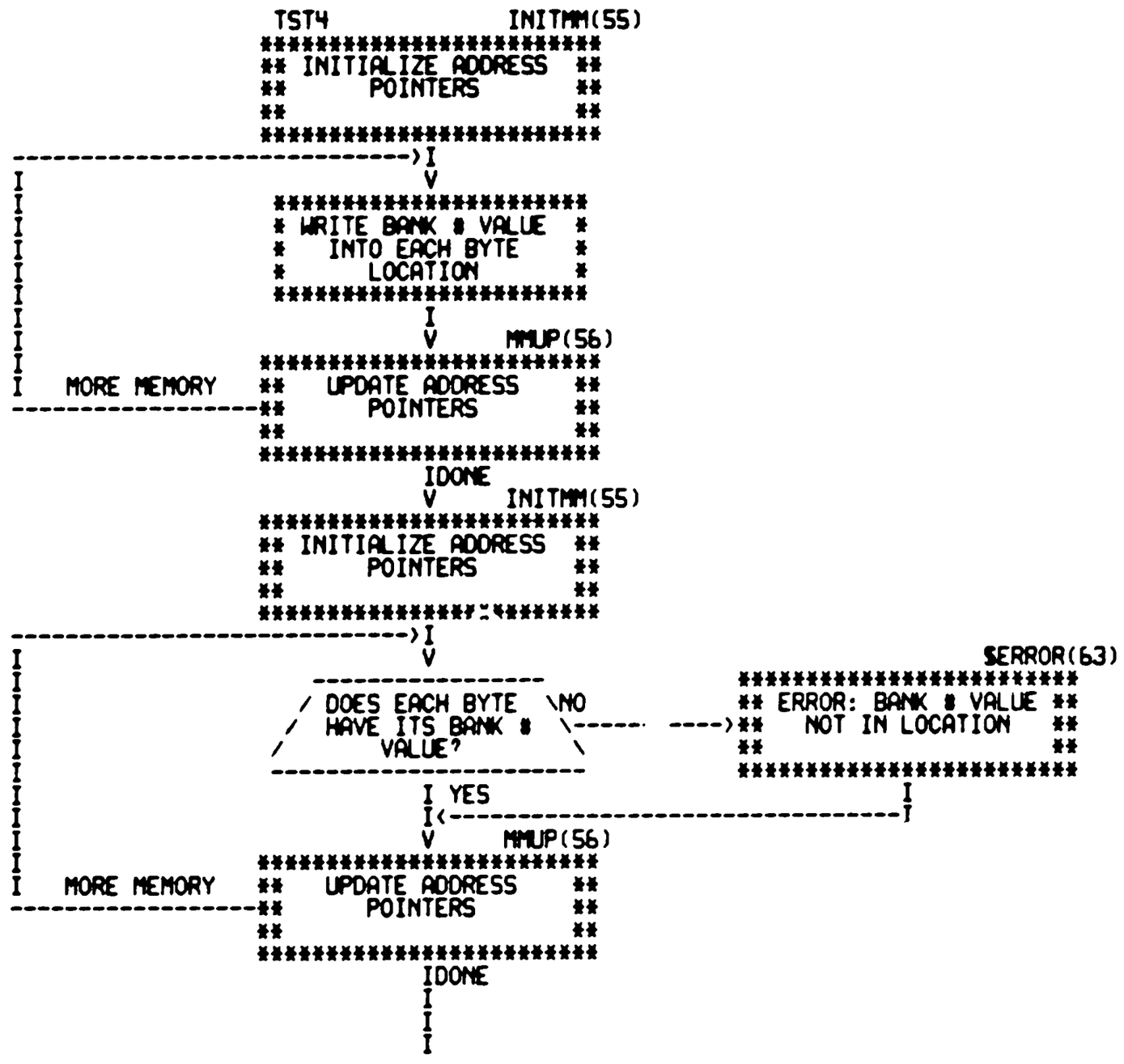

****
**MANUAL **
****
MANUAL I
SELECT PARAMETER
FLAG SET?
NO
$PRINT(63)
****
TYPE: "ADDRESS IN
UNMAPPED MEMORY"
****
I YES
MANUL1 V
$RDOCT(63)
****
GET FIRST ADDRESS
****
I
V
FIRST ADDRESS IN
MAPPED MEMORY?
NO
I YES
V
$RDOCT(63)
****
GET LAST ADDRESS
****
I
V
LAST ADDRESS
HIGHER THAN FIRST
ADDRESS?
NO
I YES
V
LAST ADDRESS IN
MAPPED MEMORY?
NO
I YES
I
****
GET SELECTABLE
CONSTANT
****

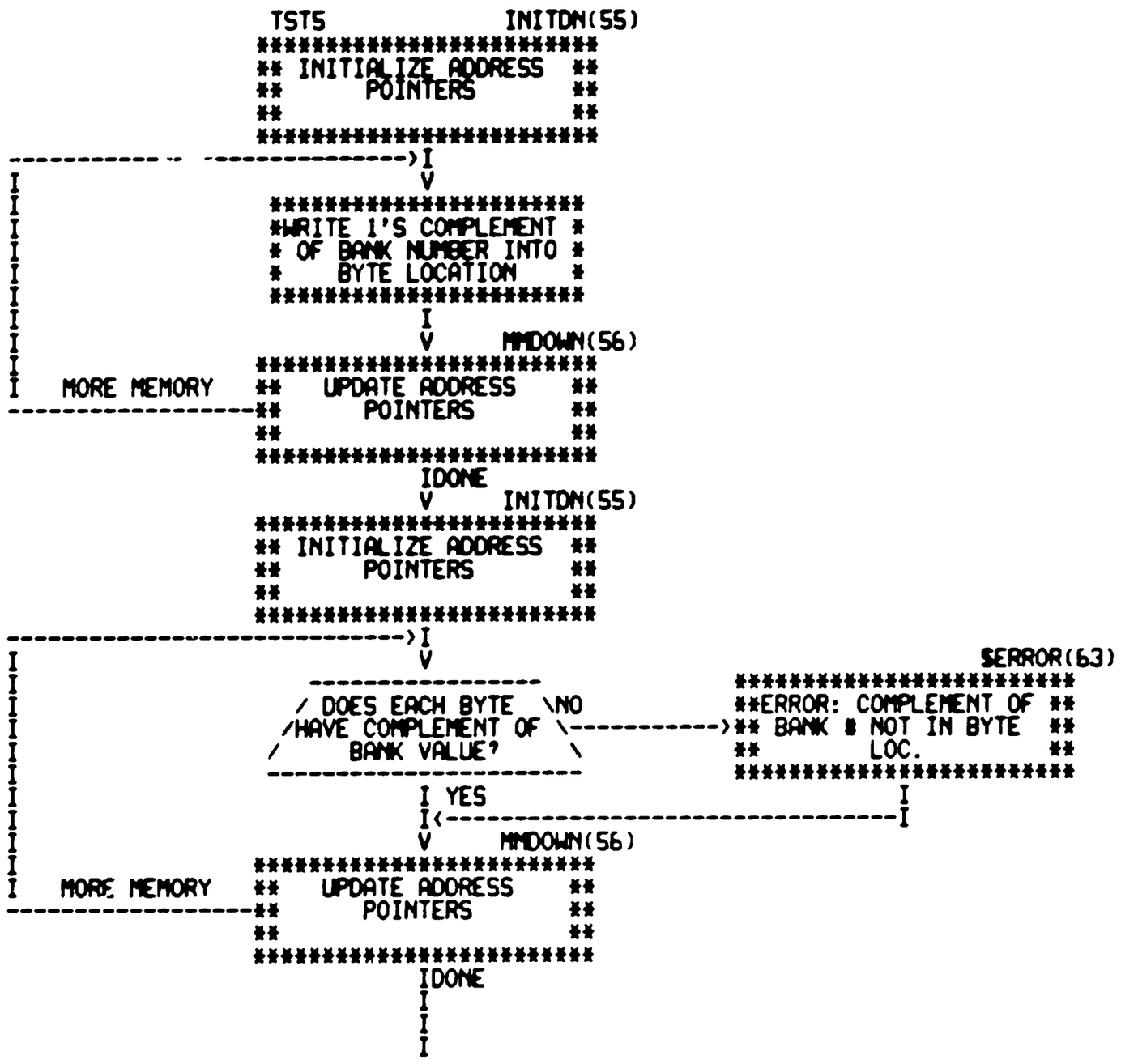

MANUL2
MAKE NECESSARY
ADJUSTMENTS TO FIRST
AND LAST ADDRESSES
START1
START1
INITIALIZE EVERYTING
FOR A NEW PASS

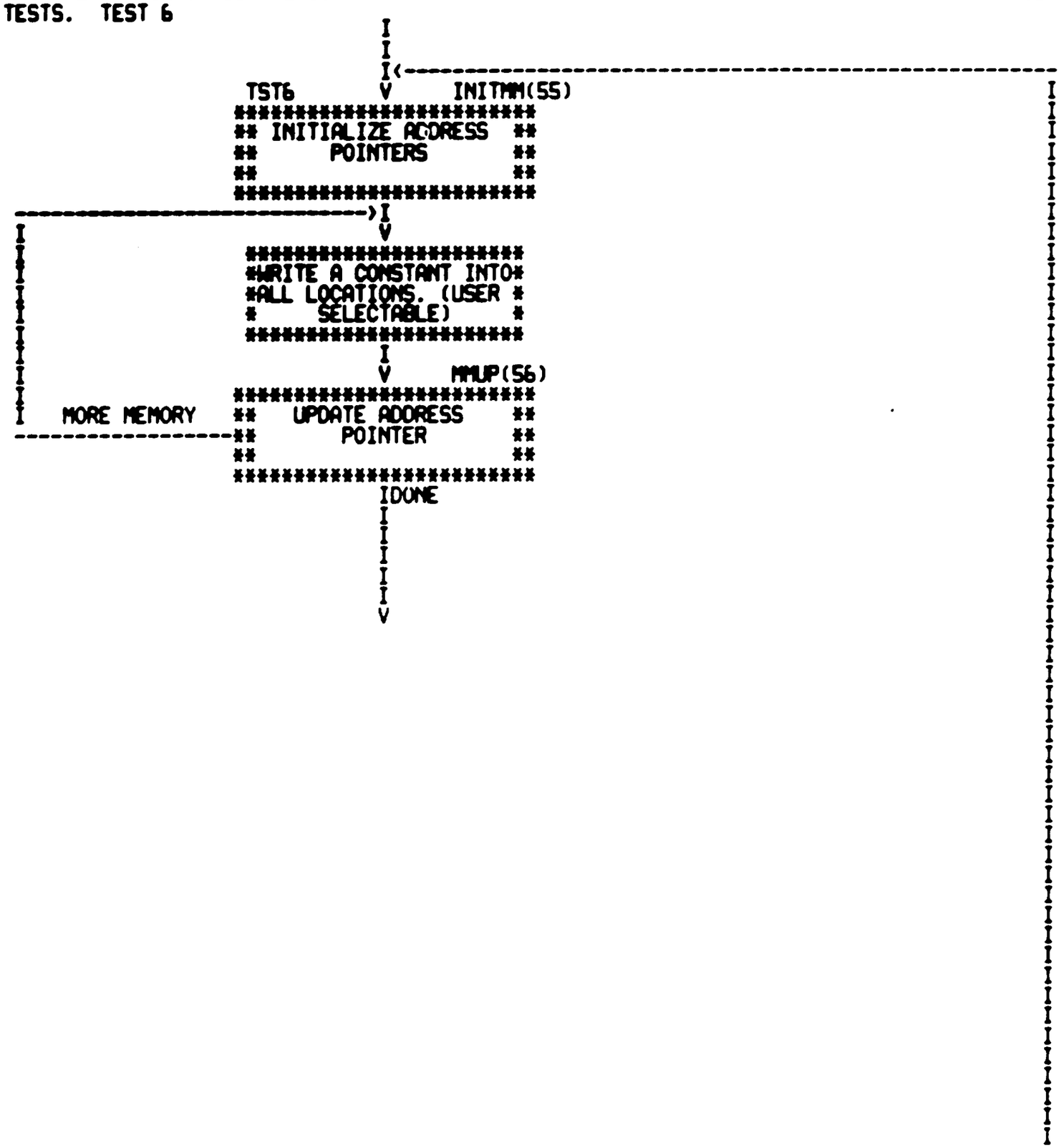

TST1
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE PHYSICAL
ADDRESS VALUE IN EACH
WORD LOCATION
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
DOES EACH
LOCATION HAVE
ADDRESS VALUE?
NO
**ERROR: ADDRESS VALUE **
** NOT IN LOCATION **
YES
MMDOWN(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

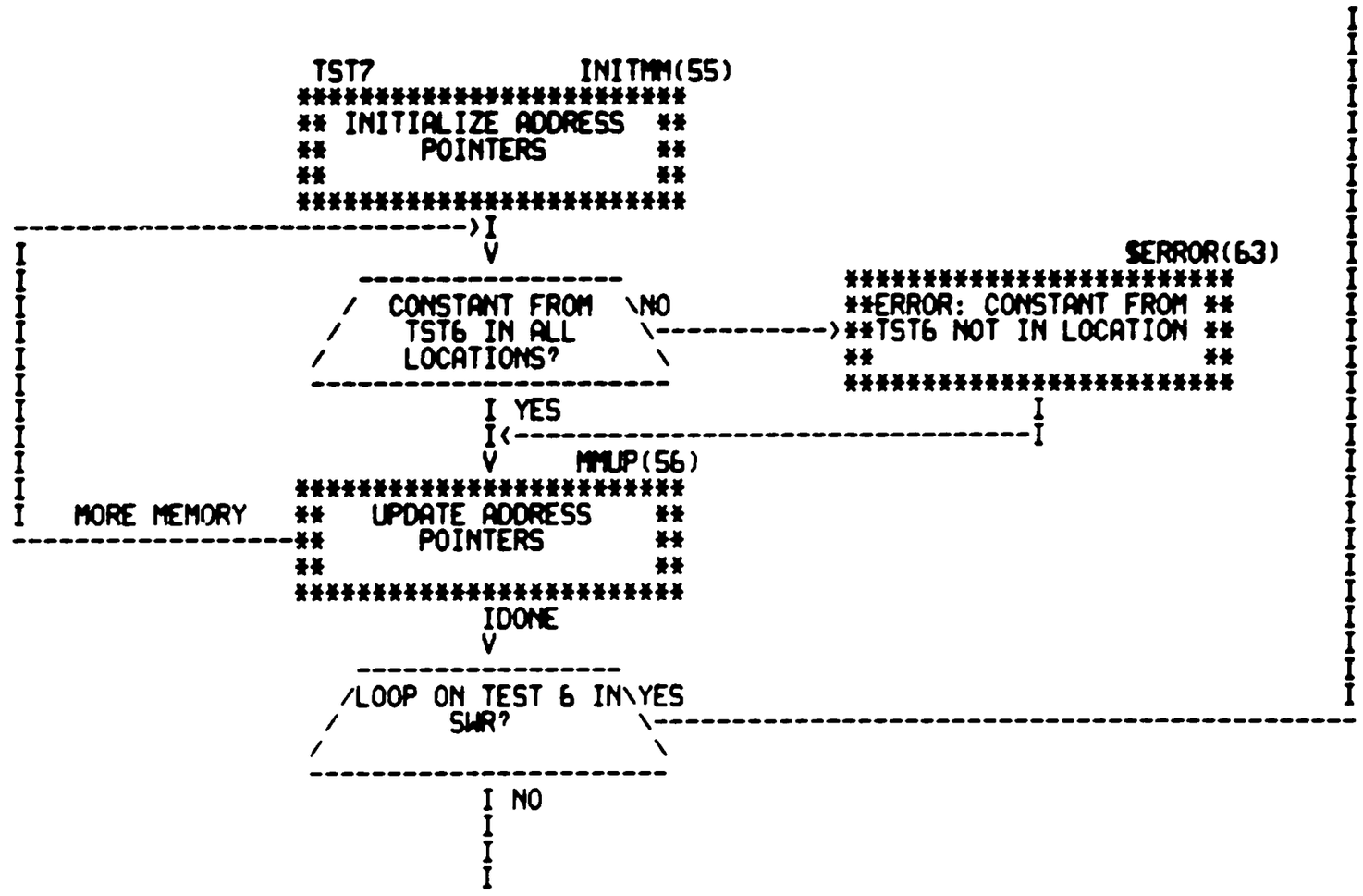

TST2
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
SERROR(63)
DOES EACH BYTE LOCATION HAVE AC ESS VALUE?
NO
**ERROR: ADDRESS VALUE **
**NOT IN BYTE LOCATION **
YES
MMDOWN(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

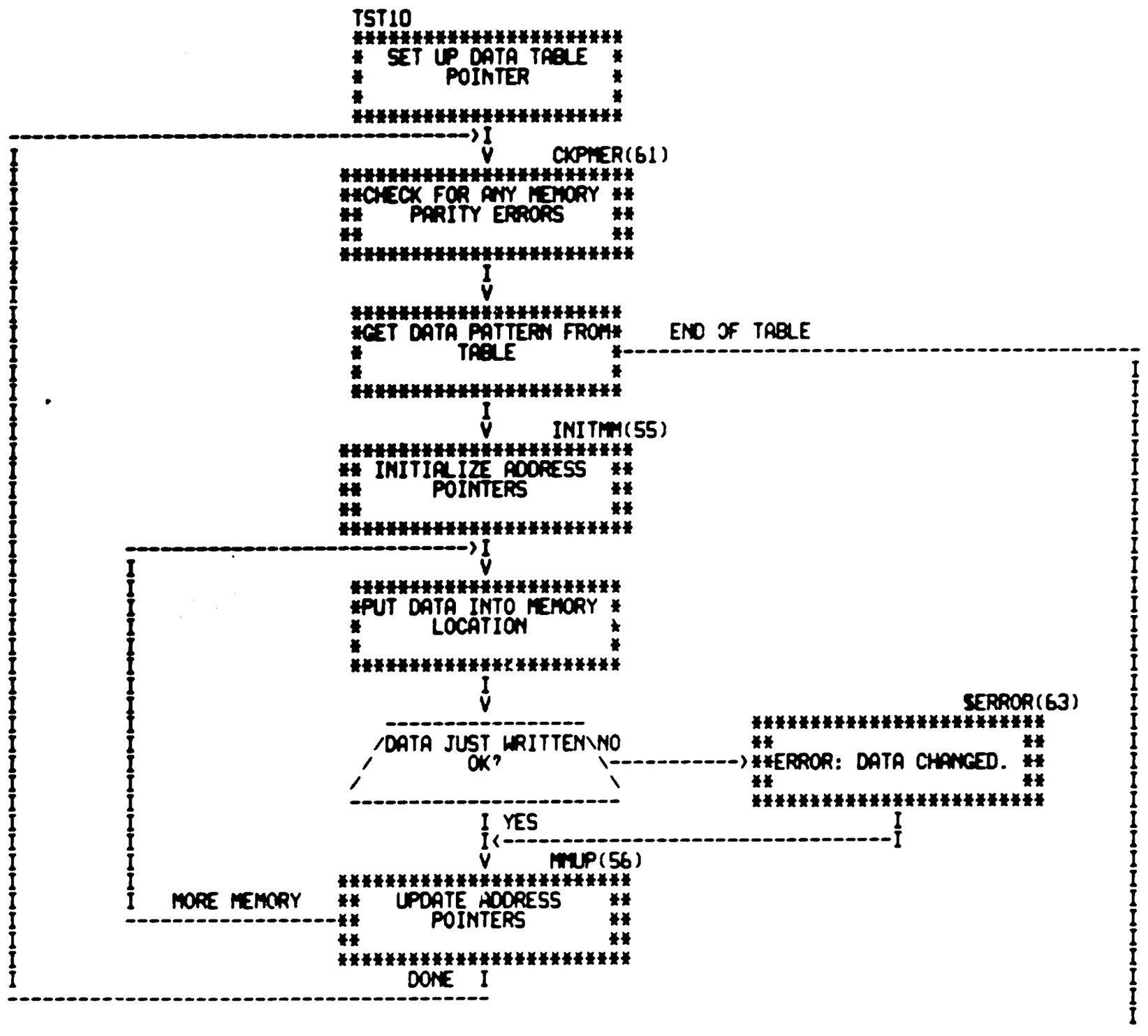

TST3
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE ONE'S
COMPLEMENT OF ADR
INTO WORD LOCATION
MMDOWN(56)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE
INITMM(55)
DOES EACH WORD HAVE COMPLEMENT OF ADR. VALUE?
NO
$ERROR(63)
**ERROR: COMPLEMENT OF **
ADR. NOT IN WORD LOC.
YES
MMUP(56)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE

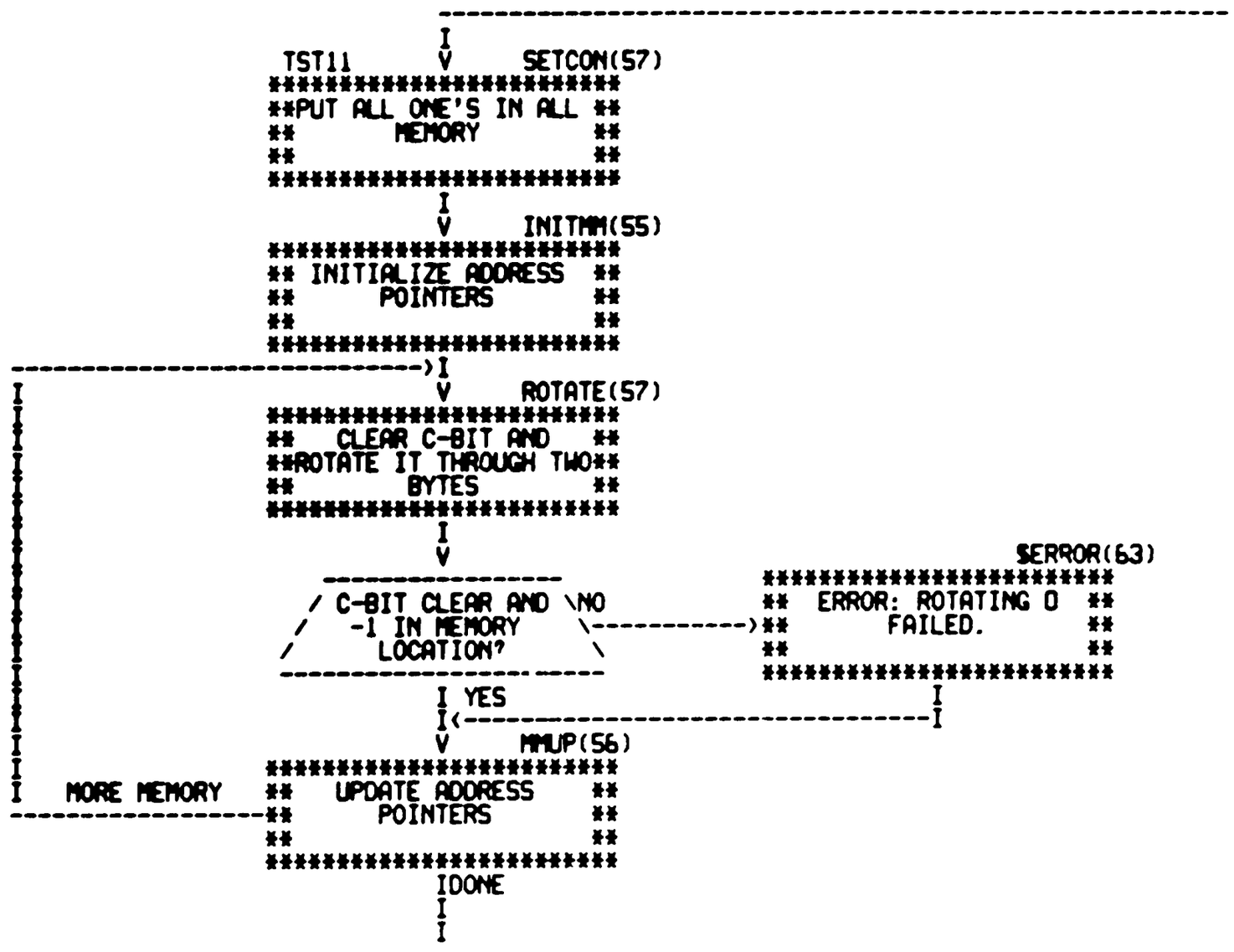

TST4
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE BANK # VALUE
INTO EACH BYTE
LOCATION
MMUP(56)
UPDATE ADDRESS
POINTERS
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
MORE MEMORY
DOES EACH BYTE HAVE ITS BANK # VALUE?
NO
YES
$ERROR(63)
** ERROR: BANK # VALUE **
** NOT IN LOCATION **
MMUP(56)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE

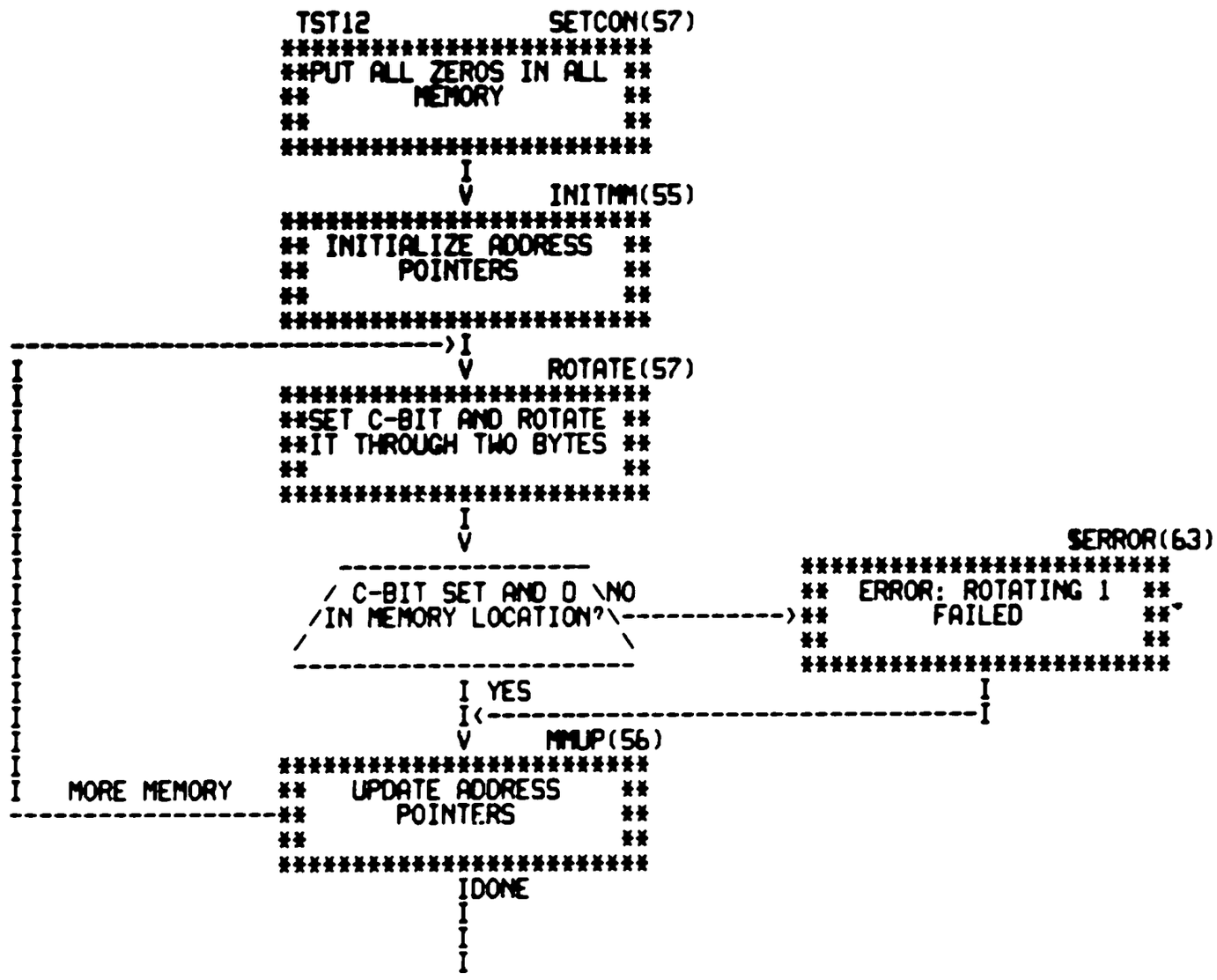

TST5
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE 1'S COMPLEMENT OF BANK NUMBER INTO BYTE LOCATION
MMDOWN(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH BYTE HAVE COMPLEMENT OF BANK VALUE?
NO
YES
$ERROR(63)
**ERROR: COMPLEMENT OF BANK # NOT IN BYTE LOC.
MMDOWN(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

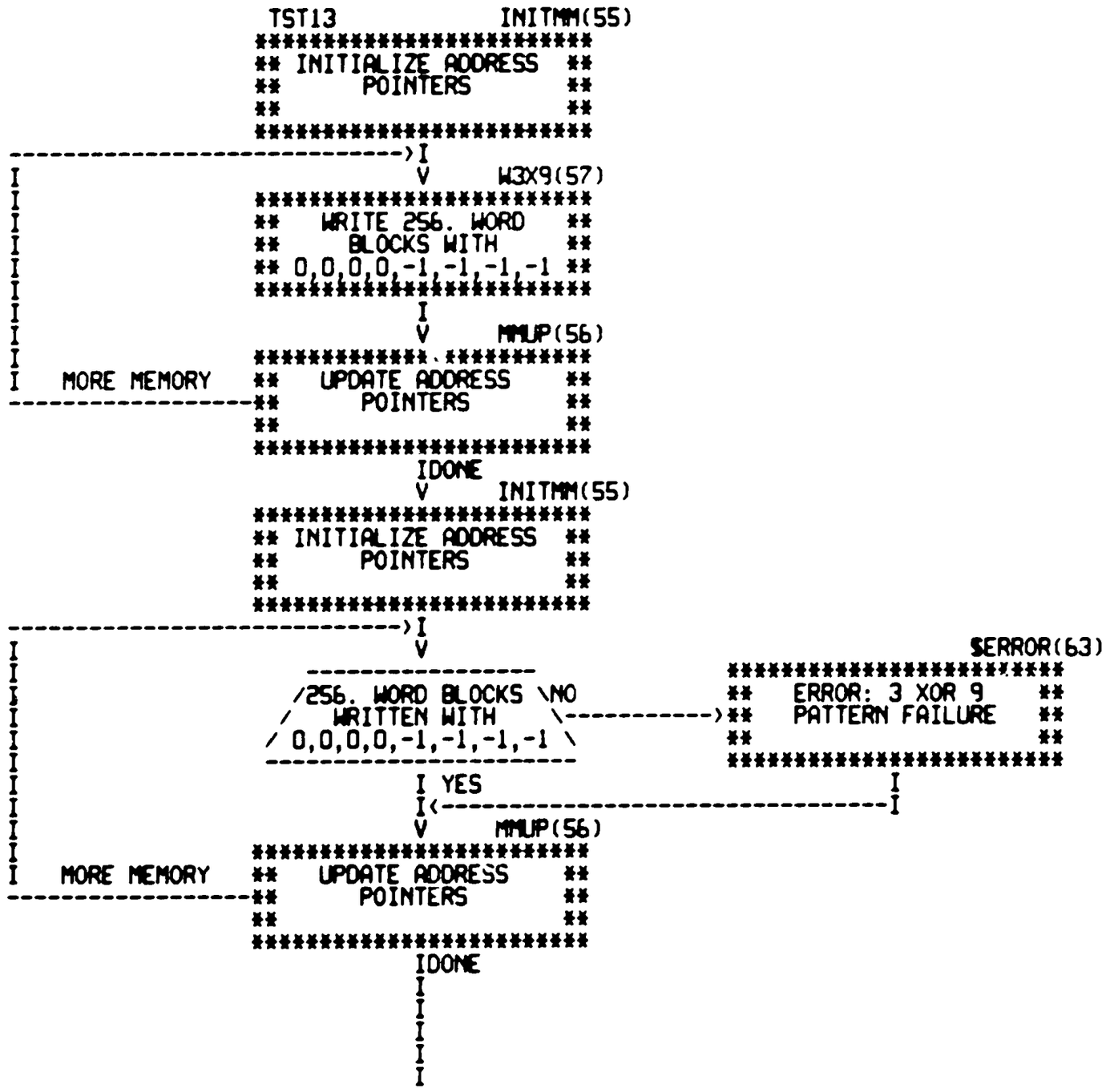

TST6
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE A CONSTANT INTO ALL LOCATIONS. (USER SELECTABLE)
MMUP(56)
** UPDATE ADDRESS **
** POINTER **
MORE MEMORY
IDONE

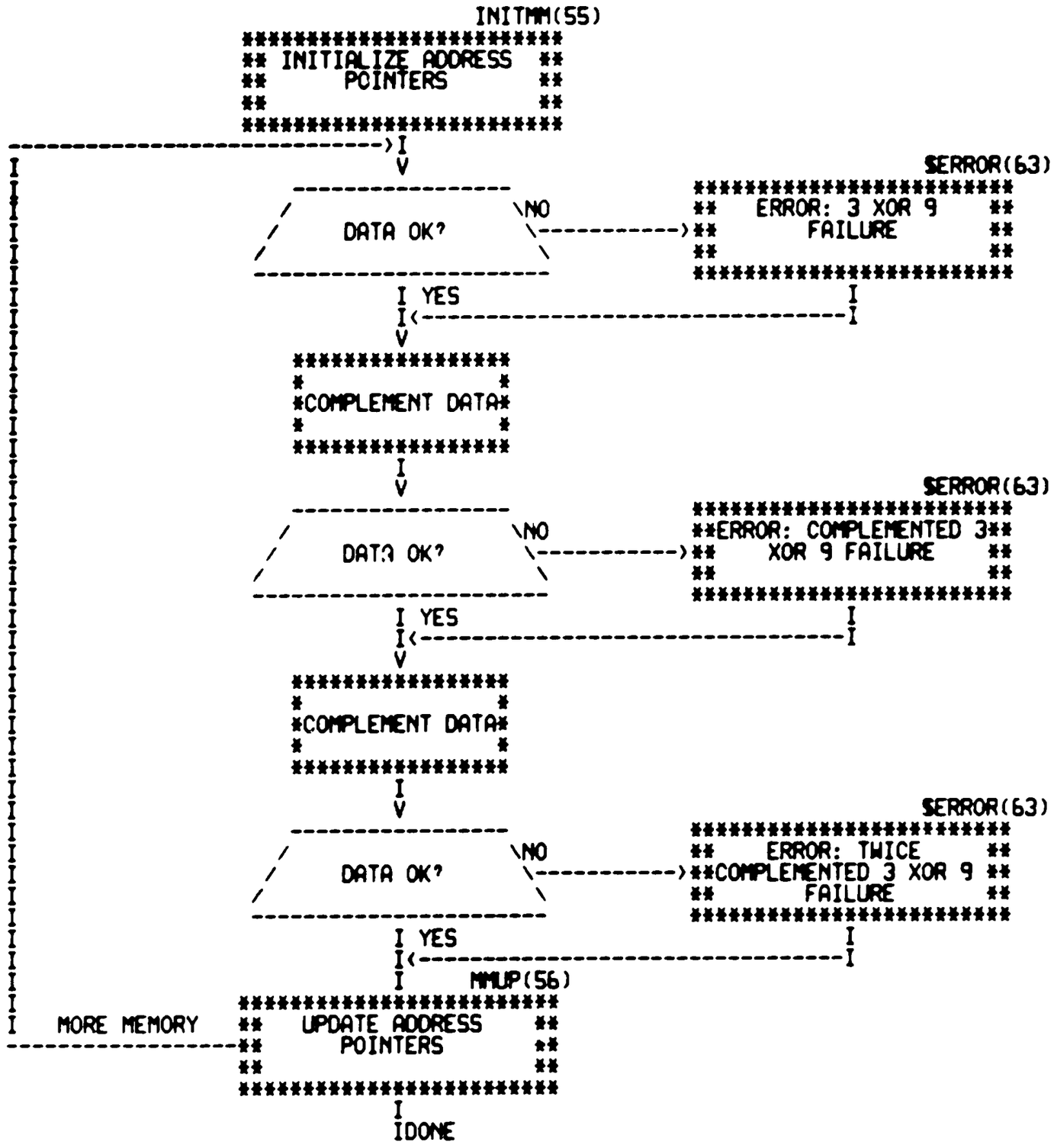

TST7
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
CONSTANT FROM TST6 IN ALL LOCATIONS?
NO
$ERROR(63)
**ERROR: CONSTANT FROM **TST6 NOT IN LOCATION **
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
IDONE
LOOP ON TEST 6 IN SWR?
YES
NO

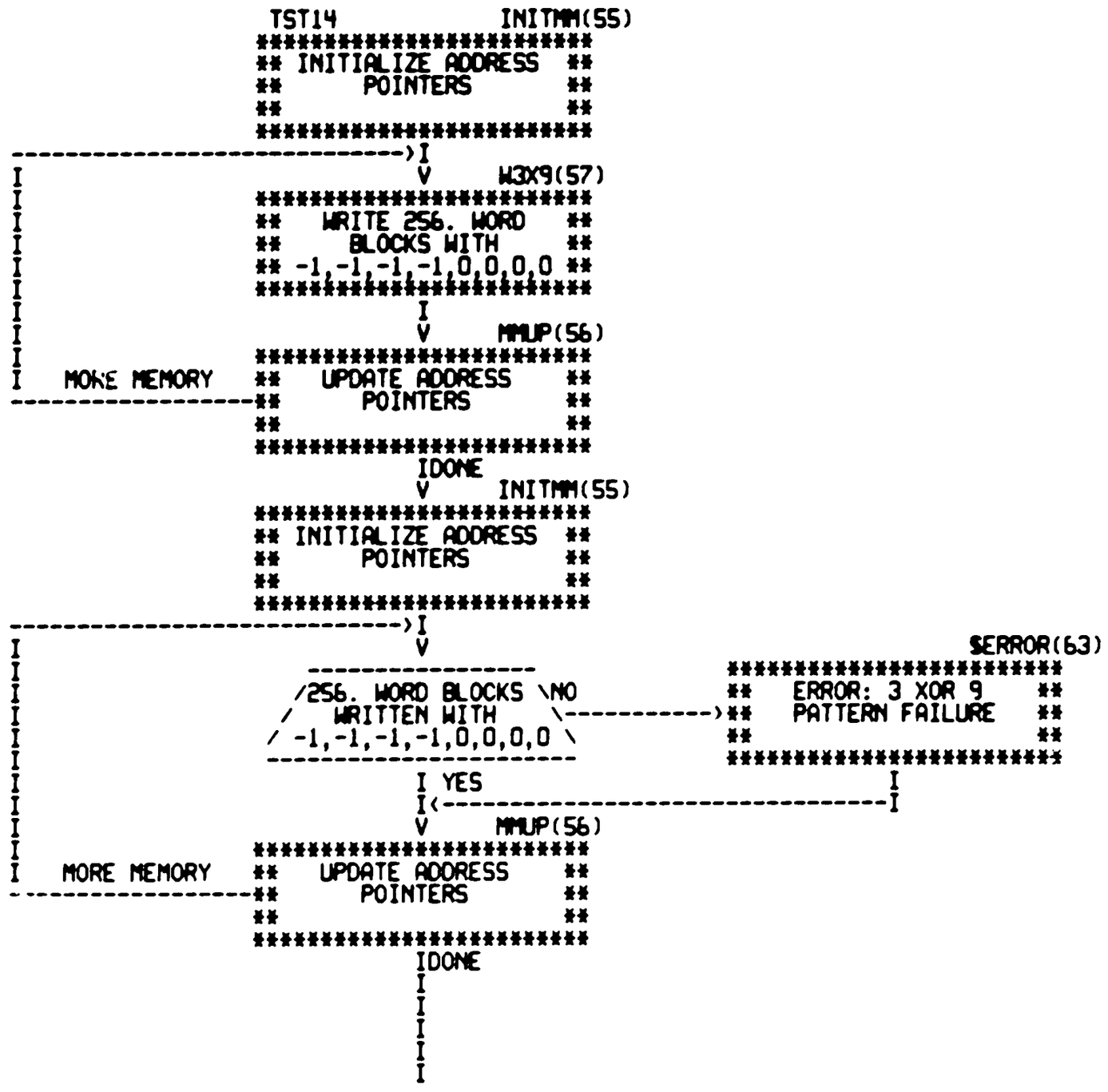

TST10
SET UP DATA TABLE POINTER
CKPMER(61)
CHECK FOR ANY MEMORY PARITY ERRORS
GET DATA PATTERN FROM TABLE
END OF TABLE
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT DATA INTO MEMORY LOCATION
DATA JUST WRITTEN OK?
NO
$ERROR(63)
ERROR: DATA CHANGED.
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

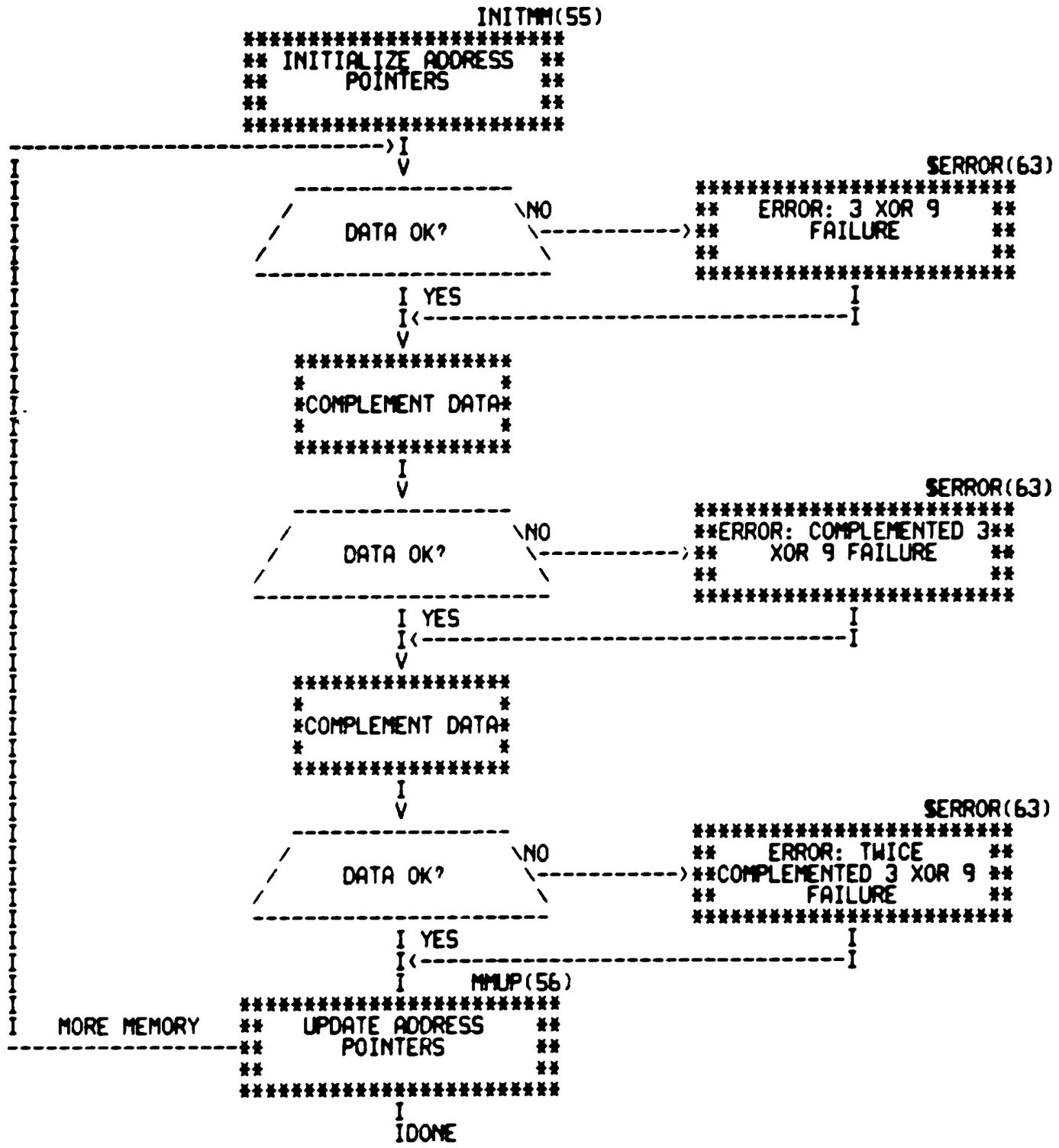

TST11
SETCON(57)
PUT ALL ONE'S IN ALL MEMORY
INITMM(55)
** INITIALIZE ADDRESS POINTERS **
ROTATE(57)
** CLEAR C-BIT AND ROTATE IT THROUGH TWO BYTES **
C-BIT CLEAR AND -1 IN MEMORY LOCATION?
NO
$ERROR(63)
** ERROR: ROTATING 0 FAILED. **
YES
MMUP(56)
** UPDATE ADDRESS POINTERS **
MORE MEMORY
IDONE

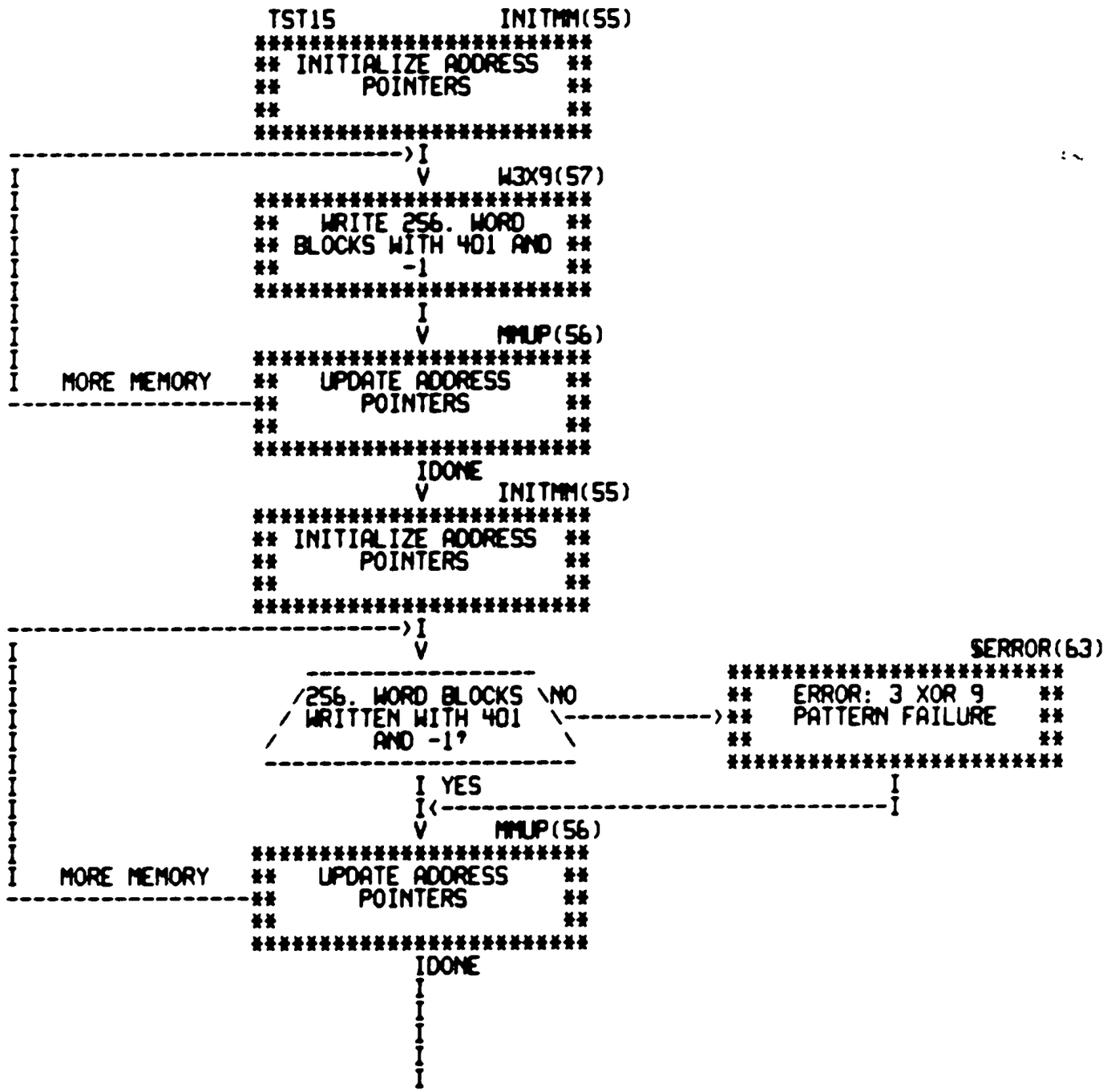

TST12
SETCON(57)
**PUT ALL ZEROS IN ALL MEMORY
INITMM(55)
** INITIALIZE ADDRESS POINTERS
ROTATE(57)
**SET C-BIT AND ROTATE IT THROUGH TWO BYTES
C-BIT SET AND 0 IN MEMORY LOCATION?
NO
$ERROR(63)
** ERROR: ROTATING 1 FAILED
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS POINTERS
IDONE

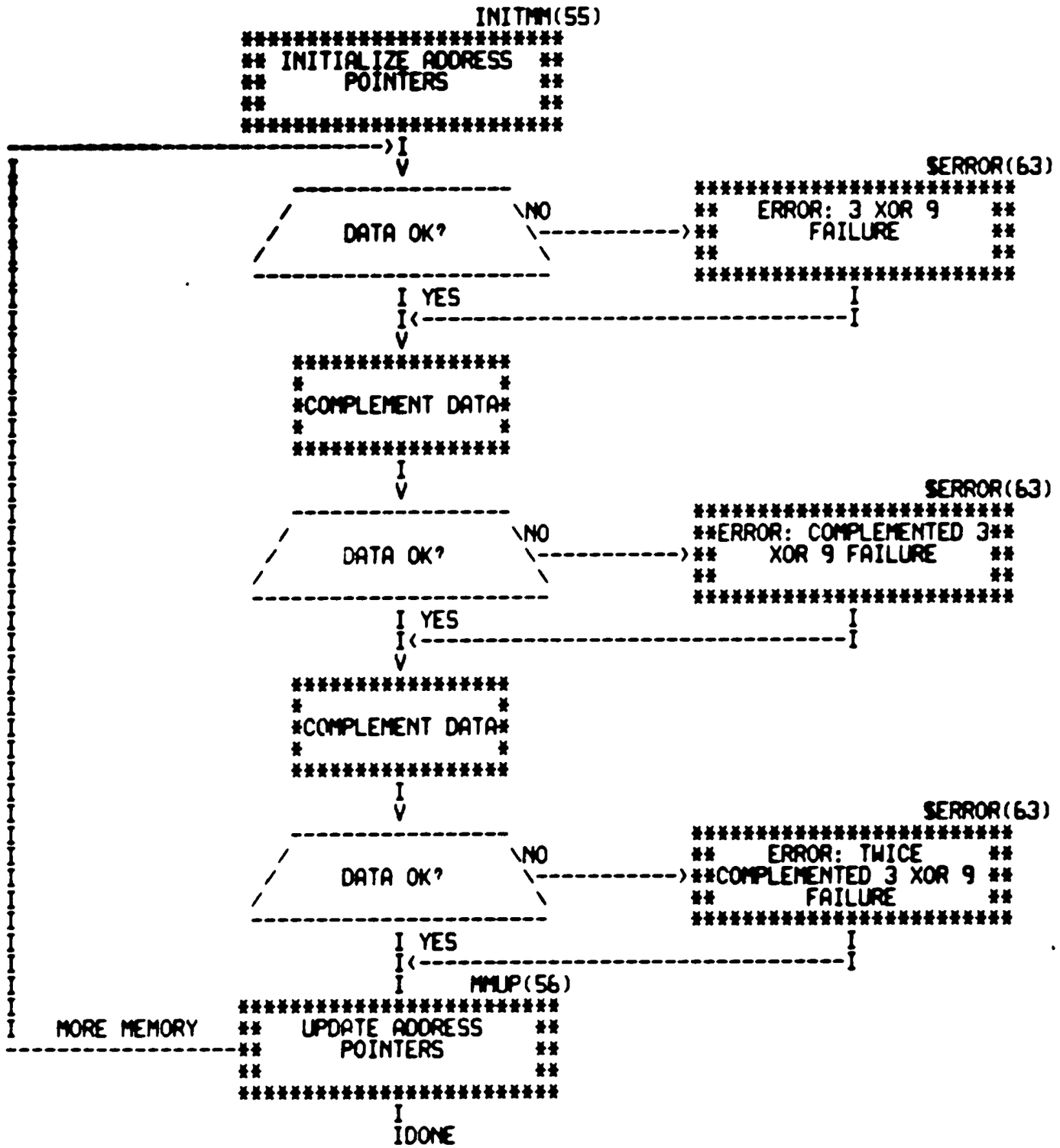

TST13
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH **
** 0,0,0,0,-1,-1,-1,-1 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/ WRITTEN WITH \
/ 0,0,0,0,-1,-1,-1,-1 \
I YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(63)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

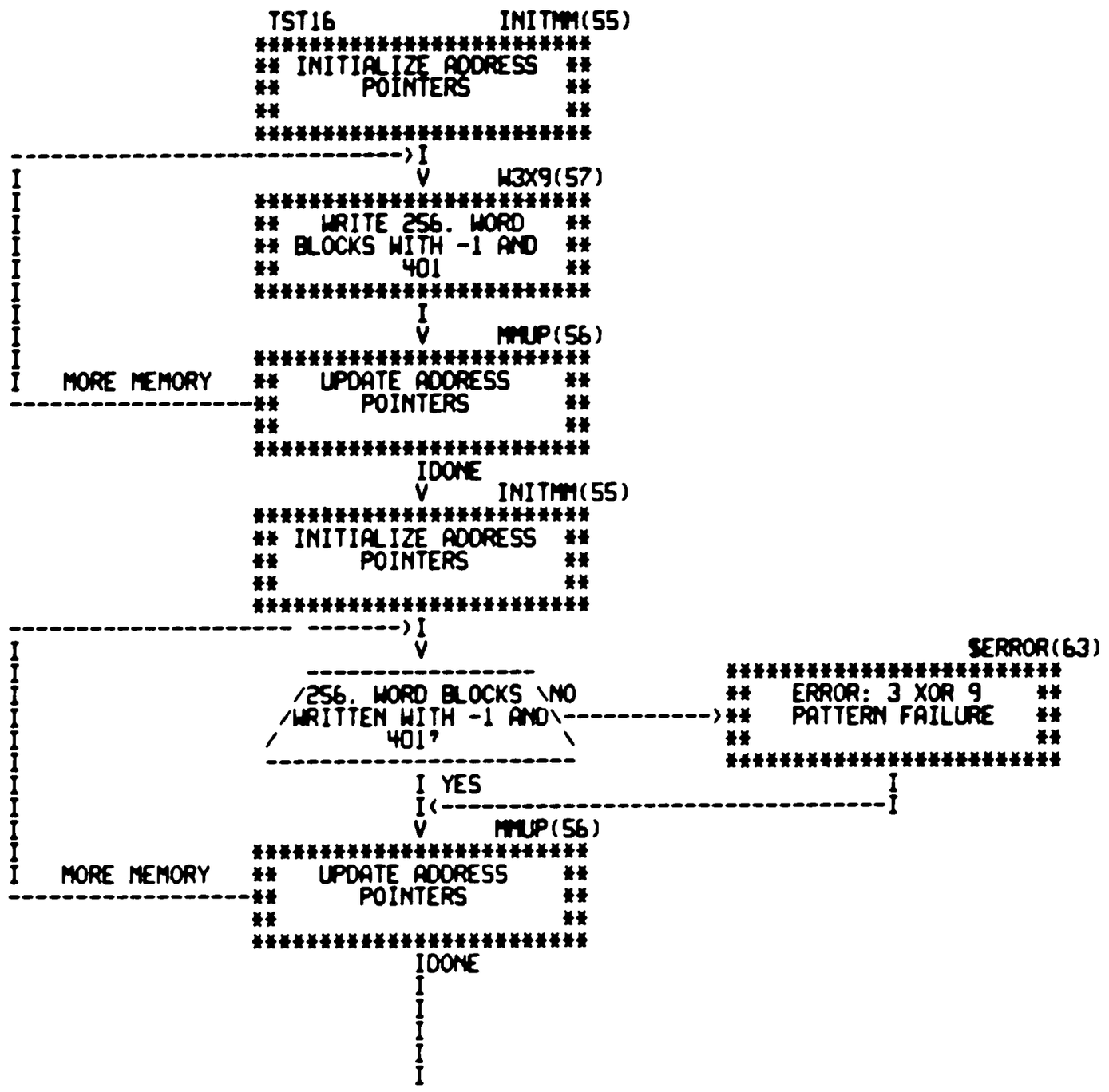

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
YES
$ERROR(63)
ERROR: 3 XOR 9
FAILURE
COMPLEMENT DATA
$ERROR(63)
ERROR: COMPLEMENTED 3
XOR 9 FAILURE
COMPLEMENT DATA
$ERROR(63)
ERROR: TWICE
COMPLEMENTED 3 XOR 9
FAILURE
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
DONE

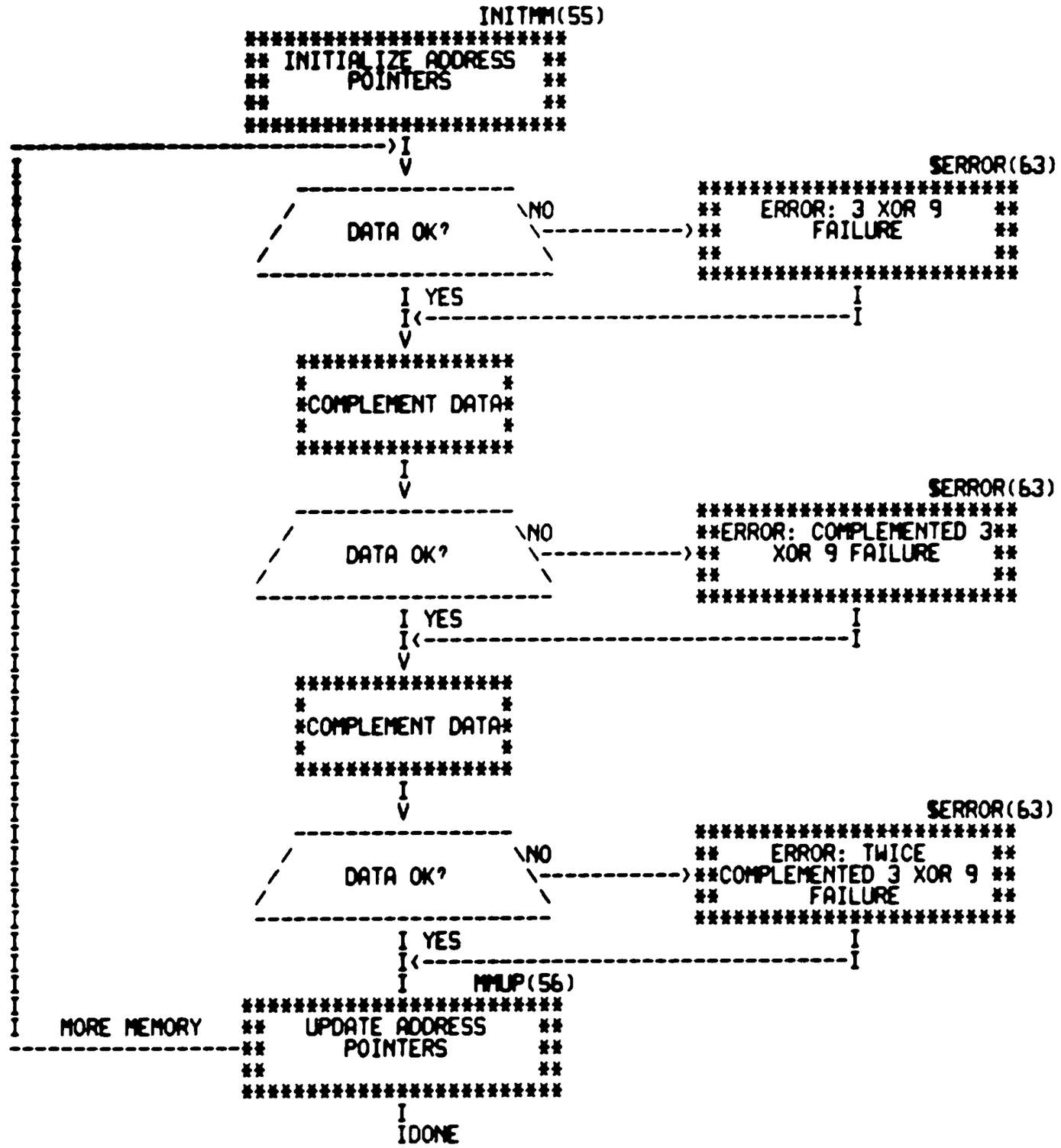

TST14
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH **
** -1,-1,-1,-1,0,0,0,0 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
/256. WORD BLOCKS \NO
/ WRITTEN WITH \
/ -1,-1,-1,-1,0,0,0,0 \
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

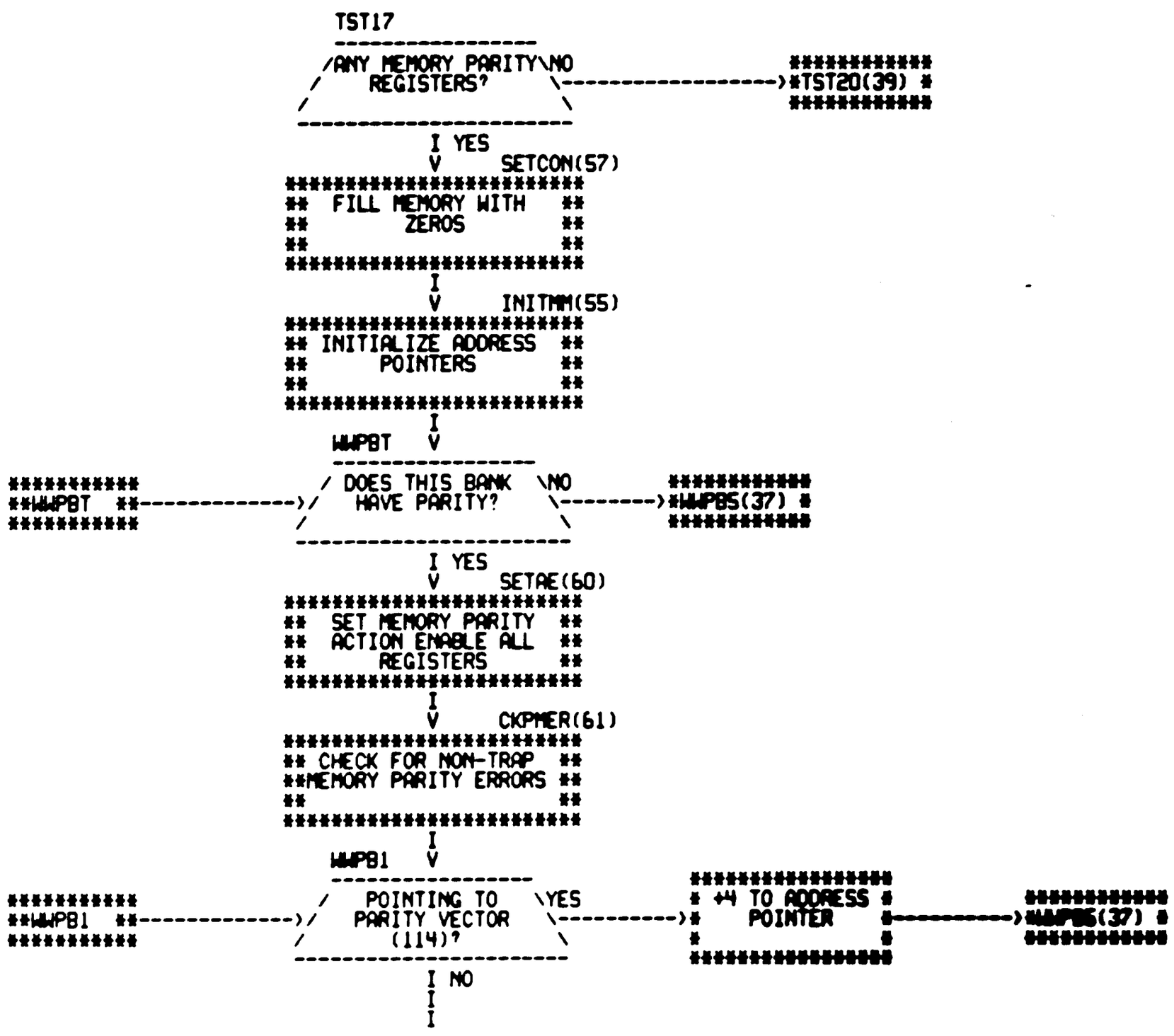

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
** ERROR: 3 XOR 9 **
** FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
DONE

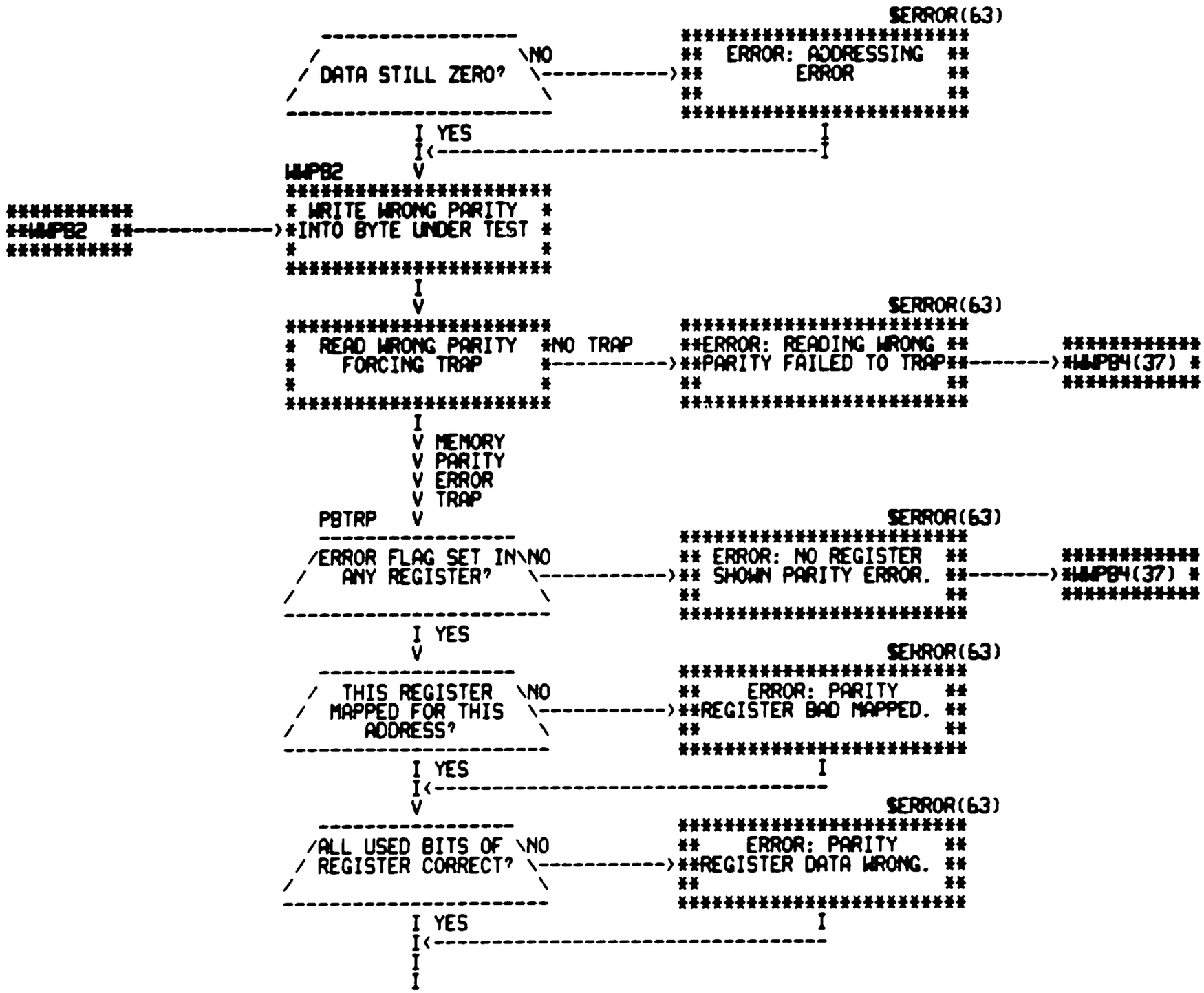

TST15
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH 401 AND **
** -1 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/ WRITTEN WITH 401 \
/ AND -1? \
I YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
SERROR(63)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

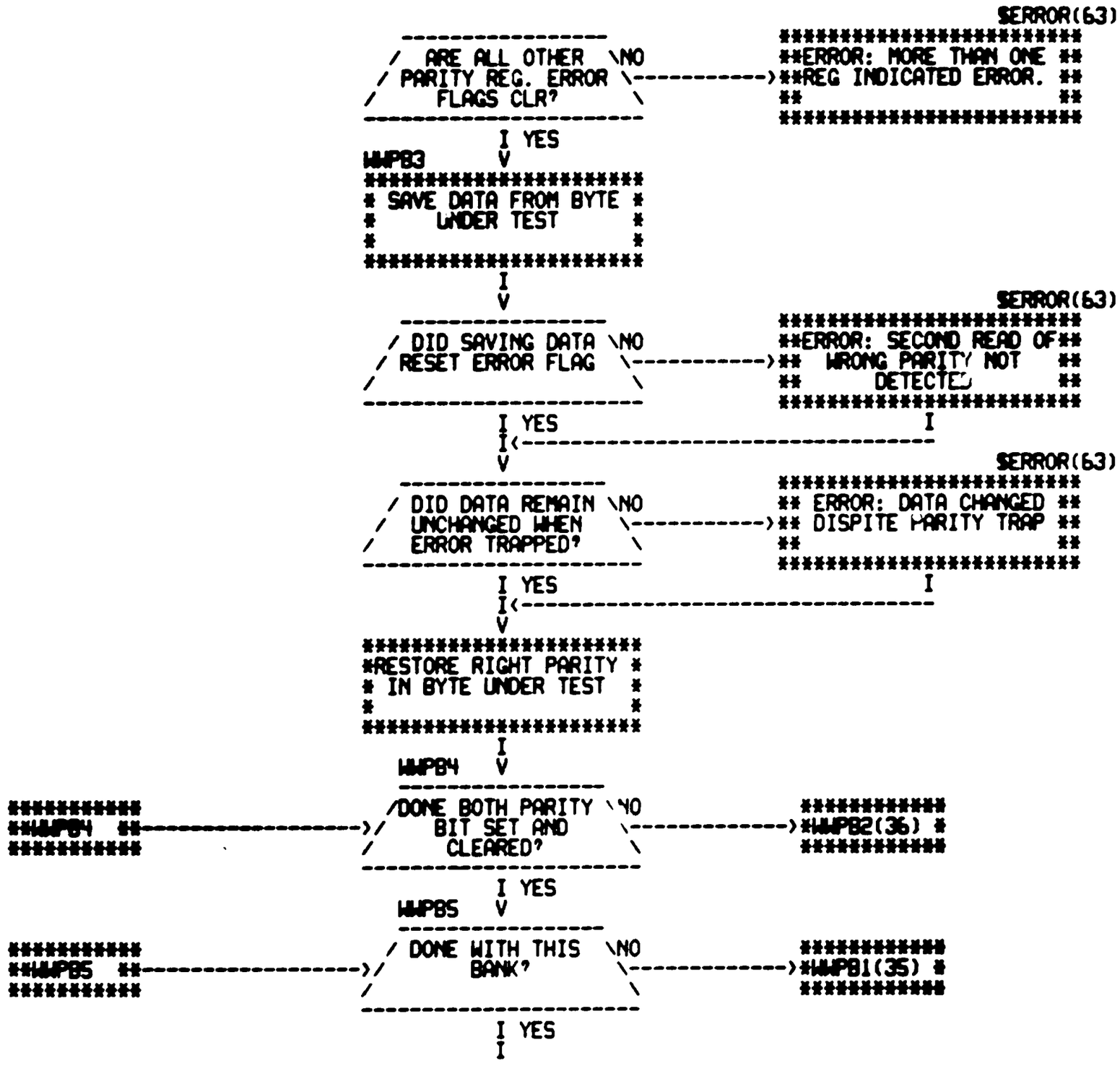

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
** ERROR: 3 XOR 9 **
** FAILURE **
YES
COMPLEMENT DATA
$ERROR(63)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
DATA OK?
NO
YES
COMPLEMENT DATA
$ERROR(63)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
DATA OK?
NO
YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE

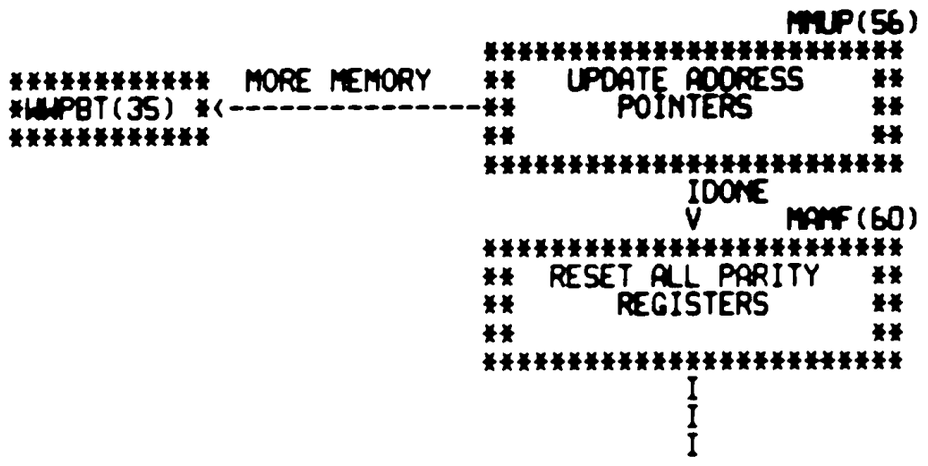

TST16
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH -1 AND **
** 401 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/WRITTEN WITH -1 AND\
/ 401? \
I YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(63)
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **

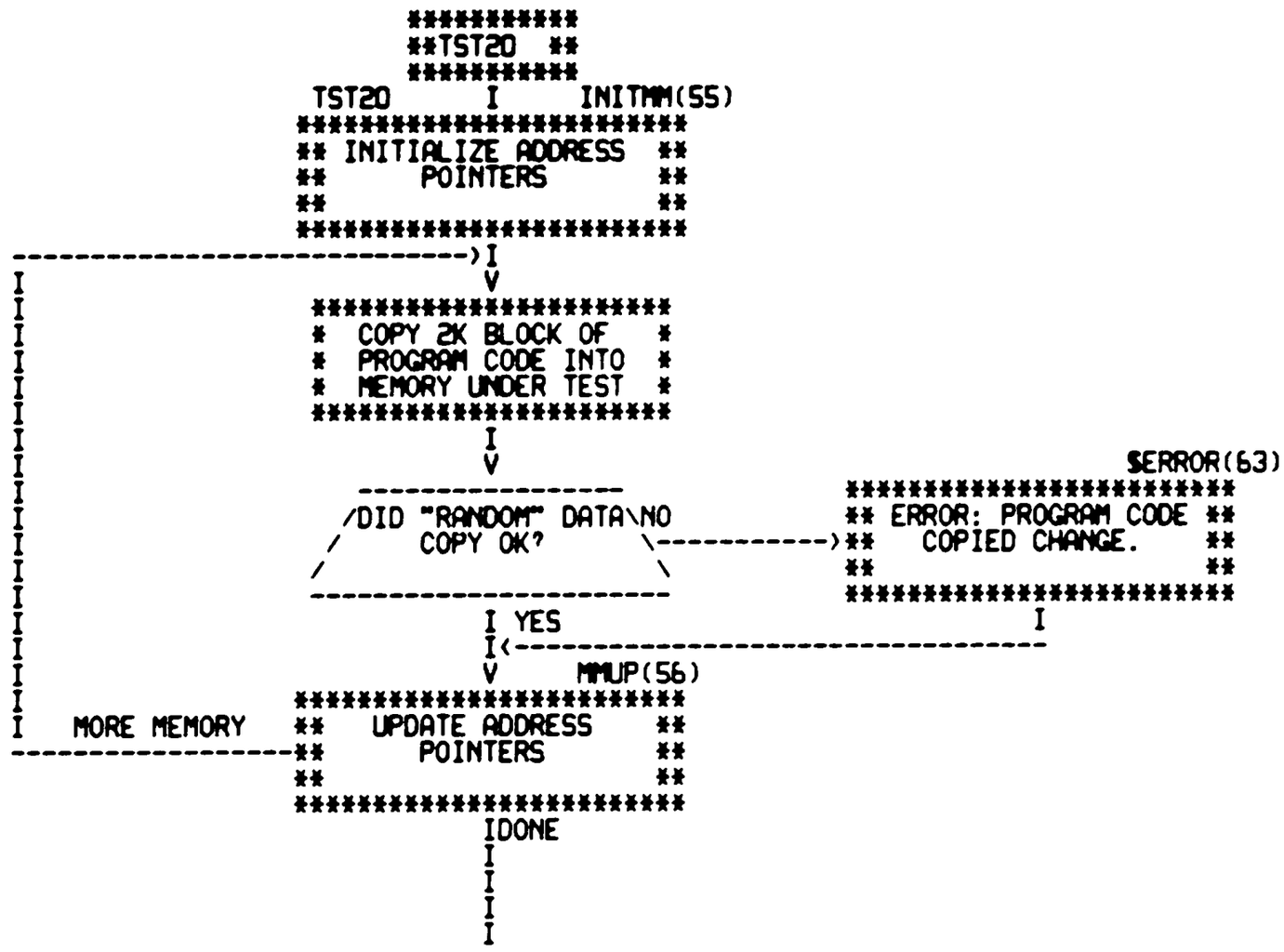

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
SERROR(63)
** ERROR: 3 XOR 9 **
** FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
SERROR(63)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
SERROR(63)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
DONE

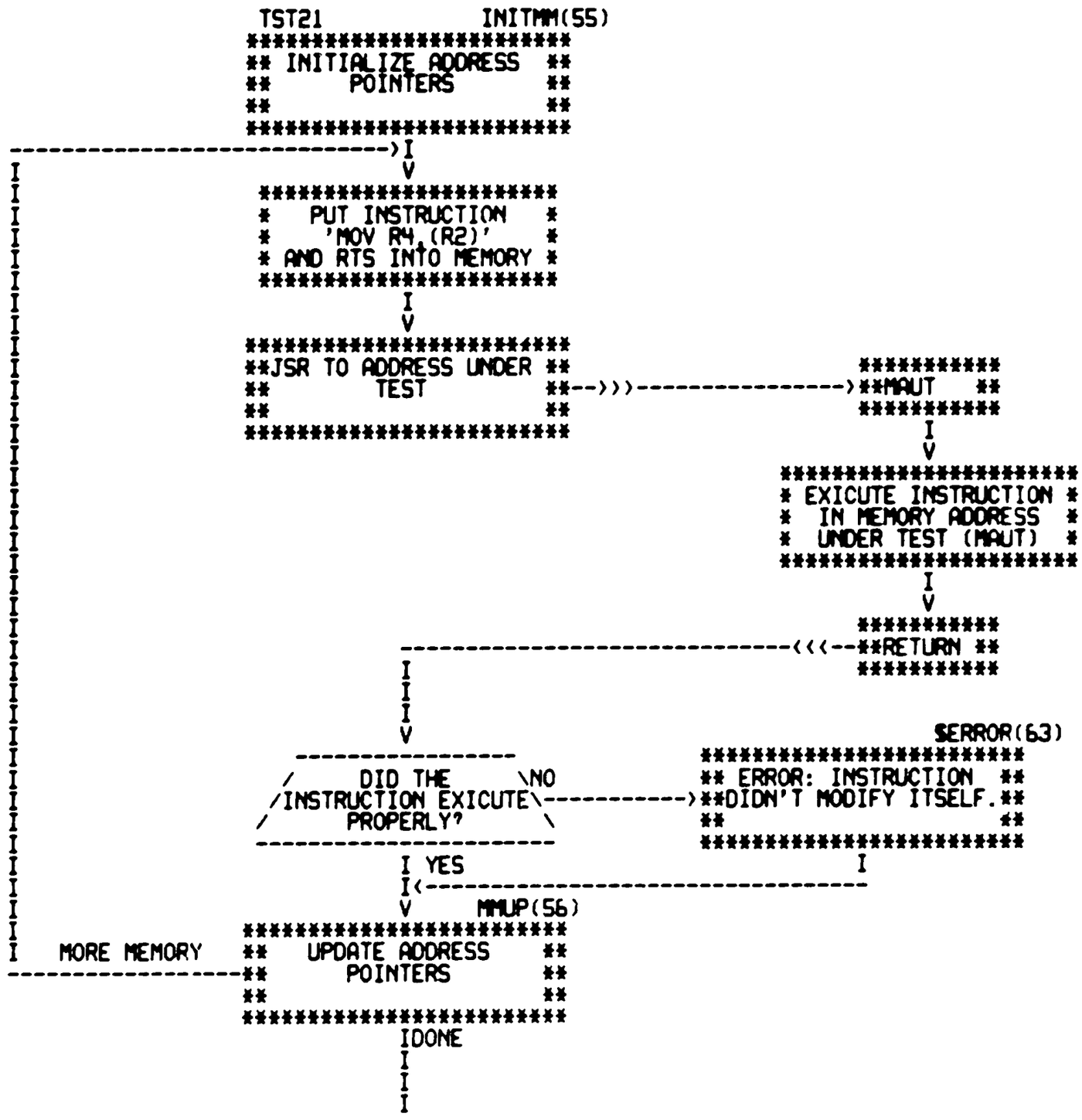

TST17
ANY MEMORY PARITY REGISTERS?
NO
*TST20(39) *
YES
SETCON(57)
FILL MEMORY WITH ZEROS
INITMM(55)
INITIALIZE ADDRESS POINTERS
WWPBT
DOES THIS BANK HAVE PARITY?
NO
*WWPBS(37) *
WWPBT
YES
SETAE(60)
SET MEMORY PARITY ACTION ENABLE ALL REGISTERS
CKPMER(61)
CHECK FOR NON-TRAP MEMORY PARITY ERRORS
WWPB1
POINTING TO PARITY VECTOR (114)?
YES
+4 TO ADDRESS POINTER
*WWPBS(37) *
WWPB1
NO

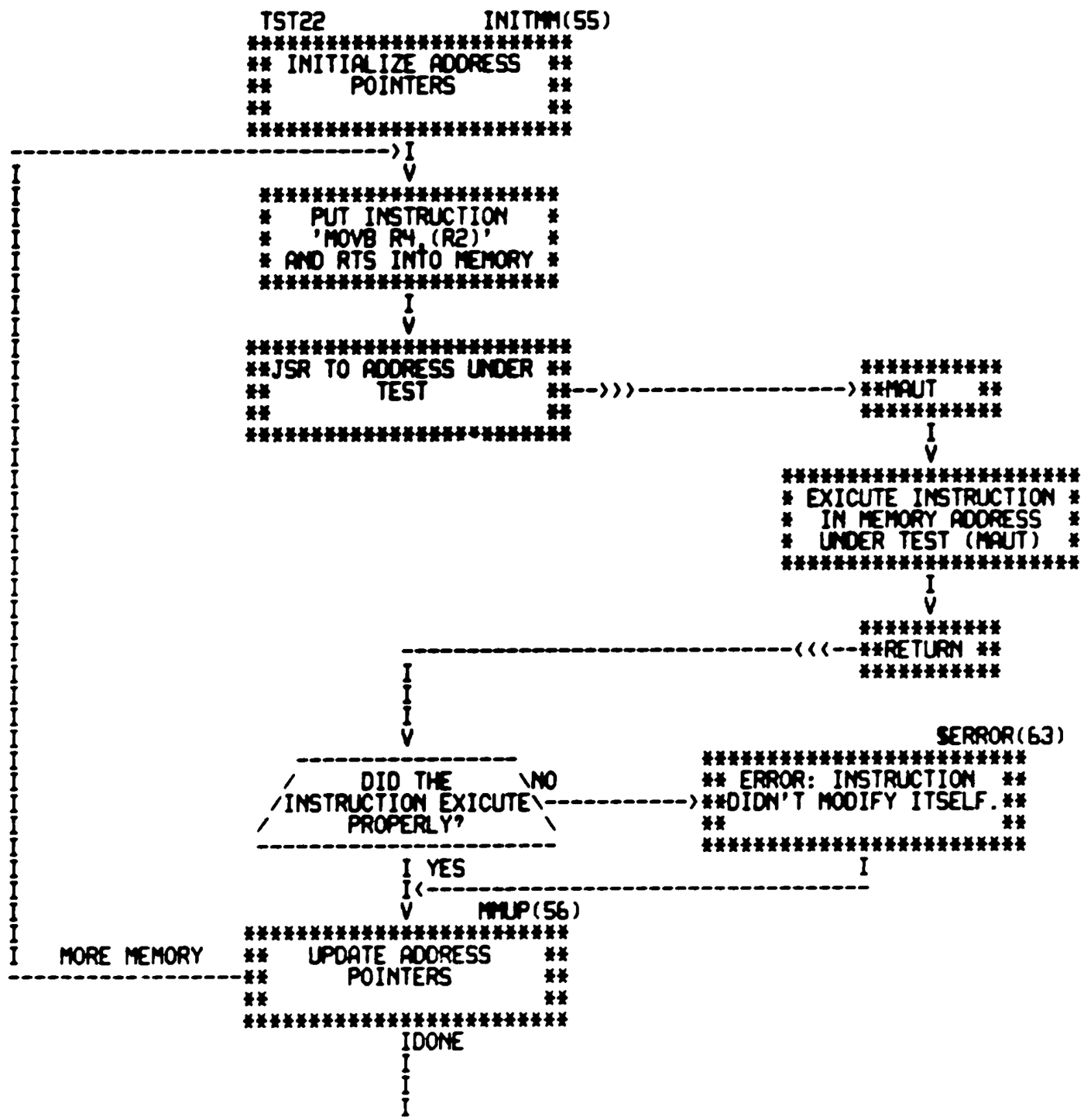

$ERROR(63)
DATA STILL ZERO?
NO
** ERROR: ADDRESSING **
** ERROR **
YES
HWPB2
* WRITE WRONG PARITY *
*INTO BYTE UNDER TEST *
**HWPB2 **
READ WRONG PARITY
FORCING TRAP
NO TRAP
**ERROR: READING WRONG **
PARITY FAILED TO TRAP
*HWPB4(37) *
MEMORY
PARITY
ERROR
TRAP
PBTRP
ERROR FLAG SET IN ANY REGISTER?
** ERROR: NO REGISTER **
** SHOWN PARITY ERROR. **
THIS REGISTER MAPPED FOR THIS ADDRESS?
** ERROR: PARITY **
**REGISTER BAD MAPPED. **
ALL USED BITS OF REGISTER CORRECT?
**REGISTER DATA WRONG. **

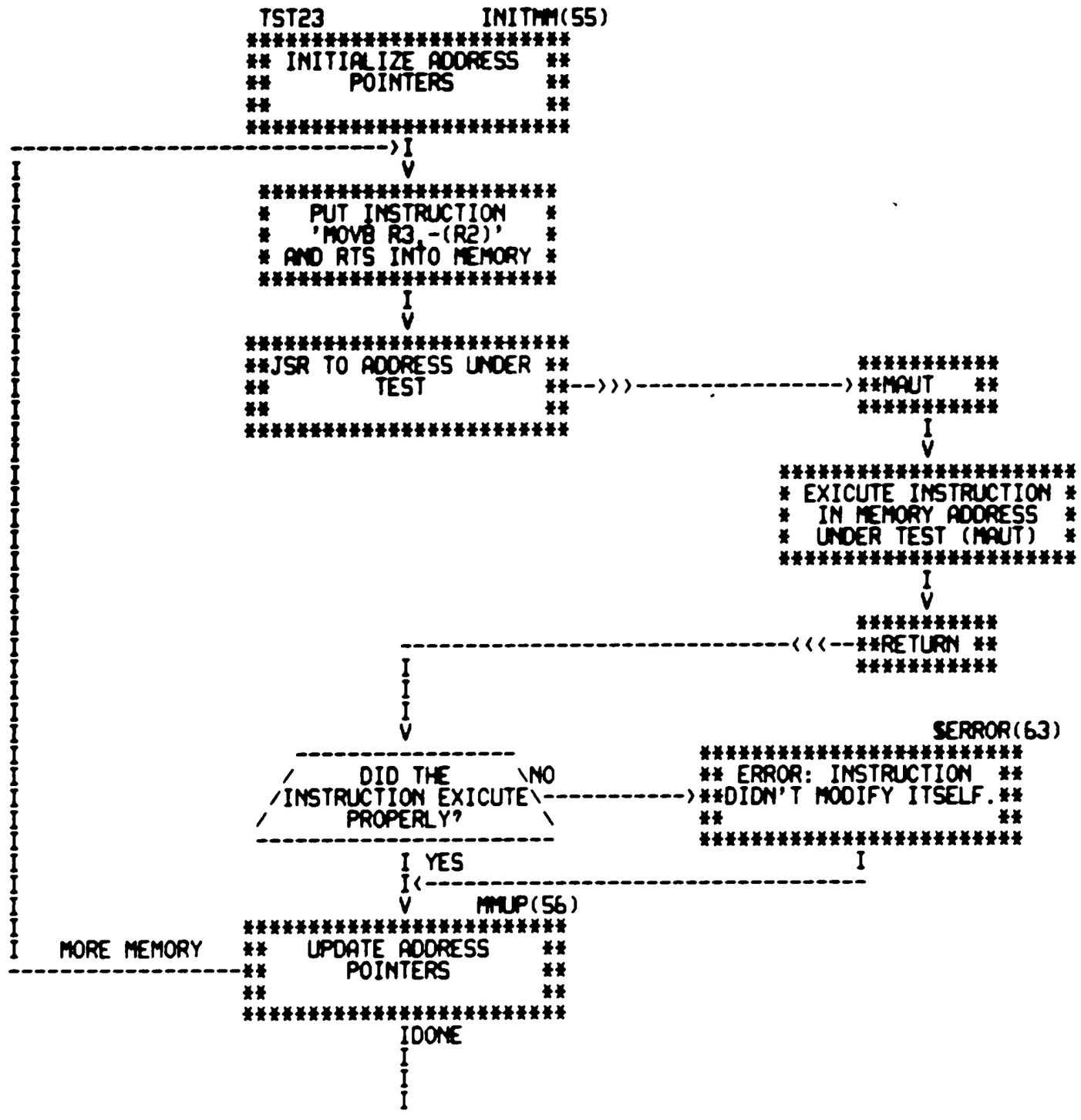

ARE ALL OTHER PARITY REG. ERROR FLAGS CLR?
NO
$ERROR(63)
**ERROR: MORE THAN ONE **REG INDICATED ERROR.
YES
HWPB3
SAVE DATA FROM BYTE UNDER TEST
DID SAVING DATA RESET ERROR FLAG
NO
$ERROR(63)
**ERROR: SECOND READ OF WRONG PARITY NOT DETECTED
YES
$ERROR(63)
DID DATA REMAIN UNCHANGED WHEN ERROR TRAPPED?
NO
** ERROR: DATA CHANGED DISPITE PARITY TRAP
YES
RESTORE RIGHT PARITY IN BYTE UNDER TEST
HWPB4
DONE BOTH PARITY BIT SET AND CLEARED?
NO
HWPB2(36)
YES
HWPB5
DONE WITH THIS BANK?
NO
HWPB1(35)
YES

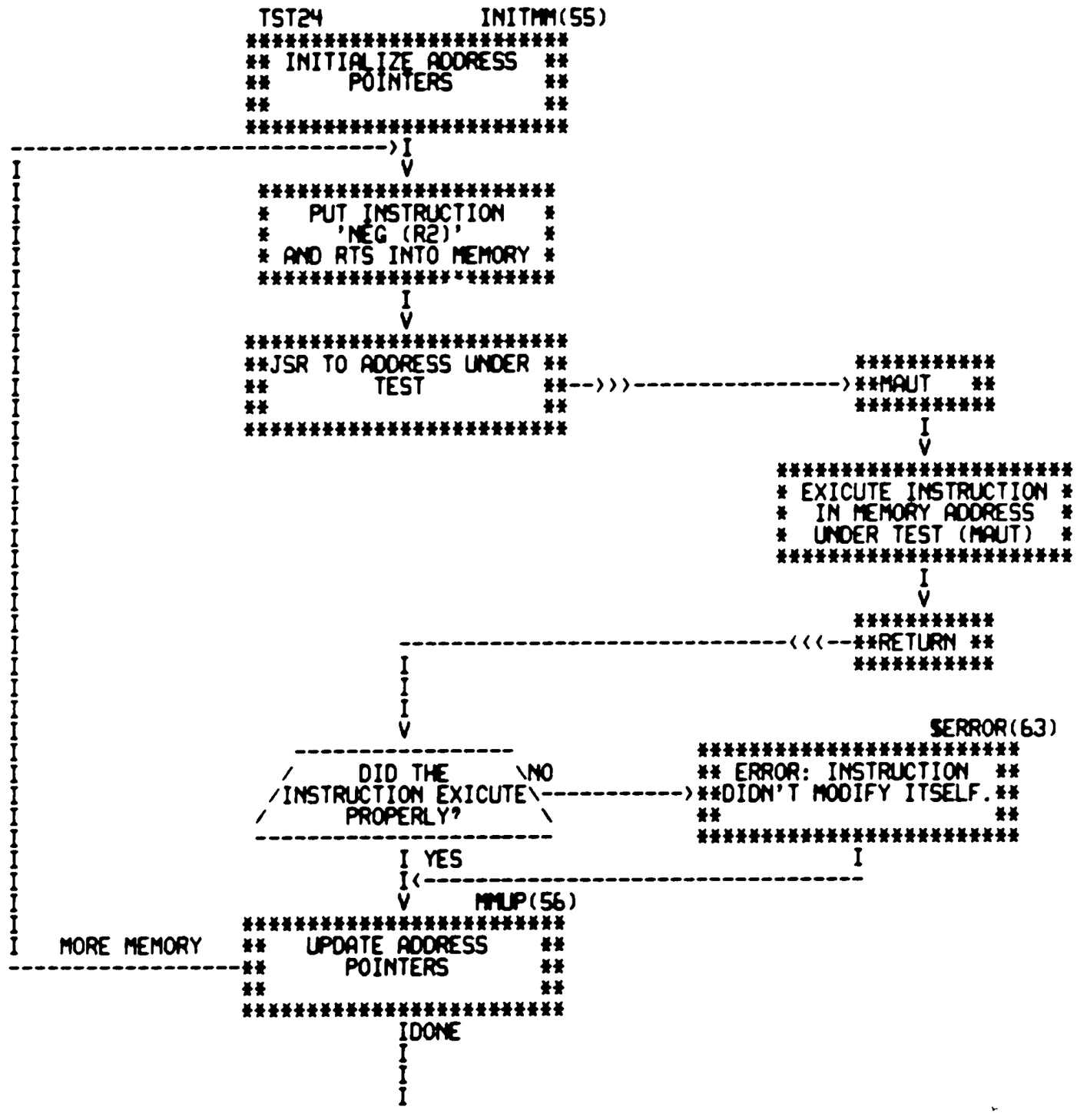

MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
WWPBT(35)
IDONE
MAMF(60)
RESET ALL PARITY REGISTERS

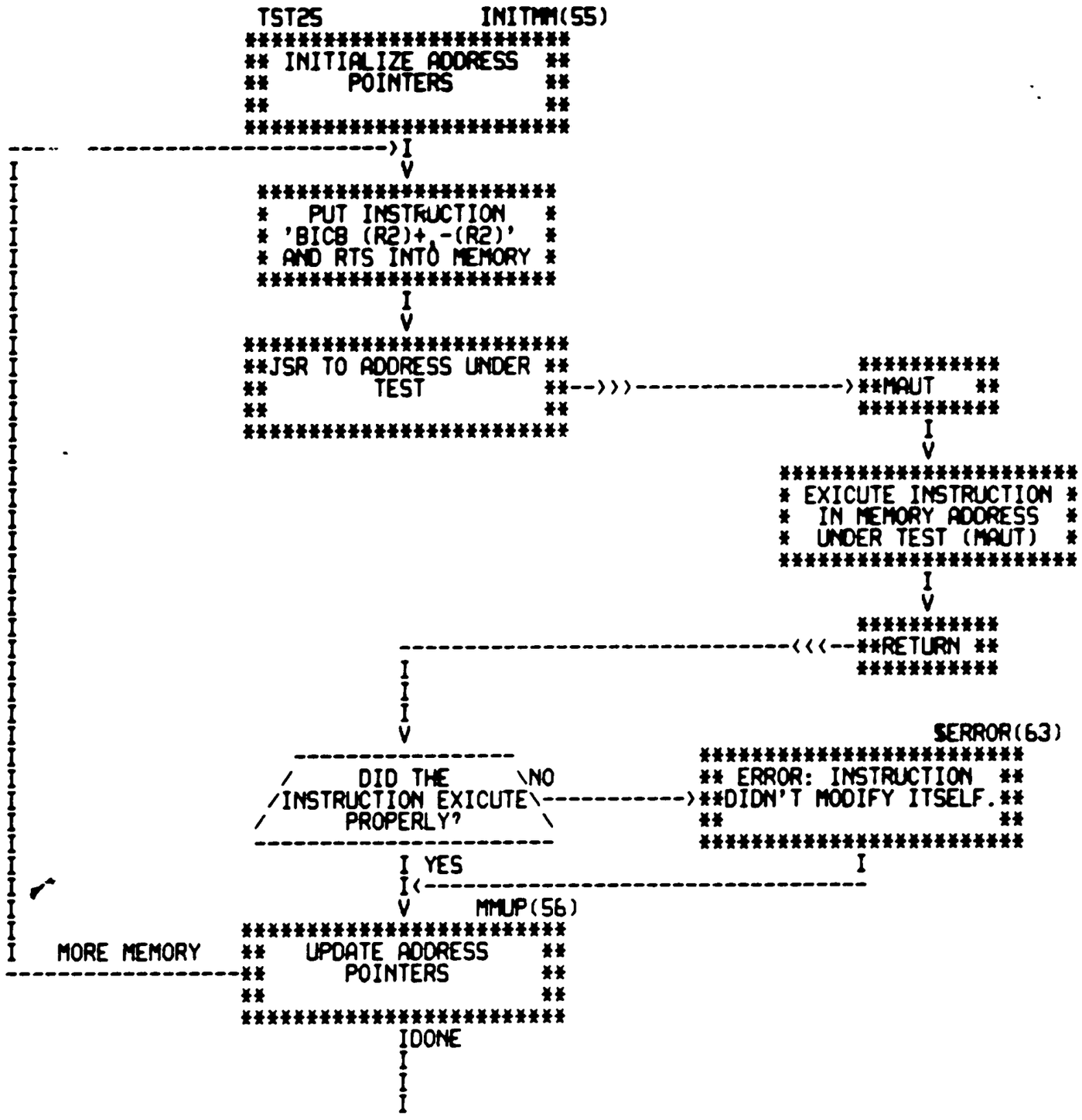

**TST20 **
TST20
I
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
* COPY 2K BLOCK OF *
* PROGRAM CODE INTO *
* MEMORY UNDER TEST *
/DID "RANDOM" DATA\NO
/ COPY OK? \
\ /
I YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
IDONE
$ERROR(63)
** ERROR: PROGRAM CODE **
** COPIED CHANGE. **

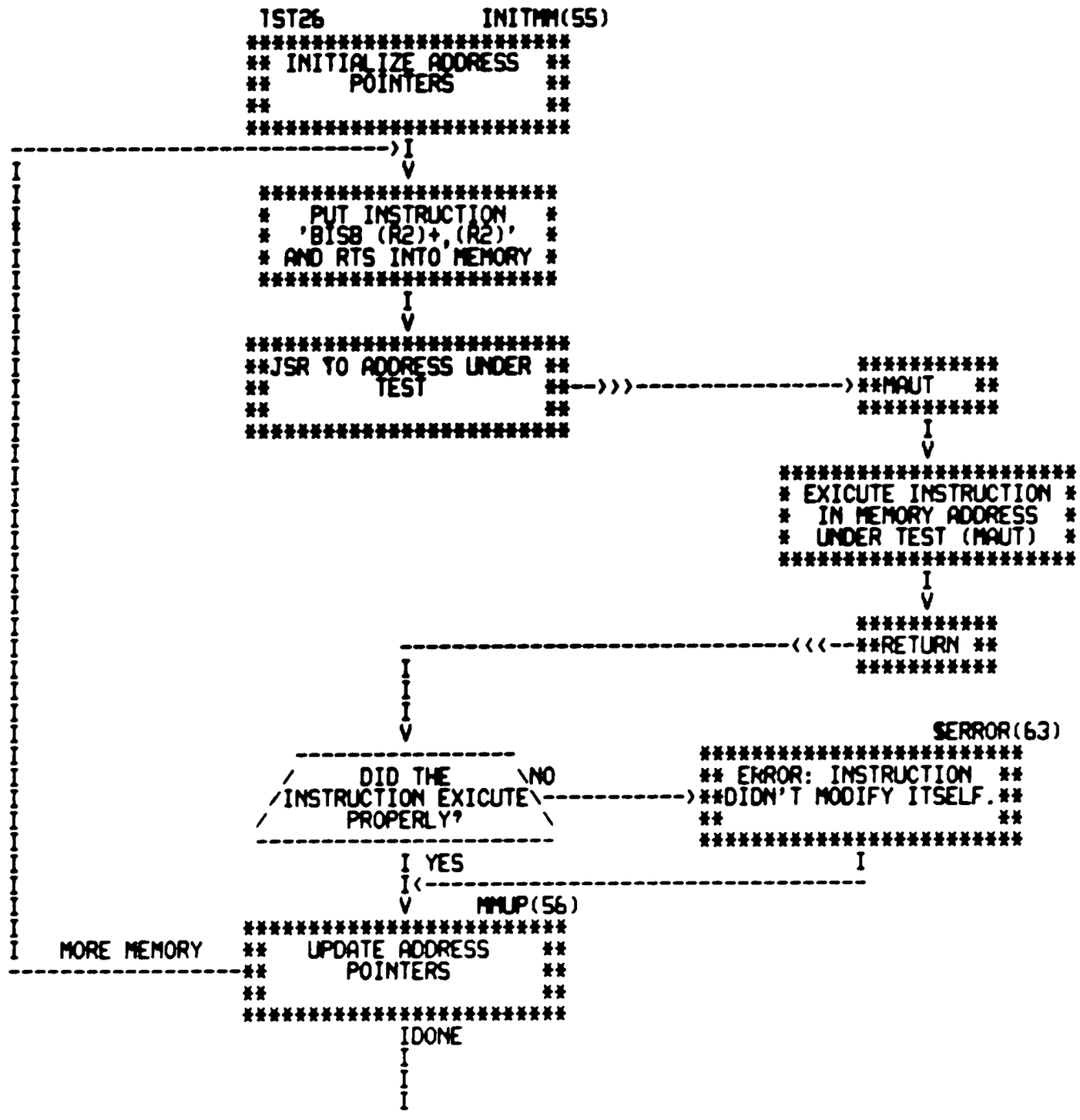

TST21
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'MOV R4,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

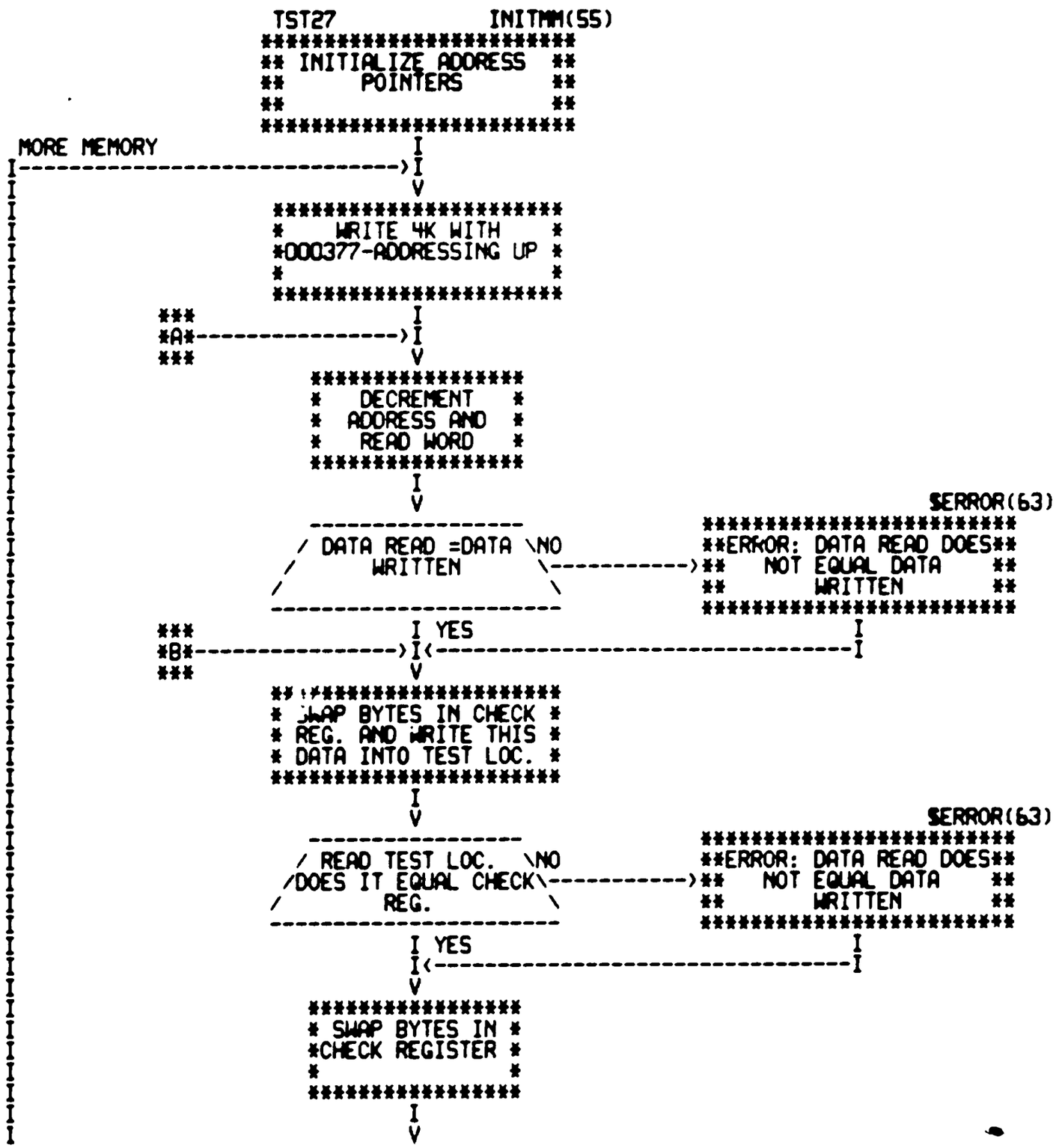

TST22
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'MOVB R4,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

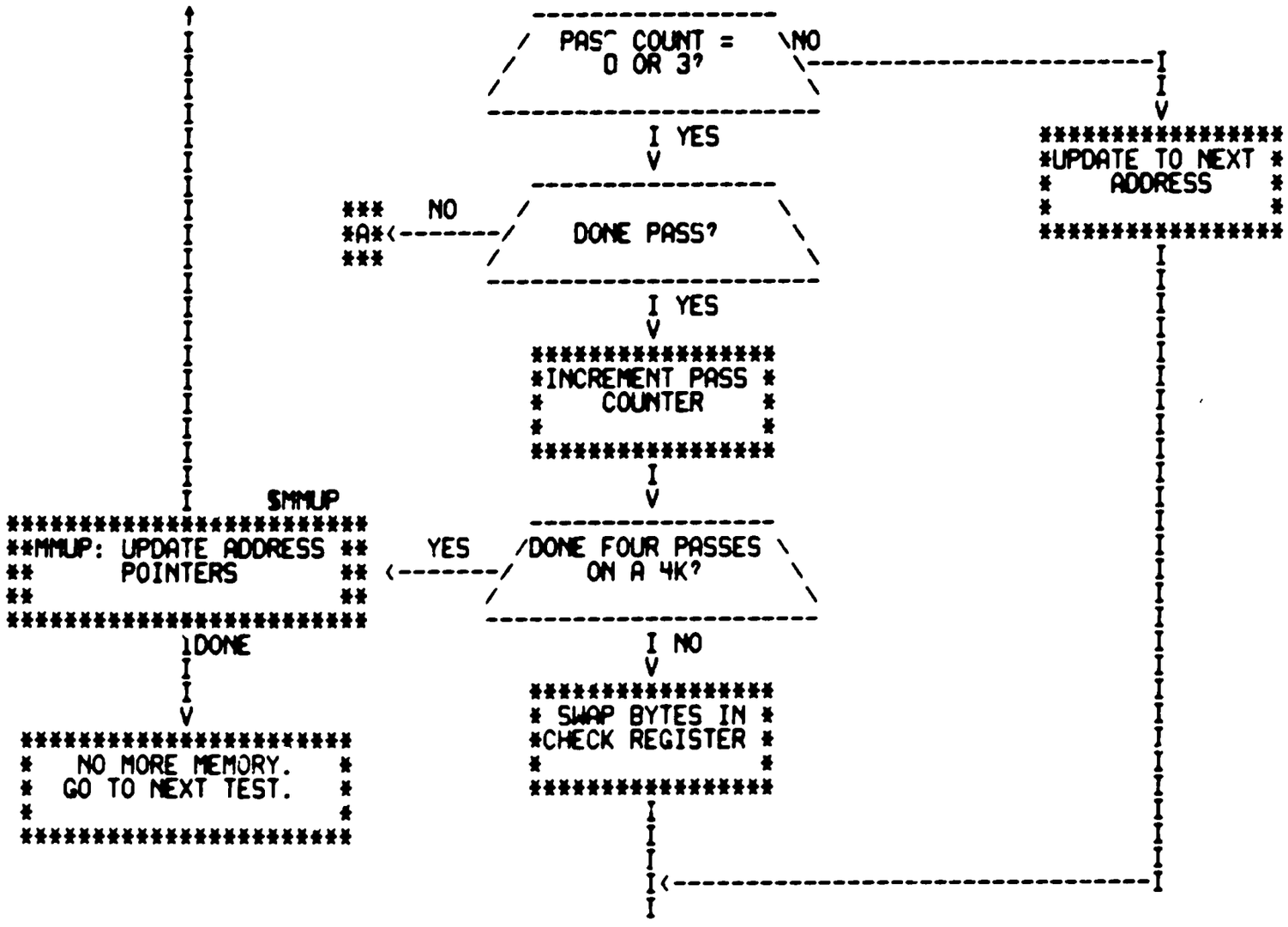

TST23
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION 'MOVB R3,-(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
SERROR(63)
** ERROR: INSTRUCTION DIDN'T MODIFY ITSELF. **
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS POINTERS **
IDONE

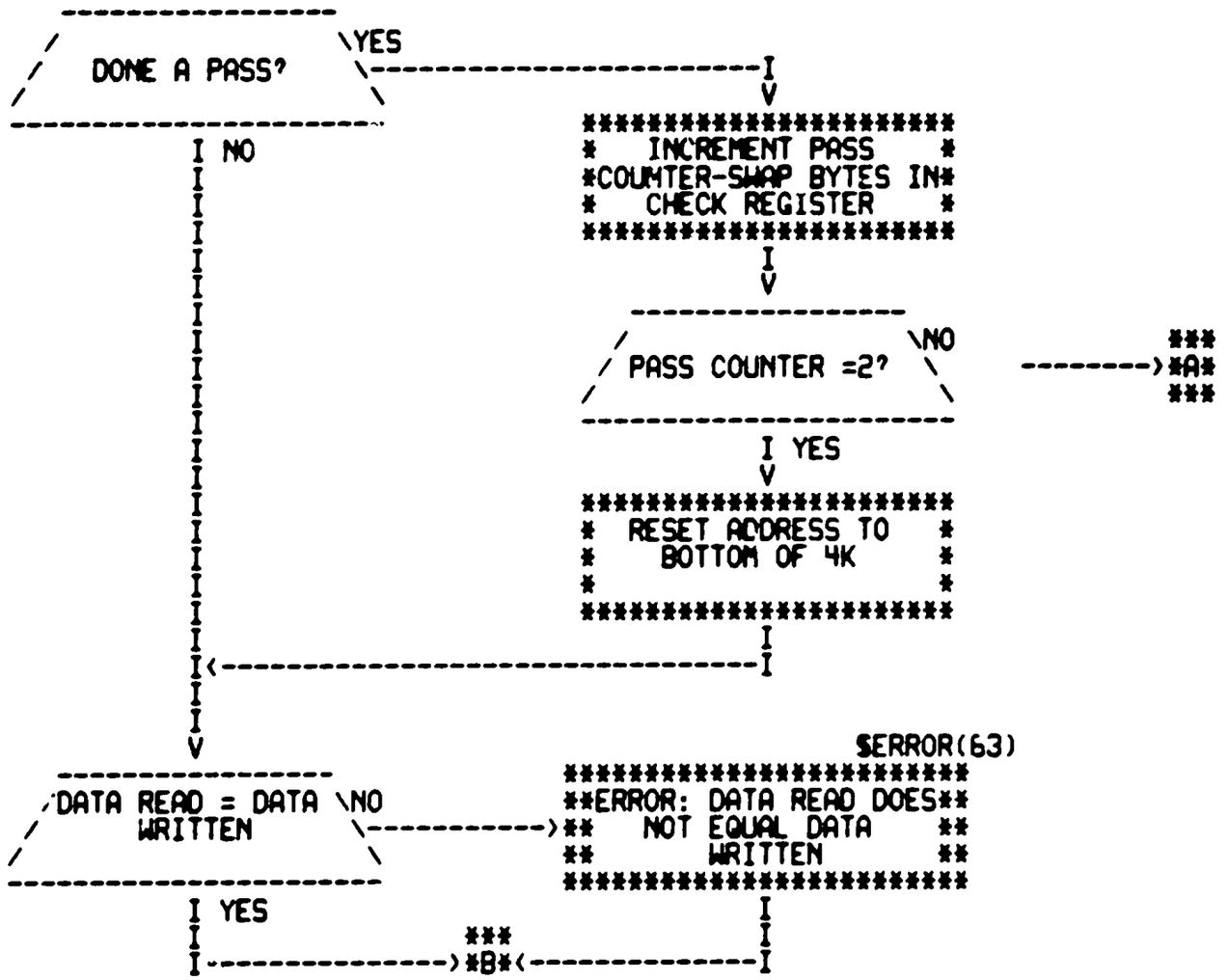

TST24
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
SERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
PUT INSTRUCTION 'NEG (R2)' AND RTS INTO MEMORY
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE

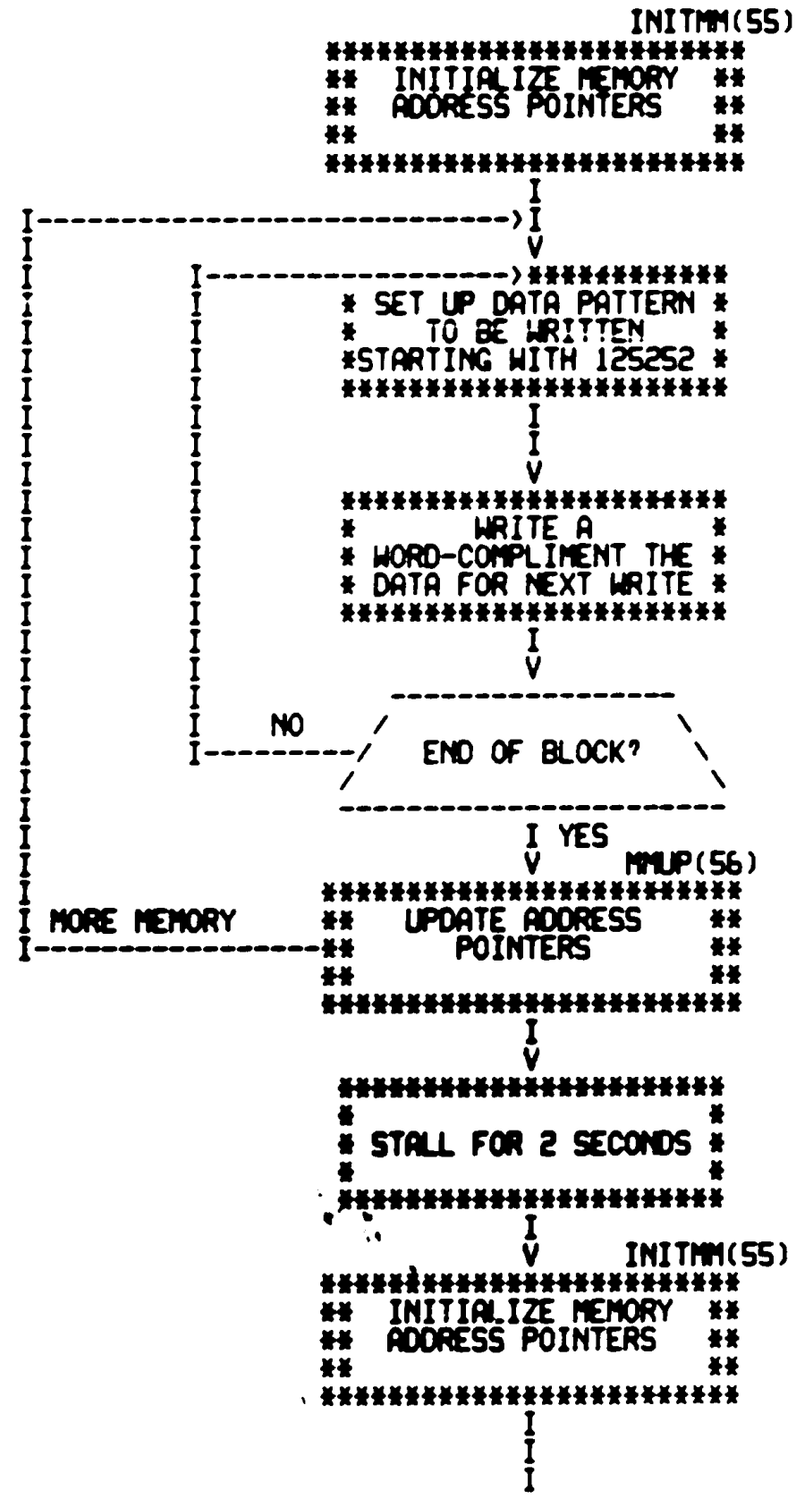

TST25
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'BICB (R2)+,-(R2)'
AND RTS INTO MEMORY
**JSR TO ADDRESS UNDER **
** TEST **
**MAUT **
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
**RETURN **
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
$ERROR(63)
** ERROR: INSTRUCTION **
DIDN'T MODIFY ITSELF.
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
IDONE

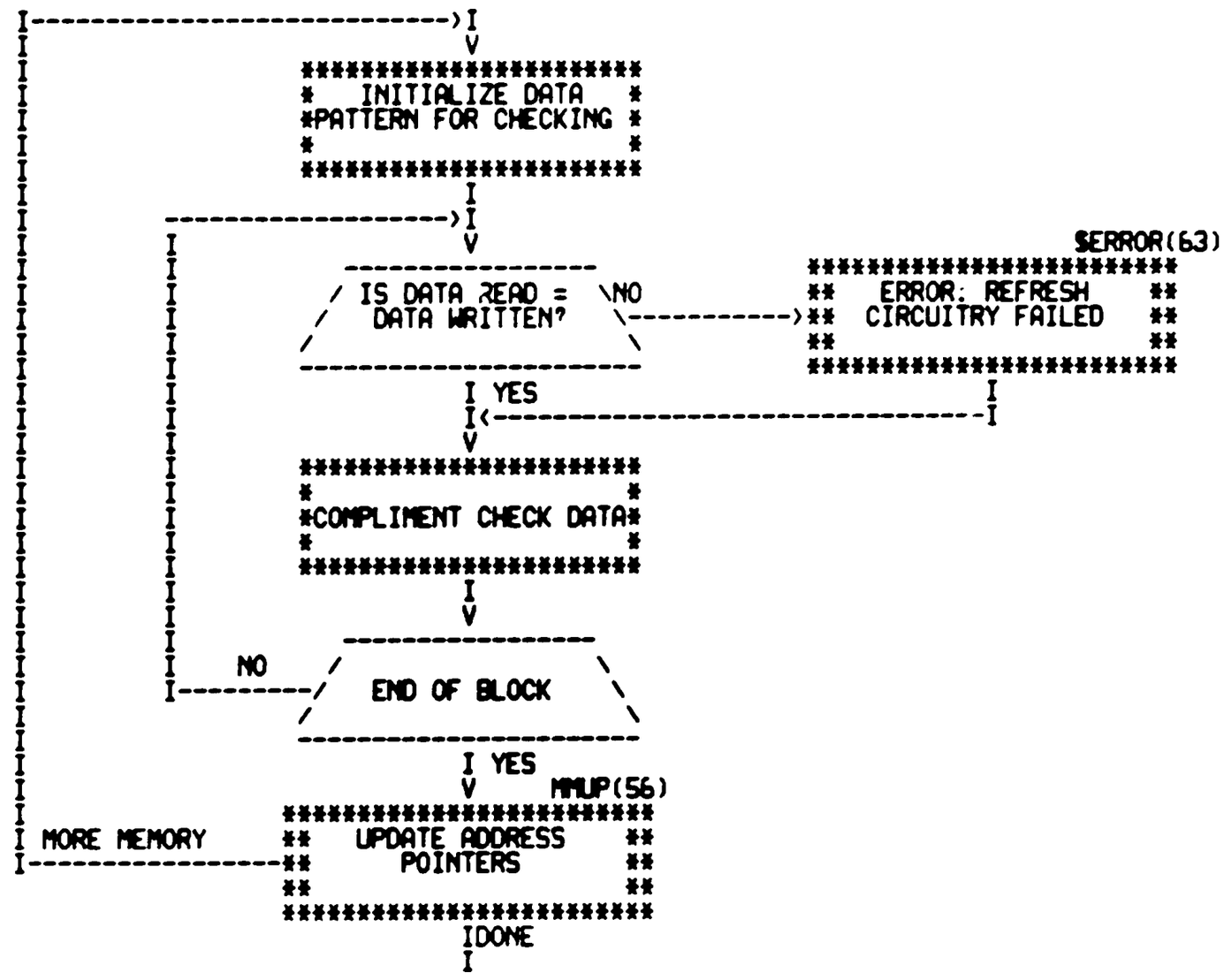

TST26
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'BISB (R2)+,(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

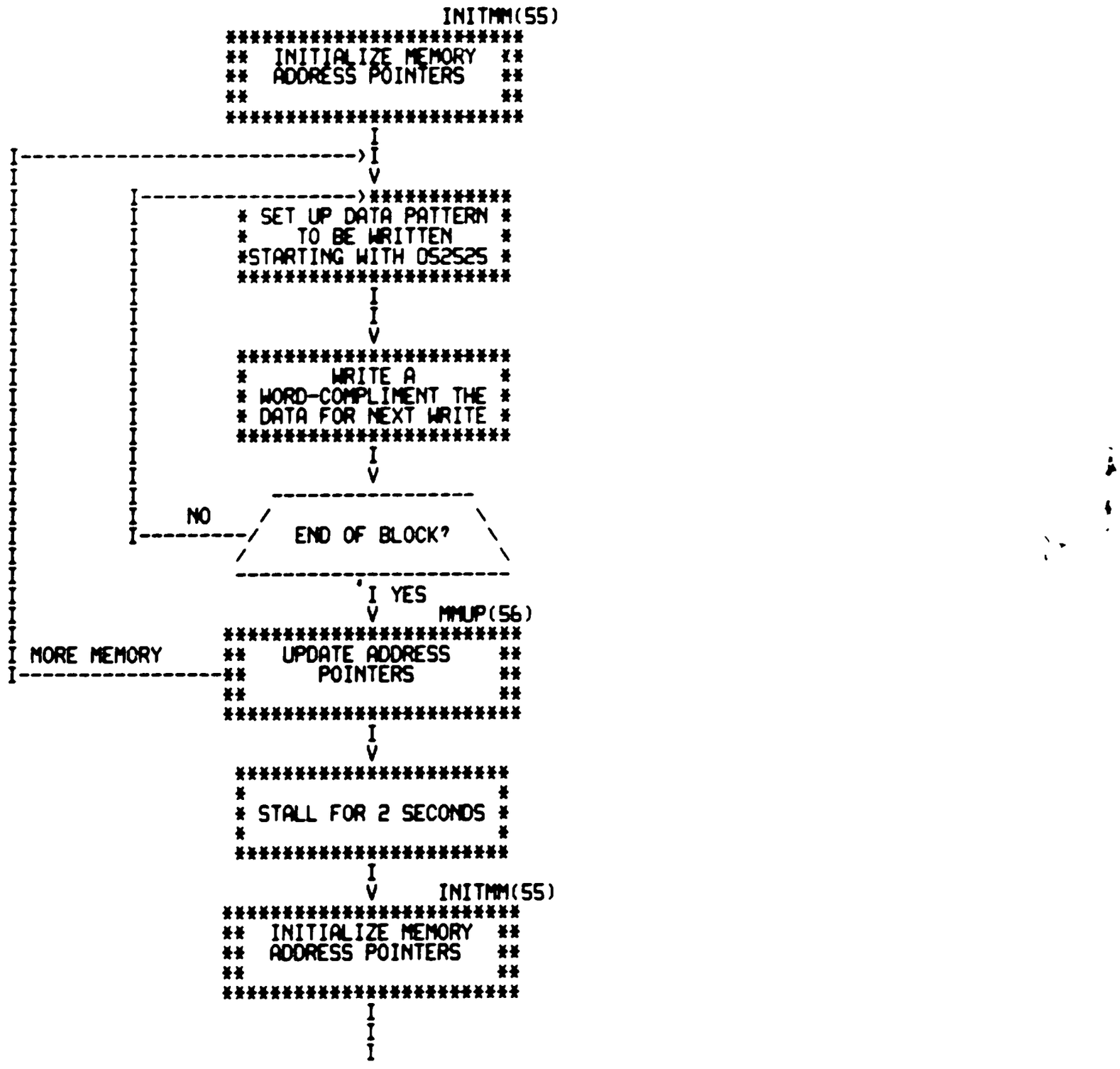

TST27
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
MORE MEMORY
WRITE 4K WITH
*000377-ADDRESSING UP *
A
DECREMENT
ADDRESS AND
READ WORD
DATA READ =DATA WRITTEN
NO
YES
$ERROR(63)
ERROR: DATA READ DOES
** NOT EQUAL DATA **
** WRITTEN **
B
SWAP BYTES IN CHECK
REG. AND WRITE THIS
DATA INTO TEST LOC.
READ TEST LOC.
DOES IT EQUAL CHECK
REG.
SWAP BYTES IN
*CHECK REGISTER *

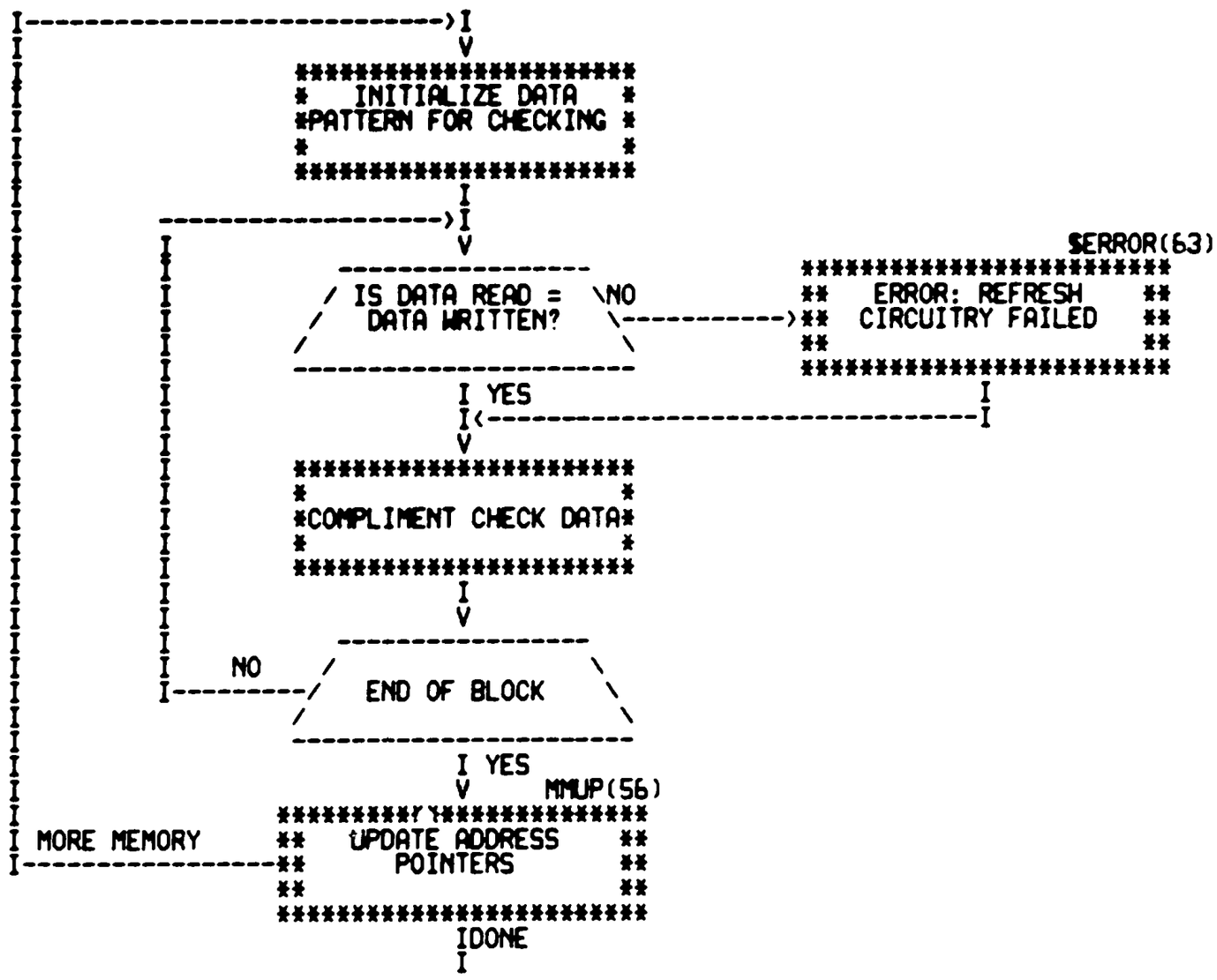

PASS COUNT = 0 OR 3?
NO
I YES
DONE PASS?
*** NO
A
***
I YES
INCREMENT PASS COUNTER
UPDATE TO NEXT ADDRESS
SMMUP
MMUP: UPDATE ADDRESS POINTERS
YES
DONE FOUR PASSES ON A 4K?
I NO
* SWAP BYTES IN CHECK REGISTER *
IDONE
* NO MORE MEMORY. GO TO NEXT TEST. *

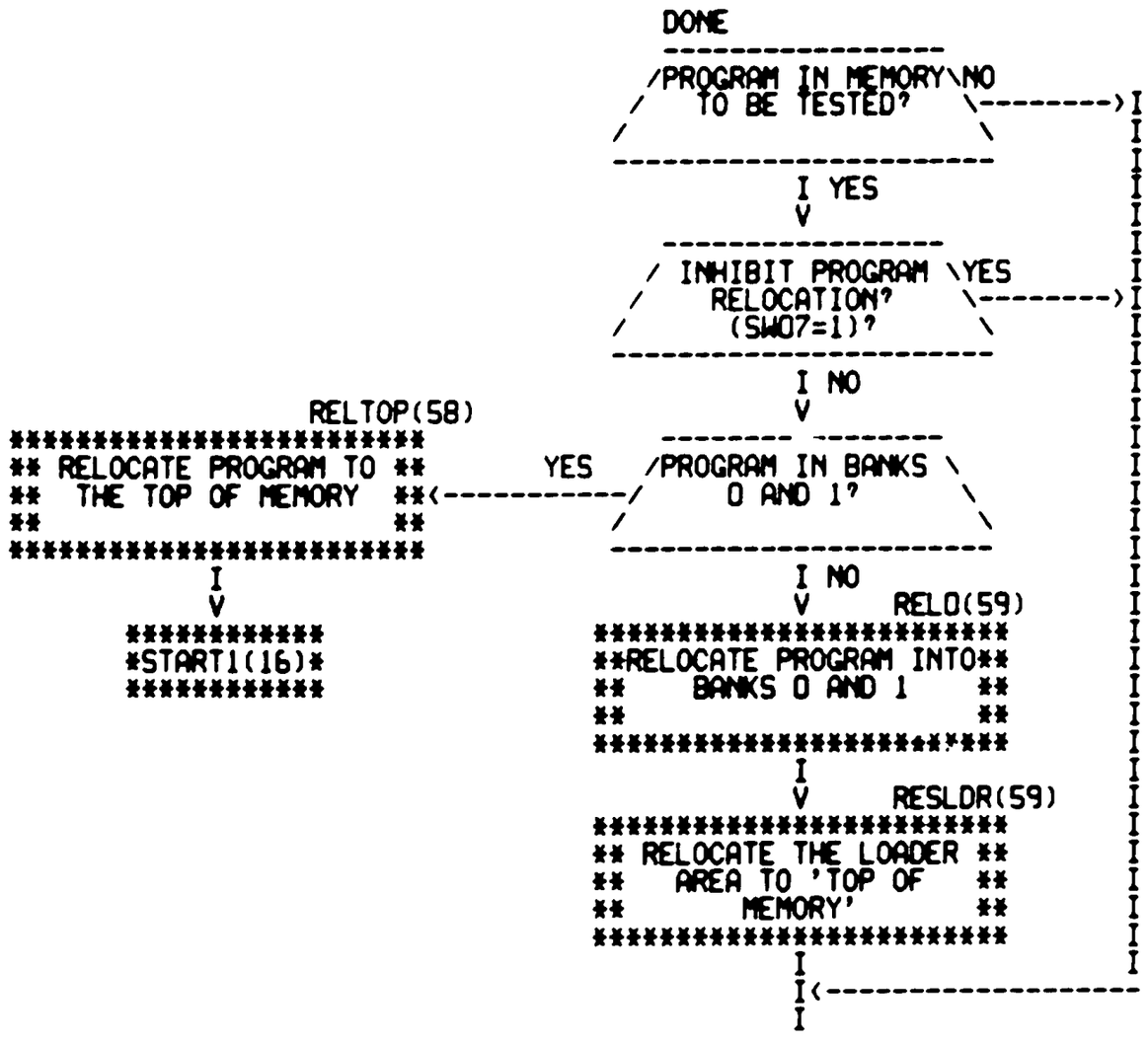

DONE A PASS?
YES
NO
INCREMENT PASS
COUNTER-SWAP BYTES IN
CHECK REGISTER
PASS COUNTER =2?
NO
YES
A
RESET ADDRESS TO
BOTTOM OF 4K
DATA READ = DATA
WRITTEN
NO
YES
$ERROR(63)
**ERROR: DATA READ DOES
NOT EQUAL DATA
WRITTEN
B

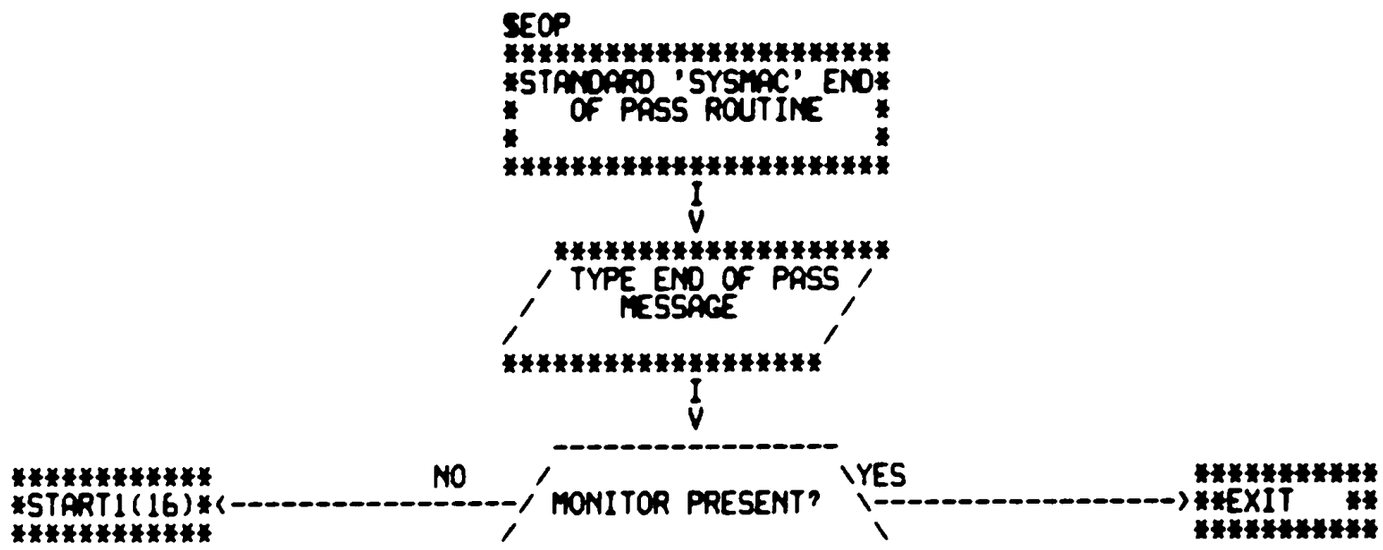

INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS
SET UP DATA PATTERN TO BE WRITTEN STARTING WITH 125252
WRITE A WORD-COMPLIMENT THE DATA FOR NEXT WRITE
NO
END OF BLOCK?
YES
MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
STALL FOR 2 SECONDS
INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS

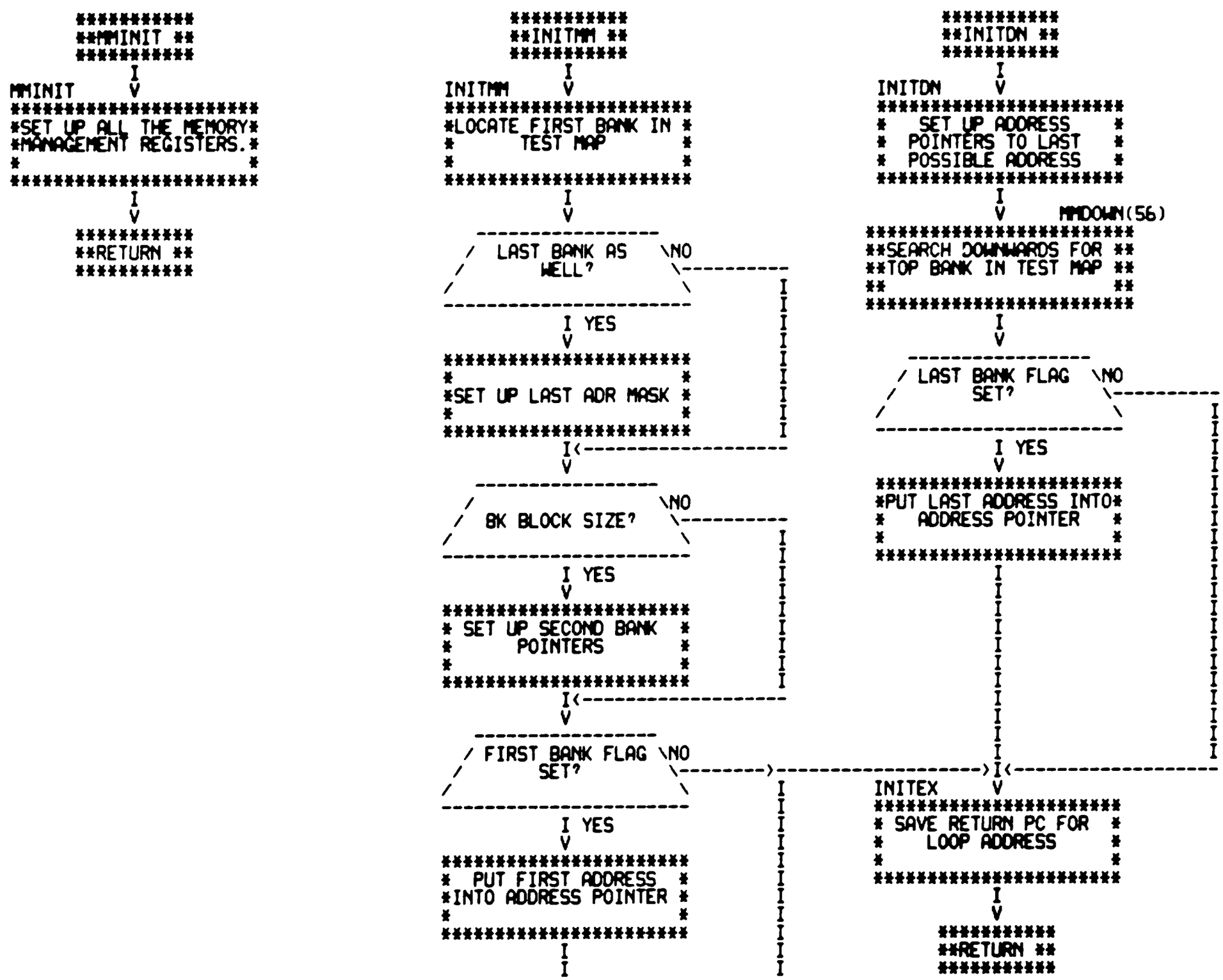

INITIALIZE DATA PATTERN FOR CHECKING
IS DATA READ = DATA WRITTEN?
NO
$ERROR(63)
ERROR: REFRESH CIRCUITRY FAILED
YES
COMPLIMENT CHECK DATA
END OF BLOCK
NO
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

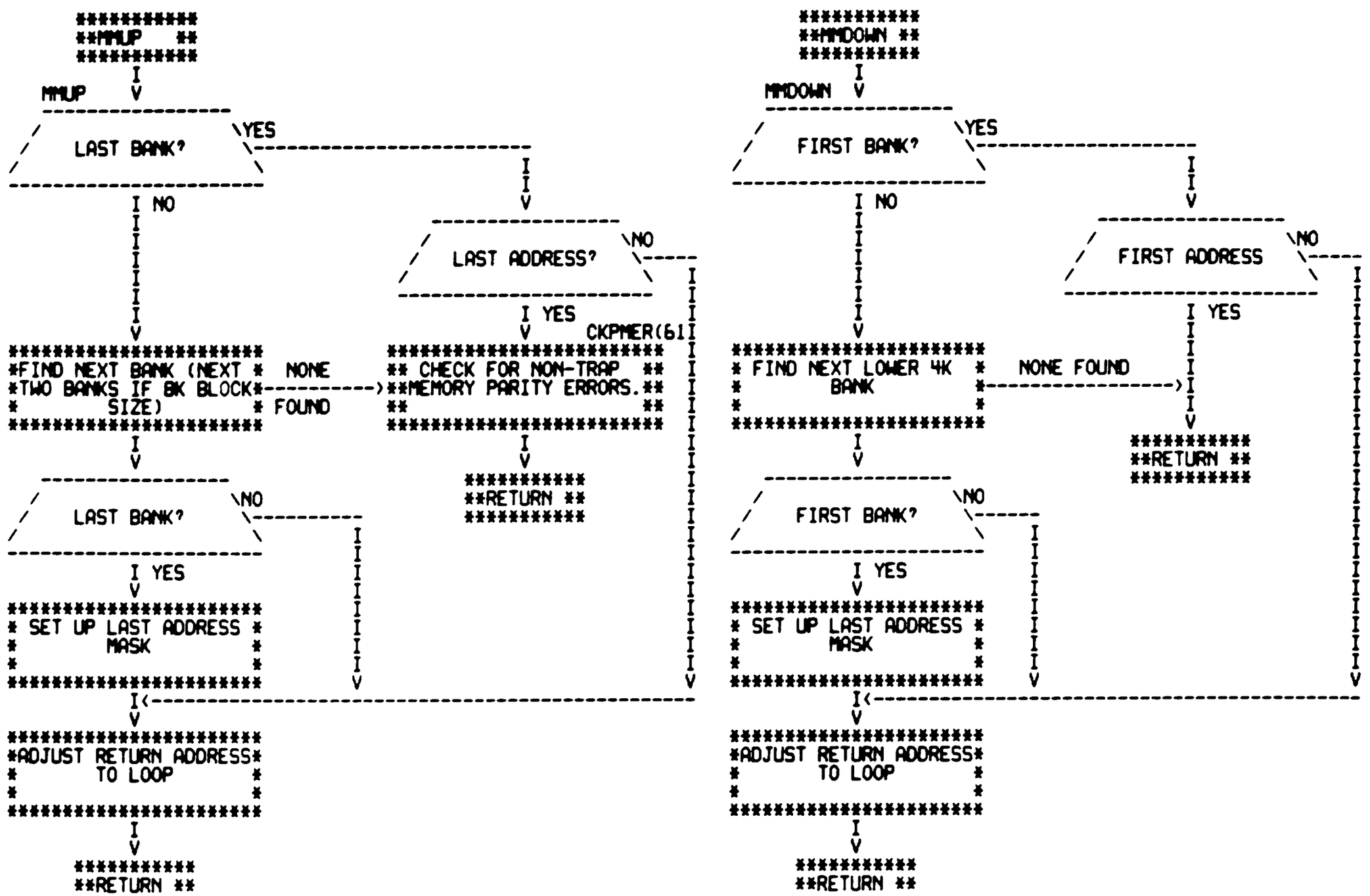

INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS
SET UP DATA PATTERN TO BE WRITTEN STARTING WITH 052525
WRITE A WORD-COMPLIMENT THE DATA FOR NEXT WRITE
NO
END OF BLOCK?
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
STALL FOR 2 SECONDS
INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS

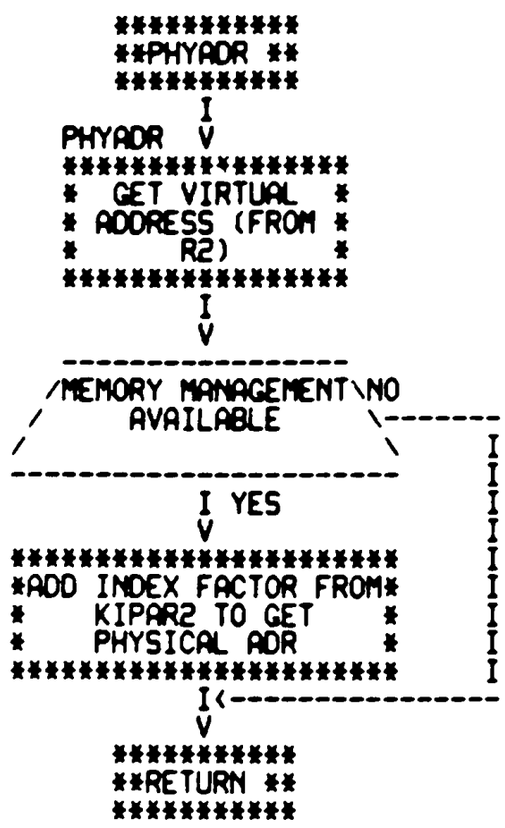

INITIALIZE DATA PATTERN FOR CHECKING
IS DATA READ = DATA WRITTEN?
NO
$ERROR(63)
ERROR: REFRESH CIRCUITRY FAILED
YES
COMPLIMENT CHECK DATA
END OF BLOCK
NO
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

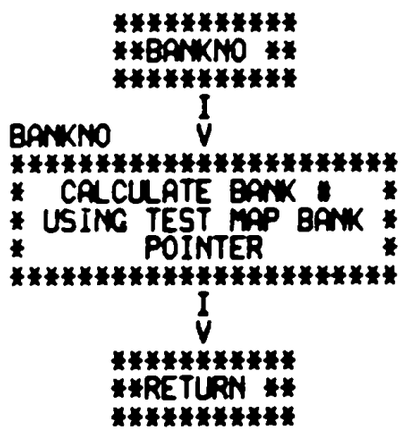

DONE
/PROGRAM IN MEMORY\NO
/ TO BE TESTED? \
/ \
I YES
V
/ INHIBIT PROGRAM \YES
/ RELOCATION? \
/ (SW07=1)? \
I NO
V
RELTOP(58)
** RELOCATE PROGRAM TO **
** THE TOP OF MEMORY **
YES /PROGRAM IN BANKS \
/ 0 AND 1? \
I NO
V RELO(59)
START1(16)
RELOCATE PROGRAM INTO
** BANKS 0 AND 1 **
I
V RESLDR(59)
** RELOCATE THE LOADER **
** AREA TO 'TOP OF **
** MEMORY' **

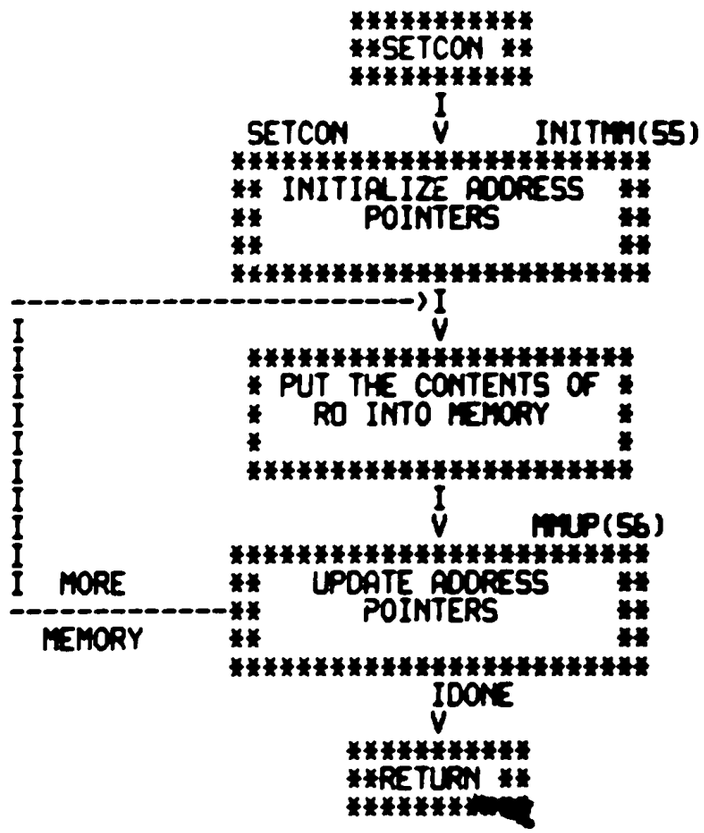

$EOP
STANDARD 'SYSMAC' END
OF PASS ROUTINE
TYPE END OF PASS
MESSAGE
NO
MONITOR PRESENT?
YES
START1(16)
EXIT

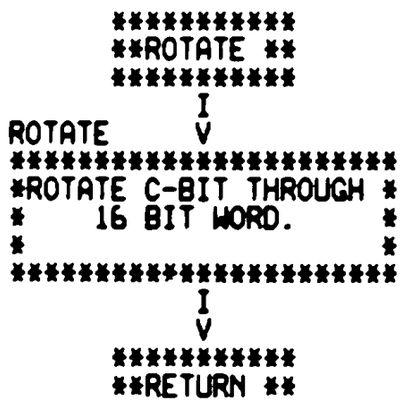

**MMINIT **
MMINIT
SET UP ALL THE MEMORY
MANAGEMENT REGISTERS.
**RETURN **
**INITMM **
INITMM
*LOCATE FIRST BANK IN *
* TEST MAP *
LAST BANK AS WELL?
NO
YES
*SET UP LAST ADR MASK *
BK BLOCK SIZE?
NO
YES
* SET UP SECOND BANK *
* POINTERS *
FIRST BANK FLAG SET?
NO
YES
* PUT FIRST ADDRESS *
*INTO ADDRESS POINTER *
**INITDN **
INITDN
* SET UP ADDRESS *
* POINTERS TO LAST *
* POSSIBLE ADDRESS *
MMDOWN(56)
**SEARCH DOWNWARDS FOR **
**TOP BANK IN TEST MAP **
LAST BANK FLAG SET?
NO
YES
PUT LAST ADDRESS INTO
* ADDRESS POINTER *
INITEX
* SAVE RETURN PC FOR *
* LOOP ADDRESS *
**RETURN **

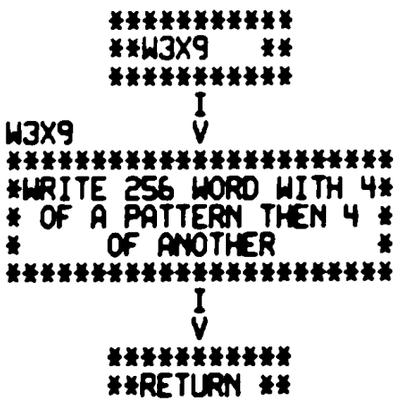

**MMUP **
MMUP
LAST BANK?
YES
NO
LAST ADDRESS?
NO
YES
CKPMER(61
#FIND NEXT BANK (NEXT *TWO BANKS IF 8K BLOCK* SIZE)
NONE
FOUND
** CHECK FOR NON-TRAP **MEMORY PARITY ERRORS.**
**RETURN **
LAST BANK?
NO
YES
* SET UP LAST ADDRESS * MASK
ADJUST RETURN ADDRESS TO LOOP
**RETURN **
**MMDOWN **
MMDOWN
FIRST 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 **

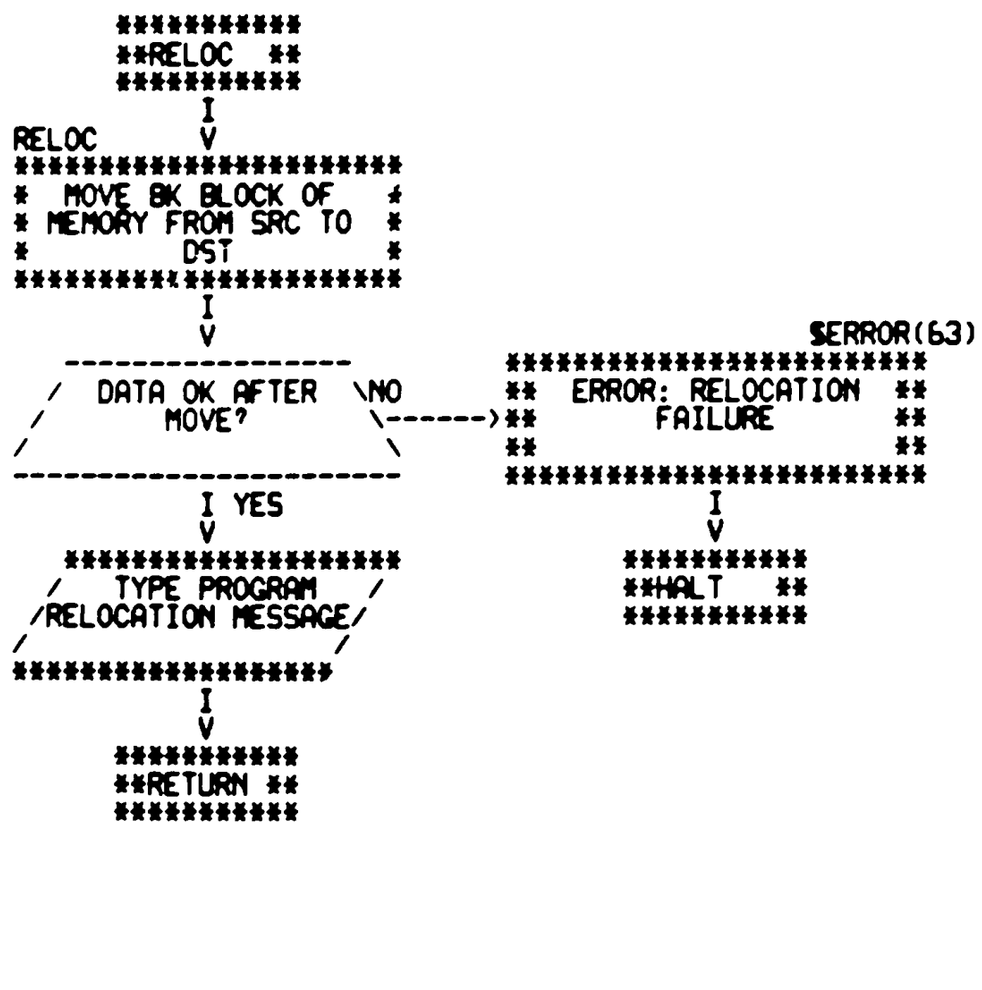

**PHYADR **
PHYADR
* GET VIRTUAL *
* ADDRESS (FROM *
* R2) *
MEMORY MANAGEMENT
AVAILABLE
NO
YES
ADD INDEX FACTOR FROM
* KIPAR2 TO GET *
* PHYSICAL ADR *
**RETURN **

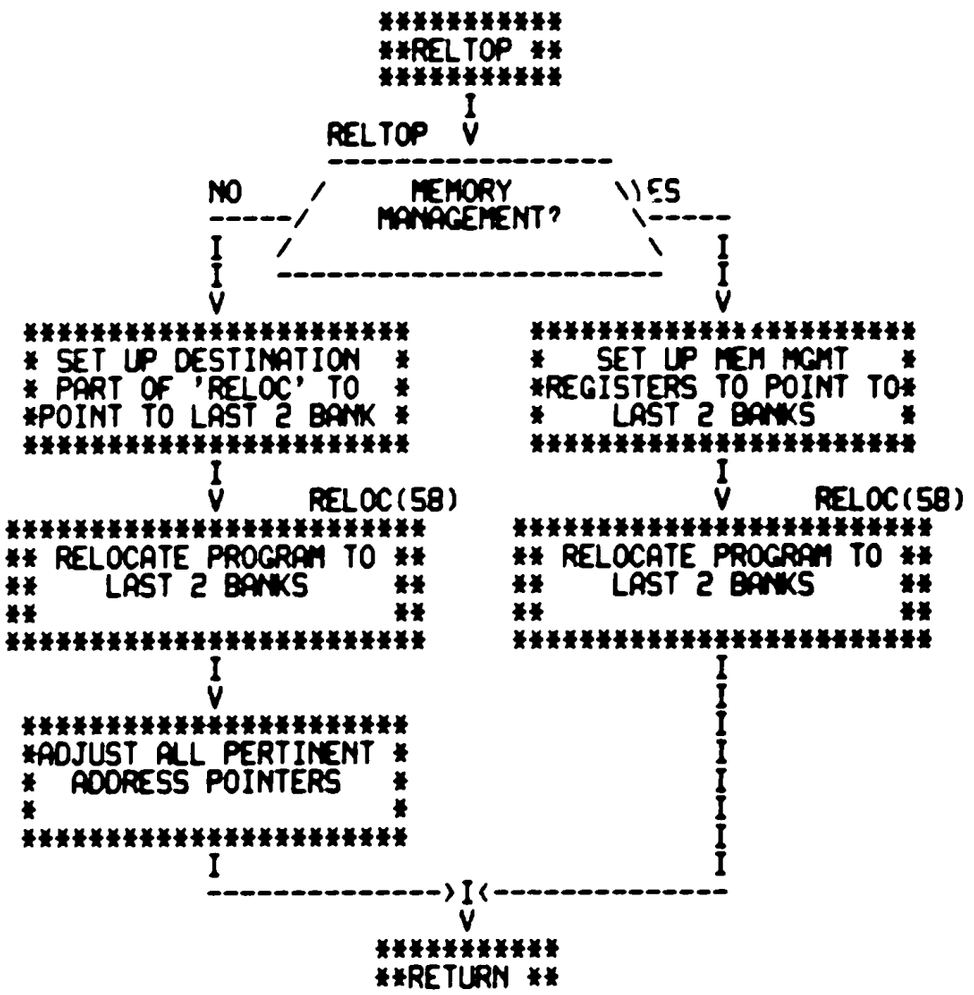

**BANKNO **
BANKNO
* CALCULATE BANK # *
* USING TEST MAP BANK *
* POINTER *
**RETURN **

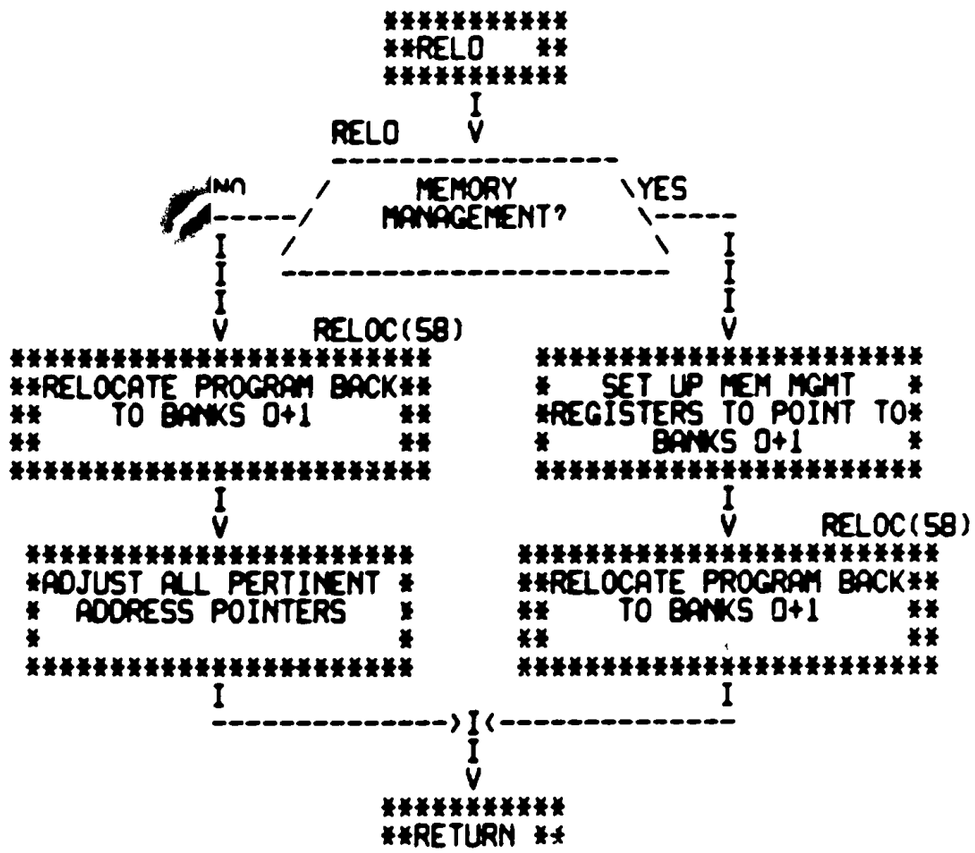

**SETCON **
SETCON
INITMM(S5)
** INITIALIZE ADDRESS **
** POINTERS **
* PUT THE CONTENTS OF *
* R0 INTO MEMORY *
MMUP(S6)
** UPDATE ADDRESS **
** POINTERS **
MORE
MEMORY
IDONE
**RETURN **

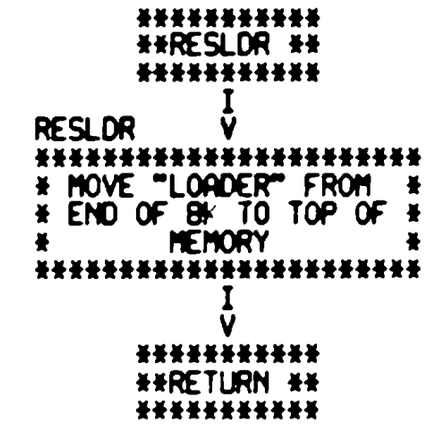

**ROTATE **
ROTATE
*ROTATE C-BIT THROUGH *
* 16 BIT WORD. *
**RETURN **

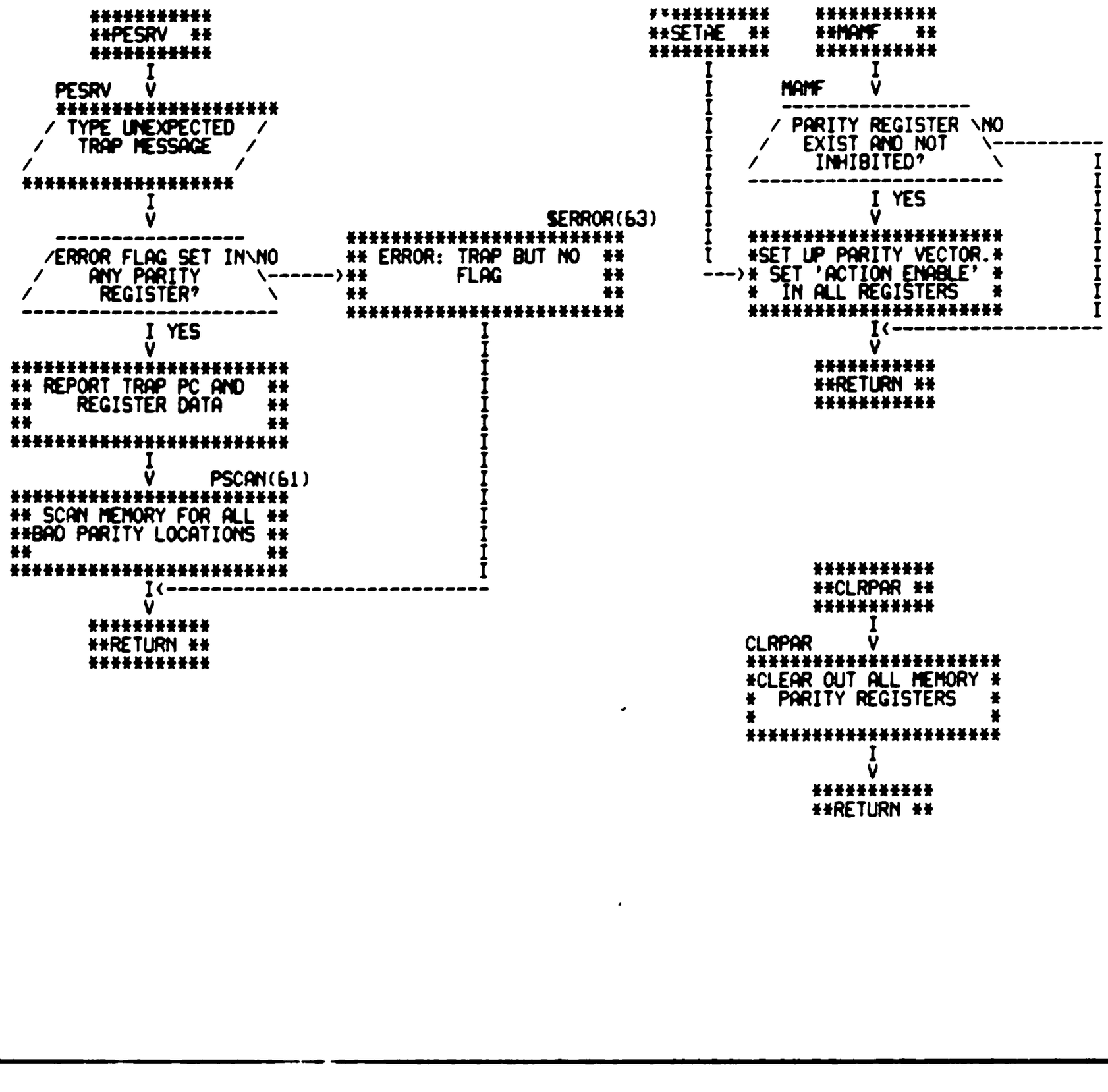

**W3X9 **
W3X9
WRITE 256 WORD WITH 4
* OF A PATTERN THEN 4 *
* OF ANOTHER *
**RETURN **

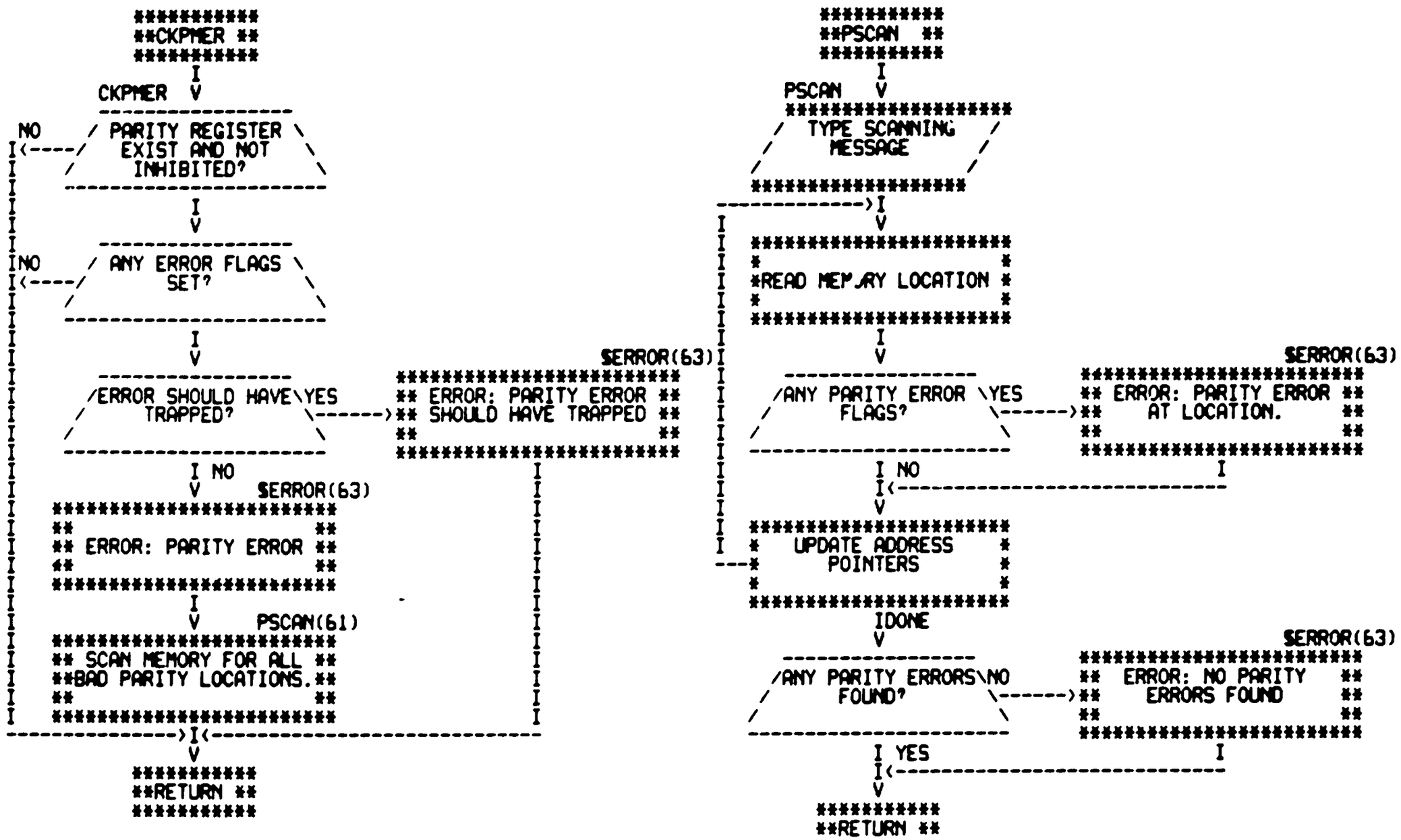

**RELOC **
RELOC
MOVE BK BLOCK OF MEMORY FROM SRC TO DST
DATA OK AFTER MOVE?
NO
YES
SERROR(63)
** ERROR: RELOCATION FAILURE **
**HALT **
TYPE PROGRAM RELOCATION MESSAGE
**RETURN **

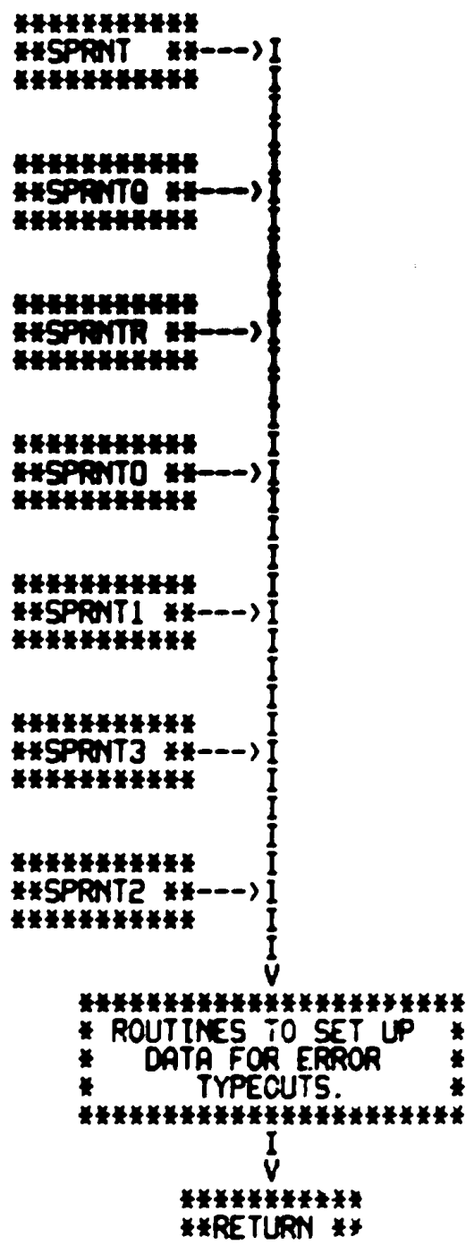

**RELTOP **
RELTOP
NO
MEMORY MANAGEMENT?
YES
SET UP DESTINATION PART OF 'RELOC' TO POINT TO LAST 2 BANK
SET UP MEM MGMT REGISTERS TO POINT TO LAST 2 BANKS
RELOC(58)
RELOC(58)
RELOCATE PROGRAM TO LAST 2 BANKS
RELOCATE PROGRAM TO LAST 2 BANKS
ADJUST ALL PERTINENT ADDRESS POINTERS
**RETURN **

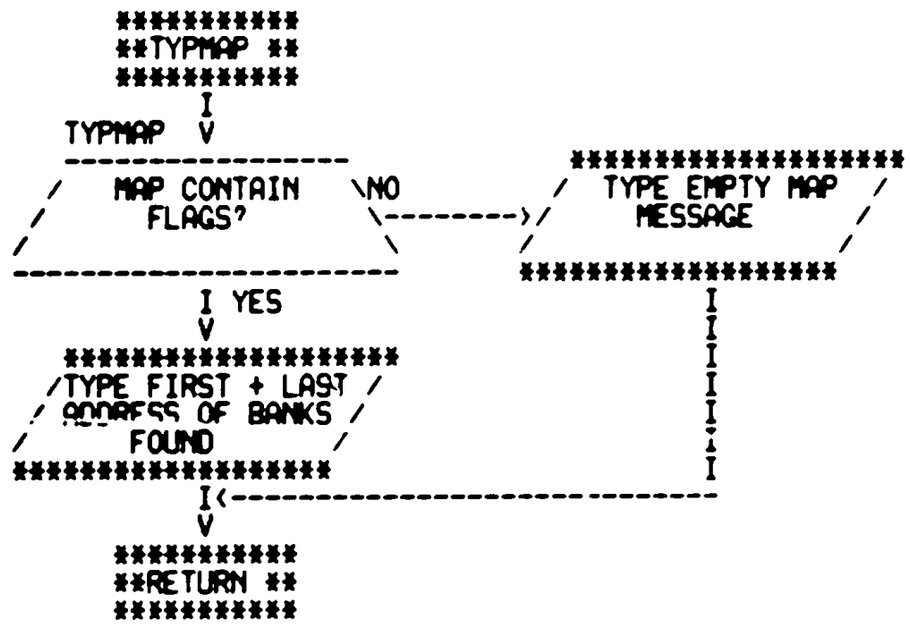

**RELO **
RELO
MEMORY MANAGEMENT?
NO
YES
RELOC(58)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
*ADJUST ALL PERTINENT *
* ADDRESS POINTERS *
* SET UP MEM MGMT *
REGISTERS TO POINT TO
* BANKS 0+1 *
RELOC(58)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
**RETURN **

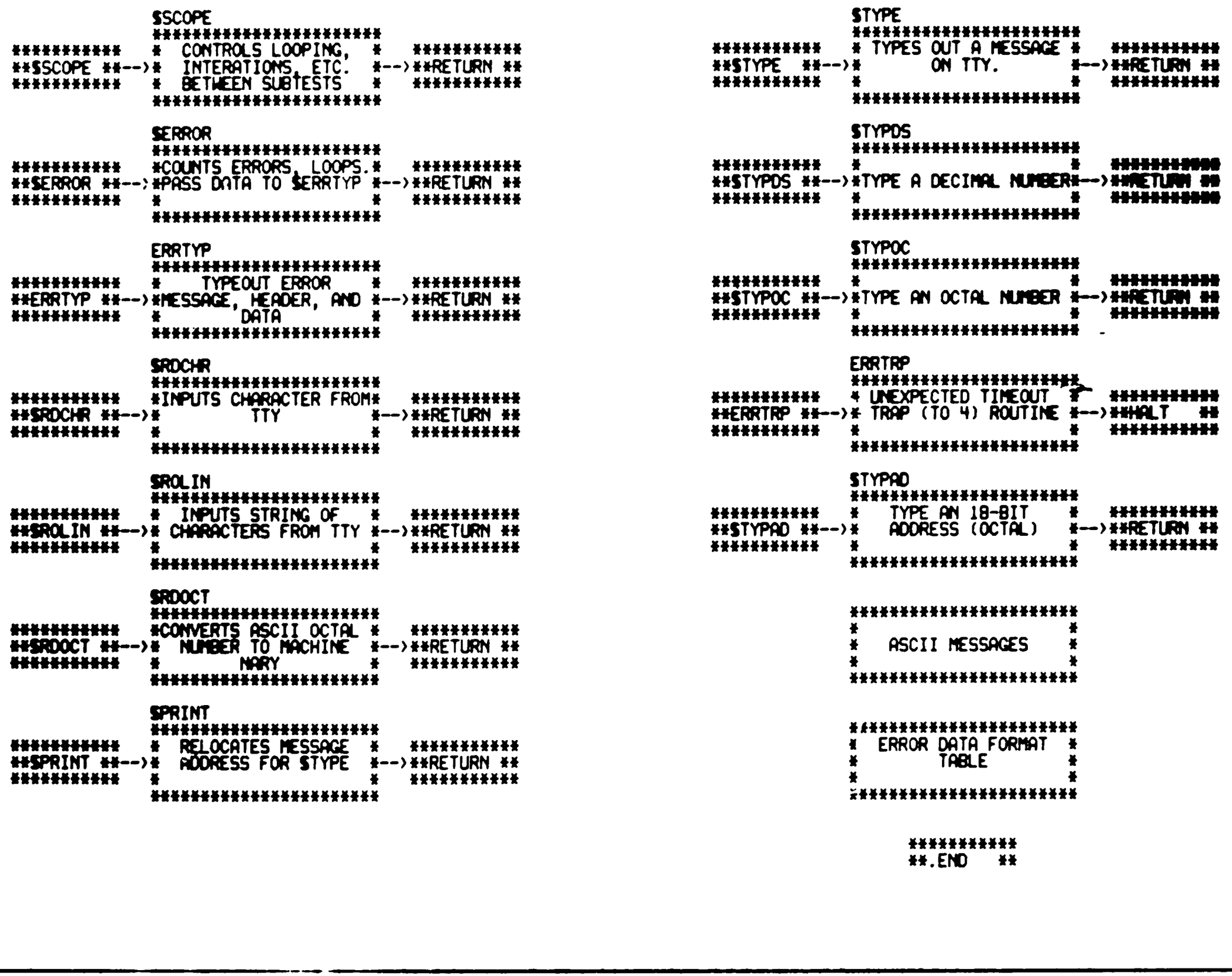

**RESLDR **
RESLDR
* MOVE "LOADER" FROM *
* END OF 8K TO TOP OF *
* MEMORY *
**RETURN **


**PESRV **
PESRV
TYPE UNEXPECTED TRAP MESSAGE
ERROR FLAG SET IN ANY PARITY REGISTER?
NO
YES
** REPORT TRAP PC AND REGISTER DATA **
PSCAN(61)
** SCAN MEMORY FOR ALL **BAD PARITY LOCATIONS **
**RETURN **
SERROR(63)
** ERROR: TRAP BUT NO FLAG **
**SETAE **
MAMF
PARITY REGISTER EXIST AND NOT INHIBITED?
NO
YES
*SET UP PARITY VECTOR. * SET 'ACTION ENABLE' * IN ALL REGISTERS *
**MAMF **
**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(63)
** ERROR: PARITY ERROR **
** SHOULD HAVE TRAPPED **
I NO
$ERROR(63)
** ERROR: PARITY ERROR **
PSCAN(61)
** SCAN MEMORY FOR ALL **
BAD PARITY LOCATIONS.
**RETURN **
**PSCAN **
PSCAN
TYPE SCANNING MESSAGE
*READ MEMORY LOCATION *
ANY PARITY ERROR FLAGS?
YES
$ERROR(63)
** ERROR: PARITY ERROR **
** AT LOCATION. **
I NO
* UPDATE ADDRESS POINTERS *
IDONE
ANY PARITY ERRORS FOUND?
NO
$ERROR(63)
** ERROR: NO PARITY ERRORS FOUND **
I YES
**RETURN **


**SPRNT **
**SPRNTQ **
**SPRNTR **
**SPRNTO **
**SPRNT1 **
**SPRNT3 **
**SPRNT2 **
* ROUTINES TO SET UP *
* DATA FOR ERROR *
* TYPECUTS. *
**RETURN **


**TYPMAP **
TYPMAP
MAP CONTAIN FLAGS?
NO
TYPE EMPTY MAP MESSAGE
YES
TYPE FIRST + LAST ADDRESS OF BANKS FOUND
**RETURN **


$SCOPE
** $SCOPE **
CONTROLS LOOPING, INTERATIONS, ETC. BETWEEN SUBTESTS
**RETURN **
$ERROR
** $ERROR **
COUNTS ERRORS, LOOPS. PASS DATA TO $ERRTYP
**RETURN **
ERRTYP
** ERRTYP **
TYPEOUT ERROR MESSAGE, HEADER, AND DATA
**RETURN **
$RDCHR
** $RDCHR **
INPUTS CHARACTER FROM TTY
**RETURN **
$ROLIN
** $ROLIN **
INPUTS STRING OF CHARACTERS FROM TTY
**RETURN **
$RDOCT
** $RDOCT **
CONVERTS ASCII OCTAL NUMBER TO MACHINE BINARY
**RETURN **
$PRINT
** $PRINT **
RELOCATES MESSAGE ADDRESS FOR $TYPE
**RETURN **
$TYPE
** $TYPE **
TYPES OUT A MESSAGE ON TTY.
**RETURN **
$TYPDS
** $TYPDS **
TYPE A DECIMAL NUMBER
**RETURN **
$TYPOC
** $TYPOC **
TYPE AN OCTAL NUMBER
**RETURN **
ERRTRP
** ERRTRP **
UNEXPECTED TIMEOUT TRAP (TO 4) ROUTINE
**HALT **
$TYPAD
** $TYPAD **
TYPE AN 18-BIT ADDRESS (OCTAL)
**RETURN **
ASCII MESSAGES
ERROR DATA FORMAT TABLE
** .END **

```
BANKNO      57#   57
CKPMER      24    35    56    61#   61
CLRPAR      60#   60
CTRLS       14
DONE        53
ERRTRP      63    63#
ERRTYP      63    63#
EXIT        54
GMPR        08    12    12
GMPRA       12
GMPRB       12
GMPRC       12
HALT        03    04    04    08    11    58    63
INITDN      17    18    19    21    21    55#   55
INITEX      55
INITMM      17    18    19    20    20    22    23    24    25    26    27    27    28    29    29    30    31    31
            32    33    33    34    35    39    40    41    42    43    44    45    46    49    49    51    51    55#
            55    57
MAMF        38    60#   60
MANUAL      12    15#   15
MANUL1      15
MANUL2      16
MAPRB       13
MAUT        40    41    42    43    44    45
MMDOWN      17    18    19    21    21    55    56#   56
MMINIT      09    55#   55
MMUP        17    18    19    20    20    22    23    24    25    26    27    27    28    29    29    30    31    31
            32    33    33    34    38    39    40    41    42    43    44    45    49    50    51    52    56#   56
            57
PBTRP       36
PESRV       60#   60
PHYADR      57#   57
PSCAN       60    61    61#   61
RELO        03    53    59#   59
RELOC       58#   58    58    58    59    59
RELTOP      53    58#   58
RESCHK      14
RESLDR      03    53    59#   59
REST1       02
REST2       02
RESTAR      02#   02    06
RESTOR      02#
RETURN      04    40    41    42    43    44    45    55    55    56    56    56    56    57    57    57    57    57
            58    58    59    59    60    60    60    61    61    62    62    63    63    63    63    63    63    63
            63    63    63    63
ROTATE      25    26    57#   57
SELECT      06#
SETAE       35    60#
SETCON      25    26    35    57#   57
SPRNT       62#
SPRNTO      62#
```

```
SPRNT2      62#
SPRNT3      62#
SPRNT0      62#
SPRNTR      62#
START       26#
START1      03    16#   16    53    54
STARTA      02    06
TIMOUT      09#
TMAP        13
TST1        17
TST11       25
TST12       26
TST13       27
TST14       29
TST15       31
TST16       33
TST17       35
TST2        18
TST20       35    39#   39
TST21       40
TST22       41
TST23       42
TST24       43
TST25       44
TST26       45
TST27       46
TST3        19
TST4        20
TST5        21
TST6        22
TST7        23
TYPMAP      13    62#   62
W3X9        27    29    31    33    57#   57
WWPB1       35    35#   37
WWPB2       36    36#   37
WWPB3       37
WWPB4       36    36    37    37#
WWPB5       35    35    37    37#
WWPBT       35    35#   38
$EOP        54
$ERROR      14    14    14    17    18    19    20    21    23    24    25    26    27    28    28    28    29    30
            30    30    31    32    32    32    33    34    34    34    36    36    36    36    36    37    37    37
            39    40    41    42    43    44    45    46    46    48    50    52    58    60    61    61    61    61
            63    63#
$ILLUP      04#
$MMUP       47
$PRINT      04    09    10    10    15    63    63#
$PWRON      04#
$PWRUP      04#
$RDCHR      63    63#
$RDOCT      15    15    63    63#
```

	63#			
$SCOPE	63	63#		
$TYPAD	10	10	63	63#
$TYPDS	63	63#		
$TYPE	63	63#		
$TYPOC	63	63#		

```
                        .TITLE  MAINDEC-11-DZQMC-D-D:  0-124K MEMORY EXERCISER, 16K VER
                        ;*COPYRIGHT (C) 1975,1977
                        ;*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-C3), JAN 19, 1977.
                        ;*

                                .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.
                        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
                        000174                          .=174
000174  000000                                  DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
000176  000000                                  SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  002632                                  JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
000204  000167  002430                                  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  003346                                  JMP     TIMOUT          ;TYPE OUT MEMORY MAP, BYTE BY BYTE.

                        000004                  .=ERRVEC
000004  025022                                  .WORD   ERRTRP
000006  000000                                  .WORD   0

                                                .SBTTL  ACT11 HOOKS

                                                ;;**************************************************************
                                                ;HOOKS REQUIRED BY ACT11
                        000010                  $SVPC=.                         ;SAVE PC
                        000046                  .=46
000046  014136                                  $ENDAD                          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                        000052                  .=52
000052  040000                                  .WORD   BIT14                   ;;2)SET LOC.52 TO BIT14
                        000010                  .=$SVPC                         ;; RESTORE PC
```

```
                000300                          .=300
                                        ;;*****************************************************************
                                        ;*  THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
                                        ;*   THEY CAN BE PROTECTED BY SELECTING SWDS (SEE DOCUMENT FOR USE OF SWDS).
                                        ;*   THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
                                        ;*   MANAGEMENT IS DISABLED BY "CONSOLE START".
                                        ;*****************************************************************
000300  005005                  RESTAR: CLR     RS              ;CLEAR FLAG TO INDICATE RESTART.
000302  000401                          BR      REST1           ;GO RESTORE PROGRAM BEFORE RESTARTING.
000304  010705                  RESTOR: MOV     PC,     RS      ;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  002322                  JMP     STARTA          ;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  000052          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.
```

```
000540  001402                          BEQ     21$             ;BR IF IN BANKS 0 AND 1.
000542  004767  016222                  JSR     PC,     RELD    ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
000546  005705                  21$:    TST     R5              ;CHECK RESTART/RESTORE FLAG.
000550  001006                          BNE     22$             ;BR IF RESTORE.
000552  005067  000412                  CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
000556  105067  000320                  CLRB    $TSTNM
000562  000167  005260                  JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
000566  004767  016404          22$:    JSR     PC,     RESLDR  ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
000572  000000                          HALT                    ;HALT AFTER RESTORING THE LOADERS.
000574  000167  002046                  JMP     STARTA          ;CONTINUE WILL RESTART THE PROGRAM.
                                ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
                                ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
                                ;*  CIRCUMSTANCES.
000600  000000          RELOCF: .WORD   0       ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
000602  000000  000000  PRGMAP: .WORD   0,0     ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
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  022452                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000746  025547          $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  003000          $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
```

```
                                ;
                                ;APT MAILBOX-ETABLE
                                ;
001212  000000          SPASS:  .WORD   APASS   ;;PASS COUNT
001214  000000          SDEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
001216  000000          SUNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
001220  000000          SMSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
001222  000000          SMSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
001224                  SETABLE:                ;;APT ENVIRONMENT TABLE
001224      000         SENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
001225      000         SENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
001226  000000          SSWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
001230  000000          SUSWR:  .WORD   AUSWR   ;;USER SWITCHES
001232  000000          SCPUOP: .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         SMAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
001235      000         SMTYP1: .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          SMADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
                        ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001240      000         SMAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
001241      000         SMTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
001242  000000          SMADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
001244      000         SMAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
001245      000         SMTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
001246  000000          SMADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
001250      000         SMAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
001251      000         SMTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
001252  000000          SMADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
001254  000000          SVECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001256  000000          SVECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001260  000000          SBASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001262  000000          SDEVM:  .WORD   ADEVM   ;;DEVICE MAP
001264  000000          SCDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
001266  000000          SCDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
001270  000000          SDDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
001272  000000          SDDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
001274  000000          SDDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
001276  000000          SDDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
001300  000000          SDDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
001302  000000          SDDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
001304  000000          SDDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
001306  000000          SDDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
001310  000000          SDDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
001312  000000          SDDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
001314  000000          SDDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
001316  000000          SDDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
001320  000000          SDDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
001322  000000          SDDW13  .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
001324  000000          SDDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
001326  000000          SDDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
```

```
                                        SETEND:
                                        .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 (SERTYP).
001516  070032          RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
                                                ;NOTE:  FOR MS11 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 "INITDM" 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.
001612  000004          WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND

                        ;;*****************************************************************
                        ;*  RELATIVE ADDRESSING TABLE.
                        ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
                        ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMENT TAGS.
                        ;;*****************************************************************
001614                  RADTAB:
```

```
557  001614  001100                  .STACK: STACK           ;STACK POINTER INITIAL ADDRESS.
558  001616  001516                  .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
559  001620  002070                  .MPRO:  MPRO            ;MEMORY PARITY REGISTER TABLE ADDRESS.
560  001622  002270                  .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
561  001624  012000                  .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
562  001626  002042                  .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
563  001630  017336                  .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
564  001632  002332                  .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
565  001634  000010                  .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
566  001636  013722                  .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
567                                   ;;* ***************************************************************
568                                   ;;  DATA CONTAINERS FOR ERROR PRINTOUT.
569                                   ;;***************************************************************
570  001640  001116  001120  001124  DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0
571  001646  001126  000000
572  001652  001514  001116  001120  DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
573  001660  001124  001126  000000
574  001666  001514  001116  001120  DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0
575  001674  001124  000000
576  001700  001514  001116  001160  DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0
577  001706  001120  000000
578  001712  001514  001116  001120  DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
579  001720  001160  001124  001126
580  001726  000000
581  001730  001514  001116  001160  DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
582  001736  001120  001124  001126
583  001744  000000
584  001746  001514  001116  001120  DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
585  001754  001122  001124  001126
586  001762  000000
587  001764  001514  001116  001122  DT24:   $VERPC,$ERRPC,$BDADR,0
588  001772  000000
589  001774  001514  001116  001122  DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
590  002002  001160  001162  000000
591  002010  001514  001116  001160  DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
592  002016  001162  000000
593  002022  001160  001162  001120  DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
594  002030  001126  000000
595  002034  001166  000000          DT31:   $TMP3,0
596  002040  177777                          .WORD   -1      ;TABLE TERMINATOR.
597
598                                   .SBTTL  MEMORY PARITY PATTERNS TABLE
599                                   ;;***************************************************************
600                                   ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
601                                   ;;***************************************************************
602
603  002042  125325                  MPPATS: 125325                  ;EVEN,000
604  002044  152652                          152652                  ;000,EVEN
605  002046  052452                          052452                  ;EVEN,000
606  002050  025125                          025125                  ;000,EVEN
607  002052  102070                          102070                  ;EVEN,EVEN
608  002054  072527                          072527                  ;000,000
609  002056  177777                          177777                  ;EVEN,EVEN
610  002060  107030                          107030                  ;000,000
611  002062  152525                          152525                  ;000,EVEN
612  002064  000000                          0                       ;EXTRA PATTERN HOLDER FOR
```

```
                                                        ;FUTURE USE
002066  000000          MPEND:  0                     ;TABLE TERMINATOR


                        .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
                        ;//////////////////////////////////////////////////////////////////////
                        ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
                        ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
                        ;*
                        ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
                        ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
                        ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
                        ;*  THE DEVICE ADDRESS,  EACH BIT REPRESENTING A 4K BLOCK.  I.E.
                        ;*  FIRST WORD BIT0 = 0 - 4K,  BIT1 = 4 - 8K,...  BIT15 = 60 - 64K
                        ;*  SECOND WORD BIT0 = 64 - 68K,... BIT14 = 120 - 124K.
                        ;//////////////////////////////////////////////////////////////////////
002070  172101  MPR0:    172100 +1                    ;PARITY STATUS REGISTER
002072  000000            0                          ;CONTROL MAP (LOW 64K)
002074  000000            0                          ;CONTROL MAP (HIGH 64K)
002076  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002100  172103  MPR1:    172102 +1                    ;PARITY STATUS REGISTER
002102  000000            0                          ;CONTROL MAP (LOW 64K)
002104  000000            0                          ;CONTROL MAP (HIGH 64K)
002106  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002110  172105  MPR2:    172104 +1                    ;PARITY STATUS REGISTER
002112  000000            0                          ;CONTROL MAP (LOW 64K)
002114  000000            0                          ;CONTROL MAP (HIGH 64K)
002116  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002120  172107  MPR3:    172106 +1                    ;PARITY STATUS REGISTER
002122  000000            0                          ;CONTROL MAP (LOW 64K)
002124  000000            0                          ;CONTROL MAP (HIGH 64K)
002126  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002130  172111  MPR4:    172110 +1                    ;PARITY STATUS REGISTER
002132  000000            0                          ;CONTROL MAP (LOW 64K)
002134  000000            0                          ;CONTROL MAP (HIGH 64K)
002136  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002140  172113  MPR5:    172112 +1                    ;PARITY STATUS REGISTER
002142  000000            0                          ;CONTROL MAP (LOW 64K)
002144  000000            0                          ;CONTROL MAP (HIGH 64K)
002146  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002150  172115  MPR6:    172114 +1                    ;PARITY STATUS REGISTER
002152  000000            0                          ;CONTROL MAP (LOW 64K)
002154  000000            0                          ;CONTROL MAP (HIGH 64K)
002156  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002160  172117  MPR7:    172116 +1                    ;PARITY STATUS REGISTER
002162  000000            0                          ;CONTROL MAP (LOW 64K)
002164  000000            0                          ;CONTROL MAP (HIGH 64K)
002166  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002170  172121  MPR8:    172120 +1                    ;PARITY STATUS REGISTER
002172  000000            0                          ;CONTROL MAP (LOW 64K)
002174  000000            0                          ;CONTROL MAP (HIGH 64K)
002176  000000            0                          ;MASK FOR MOS,CORE,MS11-K
002200  172123  MPR9:    172122 +1                    ;PARITY STATUS REGISTER
002202  000000            0                          ;CONTROL MAP (LOW 64K)
002204  000000            0                          ;CONTROL MAP (HIGH 64K)
002206  000000            0                          ;MASK FOR MOS,CORE,MS11-K
```

```
                        MPR10:  172124 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        MPR11:  172126 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        MPR12:  172130 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        MPR13:  172132 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        MPR14:  172134 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        MPR15:  172136 +1               ;PARITY STATUS REGISTER
                                0                       ;CONTROL MAP (LOW 64K)
                                0                       ;CONTROL MAP (HIGH 64K)
                                0                       ;MASK FOR MOS,CORE,MS11-K
                        ;THIS IS THE END OF THE TABLE !
002270  000021         MPRX:   .BLKW   17.             ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
                                                        ; (THE EXTRA WORD IS FOR A TERMINATOR.)
```

```
                                .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


002332                          $ERRTB:
                                ;*  ITEM 1
002332  026714                      DM1             ;PARITY REGISTER DATA ERROR.
002334  030273                      DH1             ;PC,REG,S/B,WAS
002336  001640                      DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
002340  030640                      DF1             ;16,18,16,16
                                ;*  ITEM 2
002342  026750                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002344  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002346  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002350  030644                      DF2             ;16,18,18,16,16
                                ;*  ITEM 3
002352  026750                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002354  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002356  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002360  030651                      DF3             ;16,18,18,8,8
                                ;*  ITEM 4
002362  027004                      DM4             ;CONSTANT DATA ERROR(TST6-10).
002364  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002366  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002370  030644                      DF2             ;16,18,18,16,16
                                ;*  ITEM 5
002372  027042                      DM5             ;ROTATING BIT ERROR(TST11-12).
002374  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002376  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002400  030644                      DF2             ;16,18,18,16,16
                                ;*  ITEM 6
002402  027100                      DM6             ;MOS REFRESH TEST ERROR (TST30-31).
002404  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002406  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002410  030644                      DF2             ;16,18,18,16,16
                                ;*  ITEM 7
002412  027144                      DM7             ;3 XOR 9 PATTERN ERROR(TST13-16).
002414  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002416  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002420  030644                      DF2             ;16,18,18,16,16
                                ;*  ITEM 10
002422  027205                      DM10            ;MARCHING 1'S AND 0'S ERROR(TST27).
002424  030312                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002426  001652                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002430  030644                      DF2             ;16,'8,18,16,16
                                ;*  ITEM 11
```

```
753  002432  027251                    DM11    ;PARITY MEMORY ADDRESS ERROR(TST17).
754  002434  030312                    DH2     ;V/PC,P/PC,MA,S/B,WAS
755  002436  001652                    DT2     ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
756  002440  030651                    DF3     ;16,18,18,8,8
757                          ;*  ITEM 12
758  002442  027315                    DM12    ;DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
759  002444  030337                    DH12    ;V/PC,P/PC,MA,S/B
760  002446  001666                    DT12    ;$VERPC,$ERRPC,$GDADR,$GDDAT
761  002450  030651                    DF3     ;16,18,18,8
762                          ;*  ITEM 13
763  002452  027371                    DM13    ;WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
764  002454  030337                    DH12    ;V/PC,P/PC,MA,S/B
765  002456  001666                    DT12    ;$VERPC,$ERRPC,$GDADR,$GDDAT
766  002460  030651                    DF3     ;16,18,18,8
767                          ;*  ITEM 14
768  002462  027461                    DM14    ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
769  002464  030360                    DH14    ;V/PC,P/PC,REG,MA
770  002466  001700                    DT14    ;$VERPC,$ERRPC,$TMP0,$GDADR
771  002470  030656                    DF14    ;16,18,18,18
772                          ;*  ITEM 15
773  002472  026714                    DM1     ;PARITY REGISTER DATA ERROR.
774  002474  030401                    DH15    ;V/PC,P/PC,MAUT,REG,S/B,WAS
775  002476  001712                    DT15    ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
776  002500  030656                    DF14    ;16,18,18,18,16,16
777                          ;*  ITEM 16
778  002502  027560                    DM16    ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
779  002504  030360                    DH14    ;V/PC,P/PC,REG,MA
780  002506  001700                    DT14    ;$VERPC,$ERRPC,$TMP0,$GDADR
781  002510  030656                    DF14    ;16,18,18,18
782                          ;*  ITEM 17
783  002512  027637                    DM17    ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
784                                            ;  TRAPPED(TST21).
785  002514  030312                    DH2     ;V/PC,P/PC,MA,S/B,WAS
786  002516  001652                    DT2     ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
787  002520  030651                    DF3     ;16,18,18,8,8
788                          ;*  ITEM 20
789  002522  027735                    DM20    ;RANDOM DATA ERROR(TST20).
790  002524  030312                    DH2     ;V/PC,P/PC,MA,S/B,WAS
791  002526  001652                    DT2     ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
792  002530  030644                    DF2     ;16,18,18,16,16
793                          ;*  ITEM 21
794  002532  027767                    DM21    ;INSTRUCTION EXICUTION ERROR(TST21-26).
795  002534  030434                    DH21    ;V/PC,P/PC,IUT,MA,S/B,WAS
796  002536  001730                    DT21    ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
797  002540  030664                    DF21    ;16,18,16,18,16,16
798                          ;*  ITEM 22     ;NOT USED
799                          ;*  ITEM 23
800  002542  030036                    DM23    ;PROGRAM CODE CHANGED WHEN RELOCATED.
801  002544  030465                    DH23    ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
802  002546  001746                    DT23    ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
803  002550  030656                    DF14    ;16,18,18,18,16,16
804                          ;*  ITEM 24
805  002552  030103                    DM24    ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
806  002554  030525                    DH24    ;V/PC,P/PC,TRP/PC
807  002556  001764                    DT24    ;$VERPC,$ERRPC,$BDADR
808  002560  030656                    DF14    ;16,18,18
```

```
809
810  002562  030157                ;*  ITEM 25
811  002564  030546                        DM25            ;TRAPPED TO 114.
812  002566  001774                        DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
813  002570  030656                        DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
814                                        DF14            ;16,18,18,18,16
815  002572  030177                ;*  ITEM 26
816  002574  030577                        DM26            ;FAILED TO TRAP.
817  002576  002010                        DH26            ;V/PC,P/PC,REG,WAS
818  002600  030644                        DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
819                                        DF2             ;16,18,18,16
820  002602  030217                ;*  ITEM 27
821  002604  030577                        DM27            ;(ACTION ENABLE WASN'T SET).
822  002606  002010                        DH26            ;V/PC,P/PC,REG,WAS
823  002610  030644                        DT26            ;$VERPC,$ERRPC,$TMP0,$BDDAT
824                                        DF2             ;16,18,18,16
825  002612  000000                ;*  ITEM 30
826  002614  030621                        0               ;NO MESSAGE.
827  002616  002022                        DH30            ;REG,WAS,MA,WAS
828  002620  030672                        DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
829                                        DF30            ;18,16,18,8
830  002622  030253                ;*  ITEM 31
831  002624  000000                        DM31            ;TRAPPED TO 4
832  002626  002034                        0               ;NO HEADER
833  002630  030672                        DT31            ;$TMP3
                                        DF30            ;18
```

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

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

841  002632  105067  176720          START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
842  002636  000403                          BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
843  002640  112767  177777  176710  SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
844  002646                          STARTA:
845                                  .SBTTL  INITIALIZE THE COMMON TAGS
846                                  ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
847  002646  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
848  002652  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
849  002654  022706  001140                  CMP     #SWR,R6 ;;DONE?
850  002660  001374                          BNE     .-6             ;;LOOP BACK IF NO
851  002662  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
852                                  ;;INITIALIZE A FEW VECTORS
853  002666  012737  000610  000024          MOV     #SPWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
854  002674  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
855  002702  016767  011164  011154          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
856                                  ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
857                                  ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
858  002710  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
859  002714  012737  002750  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
860  002722  012767  177570  176210          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
861  002730  012767  177570  176204          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
862  002736  022777  177777  176174          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
863  002744  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
864                                                                  ;;AND  THE HARDWARE SWR IS NOT = -1
865  002746  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
866  002750  012716  002756          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
867  002754  000002                          RTI
868  002756  012767  000176  176154  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
869  002764  012767  000174  176150          MOV     #DISPREG,DISPLAY
870  002772  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
871
872  002776  005067  176210                  CLR     $PASS           ;;CLEAR PASS COUNT
873  003002  132767  000200  176215          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
874  003010  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
875  003012  012767  001226  176120          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
876  003020                          67$:
877  003020  005067  176476                  CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
878  003024  005077  176112                  CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
879                                  .SBTTL  TYPE PROGRAM NAME
880                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
881  003030  005227  177777                  INC     #-1             ;;FIRST TIME?
882  003034  001046                          BNE     68$             ;;BRANCH IF NO
883  003036  022737  014136  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
884  003044  001442                          BEQ     68$             ;;BRANCH IF YES
885  003046  004567  020346                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
886  003052  003124                          .WORD   69$             ;ADDRESS OF MESSAGE TO BE TYPED
887                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
888  003054  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
889  003060  001015                          BNE     70$             ;;BRANCH IF YES
```

```
 890  003062  126727  176136  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
 891  003070  001411                          BEQ     70$             ;;BRANCH IF YES
 892  003072  026727  176042  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
 893  003100  001010                          BNE     71$             ;;BRANCH IF NO
 894                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
 895                                  ;*  WITHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 896  003102  013746  177776                  MOV     @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
 897  003106  004767  017232                  JSR     PC,    $GTSWR   ;GO TO THE SUBROUTINE
 898  003112  000403                          BR      71$
 899  003114  112767  000001  176012  70$:    MOVB    #1,$AUTO8       ;;SET AUTO-MODE INDICATOR
 900  003122                          71$:
 901  003122  000413                          BR      66$             ;;GET OVER THE ASCIZ
 902                                  ;;69$:  .ASCIZ  <CRLF>'MAINDEC-11-DZQMC-D'<CRLF>
 903  003152                          66$:
 904  003152  010700                          MOV     PC,    R0       ;GET CURRENT PROGRAM COUNTER.
 905  003154  022700  003154                  CMP     #.,    R0       ;CHECK IF THE PROGRAM IS RELOCATED.
 906  003160  001402                          BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
 907  003162  000167  175112                  JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
 908  003166  012767  000003  175406  10$:    MOV     #3,    PRGMAP   ;INITIALIZE PROGRAM MAP....LO 64K.
 909  003174  005067  175404                  CLR     PRGMAP+2        ;...HI 64K.
 910  003200  005067  175374                  CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
 911  003204  004767  014046                  JSR     PC,SAVLDR       ;GO SAVE LOADERS
 912
 913                                  ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
 914  003210  005067  175372                  CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG
 915  003214  032777  010000  175716          BIT     #SW12, @SWR     ;CHECK FOR INHIBIT KT11 SWITCH
 916  003222  001014                          BNE     IMPCK           ;BRANCH IF SET
 917  003224  012737  003254  000004          MOV     #IMPCK,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
 918  003232  005037  177572                  CLR     @#SR0           ;CLEAR MEM MGMT STATUS REG
 919  003236  004767  010744                  JSR     PC,    MMINIT   ;MEM MGMT INITIALIZATION ROUTINE.
 920  003242  005267  175340                  INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
 921  003246  004567  020146                  JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 922  003252  025262                          .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
 923                                                                  ;"KT11 AVAILABLE"
 924
 925                                  ;*      CHECK IF 11/60 CACHE PRESENT,IF SO TURN IT OFF!!!
 926  003254  012706  001100          IMPCK:  MOV     #STACK, SP
 927  003260  012737  003274  000004          MOV     #MAPMEM,@#ERRVEC
 928  003266  052767  000014  174452          BIS     #14,    IMPCHE
 929
 930                                  ;;*****************************************************************
 931                                  ;*  ROUTINE TO MAP ALL OF MEMORY.
 932                                  ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
 933                                  ;*      R0 = MEMMAP POINTER...LO 64K.
 934                                  ;*      R1 = MEMMAP POINTER...HI 64K.
 935                                  ;*      R2 = ADDRESS POINTER
 936                                  ;*      R3 = BANK POINTER...LO 64K.
 937                                  ;*      R4 = BANK POINTER...HI 64K.
 938                                  ;*      R5 = SCRATCH REGISTER.
 939                                  ;;*****************************************************************
 940  003274  012706  001100          MAPMEM: MOV     #STACK, SP      ;RESET THE STACK
 941  003300  012700  001524                  MOV     #MEMMAP,R0      ;SET UP MEMORY MAP POINTER...LO 64K.
 942  003304  012701  001526                  MOV     #MEMMAP+2,R1    ;...HI 64K.
 943  003310  005010                          CLR     (R0)            ;CLR MEMORY MAP...LO 64K.
 944  003312  005011                          CLR     (R1)            ;...HI 64K.
 945  003314  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
```

```
  946  003316  012703  000001              MOV     #1,     R3      ;SETUP 4K BANK POINTER...LO 64K.
  947  003322  005004                      CLR     R4              ;...HI 64K.
  948  003324  005067  175636              CLR     $TMP3           ;INIT TEMPORARY HIGH ADDRESS BITS.
  949  003330  004567  020064              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  950  003334  025327                      .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
  951                                                             ;"MEMORY MAP:"
  952  003336  012737  003452  000004      MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
  953  003344  011222          1$:         MOV     (R2),   (R2)+   ;READ+WRITE ALL MEMORY
  954  003346  032702  017777              BIT     #MASK4K,R2      ;CHECK FOR 4K BOUNDRY
  955  003352  001374                      BNE     1$              ;BRANCH IF MORE IN BANK
  956  003354  050310                      BIS     R3,     (R0)    ;SET FLAG FOR BANK...LO 64K.
  957  003356  050411                      BIS     R4,     (R1)    ;...HI 64K.
  958  003360  010267  175600              MOV     R2,     $TMP2   ;SAVE ADDRESS POINTER.
  959  003364  005367  175574              DEC     $TMP2           ;ADJUST TO LAST ADR, LAST BANK.
  960  003370  005767  175212              TST     MMAVA           ;CHECK FOR MEM MGMT.
  961  003374  001432                      BEQ     3$              ;BR IF NO MEM MGMT.
  962  003376  042767  160000  175560      BIC     #160000,$TMP2   ;CLEAR BANK BITS ON RELATIVE ADDRESS.
  963  003404  013705  172344              MOV     @#KIPAR2,R5     ;SAVE KIPAR2.
  964  003410  005067  175552              CLR     $TMP3           ;MAKE SURE HI BITS ARE INIT.
  965  003414  006305                      ASL     R5              ;SHIFT IT 6 PLACES.
  966  003416  006305                      ASL     R5
  967  003420  006305                      ASL     R5
  968  003422  006305                      ASL     R5
  969  003424  006305                      ASL     R5
  970  003426  006167  175534              ROL     $TMP3
  971  003432  006305                      ASL     R5
  972  003434  006167  175526              ROL     $TMP3
  973  003440  060567  175520              ADD     R5,     $TMP2   ;MAKE LAST ADR PHYSICAL.
  974  003444  005567  175516              ADC     $TMP3
  975  003450  000404                      BR      3$              ;GO TO UPDATE POINTERS.
  976
  977                          ;*  TIMEOUT TRAPS TO HERE
  978  003452  022626          2$:         CMP     (SP)+,  (SP)+   ;RESTORE THE STACK POINTER
  979  003454  052702  017777              BIS     #MASK4K,R2      ;LAST ADDRESS OF 4K BANK
  980  003460  005202                      INC     R2              ;FIRST ADDRESS OF NEXT BANK.
  981  003462  005767  175120  3$:         TST     MMAVA           ;CHECK FOR MEM MGMT
  982  003466  001411                      BEQ     4$              ;BRANCH IF NO MEM MGMT
  983  003470  062737  000200  172344      ADD     #200, @#KIPAR2  ;UPDATE THIRD PAR
  984  003476  012702  040000              MOV     #40000, R2      ;POINT TO START OF THIRD PAR
  985  003502  006303                      ASL     R3              ;UPDATE LO BANK POINTER.
  986  003504  006104                      ROL     R4              ;UPDATE HI BANK POINTER
  987  003506  100316                      BPL     1$              ;BRANCH IF MORE MEMORY TO MAP.
  988  003510  000402                      BR      5$              ;EXIT WHEN DONE.
  989
  990  003512  106303          4$:         ASLB    R3              ;UPDATE MAP POINTER
  991  003514  100313                      BPL     1$              ;BRANCH IF NOT YET DONE
  992  003516  012737  025022  000004  5$: MOV     #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
  993  003524  004767  014570              JSR     PC,     TYPMAP  ;GO TYPE THE MAP.
  994  003530  004567  017664              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  995  003534  001201                      .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
  996  003536  011067  175772              MOV     (R0),   SAVTST  ;SET UP TEST MAP...LO 64K.
  997  003542  011167  175770              MOV     (R1),   SAVTST+2 ;....HI 64K.
  998  003546  011000                      MOV     (R0),   R0      ;GET LOW MEM MAP
  999  003550  042700  177760              BIC     #177760, R0     ;MASK ALL BUT BOTTOM 4 BANKS
 1000  003554  020027  000017              CMP     R0,     #17     ;CHECK THAT BOTTOM 16K IS ALL THERE!
 1001  003560  001530                      BEQ     GMPR            ;BRANCH IF BOTTOM 16K EXISTS
```

```
 1002  003562  004567  017632              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1003  003566  025432                      .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
 1004                                                              ;"FIRST 16K OF MEMORY NOT ALL THERE!"
 1005  003570  000000              6$:     HALT                    ;FATAL ERROR HALT...
 1006                                                              ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
 1007                                      ;;*****************************************************************
 1008                                      ;*  SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
 1009                                      ;*   DATI, DATIP, DATO, AND DATOB.
 1010                                      ;*****************************************************************
 1011  003572  012706  001100      TIMOUT: MOV     #STACK, SP      ;SET UP THE STACK POINTER.
 1012  003576  005067  175004              CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG.
 1013  003602  032777  010000  175330      BIT     #SW12,  @SWR    ;CHECK IF MEM MGMT TO BE INHIBITED.
 1014  003610  001011                      BNE     1$              ;BR IF NO MEM MGMT.
 1015  003612  012737  003634  000004      MOV     #1$,    @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
 1016  003620  005037  177572              CLR     @#SR0           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
 1017  003624  004767  010356              JSR     PC,     MMINIT  ;INIT ALL MEM MGMT REGISTERS.
 1018  003630  005267  174752              INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
 1019  003634                      1$:
 1020  003634  004567  017560              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1021  003640  025345                      .WORD   BYTMES          ;ADDRESS OF MESSAGE TO BE TYPED
 1022                                                              ;"BYTE MEMORY MAP.
 1023  003642  005000                      CLR     R0              ;SET UP TYPE OUT FLAG.
 1024  003644  005002                      CLR     R2              ;SET ADDRESS POINTER TO ZERO.
 1025  003646  012737  003712  000004      MOV     #20$,   @#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
 1026  003654  105712              10$:    TSTB    (R2)            ;DO DATI ONLY.
 1027  003656  032702  000001              BIT     #BIT0,  R2      ;CHECK FOR WORD ADDRESS.
 1028  003662  001001                      BNE     11$             ;BR IF ODD BYTE ADDRESS.
 1029  003664  011212                      MOV     (R2),   (R2)    ;DO DATI, DATO...NOP FOR READ ONLY MAP.
 1030  003666  151212              11$:    BISB    (R2),   (R2)    ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
 1031  003670  005700                      TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 1032  003672  001023                      BNE     30$             ;BR IF ALREADY TYPED "FROM".
 1033  003674  004567  017520              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1034  003700  025415                      .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
 1035                                                              ;"FROM"
 1036  003702  010246                      MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
 1037  003704  004767  021150              JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 1038  003710  000413                      BR      29$             ;GO TO ADDRESS POINTER UPDATE.
 1039                                      ;*  TIME OUTS COME HERE.
 1040  003712  022626              20$:    CMP     (SP)+,  (SP)+   ;POP TWO OFF STACK.
 1041  003714  005700                      TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 1042  003716  001411                      BEQ     30$             ;BR IF ALREADY TYPED "TO".
 1043  003720  004567  017474              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1044  003724  025425                      .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
 1045                                                              ;"TO"
 1046  003726  005302                      DEC     R2              ;BACK UP ONE BYTE.
 1047  003730  010246                      MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
 1048  003732  004767  021122              JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 1049  003736  005202                      INC     R2              ;RESET ADDRESS POINTER.
 1050  003740  005100              29$:    COM     R0              ;RESET PREVIOUS TYPOUT FLAG.
 1051  003742  005202              30$:    INC     R2              ;UPDATE ADDRESS POINTER TO NEXT BYTE.
 1052  003744  001423                      BEQ     31$             ;EXIT IF ZERO REACHED.
 1053  003746  032702  017777              BIT     #MASK4K,R2      ;CHECK FOR 4K BANK BOUNDRY.
 1054  003752  001340                      BNE     10$             ;BR IF MORE THIS 4K BANK.
 1055  003754  005767  174626              TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
 1056  003760  001735                      BEQ     10$             ;BR IF NO MEM MGMT.
 1057  003762  022737  007600  172346      CMP     #7600,  @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
```

```
1058  003770  001411                          BEQ     31$             ;EXIT WHEN ALL DONE.
1059  003772  012702  060000                  MOV     #60000,R2       ;RESET VIRTUAL ADDRESS POINTER.
1060  003776  013737  172346  172344          MOV     @#KIPAR3,@#KIPAR2 ;SAVE MEM MGMT REG FOR TYPEOUT.
1061  004004  062737  000200  172346          ADD     #200,   @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1062  004012  000720                          BR      10$             ;BR BACK TO DO NEXT BANK.
1063  004014  005700                  31$:    TST     R0              ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1064  004016  001407                          BEQ     32$             ;BR TO EXIT IF TYPING ALL DONE.
1065  004020  004567  017374                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1066  004024  025425                          .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
1067                                                                  ;"TO"
1068  004026  005302                          DEC     R2              ;BACK ADDRESS POINTER UP ONE BYTE.
1069  004030  010246                          MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
1070  004032  004767  021022                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1071  004036  000000                  32$:    HALT                    ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1072                                                                  ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1073  004040  000654                          BR      TIMOUT          ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1074
1075                                  .SBTTL  MAP PARITY REGISTERS
1076                                  ;;**************************************************************
1077                                  ;*  SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1078                                  ;*  THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1079                                  ;;**************************************************************
1080
1081  004042  012704  002270          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
1082  004046  032777  000100  175064          BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
1083  004054  001036                          BNE     GMPRD           ;BR IF INHIBIT PARITY.
1084  004056  012703  002070                  MOV     #MPRD,  R3      ;SET UP TABLE POINTER
1085  004062  012737  004104  000004          MOV     #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1086  004070  042713  000001          GMPRA:  BIC     #1,     (R3)    ;CLEAR FLAG BIT IN TABLE
1087  004074  005773  000000                  TST     @(R3)           ;DOES THIS MEMORY PARITY REGISTER EXIST.
1088                                  ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1089  004100  012324                          MOV     (R3)+,  (R4)+   ;SAVE IT IN THE PARITY REG EXIST TABLE.
1090  004102  000403                          BR      GMPRC           ;SKIP TIMEOUT SERVICE CODE
1091                                  ;*  TIMEOUT COMES HERE
1092  004104  022626                  GMPRB:  CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
1093  004106  052723  000001                  BIS     #1,     (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1094  004112  005023                  GMPRC:  CLR     (R3)+           ;CLEAR THE MAP...LO 64K.
1095  004114  005023                          CLR     (R3)+           ;...HI 64K.
1096  004116  005023                          CLR     (R3)+           ;...AND THE MASK.
1097  004120  020327  002270                  CMP     R3,     #MPRX   ;HAVE WE CHECKED ALL REGISTERS?
1098  004124  103761                          BLO     GMPRA           ;NO - GO BACK TO CHECK NEXT ONE
1099  004126  005014                          CLR     (R4)            ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1100  004130  012737  025022  000004          MOV     #ERRTRP,@#ERRVEC ; RESTORE TRAPCATCHER
1101  004136  005767  176126                  TST     MPRX                    ;ANY PARITY REGISTERS PRESENT?
1102  004142  001006                          BNE     MPAMEM                  ;YES - GO TEST CONTROLS PRESENT
1103  004144  004567  017250                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1104  004150  025513                          .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
1105                                                                  ;"NO MEMORY PARITY REGISTERS FOUND"
1106  004152  005014                  GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
1107  004154  000167  001150                  JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING
1108
```

```
                                .SBTTL  MAP PARITY MEMORY
                                ;;******************************************************************
                                ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
                                ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
                                ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
                                ;       - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
                                ;       - PARITY GENERATE OR DETECT LOGIC FAILED
                                ;       - PARITY ERROR BIT FAILED TO SET
                                ;       - PARITY BITS IN MEMORY LOCATION FAILED
                                ;       - I.E. BIT STUCK AT GOOD PARITY VALUE
                                ;;******************************************************************

004160  004767  014012          MPAMEM: JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
004164  012767  000001  175352          MOV     #1,     BITPT   ;INITIALIZE 4K POINTER
004172  005067  175350                  CLR     BITPT+2         ;CLEAR HI 64K POINTER
004176  012702  014000                  MOV     #14000, R2      ;SET ADR POINTER TO 14000.
004202  005767  174400                  TST     MMAVA           ;CHECK FOR MEM MGMT
004206  001404                          BEQ     MAPRB           ;BRANCH IF NO MEM MGMT
004210  012702  054000                  MOV     #54000, R2      ;SET ADR POINTER TO PAR2
004214  004767  007766                  JSR     PC,     MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.

                                ;;******************************************************************
                                ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
                                ;*  THEN WRITE TEST LOCATION VIA DATO & READ TEST LOCATION VIA DATI
                                ;*  THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
                                ;;******************************************************************

004220  005067  175314          MAPRB:  CLR     PMEMAP          ;CLEAR THE PARITY MEMORY MAP
004224  005067  175312                  CLR     PMEMAP+2
004230  012703  002070          1$:     MOV     #MPRO,  R3      ;INITIALIZE TABLE ADDRESS
004234  032713  000001          2$:     BIT     #1,     (R3)    ;IS THIS REGISTER PRESENT?
004240  001052                          BNE     3$              ;NO - GET THE NEXT ONE
004242  013773  00.612  000000          MOV     @#WWP,  @(R3)   ;YES - SET WRITE WRONG PARITY
                                                                ;AND CLEAR REST OF REGISTER
004250  011212                          MOV     (R2),   (R2)    ;WRITE WRONG PARITY
004252  005712                          TST     (R2)            ;READ WRONG PARITY
004254  043773  001612  000000          BIC     @#WWP,  @(R3)   ;CLEAR WRITE WRONG PARITY
004262  005773  000000                  TST     @(R3)           ;OTHERWISE, CHECK TO SEE IF THIS
                                                                ;CONTROL REGISTER GOT A PARITY
                                                                ;ERROR
004266  100014                          BPL     6$              ;BRANCH IF IT DIDN'T AND CHECK
004270  032773  007740  000000          BIT     #7740,  @(R3)   ;IS IT A CORE PAR. REG.
004276  001404                          BEQ     5$              ;BRANCH IF NOT.
004300  012763  070032  000006          MOV     #70032, 6(R3)   ;IF IT IS SET UP MASK
004306  000413                          BR      7$              ;AND BRANCH TO SET BITS.
004310  012763  077772  000006  5$:     MOV     #77772, 6(R3)   ;IF MOS SET UP MASK
004316  000407                          BR                      ;AND BRANCH TO SET BIT.
004320  032773  007740  000000  6$:     BIT     #7?  ?, @(R3)   ;IF ANY BITS ARE SET
004326  001417                          BEQ     3$              ;THEN CSR IS MS11-K.
004330  012763  070000  000006          MOV     #70000, 6(R3)   ;IF MS11-K SET MASK.
004336  056763  175202  000002  7$:     BIS     BITPT,  2(R3)   ;SET FLAG IN MAP FOR THIS PARITY REGISTER
004344  056763  175176  000004          BIS     BITPT+2,4(R3)
004352  056767  175166  175160          BIS     BITPT,  PMEMAP  ;SET FLAG IN PARITY MAP
004360  056767  175162  175154          BIS     BITPT+2,PMEMAP+2
004366  062703  000010          3$:     ADD     #10,    R3      ;STEP UP TO NEXT REGISTER
004372  020327  002270                  CMP     R3,     #MPRX   ;ARE WE DONE WITH TABLE?
```

```
1165  004376  103716                          BLO     2$              ;GO BACK TO CHECK FOR ANY MORE!
1166  004400  011212                          MOV     (R2),   (R2)    ;CLEAR BAD PARITY
1167  004402  005767  174200                  TST     MMAVA           ;CHECK FOR MEM MGMT
1168  004406  001444                          BEQ     10$             ;BR IF NO MEM MGMT
1169  004410  062737  000200  172344  4$:     ADD     #200,   @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1170  004416  006367  175122                  ASL     BITPT           ;UPDATE BANK POINTER...LO 64K.
1171  004422  006167  175120                  ROL     BITPT+2         ;...HI 64K.
1172  004426  100441                          BMI     TMAP            ;BR IF ALL DONE.
1173  004430  023727  172344  001000          CMP     @#KIPAR2,#1000  ;THIS CODE TESTS IF MS11-K IS
1174  004436  001013                          BNE     12$             ;PRESENT AND IF IT IS I SET
1175  004440  032737  000003  002252          BIT     #3,@#MPR14+2    ;THE BIT TO DISABLE ECC IN
1176  004446  001004                          BNE     13$             ;THE LOCATION MAP THAT IS
1177  004450  032737  000003  002262          BIT     #3,@#MPR15+2    ;USED AS THE COMMAND TO
1178  004456  001400                          BEQ     13$             ;WRITE WRONG PARITY.
1179  004460  012737  020004  001612  13$:     MOV     #20004,@#MAP
1180  004466  036767  175052  175030  12$:     BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
1181  004474  001255                          BNE     1$              ;BR IF BANK EXISTS.
1182  004476  036767  175044  175022          BIT     BITPT+2,MEMMAP+2 ;...HI 64K.
1183  004504  001251                          BNE     1$              ;BR IF BANK EXISTS.
1184  004506  000740                          BR      4$              ;BR IF BANK DOESN'T EXIST.
1185  004510  036767  175030  175006  11$:     BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS.
1186  004516  001244                          BNE     1$              ;BR IF BANK EXISTS.
1187  004520  062702  020000          10$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
1188  004524  106367  175014                  ASLB    BITPT           ;MOVE POINTER TO NEXT BANK.
1189  004530  100367                          BPL     11$             ;BR IF MORE TO LOOK FOR.
1190
1191                                  ;;*****************************************************************
1192                                  ;*  ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
1193                                  ;*  AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
1194                                  ;;*****************************************************************
1195
1196  004532  004767  013440          TMAP:   JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS PRESENT
1197  004536  004567  016656                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1198  004542  025370                          .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
1199                                                                  ;"PARITY MEMORY MAP:"
1200  004544  012703  002070                  MOV     #MPR0,  R3      ;INITIALIZE TABLE POINTER
1201  004550  032713  000001          1$:     BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER IS PRESENT.
1202  004554  001046                          BNE     2$              ;BR IF NOT PRESENT.
1203  004556  022763  070032  000006          CMP     #70032, 6(R3)
1204  004564  001004                          BNE     3$
1205  004566  004567  016626                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1206  004572  026027                          .WORD   MX3             ;ADDRESS OF MESSAGE TO BE TYPED
1207                                                                  ;"CORE PARITY"
1208  004574  000417                          BR      5$
1209  004576  022763  077772  000006  3$:     CMP     #77772, 6(R3)
1210  004604  001004                          BNE     4$
1211  004606  004567  016606                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1212  004612  026046                          .WORD   MX4             ;ADDRESS OF MESSAGE TO BE TYPED
1213                                                                  ;"MOS PARITY"
1214  004614  000407                          BR      5$
1215  004616  022763  070000  000006  4$:     CMP     #70000, 6(R3)
1216  004624  001003                          BNE     5$
1217  004626  004567  016566                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1218  004632  026064                          .WORD   MX5             ;ADDRESS OF MESSAGE TO BE TYPED
1219                                                                  ;"MS11-K CSR"
1220  004634                          5$:
```

```
1221  004634  004567  016560                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1222  004640  025777                          .WORD   MX1             ;ADDRESS OF MESSAGE TO BE TYPED
1223                                                                  ;"REGISTER AT"
1224  004642  011346                          MOV     (R3),-(SP)      ;;SAVE (R3) FOR TYPEOUT
1225                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
1226                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1227  004644  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
1228  004650  004767  017742                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
1229  004654  004567  016540                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1230  004660  026014                          .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
1231                                                                  ;"CONTROLS"
1232  004662  010300                          MOV     R3,     R0      ;SET UP R0 FOR TYPMAP ROUTINE.
1233  004664  005720                          TST     (R0)+           ;UPDATE POINTER TO MAP.
1234  004666  004767  013426                  JSR     PC,     TYPMAP  ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1235  004672  062703  000010          2$:     ADD     #10,    R3      ;UPDATE TO NEXT REGISTER IN TABLE.
1236  004676  020327  002270                  CMP     R3,     #MPRX   ;ARE WE ALL DONE WITH TABLE?
1237  004702  103722                          BLO     1$              ;BRANCH IF MORE REGISTERS
1238  004704  004567  016510                  JSR     R5,     $PRINT  ;THE REASON I'M OUTPUTTING THIS CRLF
1239  004710  001201                          $CRLF                   ;IS TO GIVE THE PRINTER ENOUGH TIME TO
1240                                                                  ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
1241  004712  022737  070000  002256          CMP     #70000,@#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
1242  004720  001006                          BNE     7$              ;IF NO BRANCH
1243  004722  043727  002252  001540          BIC     @#MPR14+2,#PMEMAP ;IF YES THEN CLEAR THE BITS IN
1244  004730  043737  002254  001540          BIC     @#MPR14+4,@#PMEMAP ;THE PARITY MEMORY MAP.
1245  004736  022737  070000  002266  7$:     CMP     #70000, @#MPR15+6       ;DO WE HAVE A MS11-K
1246  004744  001031                          BNE     9$              ;IF NO GO TO TESTS NOW.
1247  004746  043737  002262  001540          BIC     @#MPR15+2,@#PMEMAP      ;IF YES I AM GOING TO
1248  004754  043737  002264  001542          BIC     @#MPR15+4,@#PMEMAP+2    ;CLEAR THE PARITY INDICATORS
1249  004762  012705  002270                  MOV     #MPRX,  R5      ;FOR THAT PORTION OF MEMORY.
1250  004766  021537  002250          6$:     CMP     (R5),@#MPR14    ;SEARCH FOR THIS MS11-K  CSR IN
1251  004772  001004                          BNE     8$              ;AND IF ITS THERE DELETE IT
1252  004774  005015                          CLR     (R5)
1253  004776  052737  000001  002250          BIS     #1,@#MPR14
1254  005004  022537  002260          8$:     CMP     (R5)+,  @#MPR15 ;SEARCH FOR MS11-K CSR IN
1255  005010  001366                          BNE     6$              ;THE AVAILABILITY TABLE,
1256  005012  005045                          CLR     -(R5)           ;AND CLEAR ITS ADDRESS FROM THE TABLE
1257  005014  052737  000001  002260          BIS     #1,     @#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
1258  005022  004567  016372                  JSR     R5,     $PRINT  ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1259  005026  026102                          .WORD   MX6
1260  005030  005737  002270          9$:     TST     @#MPRX          ;ARE THERE ANY PARITY REGISTERS TO TEST?
1261  005034  001002                          BNE     CTRLS           ;IF SO TEST THE BITS IN THE REGISTERS,
1262  005036  000167  000266                  JMP     MANUAL          ;IF NO JUMP OVER REGISTER TESTS.
1263
1264                                .SBTTL  TEST PARITY REGISTERS
1265                                ;;*****************************************************************
1266                                ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1267                                ;*  CAN BE SET AND CLEARED.
1268                                ;*  THIS IS A ONCE ONLY TEST.
1269                                ;;*****************************************************************
1270
1271  005042  012703  002070          CTRLS:  MOV     #MPR0,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1272  005046  011302                  1$:     MOV     (R3),   R2      ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
1273  005050  062703  000010                  ADD     #10,    R3      ;UPDATE POINTER TO NEXT PAR. REG. ADD.
1274  005054  032702  000001                  BIT     #1,     R2      ;IS THIS REGISTER BEING USED?
1275  005060  001372                          BNE     1$              ;GO TO NEXT IF NOT
1276  005062  020327  002270                  CMP     R3,     #MPRX   ;ARE WE AT END OF TABLE
```

```
1277  005066  003052                        BGT     RESCHK          ;GO TO NEXT TEST IF YES
1278  005070  016367  177776  174420        MOV     -2(R3), RESRVD  ;GET MASK FOR REGISTER WE ARE WORKING ON
1279  005076  012700  000001                MOV     #1,     R0      ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
1280  005102  005012                        CLR     (R2)            ;INITIALIZE THE PARITY REGISTER
1281  005104  011201                        MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
1282  005106  046701  174404                BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
1283  005112  001405                        BEQ     2$              ;CHECK OTHER BITS - BRANCH IF OK
1284  005114  004767  013100        64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1285  005120  004767  014422                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1286  005124  000001                        .WORD   1               ;ERROR TYPE CODE.
1287  005126  030067  174364        2$:     BIT     R0,     RESRVD  ;IS THIS BIT RESERVED?
1288  005132  001025                        BNE     3$              ;YES - DON'T TEST IT
1289  005134  010012                        MOV     R0,     (R2)    ;NO - SET THIS BIT IN THE PARITY REGISTER
1290  005136  011201                        MOV     (R2),   R1      ;READ & SAVE CONTENTS OF THE  PARITY REGISTER
1291  005140  005012                        CLR     (R2)            ;CLEAR THE PARITY REGISTER
1292  005142  046701  174350                BIC     RESRVD, R1      ;CLEAR BIT LOCATIONS THAT ARE RESERVED
1293  005146  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1294  005150  001405                        BEQ     66$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1295  005152  004767  013072        65$:    JSR     PC,     SPRNTD  ;SET UP VALUES FOR ERROR PRINTING.
1296  005156  004767  014364                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1297  005162  000001                        .WORD   1               ;ERROR TYPE CODE.
1298  005164                        66$:
1299  005164  011201                        MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
1300  005166  046701  174324                BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
1301  005172  001405                        BEQ     3$              ;CHECK OTHER BITS - BRANCH IF OK
1302  005174  004767  013020        67$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1303  005200  004767  014342                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1304  005204  000001                        .WORD   1               ;ERROR TYPE CODE.
1305  005206  006300                3$:     ASL     R0              ;ROTATE TO GET NEXT BIT TO BE TESTED
1306  005210  103346                        BCC     2$              ;BRANCH IF NOT DONE WITH ALL BITS
1307  005212  000715                        BR      1$              ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.
1308
1309                                ;;*****************************************************************
1310                                ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
1311                                ;*  THIS IS A ONCE ONLY TEST.
1312                                ;;*****************************************************************
1313
1314  005214  012704  002070        RESCHK: MOV     #MPRO,  R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1315  005220  010403                1$:     MOV     R4,     R3
1316  005222  062704  000010                ADD     #10,    R4
1317  005226  032713  000001        BIT     #1,     (R3)    ;IS THIS REGISTER BEING USED
1318  005232  001372                        BNE     1$              ;BRANCH IF NO
1319  005234  012773  177777  000000        MOV     #-1,    @(R3)   ;SET ALL BITS TO A 1
1320  005242  022704  002270                CMP     #MPRX,R4        ;ARE WE AT THE END OF THE TABLE
1321  005246  002764                        BLT     1$              ;IF YES THEN WE ARE READY TO TEST
1322  005250  000005                        RESET                   ;RESET THE WORLD
1323  005252  012703  002070                MOV     #MPRO,  R3      ;LOAD INITIAL ADDRESS FOR POINTER
1324  005256  011302                2$:     MOV     (R3),   R2      ;STORE PARITY REGISTER ADDRESS
1325  005260  062703  000010                ADD     #10,    R3
1326  005264  032702  000001                BIT     #1,     R2
1327  005270  001372                        BNE     2$
1328  005272  022703  002270                CMP     #MPRX,  R3
1329  005276  002014                        BGE     MANUAL
1330  005300  011201                        MOV     (R2),   R1      ;GET CONTENTS OF REGISTER
1331  005302  005012                        CLR     (R2)
1332  005304  042701  077772                BIC     #77772, R1      ;CLEAR BITS NOT EFFECTED  BY RESET
```

```
1333  005310  005701                          TST     R1              ;CHECK IF REST  WERE CLEARED BY  RESET
1334  005312  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1335  005314  004767  012700          64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
1336  005320  004767  014222                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1337  005324  000001                          .WORD   1               ;ERROR TYPE CODE.
1338  005326                          65$:
1339  005326  000753                          BR      2$              ;BRANCH BACK TO CHECK NEXT REGISTER
1340
1341
1342  005330  012700  000014          MANUAL: MOV     #12.,   R0      ;SET COUNTER TO CLEAR 12 WORDS.
1343  005334  012701  001562                  MOV     #FSTADR,R1      ;STARTING AT FSTADR.
1344  005340  005021                  1$:     CLR     (R1)+           ;CLEAR THE LOCATIONS.
1345  005342  005300                          DEC     R0              ;COUNT.
1346  005344  001375                          BNE     1$              ;BR IF MORE.
1347  005346  105767  174204                  TSTB    SELFLG          ;CHECK FOR SELECT PARAMETERS STARTUP.
1348  005352  001005                          BNE     MANUL1          ;BR IF PARAMETERS TO BE SELECTED.
1349  005354  016767  173604  174212          MOV     $TMP2,  LSTADR  ;SET UP VIRTUAL LAST ADDRESS.
1350  005362  000167  000402                  JMP     MANUL2          ;SKIP PARAMETER SELECTION SECTION.
```

```
                                .SBTTL  USER PARAMETER SELECTION SECTION
                                ;;*****************************************************************
                                ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                ;;*****************************************************************
005366  012700  000001          MANUL1: MOV     #BIT0,  R0      ;SET UP BANK POINTER.
005372  005001                          CLR     R1              ;...HI 64K.
005374  005002                          CLR     R2              ;CLEAR ADDRESS POINTER.
005376  005003                          CLR     R3              ;...HI ADDRESS BITS.
005400  004567  016014                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005404  026215                          .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**.
005406  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005412  004767  015630                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005416  042716  000001                  BIC     #BIT0,  (SP)    ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
005422  005067  174106                  CLR     SAVTST          ;INIT TEST MAP...LO 64K.
005426  005067  174104                  CLR     SAVTST+2        ;...HI 64K.
005432  062702  020000          1$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
005436  005503                          ADC     R3
005440  020367  015752                  CMP     R3,     $HIOCT  ;CHECK HI ADDRESS BITS.
005444  103403                          BLO     2$              ;BR IF NOT HI ENOUGH YET.
005446  101006                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005450  020216                          CMP     R2,     (SP)    ;CHECK THE LO ADDRESS BITS.
005452  101004                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005454  006300          2$:     ASL     R0              ;UPDATE POINTER...LO 64K.
005456  006101                          ROL     R1              ;...HI 64K.
005460  100364                          BPL     1$              ;BR BACK TO CHECK NEXT BANK.
005462  000507                          BR      17$             ;BR IF OVERFLOW.
005464  030067  174034          3$:     BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS.
005470  001003                          BNE     4$              ;BR IF BANK EXISTS.
005472  030167  174030                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K.
005476  001501                          BEQ     17$             ;BR IF ADDRESS IN UN-MAPPED BANK.
005500  016704  015712          4$:     MOV     $HIOCT, R4      ;SAVE FIRST ADR HI BITS.
005504                          10$:
005504  004567  015710                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005510  026302                          .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**.
005512  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005516  004767  015524                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005522  005716                          TST     (SP)            ;CHECK IF ADR 0 SELECTED (DEFAULT).
005524  001010                          BNE     11$             ;BR IF NOT 0 (DEFAULT)
005526  005767  015664                  TST     $HIOCT          ;CHECK HI BITS.
005532  001005                          BNE     11$             ;BR IF NOT 0 (DEFAULT).
005534  016716  173424                  MOV     $TMP2,  (SP)    ;SET UP DEFAULT LAST ADR.
005540  016767  173422  015650          MOV     $TMP3,  $HIOCT
005546  012667  174022          11$:    MOV     (SP)+,  LSTADR  ;GET THE DATA.
005552  020467  015640                  CMP     R4,     $HIOCT  ;CHECK FOR LAST ADR BELOW FIRST ADR.
005556  101352                          BHI     10$             ;BB IF LAST BELOW FIRST.
005560  103403                          BLO     12$             ;BR IF LAST ABOVE FIRST.
005562  021667  174006                  CMP     (SP),   LSTADR  ;CHECK FOR LAST BELOW FIRST.
005566  101346                          BHI     10$             ;BR IF LAST BELOW FIRST.
005570  032716  017777          12$:    BIT     #MASK4K,(SP)    ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005574  001404                          BEQ     13$             ;BR IF ON BOUNDRY.
```

```
1407  005576  010067  173766            MOV     R0,     FADMAP  ;SET UP FIRST ADDRESS MAP.
1408  005602  010167  173764            MOV     R1,     FADMAP+2
1409  005606  050067  173722    13$:    BIS     R0,     SAVTST  ;SET FLAG IN TEST MAP...LO 64K.
1410  005612  050167  173720            BIS     R1,     SAVTST+2;...HI 64K.
1411  005616  020367  015574    14$:    CMP     R3,     $HIOCT  ;CHECK FOR PAST LAST ADR.
1412  005622  103404                    BLO     15$             ;BR IF BELOW LAST ADR.
1413  005624  101020                    BHI     16$             ;BR IF GONE PAST LAST ADR.
1414  005626  020267  173742            CMP     R2,     LSTADR  ;CHECK FOR PAST LAST ADR.
1415  005632  101015                    BHI     16$             ;BR IF GONE PAST LAST ADR.
1416  005634  062702  020000    15$:    ADD     #20000, R2      ;UPDATE ADDRESS POINTER.
1417  005640  005503                    ADC     R3              ;...HI BITS.
1418  005642  006300                    ASL     R0              ;UPDATE BANK POINTER...LO 64K.
1419  005644  006101                    ROL     R1              ;...HI 64K.
1420  005646  100415                    BMI     17$             ;BR IF OVERFLOW.
1421  005650  030067  173650            BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
1422  005654  001354                    BNE     13$             ;BR IF BANK EXISTS.
1423  005656  030167  173644            BIT     R1,     MEMMAP+2;CHECK IF THIS BANK EXISTS.
1424  005662  001351                    BNE     13$             ;BR IF BANK EXISTS.
1425  005664  000754                    BR      14$             ;BR IF BANK DOESN'T EXIST.
1426  005666  030067  173632    16$:    BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
1427  005672  001010                    BNE     20$             ;BR IF IT EXISTS.
1428  005674  030167  173626            BIT     R1,     MEMMAP+2;CHECK IF THIS BANK EXISTS.
1429  005700  001005                    BNE     20$             ;BR IF IT EXISTS.
1430  005702  005726            17$:    TST     (SP)+           ;ADJUST THE STACK.
1431  005704  004567  015510            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1432  005710  026325                    .WORD   BADADR          ;ADDRESS OF MESSAGE TO BE TYPED
1433                                                            ;"?ADDRESS IN UNMAPPED BANK?"
1434  005712  000606                    BR      MANUAL          ;LOOP BACK TO THE BEGINNING.
1435  005714  010067  173662    20$:    MOV     R0,     LADMAP  ;SET UP MAP FOR LAST ADDRESS.
1436  005720  010167  173660            MOV     R1,     LADMAP+2
1437  005724  005767  172656    21$:    TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
1438  005730  001404                    BEQ     22$             ;BR IF NO MEM MGMT.
1439  005732  042716  160000            BIC     #160000,(SP)    ;ADJUST FSTADR TO VITRUAL BANK 0.
1440  005736  062716  040000            ADD     #40000, (SP)    ;...TO VIRTUAL BANK 2.
1441  005742  012667  173614    22$:    MOV     (SP)+,  FSTADR  ;SAVE FISRT ADDRESS OFF THE STACK.
1442  005746                    30$:
1443  005746  004567  015446            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1444  005752  026362                    .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
1445                                                            ;"SELECT CONSTANT: "
1446                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1447                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1448  005754  013746  177776            MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
1449  005760  004767  015262            JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
1450  005764  012667  173620            MOV     (SP)+,  .CONST  ;SAVE THE CONSTANT
1451  005770  005767  172612    MANUL2: TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
1452  005774  001406                    BEQ     31$             ;BR IF NO MEM MGMT.
1453  005776  042767  160000  173570    BIC     #160000,LSTADR  ;ADJUST LSTADR TO VIRTUAL BANK 0.
1454  006004  062767  040000  173562    ADD     #40000, LSTADR  ;...VIRTUAL BANK 2.
1455  006012  062767  000002  173554    ADD     #2,     LSTADR  ;ADJUST LAST ADDRESS UP ONE WORD.
1456  006020  042767  000001  173546    BIC     #BIT0,  LSTADR  ;MAKE SURE IT IS A WORD ADDRESS.
1457  006026  032767  017777  173540    BIT     #MASK4K,LSTADR  ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
1458  006034  001004                    BNE     START1          ;BR IF NOT ON BOUNDRY.
1459  006036  005067  173540            CLR     LADMAP          ;CLEAR OUT THE LAST ADDRESS MAP.
1460  006042  005067  173536            CLR     LADMAP+2
1461
```

```
        ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
        ;*  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
        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

  1468  006046  016706  173542          START1: MOV     .STACK, SP      ;SET STACK POINTER
  1469  006052  012767  006046  173026          MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
  1470  006060  066767  172514  173020          ADD     RELOCF, $LPADR
  1471  006066  004767  011352                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
  1472  006072  005767  172510                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
  1473  006076  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
  1474  006100  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
  1475  006106  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
  1476  006110  004767  006072                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
```

```
1477                                    .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
1478                                    ;;*****************************************************************
1479                                    ;*TEST 1         WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1480                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1481                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1482                                    ;*      R2 = VIRTUAL ADDRESS
1483                                    ;*      R3 = NOT USED
1484                                    ;*      R4 = NOT USED
1485                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1486                                    ;;*****************************************************************
1487  006114                            TST1:
1488  006114  004567  012416                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1489  006120  000001                            .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
1490                                                                    ; REQUIRED FOR THIS TEST.
1491  006122  000167  005574                   JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1492                                                                    ; AVAILABLE FOR TEST.
1493                                    ;*  UPWARDS WORD ADDRESSING.
1494  006126  004467  006202                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1495  006132  004767  007624            1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1496  006136  010012                    2$:    MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
1497  006140  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1498  006142  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1499  006144  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1500  006146  004767  012122            64$:   JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1501  006152  004767  013370                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1502  006156  000002                            .WORD   2               ;ERROR TYPE CODE.
1503  006160                            65$:
1504  006160  062700  000002                   ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
1505  006164  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1506  006166  001363                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1507  006170  004767  006716                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1508
1509                                    ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1510                                    ;*  DOWNWARDS WORD ADDRESSING.
1511  006174  004467  006572                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1512  006200  004767  007556            3$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1513  006204  162700  000002            4$:    SUB     #2,     R0      ;DEC DATA BY 2
1514  006210  014201                            MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
1515  006212  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1516  006214  001405                            BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1517  006216  004767  012026            66$:   JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
1518  006222  004767  013320                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1519  006226  000002                            .WORD   2               ;ERROR TYPE CODE.
1520  006230                            67$:
1521  006230  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1522  006232  001364                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1523  006234  004767  007342                   JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
1524                                    ;;*****************************************************************
1525                                    ;*TEST 2        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
1526                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1527                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1528                                    ;*      R2 = VIRTUAL ADDRESS
1529                                    ;*      R3 = NOT USED
1530                                    ;*      R4 = NOT USED
1531                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1532                                    ;;*****************************************************************
1533  006240                            TST2:
1534  006240  004567  012272                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1535  006244  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1536                                    ;*  UPWARDS BYTE ADDRESSING.
1537  006246  004467  006062                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1538  006252  004767  007504           1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1539  006256  110022                   2$:     MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
1540  006260  005200                            INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
1541  006262  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1542  006264  001374                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1543  006266  004767  006620                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1544
1545                                    ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
1546                                    ;*  DOWNWARDS BYTE ADDRESSING.
1547  006272  004467  006474                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1548  006276  004767  007460           3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1549  006302  005300                   4$:     DEC     R0              ;DEC DATA BY 1
1550  006304  114201                            MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
1551  006306  120001                            CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
1552  006310  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1553  006312  004767  011732           64$:    JSR     PC,     SPRNTD  ;SET UP VALUES FOR ERROR PRINTING.
1554  006316  004767  013224                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1555  006322  000003                            .WORD   3               ;ERROR TYPE CODE.
1556  006324                           65$:
1557  006324  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1558  006326  001365                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1559  006330  004767  007246                   JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
1560
1561                                    ;;*****************************************************************
1562                                    ;*TEST 3        WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
1563                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1564                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1565                                    ;*      R2 = VIRTUAL ADDRESS
1566                                    ;*      R3 = NOT USED
1567                                    ;*      R4 = NOT USED
1568                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1569                                    ;;*****************************************************************
1570  006334                            TST3:
1571  006334  004567  012176                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1572  006340  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1573                                    ;*  DOWNWARDS WORD ADDRESSING.
1574  006342  004467  006424                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1575  006346  004767  007410           1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1576  006352  005100                            COM     R0              ;COMPLEMENT THE ADR
1577  006354  062700  000002           2$:     ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
1578  006360  010042                            MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
1579  006362  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```

```
1580  006364  001373                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1581  006366  004767  007210                JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
1582
1583                                ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
1584                                ;*  UPWARDS WORD ADDRESSING.
1585  006372  004467  005736                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1586  006376  004767  007360        3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
1587  006402  005100                        COM     R0              ;COMPLEMENT IT
1588  006404                        4$:
1589  006404  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1590  006406  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1591  006410  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1592  006412  004767  011656        64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1593  006416  004767  013124                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1594  006422  000002                        .WORD   2               ;ERROR TYPE CODE.
1595  006424                        65$:
1596  006424  162700  000002                SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
1597  006430  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1598  006432  001364                        BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1599  006434  004767  006452                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
1600
1601                                ;;*****************************************************************
1602                                ;*TEST 4        WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1603                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1604                                ;*      R1 = DATA READ FROM MEMORY (WAS)
1605                                ;*      R2 = VIRTUAL ADDRESS
1606                                ;*      R3 = NOT USED
1607                                ;*      R4 = NOT USED
1608                                ;*      R5 = BLOCK BOUNDRY BIT MASK.
1609                                ;;*****************************************************************
1610  006440                        TST4:
1611  006440  004567  012072                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1612  006444  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1613                                ;*  UPWARDS BYTE ADDRESSING.
1614  006446  004467  005662                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1615  006452  004767  007360        1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
1616  006456  110022                2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
1617  006460  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1618  006462  001375                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1619  006464  004767  006422                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1620
1621                                ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1622                                ;*  UPWARDS BYTE ADDRESSING.
1623  006470  004467  005640                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1624  006474  004767  007336        3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
1625  006500  112201                4$:     MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
1626  006502  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1627  006504  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1628  006506  004767  011544        64$:    JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
1629  006512  004767  013030                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1630  006516  000003                        .WORD   3               ;ERROR TYPE CODE.
1631  006520                        65$:
1632  006520  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1633  006522  001366                        BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1634  006524  004767  006362                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
1635
```

```
1636                                    ;;*****************************************************************
1637                                    ;*TEST 5         WRITE 1'S COMPLEMENT OF BANK #.
1638                                    ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1639                                    ;*      R1 = DATA READ FROM MEMORY (WAS)
1640                                    ;*      R2 = VIRTUAL ADDRESS
1641                                    ;*      R3 = NOT USED
1642                                    ;*      R4 = NOT USED
1643                                    ;*      R5 = BLOCK BOUNDRY BIT MASK.
1644                                    ;;*****************************************************************
1645  006530                            T5T5:
1646  006530  004567  012002                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1647  006534  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1648                                    ;*  DOWNWARDS BYTE ADDRESSING.
1649  006536  004467  006230                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1650  006542  004767  007270           1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
1651  006546  005100                           COM     R0              ;1'S COMPLEMENT OF BANK #
1652  006550  110042                   2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
1653  006552  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1654  006554  001375                           BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1655  006556  004767  007020                   JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
1656
1657                                    ;*  CHECK THAT DATA WRITTEN CAN BE READ.
1658                                    ;*  DOWNWARDS BYTE ADDRESSING.
1659  006562  004467  006204                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1660  006566  004767  007244           3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
1661  006572  005100                           COM     R0              ;SET 1'S COMPLEMENT OF BANK #
1662  006574  114201                   4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
1663  006576  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1664  006600  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1665  006602  004767  011442           64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
1666  006606  004767  012734                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1667  006612  000003                           .WORD   3               ;ERROR TYPE CODE.
1668  006614                           65$:
1669  006614  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1670  006616  001366                           BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
1671  006620  004767  006756                   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.
                        ;;*****************************************************************
006624                  TST6:
006624  004567  011706          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006630  000000                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006632  016700  172752  TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
006636  004467  005472          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006642  010022  1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
006644  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006646  001375          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006650  004767  006236          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.
                        ;;*****************************************************************
006654                  TST7:
006654  004567  011656          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006660  000000                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006662  016700  172722          MOV     .CONST, R0      ;GET USER CONSTANT
006666  004467  005442          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006672                  1$:
006672  012201          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006674  020001          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006676  001405          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006700  004767  011370  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006704  004767  012636          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006710  000004                  .WORD   4               ;ERROR TYPE CODE.
006712                  65$:
006712  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006714  001366          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006716  004767  006170          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.
006722  032777  000400  172210          BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
006730  001416          BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
006732  017746  172202          MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
006736  042716  177740          BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
006742  022726  000006          CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
006746  001007          BNE     TST10           ;BRANCH IF NOT TEST 6
006750  162767  000001  172124  SUB     #1,     $TSTNM  ;RESET TEST NUM
006756  162767  000030  172122  SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
```

```
1728  006764  000722                          BR      TST6A           ;GO TO TEST 6
1729
1730                                  ;;*****************************************************************
1731                                  ;*TEST 10      WORSE CASE NOISE (PARITY) WORD TESTING
1732                                  ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
1733                                  ;;*****************************************************************
1734  006766                          TST10:
1735  006766  004567  011544                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1736  006772  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1737  006774  016704  172626                  MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
1738  007000  004767  010540          1$:     JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
1739  007004  012400                          MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
1740  007006  001420                          BEQ     TST11           ;BR IF END OF TABLE.
1741  007010  004467  005320                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1742  007014  010012                  2$:     MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
1743  007016  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1744  007020  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1745  007022  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1746  007024  004767  011244          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1747  007030  004767  012512                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1748  007034  000004                          .WORD   4               ;ERROR TYPE CODE.
1749  007036                          65$:
1750  007036  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1751  007040  001365                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
1752  007042  004767  006044                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
1753  007046  000754                          BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
1754                                    ;;*****************************************************************
1755                                    ;*TEST 11       ROTATE A "0" BIT THROUGH A FIELD OF ONES.
1756                                    ;;*****************************************************************
1757  007050                            $ST11:
1758  007050  004567  011462                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1759  007054  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1760  007056  012700  177777                   MOV     #-1,    R0      ;SET CHECK WORD
1761  007062  004767  007010                   JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
1762  007066  004467  005242                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1763  007072  000241            1$:     CLC                             ;CLEAR CARRY BIT IN PSW
1764  007074  004767  007016                   JSR     PC,     ROTATE
1765  007100  016201  177776                   MOV     -2(R2), R1      ;GET RESULT
1766  007104  103402                           BCS     63$             ;BRANCH IF 'C' BIT WAS SET
1767  007106  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1768  007110  001405                           BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1769  007112  004767  011156    63$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1770  007116  004767  012424                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1771  007122  000005                           .WORD   5               ;ERROR TYPE CODE.
1772  007124                            64$:
1773  007124  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1774  007126  001361                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
1775  007130  004767  005756                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
1776
1777                                    ;;*****************************************************************
1778                                    ;*TEST 12       ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
1779                                    ;;*****************************************************************
1780  007134                            $ST12:
1781  007134  004567  011376                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
1782  007140  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1783  007142  005000                           CLR     R0              ;SET CHECK WORD
1784  007144  004767  006726                   JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
1785  007150  004467  005160                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1786  007154  000261            1$:     SEC                             ;SET 'C' BIT IN PSW
1787  007156  004767  006734                   JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
1788  007162  016201  177776                   MOV     -2(R2), R1      ;GET RESULT
1789  007166  103002                           BCC     63$             ;BRANCH IF 'C' IS CLEAR
1790  007170  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1791  007172  001405                           BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1792  007174  004767  011074    63$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1793  007200  004767  012342                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1794  007204  000005                           .WORD   5               ;ERROR TYPE CODE.
1795  007206                            64$:
1796  007206  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1797  007210  001361                           BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
1798  007212  004767  005674                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 13       3 XOR 9 TEST PATTERN.
                                ;;*****************************************************************
                                ;
007216                          TST13:
007216  004567  011314                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007222  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007224  000167  000312                  JMP     TST14           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007230  005000                  .3X9:   CLR     R0              ;SET UP TEST DATA
007232  012703  177777                  MOV     #-1,    R3      ;SET COM DATA REG
007236  004467  005072                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007242  004767  006716          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007246  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007250  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007252  004767  005634                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
                                ;
                                ;;*****************************************************************
                                ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
                                ;;*****************************************************************
007256  005000                          CLR     R0              ;SET CHECK WORD
007260  004467  005050                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007264  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007270                          12$:
007270  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007272  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007274  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007276  004767  010772          64$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007302  004767  012240                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007306  000007                          .WORD   7               ;ERROR TYPE CODE.
007310                          65$:
007310  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007312  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007314  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007316  004767  010752          66$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007322  004767  012220                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007326  000007                          .WORD   7               ;ERROR TYPE CODE.
007330                          67$:
007330  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007332  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007334  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007336  004767  010732          68$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007342  004767  012200                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007346  000007                          .WORD   7               ;ERROR TYPE CODE.
007350                          69$:
007350  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007352  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007354  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007356  004767  010712          70$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007362  004767  012160                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007366  000007                          .WORD   7               ;ERROR TYPE CODE.
007370                          71$:
007370  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007372  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007374  001335                          BNE     12$
007376  005100                          COM     R0              ;COMPLEMENT CHECK WORD
```

```
1855  007400  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1856  007402  001330                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
1857  007404  004767  005502                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
1858
1859                                ;;*****************************************************************
1860                                ;*  CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
1861                                ;;*****************************************************************
1862  007410  005000                        CLR     R0
1863  007412  004467  004716                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1864  007416  012704  000100        21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
1865  007422  012703  000004        22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
1866  007426                        23$:
1867  007426  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1868  007430  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1869  007432  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1870  007434  004767  010634        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1871  007440  004767  012102                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1872  007444  000007                        .WORD   7               ;ERROR TYPE CODE.
1873  007446                        73$:
1874  007446  005100                        COM     R0              ;COMPLEMENT CHECK WORD
1875  007450  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
1876  007452  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1877  007454  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1878  007456  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1879  007460  004767  010610        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1880  007464  004767  012056                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1881  007470  000007                        .WORD   7               ;ERROR TYPE CODE.
1882  007472                        75$:
1883  007472  005100                        COM     R0              ;COMPLEMENT CHECK WORD
1884  007474  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
1885  007476  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
1886  007500  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
1887  007502  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
1888  007504  004767  010564        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
1889  007510  004767  012032                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1890  007514  000007                        .WORD   7               ;ERROR TYPE CODE.
1891  007516                        77$:
1892  007516  005303                        DEC     R3              ;DECREMENT 4 WORD COUNTER
1893  007520  001342                        BNE     23$             ;BR IF NOT DONE.
1894  007522  005100                        COM     R0              ;COMPLEMENT CHECK WORD
1895  007524  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
1896  007526  001335                        BNE     22$             ;BR IF NOT DONE.
1897  007530  005100                        COM     R0              ;COMPLEMENT CHECK WORD
1898  007532  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
1899  007534  001330                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
1900  007536  004767  005350                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 14       COMPLEMENT 3 XOR 9 TEST PATTERN
                                ;;*****************************************************************
007542                          TST14:
007542  004567  010770                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007546  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007550  000167  000316                  JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007554  012700  177777                  MOV     #-1,    R0      ;SET UP TEST DATA
007560  005003                          CLR     R3              ;SET COM DATA REG
007562  004467  004546                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007566  004767  006372          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007572  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007574  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007576  004767  005310                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.


                                ;;*****************************************************************
                                ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
007602  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007606  004467  004522                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007612  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007616                          12$:
007616  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007620  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007622  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007624  004767  010444          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007630  004767  011712                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007634  000007                          .WORD   7               ;ERROR TYPE CODE.
007636                          65$:
007636  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007640  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007642  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007644  004767  010424          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007650  004767  011672                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007654  000007                          .WORD   7               ;ERROR TYPE CODE.
007656                          67$:
007656  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007660  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007662  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007664  004767  010404          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007670  004767  011652                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007674  000007                          .WORD   7               ;ERROR TYPE CODE.
007676                          69$:
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     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007704  004767  010364          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007710  004767  011632                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007714  000007                          .WORD   7               ;ERROR TYPE CODE.
007716                          71$:
007716  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007720  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007722  001335                          BNE     12$
```

```
 1957  007724  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 1958  007726  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 1959  007730  001330                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
 1960  007732  004767  005154                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.  .
 1961
 1962                                ;;*****************************************************************
 1963                                ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
 1964                                ;;*****************************************************************
 1965  007736  012700  177777                MOV     #-1,    R0      ;SET UP CHECK WORD.
 1966  007742  004467  004366                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 1967  007746  012704  000100        21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
 1968  007752  012703  000004        22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
 1969  007756                        23$:
 1970  007756  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 1971  007760  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 1972  007762  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 1973  007764  004767  010304        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 1974  007770  004767  011552                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 1975  007774  000007                        .WORD   7               ;ERROR TYPE CODE.
 1976  007776                        73$:
 1977  007776  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 1978  010000  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 1979  010002  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 1980  010004  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 1981  010006  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 1982  010010  004767  010260        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 1983  010014  004767  011526                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 1984  010020  000007                        .WORD   7               ;ERROR TYPE CODE.
 1985  010022                        75$:
 1986  010022  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 1987  010024  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 1988  010026  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 1989  010030  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 1990  010032  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 1991  010034  004767  010234        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 1992  010040  004767  011502                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 1993  010044  000007                        .WORD   7               ;ERROR TYPE CODE.
 1994  010046                        77$:
 1995  010046  005303                        DEC     R3              ;DECREMENT 4 WORD COUNTER
 1996  010050  001342                        BNE     23$             ;BR IF NOT DONE.
 1997  010052  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 1998  010054  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
 1999  010056  001335                        BNE     22$             ;BR IF NOT DONE.
 2000  010060  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2001  010062  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 2002  010064  001330                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
 2003  010066  004767  005020                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 15       MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
                                ;;*****************************************************************
010072                          TST15:
010072  004567  010440                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010076  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
010100  000167  000610                  JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
010104  012700  000401                  MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN
010110  012703  177777                  MOV     #-1,    R3      ;SET COM DATA REG
010114  004467  004214                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010120  004767  006040          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
010124  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010126  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
010130  004767  004756                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
010134  012700  000401                  MOV     #401,   R0      ;RESET PARITY "ALL ZEROS" PATTERN.
010140  012703  177777                  MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
010144  004467  004164                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010150  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
010154                          12$:
010154  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010156  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010160  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010162  004767  010106          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010166  004767  011354                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010172  000007                          .WORD   7               ;ERROR TYPE CODE.
010174                          65$:
010174  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010176  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010200  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010202  004767  010066          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010206  004767  011334                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010212  000007                          .WORD   7               ;ERROR TYPE CODE.
010214                          67$:
010214  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010216  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010220  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010222  004767  010046          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010226  004767  011314                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010232  000007                          .WORD   7               ;ERROR TYPE CODE.
010234                          69$:
010234  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010236  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010240  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010242  004767  010026          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010246  004767  011274                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010252  000007                          .WORD   7               ;ERROR TYPE CODE.
010254                          71$:
010254  010046                          MOV     R0,     -(SP)   ;SAVE R0
010256  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
010260  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
```

```
2060  010262  005304                        DEC     R4              ;COUNT 256. WORDS
2061  010264  001333                        BNE     12$             ;BRANCH IF MORE
2062  010266  010046                        MOV     R0,     -(SP)   ;SAVE R0
2063  010270  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
2064  010272  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2065  010274  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2066  010276  001324                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
2067  010300  004767  004606                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
2068
2069                          ;;*****************************************************************
2070                          ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
2071                          ;;*****************************************************************
2072  010304  012700  000401                MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN.
2073  010310  012703  177777                MOV     #-1,    R3      ;SET UP ALL ONES PATTERN.
2074  010314  004467  004014                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2075  010320  012704  000100        21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
2076  010324                        22$:
2077  010324  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2078  010326  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2079  010330  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2080  010332  004767  007736        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2081  010336  004767  011204                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2082  010342  000007                        .WORD   7               ;ERROR TYPE CODE.
2083  010344                        73$:
2084  010344  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2085  010346  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2086  010350  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2087  010352  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2088  010354  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2089  010356  004767  007712        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2090  010362  004767  011160                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2091  010366  000007                        .WORD   7               ;ERROR TYPE CODE.
2092  010370                        75$:
2093  010370  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2094  010372  005142                        COM     -(R2)           ;RESTORE DATA
2095  010374  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2096  010376  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2097  010400  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2098  010402  004767  007666        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2099  010406  004767  011134                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2100  010412  000007                        .WORD   7               ;ERROR TYPE CODE.
2101  010414                        77$:
2102  010414  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2103  010416  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2104  010420  001405                        BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2105  010422  004767  007646        78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2106  010426  004767  011114                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2107  010432  000007                        .WORD   7               ;ERROR TYPE CODE.
2108  010434                        79$:
2109  010434  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2110  010436  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2111  010440  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2112  010442  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2113  010444  001405                        BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2114  010446  004767  007622        80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2115  010452  004767  011070                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
2116  010456  000007                        .WORD   7               ;ERROR TYPE CODE.
2117  010460                        81$:
2118  010460  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2119  010462  005142                        COM     -(R2)           ;RESTORE DATA
2120  010464  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2121  010466  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2122  010470  001405                        BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2123  010472  004767  007576        82$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2124  010476  004767  011044                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2125  010502  000007                        .WORD   7               ;ERROR TYPE CODE.
2126  010504                        83$:
2127  010504  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2128  010506  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2129  010510  001405                        BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2130  010512  004767  007556        84$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2131  010516  004767  011024                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2132  010522  000007                        .WORD   7               ;ERROR TYPE CODE.
2133  010524                        85$:
2134  010524  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2135  010526  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2136  010530  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2137  010532  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2138  010534  001405                        BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2139  010536  004767  007532        86$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2140  010542  004767  011000                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2141  010546  000007                        .WORD   7               ;ERROR TYPE CODE.
2142  010550                        87$:
2143  010550  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2144  010552  005142                        COM     -(R2)           ;RESTORE DATA
2145  010554  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2146  010556  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2147  010560  001405                        BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2148  010562  004767  007506        88$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2149  010566  004767  010754                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2150  010572  000007                        .WORD   7               ;ERROR TYPE CODE.
2151  010574                        89$:
2152  010574  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2153  010576  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2154  010600  001405                        BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2155  010602  004767  007466        90$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2156  010606  004767  010734                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2157  010612  000007                        .WORD   7               ;ERROR TYPE CODE.
2158  010614                        91$:
2159  010614  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2160  010616  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
2161  010620  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2162  010622  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2163  010624  001405                        BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2164  010626  004767  007442        92$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2165  010632  004767  010710                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2166  010636  000007                        .WORD   7               ;ERROR TYPE CODE.
2167  010640                        93$:
2168  010640  005100                        COM     R0              ;COMPLEMENT CHECK WORD
2169  010642  005142                        COM     -(R2)           ;RESTORE DATA
2170  010644  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2171  010646  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
2172  010650  001405                        BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2173  010652  004767  007416        94$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2174  010656  004767  010664                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2175  010662  000007                        .WORD   7               ;ERROR TYPE CODE.
2176  010664                        95$:
2177  010664  010046                        MOV     R0,     -(SP)   ;SAVE R0
2178  010666  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
2179  010670  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2180  010672  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
2181  010674  001213                        BNE     22$             ;BRANCH IF MORE.
2182  010676  010046                        MOV     R0,     -(SP)   ;SAVE R0
2183  010700  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
2184  010702  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2185  010704  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2186  010706  001204                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
2187  010710  004767  004176                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
2188
2189                                ;;*****************************************************************
2190                                ;*TEST 16       COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2191                                ;;*****************************************************************
2192  010714                        TST16:
2193  010714  004567  007616                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2194  010720  000777                        .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
2195                                                                ; REQUIRED FOR THIS TEST.
2196  010722  000167  000610                JMP     TST17           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2197                                                                ; AVAILABLE FOR TEST.
2198  010726  012700  177777                MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
2199  010732  012703  000401                MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
2200  010736  004467  003372                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2201  010742  004767  005216        1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2202  010746  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2203  010750  001374                        BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2204  010752  004767  004134                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2205
2206                                ;;*****************************************************************
2207                                ;*  CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2208                                ;;*****************************************************************
2209  010756  012700  177777                MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
2210  010762  012703  000401                MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
2211  010766  004467  003342                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2212  010772  012704  000100        11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
2213  010776                        12$:
2214  010776  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2215  011000  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2216  011002  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2217  011004  004767  007264        64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2218  011010  004767  010532                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2219  011014  000007                        .WORD   7               ;ERROR TYPE CODE.
2220  011016                        65$:
2221  011016  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2222  011020  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2223  011022  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2224  011024  004767  007244        66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2225  011030  004767  010512                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2226  011034  000007                        .WORD   7               ;ERROR TYPE CODE.
2227  011036                        67$:
```

```
                                                MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                        68$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                .WORD   7               ;ERROR TYPE CODE.
                                        69$:
                                                MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                        70$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                .WORD   7               ;ERROR TYPE CODE.
                                        71$:
                                                MOV     R0,     -(SP)   ;SAVE R0
                                                MOV     R3,     R0      ;PUT R3 INTO R0
                                                MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                                DEC     R4              ;COUNT 256. WORDS
                                                BNE     12$             ;BRANCH IF MORE
                                                MOV     R0,     -(SP)   ;SAVE R0
                                                MOV     R3,     R0      ;PUT R3 INTO R0
                                                MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
                                                BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
                                                BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
                                                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                                        ;;*****************************************************************
                                        ;;  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
                                        ;;*****************************************************************
                                                MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
                                                MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
                                                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
                                        21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
                                        22$:
                                                MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                        72$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                .WORD   7               ;ERROR TYPE CODE.
                                        73$:
                                                COM     R0              ;COMPLEMENT CHECK WORD
                                                COM     -(R2)           ;COMPLEMENT TEST DATA
                                                MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                        74$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
                                                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                                .WORD   7               ;ERROR TYPE CODE.
                                        75$:
                                                COM     R0              ;COMPLEMENT CHECK WORD
                                                COM     -(R2)           ;RESTORE DATA
                                                MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
                                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
                                                BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
                                        76$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
2284  011230  004767  010312          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2285  011234  000007                  .WORD   7               ;ERROR TYPE CODE.
2286  011236                  77$:
2287  011236  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2288  011240  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2289  011242  001405                  BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2290  011244  004767  007024  78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2291  011250  004767  010272          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2292  011254  000007                  .WORD   7               ;ERROR TYPE CODE.
2293  011256                  79$:
2294  011256  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2295  011260  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
2296  011262  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2297  011264  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2298  011266  001405                  BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2299  011270  004767  007000  80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2300  011274  004767  010246          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2301  011300  000007                  .WORD   7               ;ERROR TYPE CODE.
2302  011302                  81$:
2303  011302  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2304  011304  005142                  COM     -(R2)           ;RESTORE DATA
2305  011306  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2306  011310  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2307  011312  001405                  BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2308  011314  004767  006754  82$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2309  011320  004767  010222          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2310  011324  000007                  .WORD   7               ;ERROR TYPE CODE.
2311  011326                  83$:
2312  011326  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2313  011330  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2314  011332  001405                  BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2315  011334  004767  006734  84$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2316  011340  004767  010202          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2317  011344  000007                  .WORD   7               ;ERROR TYPE CODE.
2318  011346                  85$:
2319  011346  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2320  011350  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
2321  011352  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2322  011354  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2323  011356  001405                  BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2324  011360  004767  006710  86$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2325  011364  004767  010156          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2326  011370  000007                  .WORD   7               ;ERROR TYPE CODE.
2327  011372                  87$:
2328  011372  005100                  COM     R0              ;COMPLEMENT CHECK WORD
2329  011374  005142                  COM     -(R2)           ;RESTORE DATA
2330  011376  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2331  011400  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2332  011402  001405                  BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2333  011404  004767  006664  88$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2334  011410  004767  010132          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2335  011414  000007                  .WORD   7               ;ERROR TYPE CODE.
2336  011416                  89$:
2337  011416  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2338  011420  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2339  011422  001405                  BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

```
2340  011424  004767  006644          90$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2341  011430  004767  010112                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2342  011434  000007                          .WORD   7               ;ERROR TYPE CODE.
2343  011436                          91$:
2344  011436  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2345  011440  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
2346  011442  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2347  011444  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2348  011446  001405                          BEQ     92$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2349  011450  004767  006620          92$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2350  011454  004767  010066                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2351  011460  000007                          .WORD   7               ;ERROR TYPE CODE.
2352  011462                          93$:
2353  011462  005100                          COM     R0              ;COMPLEMENT CHECK WORD
2354  011464  005142                          COM     -(R2)           ;RESTORE DATA
2355  011466  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2356  011470  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2357  011472  001405                          BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2358  011474  004767  006574          94$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2359  011500  004767  010042                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2360  011504  000007                          .WORD   7               ;ERROR TYPE CODE.
2361  011506                          95$:
2362  011506  010046                          MOV     R0,     -(SP)   ;SAVE R0
2363  011510  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
2364  011512  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2365  011514  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
2366  011516  001213                          BNE     22$             ;BRANCH IF MORE.
2367  011520  010046                          MOV     R0,     -(SP)   ;SAVE R0
2368  011522  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
2369  011524  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2370  011526  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2371  011530  001204                          BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
2372  011532  004767  003354                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 17       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
                                ;;*****************************************************************
011536                          TST17:
011536  004567  006774                      JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
011542  000000                              .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
011544  005767  170520          WWPB0:      TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
011550  001404                              BEQ     1$              ;BR IF NO PARITY MEMORY.
011552  032777  000100  167360              BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
011560  001402                              BEQ     2$              ;BR IF NOT SET.
011562  000167  000622          1$:         JMP     TST20           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
011566  005000                  2$:         CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
011570  004767  004302                      JSR     PC,     SETCC   ;ROUTINE TO LOAD ALL MEMORY.
011574  004467  002534                      JSR     R4,     INITA   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011600  036767  167740  167732  WWPBYT:     BIT     BITPT,  PMEMAP  ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
011606  001010                              BNE     2$              ;BR IF PARITY MEM.
011610  036767  167732  167724              BIT     BITPT+2,PMEMAP+2;...HI 64K.
011616  001004                              BNE     2$              ;BR IF PARITY MEM.
011620  050502                              BIS     R5,     R2      ;POINT TO END OF BLOCK.
011622  005202                              INC     R2              ;FIRST ADR OF NEXT BLOCK.
011624  000167  000540                      JMP     WWPB5           ;BR TO FIND NEXT BLOCK.
011630  004767  005654          2$:         JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
011634  004767  005704                      JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
011640  020227  000114          WWPB1:      CMP     R2,     #114    ;CKECK IF POINTING TO PARITY ERROR VECTOR.
011644  001004                              BNE     3$              ;BR IF NOT AT VECTOR.
011646  062702  000004                      ADD     #4,     R2      ;SKIP PARITY VECTOR.
011652  000167  000512                      JMP     WWPB5           ;CHECK FOR BLOCK END.
011656  111201                  3$:         MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
011660  001405                              BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011662  004767  006332          64$:        JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
011666  004767  007654                      JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011672  000011                              .WORD   11              ;ERROR TYPE CODE.
011674                          65$:
011674  105067  167660                      CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
011700  112700  000252                      MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
011704  110012                  WWPB2:      MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
011706  016703  167710                      MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
011712  053773  001612  000000  10$:        BIS     @#WWP,@(R3)     ;SET WRITE WRONG PARITY.
011720  052733  000001                      BIS     #AE,@(R3)+
011724  005713                              TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
011726  001371                              BNE     10$             ;BR IF MORE REGS IN TABLE.
                                ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
011730  110012                              MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
011732  016703  167664                      MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
011736  043733  001612          11$:        BIC     @#WWP,  @(R3)+  ;CLEAR WRITE WRONG PARITY.
011742  005713                              TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
011744  001374                              BNE     11$             ;BR IF MORE PARITY REGISTERS.
011746  016737  167652  000114              MOV     .PBTRP, @#PARVEC;SET UP VECTOR FOR EXPECTED TRAP.
                                ;*  DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
011754  105412                              NEGB    (R2)            ;DATIP (DATOB AND COM PARITY BIT.)
                                ;*  SHOULD HAVE TRAPPED TO PBTRP.
```

```
2429  011756  016737  167646  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
2430  011764  004767  006260          64$:    JSR     PC,     SPRNTO   ;SET UP VALUES FOR ERROR PRINTING.
2431  011770  004767  007552                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2432  011774  000012                          .WORD   12               ;ERROR TYPE CODE.
2433  011776  000562                          BR      WWPB4            ;SKIP TRAP SERVICE.
2434
2435                                  ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
2436  012000  016737  167624  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2437  012006  022626                  CMP     (SP)+,  (SP)+            ;RESET THE STACK POINTER AFTER TRAP.
2438  012010  016703  167604                  MOV     .MPRQ,  R3       ;GET PARITY REG AND MAP TABLE POINTER.
2439  012014  032713  000001          21$:    BIT     #BIT0,  (R3)     ;CHECK IF THIS REGISTER EXISTS.
2440  012020  001003                          BNE     22$              ;BR IF IT DOESN'T EXIST.
2441  012022  017301  000000                  MOV     @(R3),  R1       ;GET THE CONTENTS.
2442  012026  100413                          BMI     23$              ;BR IF ERROR FLAG SET.
2443  012030  062703  000010          22$:    ADD     #10,    R3       ;MOVE POINTER TO NEXT REG.
2444  012034  020367  167562                  CMP     R3,     .MPRX    ;CHECK FOR END OF TABLE.
2445  012040  103765                          BLO     21$              ;BR IF MORE REGISTERS.
2446  012042  004767  006202          64$:    JSR     PC,     SPRNTO   ;SET UP VALUES FOR ERROR PRINTING.
2447  012046  004767  007474                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2448  012052  000013                          .WORD   13               ;ERROR TYPE CODE.
2449  012054  000533                          BR      WWPB4            ;EXIT AFTER ERROR.
2450  012056  036763  167462  000002  23$:    BIT     BITPT,  2(R3)    ;CHECK THE MAP FOR THIS REGISTER.
2451  012064  001011                          BNE     24$              ;BR IF THIS REGISTER CONTROLS THIS BANK.
2452  012066  036763  167454  000004          BIT     BITPT+2,4(R3)    ;CHECK THE HI 64K.
2453  012074  001005                          BNE     24$              ;BR IF THIS REGISTER CONTROLS THIS BANK.
2454  012076  004767  006142          65$:    JSR     PC,     SPRNTP   ;SET UP VALUES FOR ERROR PRINTING.
2455  012102  004767  007440                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2456  012106  000014                          .WORD   14               ;ERROR TYPE CODE.
2457  012110                          24$:
2458  012110  010046                          MOV     R0,-(SP)         ;;PUSH R0 ON STACK
2459  012112  010200                          MOV     R2,     R0       ;GET THE ADDRESS POINTER.
2460  012114  042700  003777                  BIC     #3777,  R0       ;CLEAR LOW ADDRESS BITS.
2461  012120  000300                          SWAB    R0               ;SHIFT 6 PLACES RIGHT.
2462  012122  006300                          ASL     R0
2463  012124  006300                          ASL     R0
2464  012126  005767  166454                  TST     MMAVA            ;CHECK FOR MEM MGMT.
2465  012132  001404                          BEQ     25$              ;BR IF NO MEM MGMT.
2466  012134  042700  177600                  BIC     #177600,R0       ;CLEAR BANK BITS
2467  012140  063700  172344                  ADD     @#KIPAR2,R0      ;ADD MEM MGMT OFFSET.
2468  012144  052700  100001          25$:    BIS     #BIT15+BIT0,R0   ;SET ERROR AND AE BIT IN CHECK WORD.
2469  012150  016367  000006  167340          MOV     6(R3),  RESRVD   ;GET APPROPIATE MASK.
2470  012156  046700  167334                  BIC     RESRVD, R0       ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2471  012162  046701  167330                  BIC     RESRVD, R1       ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2472                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2473  012166  020001                          CMP     R0,     R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
2474  012170  001405                          BEQ     67$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2475  012172  004767  006046          66$:    JSR     PC,     SPRNTP   ;SET UP VALUES FOR ERROR PRINTING.
2476  012176  004767  007344                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2477  012202  000015                          .WORD   15               ;ERROR TYPE CODE.
2478  012204                          67$:
2479  012204  005073  000000                  CLR     @(R3)            ;CLEAR REG INCLUDING ACTION ENABLE.
2480  012210  010346                          MOV     R3,-(SP)         ;;PUSH R3 ON STACK
2481  012212  062703  000010          26$:    ADD     #10,    R3       ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2482  012216  020367  167400                  CMP     R3,     .MPRX    ;CHECK FOR END OF TABLE.
2483  012222  101014                          BHI     WWPB3            ;BR IF END OF TABLE REACHED.
2484  012224  032713  000001                  BIT     #BIT0,  (R3)     ;CHECK IF NEXT REG EXISTS.
```

```
2485  012230  001370                        BNE     26$             ;BR IF THIS PARITY REG DOESN'T EXIST.
2486  012232  017301  000000                MOV     @(R3), R1       ;SAVE AND CHECK FOR ERROR FLAG.
2487  012236  100365                        BPL     26$             ;BR IF NO ERROR FLAG.
2488  012240  004767  006000        68$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2489  012244  004767  007276                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2490  012250  000016                        .WORD   16              ;ERROR TYPE CODE.
2491  012252  000757                        BR      26$             ;BR AFTER ERROR.
2492  012254  111204        WWPB3:  MOVB    (R2),   R4              ;GET THE DATA FOR CHECKING.
2493                        ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
2494                        ;*  ACTION ENABLE IS NOT SET IN CONTROLING REG, SO NO TRAP SHOULD OCCURE.
2495  012256  111212                        MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
2496                        ;NOTE:  THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
2497                        ;       WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
2498  012260  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
2499  012262  017301  000000                MOV     @(R3), R1       ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
2500  012266  046701  167224                BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2501                        ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2502  012272  042700  000001                BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
2503  012276  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2504  012300  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2505  012302  004767  005736        64$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
2506  012306  004767  007234                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2507  012312  000015                        .WORD   15              ;ERROR TYPE CODE.
2508  012314                        65$:
2509  012314  012773  000001  000000        MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
2510  012322  010401                        MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
2511  012324  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
2512  012326  120001                        CMPB    R0,     R1      ;CHECK THE DATA.
2513  012330  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2514  012332  004767  005712        66$:    JSR     PC,     SPRNTD  ;SET UP VALUES FOR ERROR PRINTING.
2515  012336  004767  007204                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2516  012342  000017                        .WORD   17              ;ERROR TYPE CODE.
2517  012344                        67$:
2518  012344  110012        WWPB4:  MOVB    R0,     (R2)            ;RESTORE DATA.
2519  012346  105712                        TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
2520  012350  012700  000253                MOV     #253,   R0      ;SET ODD PARITY DATA.
2521  012354  105167  167200                COMB    OEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
2522  012360  100002                        BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
2523  012362  000167  177316                JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
2524  012366  005202        27$:    INC     R2                      ;MOVE POINTER TO NEXT MEMORY BYTE.
2525  012370  030502        WWPB5:  BIT     R5,     R2              ;CHECK FOR END OF BLOCK.
2526  012372  001402                        BEQ     30$             ;BR IF END OF BLOCK FOUND.
2527  012374  000167  177240                JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
2528  012400  004767  002506        30$:    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
2529  012404  004767  005034                JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
                                ;;*****************************************************************
                                ;*TEST 20       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
                                ;;*****************************************************************
012410                          TST20:
012410  004567  006122                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE,
012414  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
012416  010703                  RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
012420  042703  007777                  BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
012424  004467  001704                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012430  010246                  1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
012432  010346                          MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
012434  012322                  2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
012436  032703  007777                  BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
012442  001002                          BNE     3$              ;BRANCH IF MORE
012444  162703  010000                  SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA"
012450  030502                  3$:     BIT     R5,     R2      ;CHECK FOR END OF BLOCK
012452  001370                          BNE     2$              ;BRANCH IF MORE.
012454  012603                          MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
012456  012602                          MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
012460  012300                  4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
012462  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
012464  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012466  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012470  004767  005600          64$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
012474  004767  007046                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012500  000020                          .WORD   20              ;ERROR TYPE CODE.
012502                          65$:
012502  032703  007777                  BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
012506  001002                          BNE     5$              ;BR IF MORE.
012510  162703  010000                  SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
012514                          5$:
012514  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012516  001360                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
012520  004767  002366                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
                                ;;*****************************************************************
                                ;*TEST 21       EXECUTE DATI, DATO THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000           010412          000205
                                ;*  THRU TEST /  40002           000205          000205
                                ;*
                                ;*   2ND PASS /  40002           010412          000205
                                ;*  THRU TEST /  40004           000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  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.
                                ;;*****************************************************************
012524                          TST21:
012524  004567  006006                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012530  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
012532  000167  000056                  JMP     TST22           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
012536  012703  010412          DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
012542  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
012546  010400                          MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
012550  004467  001560                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012554  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012556  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012560  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
012562  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012564  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012566  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012570  004767  005474  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012574  004767  006746          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012600  000021                  .WORD   21              ;ERROR TYPE CODE.
012602                  65$:
012602  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012604  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012606  001363                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
012610  004767  002276          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 22       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.
                                ;;*****************************************************************
012614                          T$122:
012614  004567  005716                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012620  000003                        .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
012622  000167  000060                JMP     TST23           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
012626  012703  110412        DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
012632  012704  000205                MOV     #205,   R4      ;GET 'RTS R5'
012636  012700  110605                MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
012642  004467  001466                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012646  010322                1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012650  010412                2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012652  004542                        JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
012654  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012656  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012660  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012662  004767  005402        64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012666  004767  006654                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012672  000021                        .WORD   21              ;ERROR TYPE CODE.
012674                          65$:
012674  010322                        MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012676  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012700  001363                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
012702  004767  002204                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 23       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(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          110342          161342
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          110342          161342
                                ;*  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.
                                ;*****************************************************************
012706                          TST23:
012706  004567  005624                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012712  000003                        .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
012714  000167  000064                JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
012720  012703  110342        DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
012724  012704  000205                MOV     #205,   R4      ;GET 'RTS R5'
012730  012700  161342                MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
012734  004467  001374                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012740  010322        1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012742  010412        2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012744  004562  177776                JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
012750  005302                        DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
012752  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012754  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012756  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012760  004767  005304        64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012764  004767  006556                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012770  000021                        .WORD   21              ;ERROR TYPE CODE.
012772                          65$:
012772  010322                        MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012774  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012776  001361                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013000  004767  002106                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
2713                            ;;*****************************************************************
2714                            ;*TEST 24       EXECUTE DATI, DATIP, DATO THRU MEMORY.
2715                            ;*  EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
2716                            ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
2717                            ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
2718                            ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:   .
2719                            ;*
2720                            ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
2721                            ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
2722                            ;*
2723                            ;*   1ST PASS /  40000           005412          172366
2724                            ;*  THRU TEST /  40002           000205          000205
2725                            ;*
2726                            ;*   2ND PASS /  40002           005412          172366
2727                            ;*  THRU TEST /  40004           000205          000205
2728                            ;*
2729                            ;*      ETC., ETC., ETC.
2730                            ;*
2731                            ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
2732                            ;*  R1 = DATA READ FROM MEMORY (WAS).
2733                            ;*  R2 = ADDRESS OF IUT/DATA.
2734                            ;*  R3 = INSTRUCTION UNDER TEST (IUT).
2735                            ;*  R4 = RTS R5 (CODE 205).
2736                            ;*  R5 = BLOCK BOUNDRY BIT MASK.
2737                            ;;*****************************************************************
2738  013004                    TST24:
2739  013004  004567  005526            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2740  013010  000003            .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
2741                                                    ; REQUIRED FOR THIS TEST.
2742  013012  000167  000060            JMP     TST25           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2743                                                    ; AVAILABLE FOR TEST.
2744  013016  012703  005412    DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
2745  013022  012704  000205            MOV     #205,   R4      ;GET 'RTS R5'
2746  013026  012700  172366            MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
2747  013032  004467  001276            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2748  013036  010322    1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2749  013040  010412    2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
2750  013042  004542            JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
2751  013044  012201            MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
2752  013046  020001            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2753  013050  001405            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2754  013052  004767  005212    64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
2755  013056  004767  006464            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2756  013062  000021            .WORD   21              ;ERROR TYPE CODE.
2757  013064                    65$:
2758  013064  010322            MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
2759  013066  030502            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2760  013070  001363            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
2761  013072  004767  002014            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;;*****************************************************************
                        ;*TEST 25      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.
                        ;;*****************************************************************
013076                  TST25:
013076  004567  005434          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013102  000003          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                ; REQUIRED FOR THIS TEST.
013104  000167  000060          JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                ; AVAILABLE FOR TEST.
013110  012703  142242  DPDBL:   MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
013114  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013120  012700  142000          MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
013124  004467  001204          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013130  010322  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013132  010412  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013134  004542          JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013136  012201          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013140  020001          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013142  001405          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013144  004767  005120  64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013150  004767  006372          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013154  000021          .WORD   21              ;ERROR TYPE CODE.
013156                  65$:
013156  010322          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013160  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013162  001363          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013164  004767  001722          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;*****************************************************************
                        ;*TEST 26       EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
                        ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                        ;*
                        ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                        ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                        ;*
                        ;*   1ST PASS /  40000           152212          157212
                        ;*  THRU TEST /  40002           000205          000205
                        ;*
                        ;*   2ND PASS /  40002           152212          157212
                        ;*  THRU TEST /  40004           000205          000205
                        ;*
                        ;*      ETC., ETC., ETC.
                        ;*
                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                        ;*  R2 = ADDRESS OF IUT/DATA.
                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                        ;*  R4 = RTS R5 (CODE 205).
                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
                        ;*****************************************************************
013170                  TST26:
013170  004567  005342          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013174  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
013176  000167  000062          JMP     TST27           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
013202  012703  152212  DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
013206  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013212  012700  157212          MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
013216  004467  001112          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013222  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013224  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013226  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013230  005302                  DEC     R2              ;RESET R2 TO POINT TO IUT.
013232  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013234  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013236  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013240  004767  005024  64$:    JSR     PC,     $PRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013244  004767  006276          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013250  000021                  .WORD   21              ;ERROR TYPE CODE.
013252                  65$:
013252  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013254  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013256  001362                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013260  004767  001626          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        .SBTTL SECTION 4:MOS TESTS
                        ;*****************************************************************
                        ;*TEST 27       SEARCHING 1'S AND 0'S.
                        ;* THIS TEST IS DESIGNED TO STRESS MOS MEMORIES.
                        ;* STARTING AT THE BOTTOM ADDRESS AND ADDRESSING UPWARDS A 4K BANK IS
                        ;* WRITTEN WITH 000377.THEN STARTING AT THE TOP ADDRESS OF THE BANK THE
                        ;* 000377 IS READ,THE BYTES ARE SWAPPED TO 177400 AND THE LOCATION
                        ;* REREAD TO CONFIRM THE WRITE.THIS IS REPEATED FOR EVERY LOCATION
                        ;* ADDRESSED DOWNWARD UNTIL THE BOTTOM IS REACHED. STARTING AT THE
                        ;* BOTTOM EACH LOCATION IS READ FOR 177400,THE BYTES ARE SWAPPED TO
                        ;* 000377 AND REREAD TO CONFIRM THE WRITE UNTIL THE TOP ADDRESS OF THE
                        ;* BANK IS REACHED. AGAIN STARTING AT THE BOTTOM EACH LOCATION IS READ
                        ;* FOR 000377,THE BYTES SWAPPED TO 177400 AND THE LOCATION REREAD TO
                        ;* CONFIRM THE WRITE. LASTLY STARTING FROM THE TOP AND ADDRESSING DOWN-
                        ;* WARD EACH LOCATION IS READ,THE BYTES SWAPPED TO 000377 AND THE
                        ;* LOCATION IS REREAD TO CONFIRM THE WRITE. THIS IS REPEATED FOR EVERY
                        ;*4K BANK UNDER TEST.
                        ;*
                        ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
                        ;*      R1=DATA READ FROM MEMORY(WAS)
                        ;*      R2=VIRTUAL ADDRESS
                        ;*      R3=TIMES THROUGH COUNTER
                        ;*      R4=NOT USED
                        ;*      R5=BLOCK BOUNDARY BIT MASK.
                        ;*****************************************************************
013264                  TST27:
013264  004567  005246          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013270  000000                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013272  004467  001036          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013276  005003          1$:     CLR     R3              ;CLEAR PASS COUNTER
013300  012700  000377          MOV     #000377,R0      ;SETUP TO WRITE PATTERN
013304  010022          2$:     MOV     R0,(R2)+        ;WRITE PATTERN
013306  030502                  BIT     R5,R2           ;END OF 4K?
013310  001375                  BNE     2$              ;CONTINUE WRITING IF NO.
013312  014201          3$:     MOV     -(R2),R1        ;GET DATA WRITEN
013314  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013316  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013320  004767  004750  64$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013324  004767  006216          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013330  000010                  .WORD   10              ;ERROR TYPE CODE.
013332                  65$:
013332  000300          4$:     SWAB    R0              ;SWAP BYTES OF DATA
013334  010012                  MOV     R0,(R2) ;WRITE SWAPPED WORD
013336  011201                  MOV     (R2),R1         ;GET DATA WRITEN
013340  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013342  001405                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013344  004767  004724  66$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013350  004767  006172          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013354  000010                  .WORD   10              ;ERROR TYPE CODE.
013356                  67$:
013356  000300                  SWAB    R0              ;PUT DATA BACK TO ORINGINAL
013360  005703                  TST     R3              ;IF ON PASS 0 OR PASS 3
013362  001403                  BEQ     5$              ;WE ARE ADDRESSING DOWN
013364  020327  000003          CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
013370  001010                  BNE     6$              ;UPWARD
013372  030502          5$:     BIT     R5,R2           ;DONE A PASS?
```

```
2917  013374  001346                    BNE     3$              ;IF NO CONTINUE
2918  013376  005203                    INC     R3              ;IF YES INCREMENT PASS COUNTER
2919  013400  022703  000004            CMP     #4,R3           ;ARE WE DONE ALL PASSES FOR THIS 4K?
2920  013404  001427                    BEQ     9$              ;IF YES BRANCH
2921  013406  000300                    SWAB    R0              ;ELSE SET UP NEW READ WORD
2922  013410  000404                    BR      7$              ;GO TO START OF ADDRESS UP
2923  013412  062702  000002    6$:     ADD     #2,R2           ;UPDATE TO NEXT ADDRESS
2924  013416  030502                    BIT     R5,R2           ;DONE A PASS
2925  013420  001411                    BEQ     8$              ;IF YES BRANCH
2926  013422  011201            7$:     MOV     (R2),R1         ;GET DATA WRITTEN
2927  013424  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2928  013426  001405                    BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2929  013430  004767  004640    68$:    JSR     PC,     $PRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2930  013434  004767  006106            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2931  013440  000010                    .WORD   10              ;ERROR TYPE CODE.
2932  013442                    69$:
2933  013442  000733                    BR      4$
2934  013444  005203            8$:     INC     R3              ;INCREMENT      PASS COUNTER
2935  013446  000300                    SWAB    R0              ;SET UP NEW READ WORD
2936  013450  020327  000002            CMP     R3,#2           ;ADDRESSING UP?
2937  013454  001016                    BNE     3$              ;IF NO GO TO DOWN SEQUENCE
2938  013456  012702  040000            MOV     #40000,R2       ;IF YES RESET ADDRESS TO START
2939  013462  000757                    BR      7$              ;GO TO UP SEQUENCE
2940  013464  004767  001422    9$:     JSR     PC,MMUP         ;UPDATE TO NEW BANK IF EXISTS
2941
2942                              ;;*****************************************************************
2943                              ;*TEST 30       WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2944                              ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
2945                              ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
2946                              ;* DETERIORATE BETWEEN REFRESH CYCLES.
2947                              ;*
2948                              ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
2949                              ;*      R1=DATA READ FROM MEMORY(WAS)
2950                              ;*      R2=VIRTUAL ADDRESS
2951                              ;*      R3=SMALL LOOP COUNTER FOR STALL
2952                              ;*      R4=NUMBER OF TIMES SMALL LOOP DONE
2953                              ;*      R5=BLOCK BOUNDARY BIT MASK.
2954                              ;;*****************************************************************
2955  013470                    1ST30:
2956  013470  004567  005042            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2957  013474  000000                    .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2958  013476  004467  000632            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2959  013502  012700  125252            MOV     #125252,R0      ;SETUP DATA PATTERN
2960  013506  010022            1$:     MOV     R0      ,(R2)+  ;WRITE A WORD
2961  013510  005100                    COM     R0              ;COMPLEMENT DATA
2962  013512  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2963  013514  001374                    BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2964  013516  004767  001370            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2965  013522  005003                    CLR     R3              ;SET UP COUNTER FOR STALL
2966  013524  012704  000046            MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
2967  013530  005303            2$:     DEC     R3
2968  013532  001376                    BNE     2$
2969  013534  005304                    DEC     R4
2970  013536  001374                    BNE     2$
2971  013540  004467  000570            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2972  013544  012700  125252            MOV     #125252,R0      ;INIT DATA FOR CHECKING
```

```
2973  013550                          3$:
2974  013550  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2975  013552  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2976  013554  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2977  013556  004767  004512        64$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2978  013562  004767  005760                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2979  013566  000006                          .WORD   6               ;ERROR TYPE CODE.
2980  013570                          65$:
2981  013570  005100                          COM     R0
2982  013572  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2983  013574  001365                          BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
2984  013576  004767  001310                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2985                                  ;;*****************************************************************
2986                                  ;*TEST 31        WRITE CHECKERBOARD STARTING WITH 052525 DATA
2987                                  ;;*****************************************************************
2988  013602                          TST31:
2989  013602  004567  004730                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2990  013606  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2991  013610  004467  000520                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2992  013614  012700  052525                  MOV     #052525,R0      ;SETUP DATA PATTERN
2993  013620  010022          1$:       MOV     R0,     (R2)+   ;WRITE A WORD
2994  013622  005100                          COM     R0
2995  013624  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
2996  013626  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
2997  013630  004767  001256                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
2998  013634  005003                          CLR     R3              ;SET COUNTER FOR LOOP
2999  013636  012704  000046                  MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
3000  013642  005303          2$:       DEC     R3
3001  013644  001376                          BNE     2$
3002  013646  005304                          DEC     R4
3003  013650  001374                          BNE     2$
3004  013652  004467  000456                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3005  013656  012700  052525                  MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
3006  013662                          3$:
3007  013662  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
3008  013664  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
3009  013666  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
3010  013670  004767  004400        64$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
3011  013674  004767  005646                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3012  013700  000006                          .WORD   6               ;ERROR TYPE CODE.
3013  013702                          65$:
3014  013702  005100                          COM     R0
3015  013704  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3016  013706  001365                          BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
3017  013710  004767  001176                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                                .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
013714                                          DONE:
013714  004567  004616                                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013720  000000                                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013722  005067  165242                        TST32:  CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
013726  105067  165150                                CLRB    $TSTNM          ;RESET TEST NUMBER.
013732  036767  164644  165574                1$:     BIT     PRGMAP, SAVTST  ;CHECK IF PROGRAM IS IN TEST AREA.
013740  001004                                        BNE     2$              ;BR IF IT PROG IN MEM TO BE TESTED.
013742  036767  164636  165566                        BIT     PRGMAP+2,SAVTST+2 ;CHECK HI 64K
013750  001434                                        BEQ     $EOP            ;BR IF PROG NOT IN MEM TO BE TESTED.
013752  032777  000200  165160                2$:     BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
013760  001030                                        BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
013762  022767  000003  164612                        CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IN FIRST 8K.
013770  001012                                        BNE     4$              ;BR IF NOT IN FIRST 8K.
013772  005737  000042                                TST     @#42            ;CHECK FOR A ACT11.
013776  001014                                        BNE     5$              ;BR IF A ACT11.
014000  105737  001224                                TSTB    @#$ENV          ;CHECK FOR APT
014004  001011                                        BNE     5$              ;IF APT DO NOT RELOCATE
014006  004767  002354                                JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
014012  000167  172030                        3$:     JMP     START1          ;LOOP BACK AND RUN ALL TESTS AGAIN.

014016  004767  002746                        4$:     JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
014022  005737  000042                                TST     @#42            ;TEST FOR XXDP
014026  001402                                        BEQ     6$              ;IF NOT RUNNING UNDER MON. DONT
014030  004767  003142                        5$:     JSR     PC,     RESLDR  ;RESTORE LOADERS.
014034                                          6$:
014034  004567  007360                                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
014040  001201                                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

```
                                        ;;*****************************************************************
                                        .SBTTL  END OF PASS ROUTINE

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

014042                                  $EOP:
014042  000240                                  NOP
014044  005067  165120                          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
014050  005267  165136                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
014054  042767  100000  165130                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
014062  005327                                  DEC     (PC)+           ;;LOOP?
014064  000001                          $EOPCT: .WORD   1
014066  003035                                  BGT     $DOAGN          ;;YES
014070  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
014072  000001                          $ENDCT: .WORD   1
014074  014064                                  $EOPCT
014076  004567  007316                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
014102  014166                                  .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
014104  016746  165102                          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
014110  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
014114  004767  010220                          JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
014120  004567  007274                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
014124  014203                                  .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
014126                                  $GET42:

014126  016700  163710                          MOV     42,     R0              ;;GET MONITOR ADDRESS
014132  001413                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
014134  000005                                  RESET                   ;;CLEAR THE WORLD
014136  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
014140  000240                                  NOP                     ;;SAVE ROOM
014142  000240                                  NOP                     ;;FOR
014144  000240                                  NOP                     ;;ACT11
014146  023737  000042  000046                  CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
014154  001402                                  BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
014156  004767  003074                          JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
014162                                  $DOAGN:
014162  000167  171660                          JMP     START1          ;;RETURN*****
014166  005015  047105  020104          $ENDMG: .ASCIZ  <15><12>/END PASS #/
014174  040520  051523  021440
014202     000
014203     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                        .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
                                        .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
                                        ;;*****************************************************************
                                        ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
                                        ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
                                        ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
                                        ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
                                        ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                        ;;*****************************************************************
```

```
3102  014206                          MMINIT:
3103  014206  012737  077406  172300          MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
3104  014214  012737  077406  172302          MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
3105  014222  012737  077406  172304          MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
3106  014230  012737  077406  172306          MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
3107  014236  005037  172310                  CLR     @#KIPDR4
3108  014242  005037  172312                  CLR     @#KIPDR5
3109  014246  005037  172314                  CLR     @#KIPDR6
3110  014252  012737  077406  172316          MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
3111  014260  005037  172340                  CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
3112  014264  012737  000200  172342          MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
3113  014272  005037  172344                  CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
3114  014276  005037  172346                  CLR     @#KIPAR3
3115  014302  005037  172350                  CLR     @#KIPAR4
3116  014306  005037  172352                  CLR     @#KIPAR5
3117  014312  005037  172354                  CLR     @#KIPAR6
3118  014316  012737  007600  172356          MOV     #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
3119  014324  012737  000001  177572          MOV     #1,     @#SR0   ;ENABLE MEMORY MANAGEMENT
3120  014332  000207                          RTS     PC              ;RETURN
3121
3122
3123                                  ;;*****************************************************************
3124                                  ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
3125                                  ;;*****************************************************************
3126  014334  012767  000001  165202  INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
3127  014342  005067  165200                  CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
3128  014346  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
3129  014350  016705  165232                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
3130  014354  005767  164226                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
3131  014360  001514                          BEQ     10$             ;BRANCH IF NO MEM MGMT
3132  014362  005037  172344                  CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
3133  014366  012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
3134  014372  036767  165146  165130  1$:     BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
3135  014400  001015                          BNE     2$              ;BRANCH IF MATCH
3136  014402  036767  165140  165122          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3137  014410  001011                          BNE     2$              ;BRANCH IF MATCH
3138  014412  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3139  014420  006367  165120                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
3140  014424  006167  165116                  ROL     BITPT+2         ;...HI POINTER.
3141  014430  100360                          BPL     1$              ;BR IF MORE.
3142  014432  000000                          HALT                    ;FATAL ERROR!!!  NO 4K BANK FOUND?
3143  014434  036767  165104  165140  2$:     BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
3144  014442  001004                          BNE     3$              ;BR IF LAST BANK.
3145  014444  036767  165076  165132          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
3146  014452  001405                          BEQ     4$              ;BR IF NOT LAST BANK.
3147  014454  016705  165120          3$:     MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
3148  014460  042767  020000  165110          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
3149  014466  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3150  014474  016767  165044  165046          MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
3151  014502  016767  165040  165042          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
3152  014510  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
3153  014514  001505                          BEQ     21$             ;BRANCH IF NOT 8K.
3154  014516  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
3155  014524  006367  165020                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
3156  014530  006167  165016                  ROL     TMPPT+2         ;...HI POINTER.
3157  014534  100473                          BMI     20$             ;BR IF NO MORE.
```

```
3158  014536  036767  165006  164764          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3159  014544  001004                          BNE     6$              ;BRANCH IF A MATCH.
3160  014546  036767  165000  164756          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3161  014554  001760                          BEQ     5$              ;BRANCH IF NO MEMORY
3162  014556  036767  164766  165016  6$:     BIT     TMPPT,  LADMAP  ;CHECK IF LAST BANK.
3163  014564  001004                          BNE     7$              ;BRANCH IF A MATCH
3164  014566  036767  164760  165010          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
3165  014574  001455                          BEQ     21$             ;BR IF NOT LAST BANK.
3166  014576  016705  164776          7$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
3167  014602  052767  020000  164766          BIS     #20000, TMPLAD  ;MAKE SURE LAST ADDRESS IS IN BANK 3.
3168  014610  000447                          BR      21$             ;BR TO FINISH UP.
3169
3170  014612  036767  164726  164710  10$:    BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED.
3171  014620  001006                          BNE     11$             ;BR IF MATCH.
3172  014622  062702  020000                  ADD     #20000, R2      ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3173  014626  106367  164712                  ASLB    BITPT           ;UPDATE BANK POINTER TO NEXT BANK.
3174  014632  100367                          BPL     10$             ;BR IF MORE BANKS.
3175  014634  000000                          HALT                    ;FATAL ERROR!!!  NO 4K BANK FOUND?
3176  014636  016767  164702  164704  11$:    MOV     BITPT, TMPPT    ;COPY BANK POINTER.
3177  014644  036767  164674  164730          BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
3178  014652  001021                          BNE     12$             ;BR IF LAST BANK.
3179  014654  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR 8K BLOCK SIZE.
3180  014660  001423                          BEQ     21$             ;BRANCH IF SMALLER BLOCK SIZE.
3181  014662  106367  164662                  ASLB    TMPPT           ;POINT TO NEXT BANK.
3182  014666  100416                          BMI     20$             ;BRANCH IF OVERFLOW.
3183  014670  036767  164654  164632          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3184  014676  001412                          BEQ     20$             ;BRANCH IF NOT TO BE TESTED.
3185  014700  112767  000011  164651          MOVB    #11,    FLAG8K  ;SET 8K BLOCK SIZE FLAG.
3186  014706  036767  164636  164666          BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK.
3187  014714  001403                          BEQ     20$             ;BR IF NOT LAST BANK.
3188  014716  016705  164656          12$:    MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
3189  014722  000402                          BR      21$             ;SKIP MASK RESET.
3190  014724  012705  017777          20$:    MOV     #MASK4K,R5      ;RESET MASK TO 4K BLOCK SIZE.
3191  014730  056767  164610  164612  21$:    BIS     BITPT,  TMPPT   ;SET TMPPT FOR FLAGING LAST BANK.
3192  014736  056767  164604  164606          BIS     BITPT+2,TMPPT+2
3193  014744  036767  164574  164616          BIT     BITPT,  FADMAP  ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3194  014752  001004                          BNE     22$             ;BR IF FIRST BANK.
3195  014754  036767  164566  164610          BIT     BITPT+2,FADMAP+2 ;CHECK HI 64K.
3196  014762  001450                          BEQ     INITEX          ;BR IF NOT FIRST BANK.
3197  014764  016702  164574          22$:    MOV     TMPFAD, R2      ;RESET ADDRESS POINTER TO FIRST ADR.
3198  014770  000445                          BR      INITEX
3199
3200  014772  016705  164610          INITDN: MOV     BLKMSK, R5      ;RESET R5 TO CURRENT BLOCK MASK.
3201  014776  005002                          CLR     R2              ;INIT ADDRESS POINTER.
3202  015000  005767  163602                  TST     MMAVA           ;CHECK FOR MEM MGMT
3203  015004  001411                          BEQ     31$             ;BRANCH IF NO MEM MGMT
3204  015006  012767  100000  164532          MOV     #BIT15, BITPT+2 ;SET POINTER TO TOP BIT
3205  015014  005067  164524                  CLR     BITPT
3206  015020  012737  007600  172344          MOV     #7600,  @#KIPAR2 ;SET PAR TO TOP OF MEM
3207  015026  000403                          BR      32$             ;BRANCH TO COMMON AREA
3208
3209  015030  012767  000400  164506  31$:    MOV     #BIT8,  BITPT   ;SET UP BANK POINTER
3210  015036  012767  015060  164510  32$:    MOV     #33$,   MMORE   ;SET "MMDOWN" EXIT ADDRESS.
3211  015044  066767  163530  164502          ADD     RELOCF, MMORE   ;ADD OFFSET
3212  015052  004767  000524                  JSR     PC,     MMDOWN  ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3213  015056  000000                          HALT                    ;FATAL ERROR!!!  NO MEM INDICATED IN MEM MAP ABOVE 8K!
```

```
3214  015060  036767  164460  164514  33$:    BIT     BITPT,  LADMAP  ;CHECK FOR NON BOUNDRY LAST ADDR.
3215  015066  001004                          BNE     34$             ;BR IF LAST BANK FLAG FOUND.
3216  015070  036767  164452  164506          BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
3217  015076  001402                          BEQ     INITEX          ;BR IF NO LAD FLG FOUND.
3218  015100  016702  164470          34$:    MOV     LSTADR, R2      ;SET UP R2.
3219  015104  010467  164444          INITEX: MOV     R4,     MMORE   ;PUT RETURN PC INTO "MMORE"
3220  015110  000204                          RTS     R4              ;RETURN
3221
3222                                  ;*****************************************************************
3223                                  ;*  COMMON UPWARDS ADDRESSING ROUTINE
3224                                  ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3225                                  ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3226                                  ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3227                                  ;*****************************************************************
3228  015112  036767  164432  164462  MMUP:   BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3229  015120  001122                          BNE     10$             ;BR IF LAST BANK.
3230  015122  036767  164424  164454          BIT     TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3231  015130  001116                          BNE     10$             ;BR IF LAST BANK.
3232  015132  016705  164450                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
3233  015136  005767  163444                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
3234  015142  001515                          BEQ     20$             ;BRANCH IF NO MEM MGMT
3235  015144  012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
3236  015150  062737  000200  172344  1$:     ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3237  015156  006367  164362                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
3238  015162  006167  164360                  ROL     BITPT+2         ;...HI POINTER.
3239  015166  100577                          BMI     32$             ;BR IF ALL DONE.
3240  015170  036767  164350  164332          BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK EXISTS
3241  015176  001004                          BNE     2$              ;BRANCH IF MATCH
3242  015200  036767  164342  164324          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3243  015206  001760                          BEQ     1$              ;BRANCH IF NO MATCH
3244  015210  036767  164330  164364  2$:     BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3245  015216  001004                          BNE     3$              ;BRANCH IF LAST BANK FLAG.
3246  015220  036767  164322  164356          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3247  015226  001405                          BEQ     4$              ;BR IF NOT LAST BANK.
3248  015230  016705  164344          3$:     MOV     LADMSK, R5      ;RESET MASK.
3249  015234  042767  020000  164334          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3250  015242  016767  164276  164300  4$:     MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
3251  015250  016767  164272  164274          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
3252  015256  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
3253  015262  001530                          BEQ     31$             ;BRANCH IF NOT.
3254  015264  013737  172344  172346          MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3255  015272  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
3256  015300  006367  164244                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
3257  015304  006167  164242                  ROL     TMPPT+2         ;...HI POINTER.
3258  015310  100513                          BMI     30$             ;BR IF NO MORE.
3259  015312  036767  164232  164210  6$:     BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3260  015320  001004                          BNE     7$              ;BRANCH IF A MATCH.
3261  015322  036767  164224  164202          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3262  015330  001760                          BEQ     5$              ;BRANCH IF NO MEMORY
3263  015332  036767  164212  164242  7$:     BIT     TMPPT,LADMAP    ;CHECK FOR LAST BANK FLAG.
3264  015340  001004                          BNE     8$              ;BRANCH IF A MATCH
3265  015342  036767  164204  164234          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
3266  015350  001475                          BEQ     31$             ;BR IF NO LAST BANK FLAG.
3267  015352  016705  164226          8$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADDRESS.
3268  015356  052767  020000  164212          BIS     #20000, TMPLAD  ;SET VIRTUAL ADR TO BANK 3.
3269  015364  000467                          BR      31$
```

```
3270
3271  015366  026702  164204          10$:    CMP     TMPLAD, R2      ;CHECK IF LAST ADR REACHED.
3272  015372  001064                          BNE     31$             ;BR IF MORE.
3273  015374  000474                          BR      32$             ;BR IF ALL DONE.
3274
3275  015376  106267  164155          20$:    ASRB    FLAG8K          ;SHIFT 8K FLAG
3276  015402  001407                          BEQ     22$             ;BR IF NOT 8K BLOCK.
3277  015404  103455                          BCS     30$             ;BR IF ANOTHER 4K.
3278  015406  105067  164145                  CLRB    FLAG8K          ;CLEAR OUT ALL FLAGS.
3279  015412  162702  040000                  SUB     #40000, R2      ;BACK UP 8K.
3280  015416  062702  020000          21$:    ADD     #20000, R2      ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3281  015422  106367  164116          22$:    ASLB    BITPT           ;UPDATE POINTER.
3282  015426  100457                          BMI     32$             ;BRANCH WHEN END IS REACHED.
3283  015430  036767  164110  164072          BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK EXISTS.
3284  015436  001767                          BEQ     21$             ;BRANCH IF NO MATCH.
3285  015440  036767  164100  164134          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3286  015446  001402                          BEQ     23$             ;BR IF NO MATCH.
3287  015450  016705  164124                  MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
3288  015454  016767  164064  164066  23$:    MOV     BITPT,  TMPPT   ;SET UP TMP POINTER.
3289  015462  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR 8K BLOCK SIZE.
3290  015466  001426                          BEQ     31$             ;BRANCH IF SMALLER BLOCK SIZE.
3291  015470  106367  164054                  ASLB    TMPPT           ;POINT TO NEXT BANK.
3292  015474  100421                          BMI     30$             ;BRANCH IF OVERFLOW.
3293  015476  036767  164046  164024          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
3294  015504  001415                          BEQ     30$             ;BRANCH IF NOT TO BE TESTED.
3295  015506  036767  164032  164066          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3296  015514  112767  000011  164035          MOVB    #11,    FLAG8K  ;SET 8K BLOCK FLAG.
3297  015522  036767  164016  164052          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
3298  015530  001403                          BEQ     30$             ;BR IF NO FLAG.
3299  015532  016705  164042                  MOV     LADMSK, R5      ;RESET MASK TOFIND LAST ADR.
3300  015536  000402                          BR      31$
3301  015540  012705  017777          30$:    MOV     #MASK4K,R5      ;SET MASK TO 4K.
3302  015544  056767  163774  163776  31$:    BIS     BITPT,  TMPPT   ;SET TMPPT FOR FINDING LAST ADR.
3303  015552  056767  163770  163772          BIS     BITPT+2,TMPPT+2
3304  015560  016716  163770                  MOV     MMORE,  (SP)    ;FUDGE RETURN ADDRESS TO LOOP.
3305  015564  000207                          RTS     PC              ;RETURN
3306                                ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3307  015566  005767  164476          32$:    TST     MPRX            ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3308  015572  001402                          BEQ     33$             ;BR IF NONE.
3309  015574  004767  001744                  JSR     PC,     CKPMER  ;CHECK FOR PARITY MEMORY ERRORS.
3310  015600  000207                  33$:    RTS     PC              ;STRAIGHT RETURN.
3311
3312                                ;;*****************************************************************
3313                                ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
3314                                ;*  FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
3315                                ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
3316                                ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3317                                ;;*****************************************************************
3318  015602  036767  163736  163760  MMDOWN: BIT     BITPT,  FADMAP  ;CHECK FOR FIRST ADR FLAG.
3319  015610  001004                          BNE     1$              ;BR IF FIRST ADR IN THIS BANK.
3320  015612  036767  163730  163752          BIT     BITPT+2,FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
3321  015620  001404                          BEQ     2$              ;BR IF NO FLAG
3322  015622  026702  163736          1$:     CMP     TMPFAD, R2      ;CHECK IF FIRST ADDRESS REACHED.
3323  015626  001052                          BNE     9$              ;BR IF MORE.
3324  015630  000453                          BR      10$             ;BR IF ALL DONE.
3325  015632  005767  162750          2$:     TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE
```

```
3326  015636  001425                          BEQ     6$              ;BRANCH IF NOT
3327  015640  162737  000200  172344  3$:     SUB     #200, @#KIPAR2  ;LOWER MEM MGMT PAR BY 4K
3328  015646  006067  163674                  ROR     BITPT+2         ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
3329  015652  006067  163666                  ROR     BITPT           ;...LO MAP.
3330  015656  103440                          BCS     10$             ;BR IF NO MORE.
3331  015660  036767  163660  163642          BIT     BITPT,  TSTMAP  ;CHECK FOR BANK EXISTING
3332  015666  001004                          BNE     4$              ;BR IF BANK TO BE TESTED.
3333  015670  036767  163652  163634          BIT     BITPT+2,TSTMAP+2 ;CHECK FOR BANK IN HI MAP.
3334  015676  001760                          BEQ     3$              ;BR IF NOT THERE.
3335  015700  012702  060000          4$:     MOV     #60000, R2      ;SET ADR POINTER TO TOP OF BANK
3336  015704  000411                          BR      7$              ;GO TO COMMON EXIT
3337  015706  162702  020000          5$:     SUB     #20000, R2      ;BACK POINTER DOWN ONE BANK
3338  015712  006267  163626          6$:     ASR     BITPT           ;MOVE POINTER TO NEXT LOWER BANK
3339  015716  103420                          BCS     10$             ;BRANCH TO EXIT IF NO MORE  MEM
3340  015720  036767  163620  163602          BIT     BITPT,  TSTMAP  ;CHECK IF BANK EXISTS
3341  015726  001767                          BEQ     5$              ;BRANCH IF BANK DOESN'T EXIST
3342  015730  036767  163610  163632  7$:     BIT     BITPT,  FADMAP  ;CHECK IF FIRST BANK FLAG.
3343  015736  001004                          BNE     8$              ;BR IF FIRST BANK.
3344  015740  036767  163602  163624          BIT     BITPT+2,FADMAP+2 ;CHECK IF FIRST BANK FLAG.
3345  015746  001402                          BEQ     9$              ;BR IF NO FLAG FOUND.
3346  015750  016705  163612          8$:     MOV     FADMSK, R5      ;SET UP R5 TO FIND FIRST ADDRESS.
3347  015754  016716  163574          9$:     MOV     MMORE,  (SP)    ;RESET RETURN ADDRESS
3348  015760  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.
                                ;;*****************************************************************
016200                          PHYADR: MOV     R2,     R0      ;VITRUAL INTO R0
005067  163170                          CLR     $TMP0           ;CLEAR TEMP SAVE OF HIGH BITS
005767  162612                          TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
001417                                  BEQ     1$              ;BRANCH IF NO MEM MGMT
010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
013701  172344                          MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
006301                                  ASL     R1              ;SHIFT IT 6 TIMES
006301                                  ASL     R1
006301                                  ASL     R1
006301                                  ASL     R1
006301                                  ASL     R1
006167  163136                          ROL     $TMP0           ;SAVE EXTRA BITS
006301                                  ASL     R1
006167  163130                          ROL     $TMP0
060100                                  ADD     R1,     R0      ;ADD SHIFTED PAR TO VIRTUAL
012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
000207                          1$:     RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
                                ;;*****************************************************************
005000                          BANKNO: CLR     R0              ;INIT R0
010146                                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
010246                                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016701  163474                          MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
016702  163472                          MOV     BITPT+2,R2      ;...HI 64K.
006202                          1$:     ASR     R2              ;SHIFT POINTER...HI
006001                                  ROR     R1              ;...LO
103403                                  BCS     2$              ;BR WHEN POINTER FOUND.
105200                                  INCB    R0              ;COUNT BANKS.
100373                                  BPL     1$              ;BR IF NOT OVERFLOW.
000000                                  HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
016070                          2$:
012602                                  MOV     (SP)+,R2        ;;POP STACK INTO R2
012601                                  MOV     (SP)+,R1        ;;POP STACK INTO R1
000207                                  RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
                                ;;*****************************************************************
016076                          SETCON:
004467  176232                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010022                          2$:     MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
001375                                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
004767  176776                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
000207                                  RTS     PC              ;RETURN
```

```
                                ;;*****************************************************************
                                ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
                                ;;*****************************************************************
016116  106112          ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
016120  106112                  ROLB    (R2)            ;(R2)=177775 OR 000002
016122  106112                  ROLB    (R2)            ;(R2)=177773 OR 000004
016124  106112                  ROLB    (R2)            ;(R2)=177767 OR 000010
016126  106112                  ROLB    (R2)            ;(R2)=177757 OR 000020
016130  106112                  ROLB    (R2)            ;(R2)=177737 OR 000040
016132  106112                  ROLB    (R2)            ;(R2)=177677 OR 000100
016134  106112                  ROLB    (R2)            ;(R2)=177777 OR 000000
016136  106122                  ROLB    (R2)+           ;(R2)=177577 OR 000200
016140  106112                  ROLB    (R2)            ;(R2)=177377 OR 000400
016142  106112                  ROLB    (R2)            ;(R2)=176777 OR 001000
0161+4  106112                  ROLB    (R2)            ;(R2)=175777 OR 002000
016146  106112                  ROLB    (R2)            ;(R2)=173777 OR 004000
016150  106112                  ROLB    (R2)            ;(R2)=167777 OR 010000
016152  106112                  ROLB    (R2)            ;(R2)=157777 OR 020000
016154  106112                  ROLB    (R2)            ;(R2)=137777 OR 040000
016156  106112                  ROLB    (R2)            ;(R2)=077777 OR 100000
016160  106122                  ROLB    (R2)+           ;(R2)=177777 OR 000000
016162  000207                  RTS                     ;RETURN

                                ;;*****************************************************************
                                ;* SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
                                ;;*****************************************************************
016164  012704  000020  W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS

016170  010022          2$:     MOV     R0,(R2)+
016172  010022                  MOV     R0,(R2)+
016174  010022                  MOV     R0,(R2)+
016176  010022                  MOV     R0,(R2)+

016200  010322                  MOV     R3,(R2)+
016202  010322                  MOV     R3,(R2)+
016204  010322                  MOV     R3,(R2)+
016206  010322                  MOV     R3,(R2)+

016210  010022                  MOV     R0,(R2)+
016212  010022                  MOV     R0,(R2)+
016214  010022                  MOV     R0,(R2)+
016216  010022                  MOV     R0,(R2)+

016220  010322                  MOV     R3,(R2)+
016222  010322                  MOV     R3,(R2)+
016224  010322                  MOV     R3,(R2)+
016226  010322                  MOV     R3,(R2)+

016230  005304                  DEC     R4
016232  001356                  BNE     2$
016234  010046                  MOV     R0,     -(SP)   ;SAVE R0
016236  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
016240  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
016242  000207                  RTS     PC              ;RETURN
```

```
                                .SBTTL  RELOCATION SUBROUTINES.
                                ;;**********************************************************************
                                ;* ROUTINE TO RELOCATE PROGRAM CODE
                                ;;**********************************************************************
016244                          RELOC:
016244  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016246  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
016250  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
016252  012502                  4$:     MOV     (R5)+,  R2      ;GET FIRST LOCATION.
016254  012503                          MOV     (R5)+,  R3      ;GET FIRST LOCATION OF DESTINATION.
016256  012704  020000                  MOV     #20000, R4      ;SET UP BK COUNTER.
016262  012223                  1$:     MOV     (R2)+,  (R3)+   ;MOV THE DATA.
016264  005304                          DEC     R4              ;COUNT THE WORDS.
016266  001375                          BNE     1$              ;BR IF MORE.
016270  012704  020000                  MOV     #20000, R4      ;RESET THE COUNTER.
016274  024243                  2$:     CMP     -(R2),  -(R3)   ;CHECK THE DATA JUST MOVED.
016276  001417                          BEQ     3$              ;BR IF DATA OK.
016300  011267  162620                  MOV     (R2),   $GDDAT  ;GET SOURCE DATA.
016304  011367  162616                  MOV     (R3),   $BDDAT  ;GET DESTINATION DATA.
016310  010267  162604                  MOV     R2,     $GDADR  ;GET SOURCE ADDRESS.
016314  010367  162602                  MOV     R3,     $BDADR  ;GET DESTINATION ADDRESS.
016320  004767  003222                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
016324  000023                          .WORD   23              ;ERROR TYPE CODE.
016326  000000                          HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
016330  162705  000004                  SUB     #4,     R5      ;ADJUST RETURN POINTER.
016334  000746                          BR      4$              ;GO BACK AND TRY AGAIN.
016336  005304                  3$:     DEC     R4              ;COUNT WORDS.
016340  001355                          BNE     2$              ;BR IF MORE.
016342  004567  005052                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
016346  026446                          .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"PROGRAM RELOCATED TO "
016350  010346                          MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
016352  004767  006502                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
016356  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
016360  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
016362  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016364  000205                          RTS     R5              ;RETURN
                                ;;**********************************************************************
                                ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
                                ;;**********************************************************************
016366  022767  000003  162206  RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
016374  001401                          BEQ     1$              ;BR IF OK
016376  000000                          HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
016400                          1$:
016400  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016402  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016404  005767  162176                  TST     MMAVA
016410  001465                          BEQ     10$
016412  012737  007600  172346          MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
016420  005000                          CLR     R0              ;INIT BANK POINTER...LO 64K
016422  012701  100000                  MOV     #BIT15, R1      ;...HI 64K.
016426  162737  000200  172346  2$:     SUB     #200,   @#KIPAR3        ;BACK DOWN ONE BANK.
016434  006001                          ROR     R1              ;MOVE POINTER...HI 64K.
016436  006000                          ROR     R0              ;..LO 64K.
016440  103500                          BCS     90$
016442  030167  163060                  BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
3511  016446  001003                        BNE     3$              ;BR IF AVAILABLE
3512  016450  030067  163050                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
3513  016454  001764                        BEQ     2$              ;BR IF NO BANK FOUND.
3514  016456  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3515  016464  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3516  016466  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3517  016470  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3518  016476  006001                        ROR     R1              ;SHIFT POINTER.
3519  016500  006000                        ROR     R0              ;...LO 64K.
3520  016502  103457                        BCS     90$             ;BR IF OVERFLOW.
3521  016504  030167  163016          5$:   BIT     R1,     MEMMAP+2 ;       CHECK IF BANK EXISTS...HI 64K.
3522  016510  001003                        BNE     6$              ;BR IF BANK EXISTS.
3523  016512  030067  163006                BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
3524  016516  001764                        BEQ     4$              ;BR IF BANK DOESN'T EXIST.
3525  016520  052601                  6$:   BIS     (SP)+,  R1      ;GET SECOND BANK POINTER.
3526  016522  052600                        BIS     (SP)+,  R0      ;...LO 64K.
3527  016524  030067  162052                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
3528  016530  001044                        BNE     90$             ;ABORT IF DESTINATION OVERLAYS SOURCE.
3529  016532  004567  177506                JSR     R5,     RELOC   ;GO RELOCATE PROGRAM.
3530  016536  000000                        .WORD   0               ;SOURCE FIRST ADDRESS.
3531  016540  040000                        .WORD   40000           ;DESTINATION FIRST ADDRESS.
3532  016542  013737  172344  172340        MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3533  016550  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
3534                            ;*  PROGRAM SHOULD NOW BE EXICUTING OUT OF LAST TWO BANKS OF MEMORY.
3535  016556  010167  162022                MOV     R1,     PRGMAP+2 ;RESET PROGRAM MAP.
3536  016562  000473                        BR      30$             ;BR TO COMMON EXIT.
3537
3538  016564  012700  000400          10$:  MOV     #BIT8,  R0      ;SET BANK POINTER TO TOP OF MEM.
3539  016570  005001                        CLR     R1              ;SET ADDRESS POINTER TO TOP.
3540  016572  162701  020000          11$:  SUB     #20000, R1      ;BACK DOWN ONE BANK.
3541  016576  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
3542  016600  103420                        BCS     90$             ;BR IF OVERFLOW.
3543  016602  030067  162716                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3544  016606  001771                        BEQ     11$             ;BR IF NON-EXISTANT BANK.
3545  016610  162701  020000                SUB     #20000, R1      ;BACK DOWN TO NEXT BANK.
3546  016614  006200                        ASR     R0              ;MOV POINTER DOWN ONE BANK.
3547  016616  103411                        BCS     90$             ;BR IF OVERFLOW.
3548  016620  030067  162700                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3549  016624  001762                        BEQ     11$             ;BR TO START OVER IF NO LOWER BANK.
3550  016626  010046                        MOV     R0,     -(SP)   ;SAVE THE POINTER.
3551  016630  006300                        ASL     R0              ;RESET POINTER TO HI BANK.
3552  016632  052600                        BIS     (SP)+,  R0      ;SET BIT FOR LO BANK.
3553  016634  030067  161742                BIT     R0,     PRGMAP  ;CHECK FOR A PROGRAM CONFLICT.
3554  016640  001401                        BEQ     12$             ;BR IF NO CONFLICT.
3555  016642                          90$:
3556  016642  000000                        HALT                    ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
3557  016644  010167  000006          12$:  MOV     R1,     13$     ;SET DATA FOR RELOCATION SUBROUTINE.
3558  016650  004567  177370                JSR     R5,     RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3559  016654  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
3560  016656  000000                  13$:  .WORD   0               ;DESTINATION STARTING ADDRESS.
3561  016660  010167  161714                MOV     R1,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3562  016664  060107                        ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
3563                            ;*  PROGRAM NOW EXICUTING OUT OF TOP OF MEMORY.
3564  016666  060106                        ADD     R1,     SP      ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3565  016670  010167  161704                MOV     R1,     RELOCF  ;SET THE RELOCATION FACTOR.
3566  016674  060137  000004                ADD     R1,     @#ERRVEC ;ADJUST ERROR VECTOR.
```

```
3567  016700  060137  000024                  ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3568  016704  060137  000114                  ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3569  016710  026727  162224  177570          CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
3570  016716  001404                          BEQ     14$             ;BR IF HARDWARE SWITCH REGISTER.
3571  016720  060167  162214                  ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3572  016724  060167  162212                  ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3573  016730  062701  001614          14$:    ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
3574  016734  066721  161640          15$:    ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3575  016740  005721                  16$:    TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
3576  016742  001776                          BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
3577  016744  024127  177777                  CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
3578  016750  001371                          BNE     15$             ;BR IF MORE IN TABLE.
3579  016752  010067  161624          30$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
3580  016756  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3581  016760  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3582  016762  066716  161612                  ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
3583  016766  000207                          RTS     PC              ;RETURN
3584
3585                                  ;;*****************************************************************
3586                                  ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3587                                  ;;*****************************************************************
3588  016770  032767  000003  161604  RELO:   BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3589  016776  001401                          BEQ     1$              ;BR IF NO CONFLICT.
3590  017000  000000                          HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3591  017002  005767  161600          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
3592  017006  001417                          BEQ     10$             ;BR IF NO MEMMGMT.
3593  017010  005037  172344                  CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
3594  017014  012737  000200  172346          MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
3595  017022  004567  177216                  JSR     R5,     RELOC   ;GO MOVE BK INTO BANKS 0 AND 1.
3596  017026  000000                          .WORD   0               ;SOURCE STARTING ADDRESS.
3597  017030  040000                          .WORD   40000           ;DESTINATION STARTING ADDRESS.
3598  017032  005037  172340                  CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
3599  017036  012737  000200  172342          MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
3600                                  ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3601  017044  000444                          BR      30$             ;BR TO COMMON EXIT.
3602
3603  017046  016746  161526          10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
3604  017052  011667  000004                  MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
3605  017056  004567  177162                  JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3606  017062  000000                  20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
3607  017064  000000                          .WORD   0               ;DESTINATION STARTING ADDRESS.
3608  017066  161607                          SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
3609                                  ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3610  017070  161606                          SUB     (SP),   SP      ;RESET THE STACK POINTER.
3611  017072  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3612  017074  012700  001614                  MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3613  017100  166620  000002          21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
3614  017104  005720                  22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
3615  017106  001776                          BEQ     22$             ;BR OVER TERMINATORS.
3616  017110  024027  177777                  CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
3617  017114  001371                          BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
3618  017116  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
3619  017120  161637  000004                  SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
3620  017124  161637  000024                  SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3621  017130  161637  000114                  SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3622  017134  026727  162000  177570          CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
```

```
3623  017142  001404                          BEQ     23$             ;BR IF HARDWARE SWITCH REGISTER.
3624  017144  161667  161770                  SUB     (SP), SWR       ;ADJUST SOFTWARE SWITCH REGISTER.
3625  017150  161667  161766                  SUB     (SP), DISPLAY   ;ADJUST SOFTWARE DISPLAY REGISTER.
3626  017154  162616                  23$:    SUB     (SP)+, (SP)     ;ADJUST RETURN ADDRESS.
3627  017156  005067  161416          30$:    CLR     RELOCF          ;RESET RELOCATION FACTOR.
3628  017162  012767  000003  161412          MOV     #3, PRGMAP      ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3629  017170  005067  161410                  CLR     PRGMAP+2        ;...HI 64K.
3630  017174  000207                          RTS     PC              ;RETURN.
3631
3632                                  ;;*****************************************************************
3633                                  ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3634                                  ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
3635                                  ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
3636                                  ;;*****************************************************************
3637  017176  016700  162316          RESLDR: MOV     LMAD, R0        ;CHECK IF THE LOADERS WERE SAVED.
3638  017202  001...1                         BNE     1$              ;BR IF LOADER AREA WAS SAVED.
3639  017204  000000                          HALT                    ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3640  017206  005767  161374          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
3641  017212  001402                          BEQ     2$              ;SKIP IF NO MEM MGMT.
3642  017214  005037  177572                  CLR     @#SR0           ;DISABLE MEM MGMT.
3643  017220  012701  040000          2$:     MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
3644  017224  012702  002734                  MOV     #1500., R2      ;GET COUNTER.
3645  017230  014140                  3$:     MOV     -(R1), -(R0)    ;MOVE THE LOADER AREA.
3646  017232  005302                          DEC     R2              ;COUNT HOW LONG THE AREA IS.
3647  017234  001375                          BNE     3$              ;BR IF NOT MORE TO MOVE.
3648  017236  005067  162256                  CLR     LMAD            ;CLEAR MONITOR SAVED FLAG
3649  017242  005767  161340                  TST     MMAVA           ;CHECK FOR MEM MGMT.
3650  017246  001402                          BEQ     4$              ;BR IF NO MEM MGMT.
3651  017250  005237  177572                  INC     @#SR0           ;ENABLE MEM MGMT.
3652  017254  000207                  4$:     RTS     PC              ;RETURN.
3653
3654                                  ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
3655  017256  005767  162236          SAVLDR: TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3656  017262  001024                          BNE     4$              ;BRANCH IF ALREADY SAVED
3657  017264  012700  040000                  MOV     #40000, R0      ;GET END OF 8K
3658  017270  010001                          MOV     R0, R1          ;GET END OF 8K
3659  017272  012737  017304  000004          MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
3660  017300  011020                  1$:     MOV     (R0), (R0)+     ;SEARCH FOR END OF MEMORY
3661  017302  000776                          BR      1$              ;KEEP SEARCHING
3662  017304  022626                  2$:     CMP     (SP)+, (SP)+    ;RESTORE STACK POINTER
3663  017306  012737  025022  000004          MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
3664  017314  010046                          MOV     R0, -(SP)       ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3665  017316  012702  002734                  MOV     #1500., R2      ;SET UP WORD COUNTER
3666  017322  014041                  3$:     MOV     -(R0), -(R1)    ;SAVE THE LOADERS
3667  017324  005302                          DEC     R2              ;COUNT THE WORDS
3668  017326  001375                          BNE     3$              ;BRANCH IF MORE WORDS
3669  017330  012667  162164                  MOV     (SP)+, LMAD     ;SAVE LAST MEMORY ADDRESS
3670  017334  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.
                                ;;*****************************************************************
017336  011667  161560        PESRV:  MOV     (SP),   $BOADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
017342  004567  004052                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
017346  026405                        .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"UNEXPECTED MEMORY PARITY TRAP."
017350  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
017352  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
017354  016703  162242                MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
017360  005733                1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017362  100415                        BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
017364  005713                        TST     (R3)            ;CHECK FOR TABLE TERMINATOR
017366  001374                        BNE     1$              ;BR IF MORE REGISTERS.
017370  004767  002152                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                ;***ERROR***    NO REGISTER INDICATED ERROR
017374  000024                        .WORD   24              ;ERROR TYPE CODE.
017376  000417                        BR      4$              ;EXIT
017400  005713                2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
017402  001415                        BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
017404  005733                        TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017406  100374                        BPL     2$              ;BR IF NO ERROR FLAG.
017410  004567  004004                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
017414  026476                        .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"MORE THAN ONE ERROR FOUND."
017416                          3$:
017416  004767  000610        64$:    JSR     PC,     $PRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
017422  004767  002120                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
017426  000025                        .WORD   25              ;ERROR TYPE CODE.
017430  004767  000216                JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
017434  000761                        BR      2$              ;GO LOOK FOR MORE ERRORS.
017436                          4$:
017436  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
017440  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
017442  000002                        RTI                     ;RETURN.

                                ;;*****************************************************************
                                ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
                                ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                ;;*****************************************************************
017444  005767  162620        MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
017450  001434                        BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
017452  032777  000100  161460        BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
017460  001030                        BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
017462  005767  161112                TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
017466  001410                        BEQ     SETAE           ;BR IF PROG IN BANK 0.
017470  032777  000040  161442        BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
017476  001004                        BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
017500  026727  162056  001000        CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
017506  103415                        BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
017510  016737  162114  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
017516  005037  000116                CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
017522  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
```

```
3727  017524  016703  162072          MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
3728  017530  052733  000001  MAMF1:  BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
3729  017534  005713                  TST     (R3)            ;CHECK FOR END OF TABLE.
3730  017536  001374                  BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
3731  017540  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3732  017542  000207  MAMF2:  RTS     PC                      ;RETURN.
3733
3734                   ;;*****************************************************************
3735                   ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3736                   ;;*****************************************************************
3737  017544  005767  162520  CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3738  017550  001437                  BEQ     4$              ;BR IF NO PARITY REGISTERS.
3739  017552  032777  000100  161360  BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3740  017560  001033                  BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
3741  017562  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3742  017566  016703  162032          MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
3743  017572  005733          1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3744  017574  100023                  BPL     3$              ;BR IF NO ERROR
3745  017576  032773  000001  177776  BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3746  017602  001010                  BNE     2$              ;BR IF NO ACTION ENABLE.
3747  017604  004767  000422  64$:    JSR     PC,     $PRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3748  017610  004767  001732          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3749  017614  000026                  .WORD   26              ;ERROR TYPE CODE.
3750  017616  000411                  BR      3$
3751  017620  004767  000026          JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3752  017624                  2$:
3753  017624  004767  000402  65$:    JSR     PC,     $PRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3754  017630  004767  001712          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3755  017634  000027                  .WORD   27              ;ERROR TYPE CODE.
3756  017636  004767  000010          JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3757  017642  005713          3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3758  017644  001351                  BNE     1$              ;BR IF MORE.
3759  017646  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
3760  017650  000207  4$:     RTS     PC                      ;RETURN.
3761
3762                   ;;*****************************************************************
3763                   ;*  THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY,
3764                   ;*  TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
3765                   ;*  LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
3766                   ;;*****************************************************************
3767  017652          PSCAN:
3768  017652  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3769  017654  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3770  017656  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3771  017660  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3772  017662  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3773  017664  013746  000114          MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
3774  017670  013746  000116          MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
3775  017674  004567  003520          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3776  017700  026542                  .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
3777                                                          ;"SCANNING MEMORY FOR BAD PARITY."
3778  017702  012700  000001          MOV     #BIT0,  R0      ;SET BIT POINTER TO FIRST BANK.
3779  017706  005001                  CLR     R1              ;CLR HI 64K POINTER.
3780  017710  005002                  CLR     R2              ;INIT ADDRESS POINTER.
3781  017712  005004                  CLR     R4              ;INIT ERROR DETECTED FLAG.
3782  017714  004767  000256          JSR     PC,     CLRPAR  ;CLEAR THE PARITY REGISTERS.
```

```
3783  017720  012737  000116  000114          MOV     #116,   @#114   ;HALT IF ANOTHER PARITY TRAP.
3784  017726  005037  000116                  CLR     @#116
3785  017732  005767  160650                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
3786  017736  001406                          BEQ     1$              ;BR IF NO MEM MGMT.
3787  017740  013746  172344                  MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
3788  017744  005037  172344                  CLR     @#KIPAR2        ;INIT MEM MGMT TO POINT TO BANK 0.
3789  017750  012702  040000                  MOV     #40000, R2      ;SET ADR POINTER TO PAR2.
3790  017754  030067  161544          1$:     BIT     R0,     MEMMAP  ;CHECK IF THIS BANK OF MEM EXISTS.
3791  017760  001003                          BNE     2$              ;BR IF THIS BANK EXISTS.
3792  017762  030167  161540                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K MAP.
3793  017766  001442                          BEQ     10$             ;BR IF THIS BANK DOESN'T EXIST.
3794  017770                          2$:
3795  017770  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3796  017772  111201                  3$:     MOVB    (R2),   R1      ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
3797  017774  016703  161622                  MOV     .MPRX,  R3      ;SET UP POINTER TO PARITY REGISTERS.
3798  020000  005733                  4$:     TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
3799  020002  100024                          BPL     6$              ;BR IF NO ERROR FLAG.
3800  020004  005704                          TST     R4              ;CHECK IF FIRST ERROR, THIS SCAN.
3801  020006  001003                          BNE     5$              ;BR IF MORE THAN ONE ERROR FOUND.
3802  020010  005367  161076                  DEC     $ERTTL          ;ADJUST ERROR COUNT.
3803  020014  005204                          INC     R4              ;SET FLAG TO INDICATE ERROR FOUND.
3804  020016                          5$:
3805  020016  004767  000210          64$:    JSR     PC,     $ERTYP  ;SET UP VALUES FOR ERROR PRINTING.
3806  020022  004767  001520                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3807  020026  000030                          .WORD   30              ;ERROR TYPE CODE.
3808  020030  111212                          MOVB    (R2),   (R2)    ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
3809  020032  005053                          CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
3810  020034  105712                          TSTB    (R2)            ;CHECK IF THE PARITY ERROR WAS CLEARED.
3811  020036  005733                          TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
3812  020040  100005                          BPL     6$              ;BR IF IT IS OK.
3813  020042  004567  003352                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3814  020046  026604                          .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
3815                                                                  ;"PARITY ERROR WILL NOT CLEAR."
3816  020050  005073  177776                  CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
3817  020054  005713                  6$:     TST     (R3)            ;CHECK FOR THE END OF REG ADR TABLE.
3818  020056  001350                          BNE     4$              ;BR IF MORE PARITY REGISTERS.
3819  020060  005202                          INC     R2              ;GO TO NEXT MEMORY ADDRESS.
3820  020062  032702  017777                  BIT     #MASK4K R2      ;CHECK FOR END OF 4K BANK.
3821  020066  001341                          BNE     3$              ;BR IF MORE MEMORY THIS BANK.
3822  020070  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3823  020072  000402                          BR      11$             ;BR TO CHECK FOR NEXT BANK.
3824  020074  062702  020000          10$:    ADD     #20000, R2      ;SKIP BANKS THAT AREN'T THERE.
3825  020100  005767  160502          11$:    TST     MMAVA           ;CHECK FOR MEM MGMT.
3826  020104  001413                          BEQ     12$             ;BR IF NO MEM MGMT.
3827  020106  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
3828  020114  012702  040000                  MOV     #40000, R2      ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
3829  020120  006300                          ASL     R0              ;UPDATE BANK POINTER.
3830  020122  006101                          ROL     R1              ;...HI 64K.
3831  020124  100313                          BPL     1$              ;BR IF MORE BANKS.
3832  020126  012637  172344                  MOV     (SP)+,@#KIPAR2          ;;POP STACK INTO @#KIPAR2
3833  020132  000402                          BR      20$             ;GO CHECK IF ANY ERRORS FOUND.
3834  020134  106300                  12$:    ASLB    R0              ;UPDATE POINTER TO NEXT BANK.
3835  020136  100306                          BPL     1$              ;BR IF MORE BANKS.
3836  020140  005704                  20$:    TST     R4              ;CHECK IF ANY PARITY ERRORS DETECTED.
3837  020142  001003                          BNE     21$             ;BR IF ERRORS DETECTED.
3838  020144  004567  003250                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
                        020150  025614                  .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
                        020152                  21$:
                        020152  012637  000116          MOV     (SP)+,@#116     ;;POP STACK INTO @#116
                        020156  012637  000114          MOV     (SP)+,@#114     ;;POP STACK INTO @#114
                        020162  012604                  MOV     (SP)+,R4        ;;POP STACK INTO R4
                        020164  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
                        020166  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
                        020170  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
                        020172  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
                        020174  000207                  RTS     PC              ;RETURN.

                                                ;;*****************************************************************
                                                ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
                                                ;;*****************************************************************
                        020176                  CLRPAR:
                        020176  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
                        020200  016703  161416          MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
                        020204  005713          1$:     TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
                        020206  001402                  BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
                        020210  005033                  CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
                        020212  000774                  BR      1$              ;BR FOR MORE
                        020214                  2$:
                        020214  012603                  MOV     (SP)+,R3        ;;PC STACK INTO R3
                        020216  000207                  RTS     PC              ;RET.N.

                                                .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
                                                ;;*****************************************************************
                                                ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
                                                ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
                                                ;;*****************************************************************
                        020220  010267  160674  SPRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
                        020224  005067  160674          CLR     $GDDAT          ;SHOULD BE DATA IS "0".
                        020230  000430                  BR      SPRNTB

                        020232  014367  160722  SPRNTQ: MOV     -(R3),  $TMP0   ;GET THE PARITY REGISTER ADDRESS.
                        020236  013367  160720          MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
                        020242  000402                  BR      SPRNTO

                        020244  011367  160710  SPRNTP: MOV     (R3),   $TMP0   ;GET THE PARITY REGISTER ADDRESS.
                        020250  010267  160644  SPRNTO: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        020254  000414                  BR      SPRNTA          ;BR TO COMMON SECTION.

                        020256  010267  160636  SPRNT1: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        020262  005367  160632          DEC     $GDADR          ;ADJUST IT FOR PRINTOUT.
                        020266  000407                  BR      SPRNTA          ;BR TO COMMON SECTION.

                        020270  010367  160664  SPRNT3: MOV     R3,     $TMP0   ;GET THE DATA IN R3.
                        020274  010267  160620  SPRNT2: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
                        020300  162767  000002  160612  SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
                        020306  010067  160612  SPRNTA: MOV     R0,     $GDDAT  ;GET WHAT THE DATA SHOULD BE
                        020312  010167  160610  SPRNTB: MOV     R1,     $BDDAT  ;GET WHAT THE DATA WAS
                        020316  000207                  RTS     PC              ;RETURN TO ENTER ERROR ROUTINES

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

```
3895
3896  020320  005710                ;;*****************************************************************
3897  020322  001007        TYPMAP: TST     (R0)            ;CHECK IF ANY MEMORY IN MAP...LO 64K.
3898  020324  005760  000002        BNE     1$              ;BR IF MEMORY IN MAP.
3899  020330  001004                TST     2(R0)           ;...HI 64K.
3900  020332  004567  003062        BNE     1$              ;BR IF MEMORY IN MAP.
3901  020336  026172                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3902                                .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
3903  020340  000475                                        ;"NO MEMORY FOUND."
3904  020342                        BR      6$              ;EXIT
3905  020342  010146        1$:     MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3906  020344  010246                MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3907  020346  010346                MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3908  020350  010446                MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3909  020352  012701  000001        MOV     #BIT0,  R1      ;SET UP BANK POINTER...LO 64K.
3910  020356  005002                CLR     R2              ;...HI 64K.
3911  020360  012703  177777        MOV     #-1,    R3      ;SET UP ADDRESS POINTER TO -1.
3912  020364  010304                MOV     R3,     R4      ;HI BITS OF ADDRESS AS WELL.
3913  020366  030110        2$:     BIT     R1,     (R0)    ;CHECK THE MAP FOR THIS BANK.
3914  020370  001014                BNE     3$              ;BR IF THIS BANK PRESENT.
3915  020372  030260  000002        BIT     R2,     2(R0)   ;CHECK HI 64K MAP.
3916  020376  001011                BNE     3$              ;BR IF THIS BANK PRESENT.
3917  020400  105703                TSTB    R3              ;CHECK FOR PREVIOUS PRINTOUT.
3918  020402  001042                BNE     5$              ;BR IF ALREADY TYPED "TO"
3919  020404  162703  000001        SUB     #1,     R3      ;BACK UP TO LAST ADR OF PREVIOUS BANK.
3920  020410  005604                SBC     R4              ;...HI ADDRESS BITS.
3921  020412  004567  003002        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3922  020416  025425                .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
3923  020420  000410                BR      4$              ;GO TO TYPE THE ADDRESS.
3924  020422  105703        3$:     TSTB    R3              ;CHECK FOR PREVIOUS TYPEOUT.
3925  020424  001431                BEQ     5$              ;BR IF ALREADY TYPE "FROM".
3926  020426  062703  000001        ADD     #1,     R3      ;POINT TO FIRST ADDRESS OF THIS BANK.
3927  020432  005504                ADC     R4              ;...HI BITS OF ADDRESS.
3928  020434  004567  002760        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3929  020440  025415                .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
3930  020442                4$:
3931  020442  010346                MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3932  020444  010446                MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3933  020446  006303                ASL     R3              ;BIT 15 INTO C-BIT
3934  020450  006104                ROL     R4              ;BIT 15 INTO R4.
3935  020452  006003                ROR     R3              ;RESTORE BITS 14-0.
3936  020454  010446                MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
3937                                                        ;;TYPE ADDRESS BITS 21-15
3938                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3939                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3940  020456  013746  177776        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
3941  020462  004767  004104        JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3942  020466     003                .BYTE   3               ;;TYPE 3 DIGIT(S)
3943  020467     000                .BYTE   0               ;;SUPPRESS LEADING ZEROS
3944  020470  010346                MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
3945                                                        ;;TYPE ADDRESS BITS 14-0
3946                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3947                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3948  020472  013746  177776        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
3949  020476  004767  004070        JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
3950  020502     005                .BYTE   5               ;;TYPE 5 DIGIT(S)
```

```
3951  020503  001                        .BYTE   1               ;;TYPE LEADING ZEROS
3952  020504  012604                     MOV     (SP)+,R4        ;;POP STACK INTO R4
3953  020506  012603                     MOV     (SP)+,R3        ;;POP STACK INTO R3
3954  020510  062703  020000     5$:     ADD     #20000, R3      ;UPDATE TO NEXT BANK.
3955  020514  005504                     ADC     R4              ;...HI ADDRESS BITS.
3956  020516  006301                     ASL     R1              ;SHIFT POINTER...LO 64K.
3957  020520  006102                     ROL     R2              ;...HI 64K.
3958  020522  103321                     BCC     2$              ;BR IF MORE BANKS.
3959  020524  012604                     MOV     (SP)+,R4        ;;POP STACK INTO R4
3960  020526  012603                     MOV     (SP)+,R3        ;;POP STACK INTO R3
3961  020530  012602                     MOV     (SP)+,R2        ;;POP STACK INTO R2
3962  020532  012601                     MOV     (SP)+,R1        ;;POP STACK INTO R1
3963  020534  000207             6$:     RTS     PC              ;RETURN.
3964
3965                              .SBTTL  SCOPE HANDLER ROUTINE
3966
3967                              ;;**************************************************************
3968                              ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3969                              ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3970                              ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3971                              ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3972                              ;*SW14=1        LOOP ON TEST
3973                              ;*SW11=1        INHIBIT ITERATIONS
3974                              ;*SW09=1        LOOP ON ERROR
3975                              ;*SW08=1        LOOP ON TEST IN SWR<4:0>
3976                              ;*CALL
3977                              ;*      SCOPE           ;;SCOPE=IOT
3978
3979  020536                      $SCOPE:
3980                              ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
3981                              ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3982  020536  013746  177776              MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
3983  020542  004767  001524              JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
3984  020546  012504                      MOV     (R5)+,  R4      ;SAVE MINIMUM BLOCK MASK NEXT TEST.
3985  020550  010516                      MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
3986  020552  032777  040000  160360 1$:  BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
3987  020560  001117                      BNE     $OVER           ;;YES IF SW14=1
3988                              ;#####START OF CODE FOR THE XOR TESTER#####
3989  020562  000416              $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
3990                                                              ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
3991  020564  013746  000004              MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
3992  020570  012737  020610  000004      MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
3993  020576  005737  177060              TST     @#177060        ;;TIME OUT ON XOR?
3994  020602  012637  000004              MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3995  020606  000466                      BR      $SVLAD          ;;GO TO THE NEXT TEST
3996  020610  022526              5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
3997  020612  012637  000004              MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
3998  020616  000426                      BR      7$              ;;LOOP ON THE PRESENT TEST
3999  020620                      6$:;#####END OF CODE FOR THE XOR TESTER#####
4000  020620  032777  000400  160312      BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
4001  020626  001407                      BEQ     2$              ;;BR IF NO
4002  020630  017746  160304              MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
4003  020634  042716  000340              BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
4004  020640  122667  160236              CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
4005  020644  001465                      BEQ     $OVER           ;;BR IF YES
4006  020646  105767  160231      2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
```

```
4007  020652  001421                        BEQ     3$              ;;BR IF NO
4008  020654  126767  160235  160221        CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
4009  020662  101015                        BHI     3$              ;;BR IF NO
4010  020664  032777  001000  160246        BIT     #BIT09,@$WR     ;;LOOP ON ERROR?
4011  020672  001404                        BEQ     4$              ;;BR IF NO
4012  020674  016767  160210  160204  7$:   MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
4013  020702  000446                        BR      $OVER
4014  020704  105067  160173          4$:   CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
4015  020710  005067  160254                CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
4016  020714  000415                        BR      1$              ;;ESCAPE TO THE NEXT TEST
4017  020716  032777  004000  160214  3$:   BIT     #BIT11,@$WR     ;;INHIBIT ITERATIONS?
4018  020724  001011                        BNE     1$              ;;BR IF YES
4019  020726  005767  160260                TST     $PASS           ;;IF FIRST PASS OF PROGRAM
4020  020732  001406                        BEQ     1$              ;;      INHIBIT ITERATIONS
4021  020734  005267  160144                INC     $ICNT           ;;INCREMENT ITERATION COUNT
4022  020740  026767  160224  160136        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
4023  020746  002024                        BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
4024  020750  012767  000001  160126  1$:   MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
4025  020756  016767  000552  160204        MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
4026  020764  105267  160112          $SVLAD: INCB  $TSTNM          ;;COUNT TEST NUMBERS
4027  020770  116767  160106  160212        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
4028  020776  011667  160104                MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
4029  021002  011667  160102                MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
4030  021006  005067  160160                CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
4031  021012  112767  000001  160075        MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4032  021020  016777  160056  160114  $OVER: MOV    $TSTNM,@$DISPLAY ;;DISPLAY TEST NUMBER
4033  021026  016716  160054                MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
4034  021032  020516                  INSERT: CMP   R5,     (SP)    ;CHECK FOR LOOP ON TEST.
4035  021034  001402                        BEQ     1$              ;BR IF START NEXT TEST.
4036  021036  000167  000470                JMP     ENDINS          ;JMP IF LOOP ON LAST TEST.
4037  021042  012767  037777  160536  1$:   MOV     #37777, BLKMSK  ;SET 8K BOUNDRY MASK.
4038  021050  005767  160136                TST     $PASS           ;CHECK FOR PASS 0.
4039  021054  001404                        BEQ     2$              ;BR IF PASS 0
4040  021056  126727  160020  000021        CMPB    $TSTNM, #21     ;CHECK IF IN SECTION 3.
4041  021064  103002                        BHIS    3$              ;BR IF IN SECTION 3.
4042  021066  006267  160514          2$:   ASR     BLKMSK          ;RESET BOUNDRY TO 4K.
4043  021072  016767  160464  160464  3$:   MOV     FSTADR, TMPFAD  ;GET FIRST ADDRESS.
4044  021100  005767  157474                TST     RELOCF          ;CHECK IF PRG RELOCATED.
4045  021104  001430                        BEQ     4$              ;BR IF NOT RELOCATED.
4046  021106  032777  000040  160024        BIT     #SW05,  @$WR    ;CHECK IF LOC 0-776 TO BE PROTECTED.
4047  021114  001424                        BEQ     4$              ;BR IF SW NOT SET.
4048  021116  026727  160442  001000        CMP     TMPFAD, #1000   ;CHECK IF NOT BEING TESTED.
4049  021124  103020                        BHIS    4$              ;BR IF ALREADY PROTECTED.
4050  021126  012767  001000  160430        MOV     #1000,  TMPFAD  ;RESET FIRST ADDRESS.
4051  021134  052767  000001  160426        BIS     #BIT0,  FADMAP  ;SET FLAG IN FIRST BANK.
4052  021142  026727  160426  001000        CMP     LSTADR, #1000   ;CHECK IF GONE PAST LAST ADR.
4053  021150  101006                        BHI     4$              ;BR IF ENOUGH MEMORY.
4054  021152  004567  002242                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4055  021156  026643                        .WORD   NOMTST          ;ADDRESS OF MESSAGE TO BE TYPED
4056                                                                ;"NO MEMORY TESTED"
4057  021160  016716  160452                MOV     .TST32, (SP)    ;ADJUST RETURN ADR FOR ABORT.
4058  021164  000207                        RTS     PC              ;ABORT.
4059  021166  016767  160402  160402  4$:   MOV     LSTADR, TMPLAD  ;GET LAST ADDRESS.
4060  021174  016767  160334  160326        MOV     SAVTST, TSTMAP  ;GET TEST MAP, LO 64K.
4061  021202  016767  160330  160322        MOV     SAVTST+2,TSTMAP+2 ;....HI 64K.
4062  021210  046767  157366  160312        BIC     PRGMAP, TSTMAP  ;DON'T TEST OVER THE PROGRAM.
```

```
4063  021216  046767  157362  160306          BIC     PRGMAP+2,TSTMAP+2
4064  021224  005767  157762                  TST     $PASS           ;CHECK FOR FIRST PASS
4065  021230  001011                          BNE     10$             ;BR IF NOT FIRST PASS.
4066  021232  032767  000003  160270          BIT     #3,     TSTMAP  ;CHECK IF FIRST TWO BANKS AVAILABLE.
4067  021240  001405                          BEQ     10$             ;NOT TESTING FIRST 2 BANKS.
4068  021242  042767  177774  160260          BIC     #177774,TSTMAP  ;CLR ALL BUT FIRST 2 BANKS.
4069  021250  005067  160256                  CLR     TSTMAP+2
4070  021254  005704                  10$:    TST     R4              ;CHECK FOR A MINIMUM BLOCK SIZE.
4071  021256  001503                          BEQ     20$             ;BR IF NO MIN BLOCK SIZE.
4072  021260  030467  160300                  BIT     R4,     TMPFAD  ;CHECK IF FIRST ADR ON BLOCK BOUNDRY.
4073  021264  001416                          BEQ     11$             ;BR IF FIRST ADR ON BLOCK BOUNDRY.
4074  021266  050467  160272                  BIS     R4,     TMPFAD  ;ADJUST FIRST ADR TO END OF BLOCK.
4075  021272  005267  160266                  INC     TMPFAD          ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4076  021276  032767  017777  160260          BIT     #MASK4K,TMPFAD  ;CHECK IF FIRST ADR REACHED 4K BOUNDRY.
4077  021304  001006                          BNE     11$             ;BR IF NOT ON 4K BOUNDRY.
4078  021306  046767  160256  160214          BIC     FADMAP, TSTMAP  ;DON'T TEST FIRST BANK.
4079  021314  046767  160252  160210          BIC     FADMAP+2,TSTMAP+2
4080  021322  030467  160250          11$:    BIT     R4,     TMPLAD  ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
4081  021326  001414                          BEQ     12$             ;BR IF ON BLOCK BOUNDRY.
4082  021330  040467  160242                  BIC     R4,     TMPLAD  ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
4083  021334  032767  017777  160234          BIT     #MASK4K,TMPLAD  ;CHECK IF ADJUSTED TO 4K BOUNDRY.
4084  021342  001006                          BNE     12$             ;BR IF NOT ON 4K BOUNDRY.
4085  021344  046767  160232  160156          BIC     LADMAP, TSTMAP  ;SKIP TESTING LAST BANK.
4086  021352  046767  160226  160152          BIC     LADMAP+2,TSTMAP+2
4087  021360  036767  160204  160214  12$:    BIT     FADMAP, LADMAP  ;CHECK IF FIRST AND LAST IN SAME BANK.
4088  021366  001004                          BNE     13$             ;BR IF IN SAME BANK.
4089  021370  036767  160176  160206          BIT     FADMAP+2,LADMAP+2 ;...UPPER 64K.
4090  021376  001404                          BEQ     14$             ;BR IF FIRST AND LAST NOT SAME BANK.
4091  021400  026767  160172  160156  13$:    CMP     TMPLAD, TMPFAD  ;CHECK IF ANY MEMORY LEFT.
4092  021406  101406                          BLOS    15$             ;BR IF NO MEMORY TO TEST.
4093  021410  005767  160114          14$:    TST     TSTMAP          ;CHECK IF ANY BANKS LEFT TO TEST!!
4094  021414  001017                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
4095  021416  005767  160110                  TST     TSTMAP+2        ;CHECK FOR ANY BANKS.
4096  021422  001014                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
4097  021424                          15$:
4098  021424  004567  001770                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4099  021430  026667                          .WORD   SKPMES          ;ADDRESS OF MESSAGE TO BE TYPED
4100                                                                  ;"SKIPPING TEST #"
4101  021432  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
4102  021434  116716  157442                  MOVB    $TSTNM, (SP)    ;PUT THE DATA ON THE STACK.
4103                          ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4104                          ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4105  021440  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4106  021444  004767  003122                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
4107  021450     003                          .BYTE   3               ;TYPE 3 DIGITS.
4108  021451     001                          .BYTE   1               ;TYPE LEADING ZEROS.
4109  021452  000427                          BR      ENDINS          ;RETURN TO SKIP TEST.
4110  021454  062716  000004          16$:    ADD     #4,     (SP)    ;SKIP THE SKIP ON RETURN.
4111  021460  062767  000004  157420          ADD     #4,     $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
4112  021466  012767  017777  160072  20$:    MOV     #MASK4K,FADMSK  ;GET 4K MASK.
4113  021474  016705  160064                  MOV     TMPFAD, R5      ;GET FIRST ADR.
4114  021500  040567  160062          21$:    BIC     R5,     FADMSK  ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4115  021504  006305                          ASL     R5              ;MOVE LOWEST BIT UP ONE.
4116  021506  001374                          BNE     21$             ;LOOP UNTIL OVERFLOW.
4117  021510  012767  017777  160062          MOV     #MASK4K,LADMSK  ;SET MASK BITS
4118  021516  016705  160054                  MOV     TMPLAD, R5      ;GET LAST ADR.
```

```
4119  021522  040567  160052          22$:    BIC     R5,     LADMSK  ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4120  021526  006305                          ASL     R5              ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4121  021530  001374                          BNE     22$             ;LOOP UNTIL OVERFLOW.
4122  021532  000207                  ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
4123  021534  000004                  $MXCNT: 4                       ;;MAX. NUMBER OF ITERATIONS
4124                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4125                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4126  021536  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4127  021542  004767  000524                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
4128                                  .SBTTL  ERROR HANDLER ROUTINE
4129
4130                                  ;;**************************************************************
4131                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4132                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4133                                  ;*AND GO TO $ERRTYP ON ERROR
4134                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4135                                  ;*SW15=1         HALT ON ERROR
4136                                  ;*SW13=1         INHIBIT ERROR TYPEOUTS
4137                                  ;*SW10=1         BELL ON ERROR
4138                                  ;*SW09=1         LOOP ON ERROR
4139                                  ;*CALL
4140                                  ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4141
4142  021546                          $ERROR:
4143                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4144                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4145  021546  013746  177776              MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4146  021552  004767  000514              JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
4147  021556  062716  000002              ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
4148  021562  105267  157315      7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
4149  021566  001775                      BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
4150  021570  016777  157306  157344      MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
4151  021576  032777  002000  157334      BIT     #BIT10,@SWR     ;;BELL ON ERROR?
4152  021604  001403                      BEQ     1$              ;;NO - SKIP
4153  021606  004567  001606              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4154  021612  001174                      .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
4155  021614  005267  157272      1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
4156  021620  011667  157272              MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
4157  021624  162767  000002  157264      SUB     #2,$ERRPC
4158  021632  117767  157260  157254      MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
4159  021640  032777  020000  157272      BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
4160  021646  001005                      BNE     20$             ;;SKIP TYPEOUTS
4161  021650  004767  000116              JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
4162  021654  004567  001540              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4163  021660  001201                      .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4164  021662                      20$:
4165  021662  122767  000001  157334      CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4166  021670  001007                      BNE     2$              ;;NO,SKIP APT ERROR REPORT
4167  021672  116767  157216  000004      MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
4168  021700  004767  002044              JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
4169  021704     000              21$:    .BYTE   0
4170  021705     000                      .BYTE   0
4171  021706  000777              22$:    BR      22$             ;;APT ERROR LOOP
4172  021710  005777  157224      2$:     TST     @SWR            ;;HALT ON ERROR
4173  021714  100005                      BPL     3$              ;;SKIP IF CONTINUE
4174  021716  000000                      HALT                    ;;HALT ON ERROR!
```

```
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021720  013746  177776                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021724  004767  000342                        JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
021730  032777  001000  157202        3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
021736  001402                              BEQ     4$              ;;BR IF NO
021740  016716  157144                        MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
021744  005767  157222                4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
021750  001402                              BEQ     5$              ;;BR IF NONE
021752  016716  157214                        MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
021756                                  5$:
021756  022737  014136  000042                CMP     #SENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
021764  001001                              BNE     6$              ;;BRANCH IF NO
021766  000000                              HALT                    ;;YES
021770                                  6$:
021770  000207                              RTS     PC
                                        ;;**************************************************************
                                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
021772                                  $ERRTYP:
021772  004567  001422                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021776  001201                              .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022000  010046                              MOV     R0,-(SP)                ;SAVE R0
022002  005000                              CLR     R0                      ;PICKUP THE ITEM INDEX
022004  156700  157104                        BISB    $ITEMB,R0
022010  001007                              BNE     1$                      ;IF ITEM NUMBER IS ZERO, JUST
                                                                            ;TYPE THE PC OF THE ERROR
022012  016746  157100                        MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                            ;;ERROR ADDRESS
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022016  013746  177776                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022022  004767  002570                        JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022026  000513                              BR      10$                     ;GET OUT
022030  016767  157062  157456        1$:     MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
022036  166767  156536  157450                SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
022044  005300                              DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
022046  006300                              ASL     R0                      ;       WORK FOR THE ERROR TABLE
022050  006300                              ASL     R0
022052  006300                              ASL     R0
022054  066700  157552                        ADD     .ERRTB, R0      ;FORM TABLE POINTER
022060  012067  000006                        MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
022064  001406                              BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
022066  004567  001326                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022072  000000                          2$:     .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
022074  004567  001320                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022100  001201                              .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022102  012067  000006                  3$:     MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
022106  001406                              BEQ     5$                      ;SKIP TYPEOUT IF 0
022110  004567  001304                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022114  000000                          4$:     .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
```

```
4231  022116  004567  001276                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4232  022122  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4233  022124  010146                    5$:     MOV     R1,-(SP)                ;SAVE R1
4234  022126  012001                            MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
4235  022130  001451                            BEQ     9$                      ;BR IF NO DATA TO BE TYPED
4236  022132  066701  156442                    ADD     RELOCF, R1      ;ADJUST POINTER
4237  022136  012000                            MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
4238  022140  066700  156434                    ADD     RELOCF, R0      ;ADJUST POINTER.
4239  022144  105720                    6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
4240  022146  001006                            BNE     7$                      ;BR IF NOT 16-BIT OCTAL
4241  022150  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4242                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4243                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4244  022152  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4245  022156  004767  002434                    JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
4246  022162  000426                            BR      8$
4247  022164  100406                    7$:     BMI     17$             ;BRANCH IF NOT DECIMAL
4248  022166  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
4249                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
4250                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4251  022170  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4252  022174  004767  002140                    JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
4253  022200  000417                            BR      8$              ;SKIP
4254  022202  122760  177777  177777  17$:     CMPB    #-1,    -1(R0)  ;CHECK FOR 18-BIT ADDRESS FORMAT.
4255  022210  001004                            BNE     18$             ;BR IF NOT 18-BIT ADDRESS FORMAT.
4256  022212  013146                            MOV     @(R1)+, -(SP)   ;PUT THE DATA ON THE STACK.
4257  022214  004767  002640                    JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4258  022220  000407                            BR      8$              ;SK
4259  022222                            18$:
4260  022222  005046                            CLR     -(SP)                   ;  THE STACK.
4261  022224  113116                            MOVB    @(R1)+, (SP)            TA    STACK.
4262                                    ;*  THE NEXT TWO INSTRUCTIONS PR          O THE $TYPOS ROUTINE
4263                                    ;*  WIHTOUT USING A "TRAP" INSTRU           BY **SYSMAC**.
4264  022226  013746  177776                    MOV     @#PSW,  -(SP)   ;P      R STATUS ON THE STACK
4265  022232  004767  002334                    JSR     PC,     $TYPOS  ;GO     OUTINE
4266  022236     003                            .BYTE   3               ;TYPE 3
4267  022237     001                            .BYTE   1               ;TYPE LE      ROS.
4268  022240  005711                    8$:     TST     (R1)                    ;    ANOTHER NUMBER?
4269  022242  001404                            BEQ     9$                      ;B      O
4270  022244  004567  001150                    JSR     R5,     $PRINT  ;GO PRINT OU    E FOLLOWING MESSAGE.
4271  022250  022270                            .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
4272  022252  000734                            BR      6$                      ;LOOP
4273
4274  022254  012601                    9$:     MOV     (SP)+,R1                ;RESTORE R1
4275  022256  012600                    10$:    MOV     (SP)+,R0                ;RESTORE R0
4276  022260  004567  001134                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4277  022264  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4278  022266  000207                            RTS     PC                      ;RETURN
4279  022270  000011                    11$:    .ASCIZ  /       /       ;TAB CHARACTER.
4280                                            .EVEN
4281                                    .SBTTL  TTY INPUT ROUTINE
4282
4283                                    ;;**********************************************************************
4284                                    .ENABL  LSB
4285
4286                                    ;;**********************************************************************
```

```
4287                                    ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4288                                    ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4289                                    ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4290                                    ;*WHEN OPERATING IN TTY FLAG MODE.
4291  022272  022767  000176  156640  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
4292  022300  001104                          BNE     15$             ;;BRANCH IF NO
4293  022302  105777  156636                  TSTB    @$TKS           ;;CHAR THERE?
4294  022306  100101                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
4295  022310  117746  156632                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
4296  022314  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
4297  022320  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
4298  022324  001072                          BNE     15$             ;;NO, RETURN TO USER
4299  022326  126727  156602  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
4300  022334  001466                          BEQ     15$             ;;BRANCH IF YES
4301
4302  022336  004567  001056                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4303  022342  023217                          .WORD   $CNTLG          ;ADDRESS OF MESSAGE TO BE TYPED
4304  022344                          $GTSWR:
4305  022344  004567  001050                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4306  022350  023224                          .WORD   $MSWR           ;ADDRESS OF MESSAGE TO BE TYPED
4307  022352  016746  155620                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
4308                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4309                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC.**.
4310  022356  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4311  022362  004767  002230                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
4312  022366  004567  001026                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4313  022372  023235                          .WORD   $MNEW           ;ADDRESS OF MESSAGE TO BE TYPED
4314  022374  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
4315  022376  005046                          CLR     -(SP)           ;;THE NEW SWR
4316  022400  105777  156540          7$:     TSTB    @$TKS           ;;CHAR THERE?
4317  022404  100375                          BPL     7$              ;;IF NOT TRY AGAIN
4318
4319  022406  117746  156534                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
4320  022412  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
4321
4322
4323
4324  022416  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
4325  022422  001006                          BNE     10$             ;;BRANCH IF NOT
4326  022424  004567  000770                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4327  022430  023212                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
4328  022432  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
4329  022436  000756                          BR      19$             ;;LET'S TRY IT AGAIN
4330
4331
4332  022440  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
4333  022444  001023                          BNE     16$             ;;BRANCH IF NO
4334  022446  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
4335  022452  001403                          BEQ     11$             ;;BRANCH IF YES
4336  022454  016677  000002  156456          MOV     2(SP),@SWR      ;;SAVE NEW SWR
4337  022462  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
4338  022466                          14$:
4339  022466  004567  000726                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4340  022472  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4341  022474  126727  156435  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
4342  022502  001003                          BNE     15$             ;;BRANCH IF NOT
```

```
4343  022504  012777  000100  156432          MOV     #100,@#TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
4344  022512  000002                  15$:     RTI                     ;;RETURN
4345  022514  004767  001142          16$:     JSR     PC,$TYPEC       ;;ECHO CHAR
4346  022520  021627  000060                   CMP     (SP),#60        ;;CHAR < 0?
4347  022524  002420                           BLT     18$             ;;BRANCH IF YES
4348  022526  021627  000067                   CMP     (SP),#67        ;;CHAR > 7?
4349  022532  003015                           BGT     18$             ;;BRANCH IF YES
4350  022534  042726  000060                   BIC     #60,(SP)+       ;;STRIP-OFF ASCII
4351  022540  005766  000002                   TST     2(SP)           ;;IS THIS THE FIRST CHAR
4352  022544  001403                           BEQ     17$             ;;BRANCH IF YES
4353  022546  006316                           ASL     (SP)            ;;NO, SHIFT PRESENT
4354  022550  006316                           ASL     (SP)            ;;   CHAR OVER TO MAKE
4355  022552  006316                           ASL     (SP)            ;;   ROOM FOR NEW ONE.
4356  022554  005266  000002          17$:     INC     2(SP)           ;;KEEP COUNT OF CHAR
4357  022560  056616  177776                   BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
4358  022564  000705                           BR      7$              ;;GET THE NEXT ONE
4359  022566                          18$:
4360  022566  004567  000626                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4361  022572  001200                           .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
4362  022574  000716                           BR      20$             ;;SIMULATE CONTROL-U
4363                                  .DSABL   LSB
4364
4365
4366                                  ;;**************************************************************
4367                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4368                                  ;*CALL:
4369                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
4370                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
4371                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
4372                                  ;
4373
4374  022576  011646                  $RDCHR:  MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
4375  022600  016666  000004  000002           MOV     4(SP),2(SP)     ;;SAVE THE PS
4376  022606  105777  156332          1$:      TSTB    @#TKS           ;;WAIT FOR
4377  022612  100375                           BPL     1$              ;;A CHARACTER
4378  022614  117766  156326  000004           MOVB    @#TKB,4(SP)     ;;READ THE TTY
4379  022622  042766  177600  000004           BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
4380  022630  026627  000004  000023           CMP     4(SP),#23       ;;IS IT A CONTROL-S?
4381  022636  001013                           BNE     3$              ;;BRANCH IF NO
4382  022640  105777  156300          2$:      TSTB    @#TKS           ;;WAIT FOR A CHARACTER
4383  022644  100375                           BPL     2$              ;;LOOP UNTIL ITS THERE
4384  022646  117746  156274                   MOVB    @#TKB,-(SP)     ;;GET CHARACTER
4385  022652  042716  177600                   BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
4386  022656  022627  000021                   CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
4387  022662  001366                           BNE     2$              ;;IF NOT DISCARD IT
4388  022664  000750                           BR      1$              ;;YES, RESUME
4389  022666  026627  000004  000140  3$:      CMP     4(SP),#140      ;;IS IT UPPER CASE?
4390  022674  002407                           BLT     4$              ;;BRANCH IF YES
4391  022676  026627  000004  000175           CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
4392  022704  003003                           BGT     4$              ;;BRANCH IF YES
4393  022706  042766  000040  000004           BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
4394  022714  000002                  4$:      RTI                     ;;GO BACK TO USER
4395                                  ;;**************************************************************
4396                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4397                                  ;*CALL:
4398                                  ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
```

```
4399                                    ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4400                                    ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
4401
4402  022716  010346                    $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
4403  022720  005046                            CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
4404  022722  012703  023202            1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
4405  022726  022703  023212            2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
4406  022732  101467                            BLOS    4$              ;;BR IF YES
4407                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4408                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4409  022734  013746  177776                    MOV     @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
4410  022740  004767  177632                    JSR     PC,    $RDCHR   ;GO TO THE SUBROUTINE
4411  022744  112613                            MOVB    (SP)+,(R3)      ;;GET CHARACTER
4412  022746  122713  000177            10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
4413  022752  001024                            BNE     5$              ;;BR IF NO
4414  022754  005716                            TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
4415  022756  001010                            BNE     6$              ;;BR IF NO
4416  022760  112767  000134  000212          MOVB    #'\,9$          ;;TYPE A BACK SLASH
4417  022766  004567  000426                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4418  022772  023200                            .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4419  022774  012716  177777                    MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
4420  023000  005303                    6$:     DEC     R3              ;;BACKUP BY ONE
4421  023002  020327  023202                    CMP     R3,#$TTYIN      ;;STACK EMPTY?
4422  023006  103441                            BLO     4$              ;;BR IF YES
4423  023010  111367  000164                    MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
4424  023014  004567  000400                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4425  023020  023200                            .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4426  023022  000741                            BR      2$              ;;GO READ ANOTHER CHAR.
4427  023024  005716                    5$:     TST     (SP)            ;;RUBOUT KEY SET?
4428  023026  001407                            BEQ     7$              ;;BR IF NO
4429  023030  112767  000134  000142          MOVB    #'\,9$          ;;TYPE A BACK SLASH
4430  023036  004567  000356                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4431  023042  023200                            .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4432  023044  005016                            CLR     (SP)            ;;CLEAR THE RUBOUT KEY
4433  023046  122713  000025            7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
4434  023052  001004                            BNE     8$              ;;BR IF NO
4435  023054  004567  000340                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4436  023060  023212                            .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
4437  023062  000717                            BR      1$              ;;GO START OVER
4438  023064  122713  000022            8$:     CMPB    #22,(R3)        ;;IS CHARACTER A "↑R"?
4439  023070  001014                            BNE     3$              ;;BRANCH IF NO
4440  023072  105013                            CLRB    (R3)            ;;CLEAR THE CHARACTER
4441  023074  004567  000320                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4442  023100  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
4443  023102  004567  000312                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4444  023106  023202                            .WORD   $TTYIN          ;ADDRESS OF MESSAGE TO BE TYPED
4445  023110  000706                            BR      2$              ;;GO PICKUP ANOTHER CHACTER
4446  023112                            4$:
4447  023112  004567  000302                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4448  023116  001200                            .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
4449  023120  000700                            BR      1$              ;;CLEAR THE BUFFER AND LOOP
4450  023122  111367  000052            3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
4451  023126  004567  000266                    JSR     R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
4452  023132  023200                            .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
4453  023134  122723  000015                    CMPB    #15,(R3)+       ;;CHECK FOR RETURN
4454  023140  001272                            BNE     2$              ;;LOOP IF NOT RETURN
```

```
                                        CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
                                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
                                        .WORD   $LF             ;ADDRESS OF MESSAGE TO BE TYPED
                                        TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
                                        MOV     (SP)+,R3        ;;RESTORE R3
                                        MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
                                        MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
                                        MOV     #$TTYIN,4(SP)
                                        RTI                     ;;RETURN
                                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
                                        .BYTE   0               ;;TERMINATOR
                                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
                                $CNTLU: .ASCIZ  /↑U/<15><12>    ;;CONTROL "U"
                                $CNTLG: .ASCIZ  /↑G/<15><12>    ;;CONTROL "G"
                                $MSWR:  .ASCIZ  <15><12>/SWR = /
                                $MNEW:  .ASCIZ  /  NEW = /

                                .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

                                ;;**************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
                                ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
                                ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
                                ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
                                ;*CALL:
                                ;*      RDOCT                   ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT

                                $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
                                        MOV     4(SP),2(SP)     ;;INPUT NUMBER
                                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
                                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
                                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
                                1$:
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
                                        JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
                                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
                                        MOV     R0,5$           ;;AND SAVE IT
                                        CLR     R1              ;;CLEAR DATA WORD
                                        CLR     R2
                                2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
                                        BEQ     3$              ;;IF ZERO GET OUT
                                        CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
                                        BGT     4$              ;;IS AN OCTAL DIGIT
                                        CMPB    #'7,(SP)
                                        BLT     4$
                                        ASL     R1              ;;*2
                                        ROL     R2
                                        ASL     R1              ;;*4
                                        ROL     R2
```

```
4511  023336  006301                          ASL     R1              ;;*8
4512  023340  006102                          ROL     R2
4513  023342  042716  177770                  BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
4514  023346  062601                          ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
4515  023350  000756                          BR      2$              ;;LOOP
4516  023352  005726                  3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
4517  023354  010166  000012                  MOV     R1,12(SP)       ;;SAVE THE RESULT
4518  023360  010267  000032                  MOV     R2,$HIOCT
4519  023364  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
4520  023366  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4521  023370  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4522  023372  000002                          RTI                     ;;RETURN
4523  023374  005726                  4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
4524  023376  105010                          CLRB    (R0)            ;;SET A TERMINATOR
4525  023400  004567  000014                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4526  023404  000000                  5$:     .WORD   0
4527  023406  004567  000006                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4528  023412  001200                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
4529  023414  000723                          BR      1$              ;;TRY AGAIN
4530  023416  000000                  $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
4531
4532                                  ;;*****************************************************************
4533                                  ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
4534                                  ;*  CALL:       JSR     R5,     $PRINT
4535                                  ;*              <MESSAGE VIRTUAL ADDRESS>
4536                                  ;;*****************************************************************
4537  023420  012567  000016          $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
4538  023424  066767  155150  000010          ADD     RELOCF, 1$      ;MAKE IT PHYSICAL.
4539                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4540                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4541  023432  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
4542  023436  004767  000004                  JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
4543  023442  000000                  1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
4544  023444  000205                          RTS     R5              ;RETURN.
4545
4546                                  .SBTTL  TYPE ROUTINE
4547
4548                                  ;;*****************************************************************
4549                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
4550                                  ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
4551                                  ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4552                                  ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4553                                  ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4554                                  ;*
4555                                  ;*CALL:
4556                                  ;*1) USING A TRAP INSTRUCTION
4557                                  ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4558                                  ;*OR
4559                                  ;*      TYPE
4560                                  ;*      MESADR
4561                                  ;*
4562
4563  023446  105767  155505          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
4564  023452  100002                          BPL     1$              ;;BR IF YES
4565  023454  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
4566  023456  000430                          BR      3$              ;;LEAVE
```

```
4567  023460  010046                    1$:     MOV     R0,-(SP)        ;;SAVE R0
4568  023462  017600  000002                    MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
4569  023466  122767  000001  155530            CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
4570  023474  001011                            BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
4571  023476  132767  000100  155521            BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
4572  023504  001405                            BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
4573  023506  010067  000004                    MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
4574  023512  004767  000222                    JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
4575  023516  000000                    61$:    .WORD   0               ;;MESSAGE ADDRESS
4576  023520  132767  000040  155477    62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
4577  023526  001003                            BNE     60$             ;;YES,SKIP TYPE OUT
4578  023530  112046                    2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
4579  023532  001005                            BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
4580  023534  005726                            TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
4581  023536  012600                    60$:    MOV     (SP)+,R0        ;;RESTORE R0
4582  023540  062716  000002            3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
4583  023544  000002                            RTI                     ;;RETURN
4584  023546  122716  000011            4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
4585  023552  001431                            BEQ     8$
4586  023554  122716  000200                    CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
4587  023560  001007                            BNE     5$
4588  023562  005726                            TST     (SP)+           ;;POP  <CR><LF> EQUIV
4589  023564  004567  177630                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4590  023570  001201                            $CRLF
4591  023572  105067  000130                    CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
4592  023576  000754                            BR      2$              ;;GET NEXT CHARACTER
4593  023600  004767  000056            5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
4594  023604  126726  155346            6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
4595  023610  001347                            BNE     2$              ;;IF NO GO GET NEXT CHAR.
4596  023612  016746  155336                    MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
4597                                                                    ;;AND THE NULL CHAR.
4598  023616  105366  000001            7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
4599  023622  002770                            BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
4600  023624  004767  000032                    JSR     PC,$TYPEC       ;;GO TYPE A NULL
4601  023630  105367  000072                    DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
4602  023634  000770                            BR      7$              ;;LOOP
4603
4604                                    ;HORIZONTAL TAB PROCESSOR
4605
4606  023636  112716  000040            8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
4607  023642  004767  000014            9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
4608  023646  132767  000007  000052            BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
4609  023654  001372                            BNE     9$              ;;TAB STOP
4610  023656  005726                            TST     (SP)+           ;;POP SPACE OFF STACK
4611  023660  000723                            BR      2$              ;;GET NEXT CHARACTER
4612  023662  105777  155262            $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
4613  023666  100375                            BPL     $TYPEC
4614  023670  116677  000002  155254            MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4615  023676  122766  000015  000002            CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
4616  023704  001003                            BNE     1$              ;;BRANCH IF NO
4617  023706  105067  000014                    CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
4618  023712  000406                            BR      $TYPEX          ;;EXIT
4619  023714  122766  000012  000002    1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
4620  023722  001402                            BEQ     $TYPEX          ;;BRANCH IF YES
4621  023724  105227                            INCB    (PC)+           ;;COUNT THE CHARACTER
4622  023726  000000            $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
```

```
 4623  023730  000207                    $TYPEX: RTS     PC
 4624
 4625                                    .SBTTL  APT COMMUNICATIONS ROUTINE
 4626
 4627                                    ;;**********************************************************************
 4628  023732  112767  000001  000376  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
 4629  023740  112767  000001  000366  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
 4630  023746  000403                          BR      $ATYC
 4631  023750  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 4632  023756                          $ATYC:
 4633  023756  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4634  023760  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4635  023762  105767  000346                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
 4636  023766  001450                          BEQ     5$              ;;IF NOT:  BR
 4637  023770  122767  000001  155226          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
 4638  023776  001031                          BNE     3$              ;;IF NOT:  BR
 4639  024000  132767  000100  155217          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 4640  024006  001425                          BEQ     3$              ;;IF NOT:  BR
 4641  024010  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
 4642  024014  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 4643  024022  005767  155156          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
 4644  024026  001375                          BNE     1$              ;;IF NOT:  WAIT
 4645  024030  010067  155164                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 4646  024034  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 4647  024036  001376                          BNE     2$
 4648  024040  166700  155154                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
 4649  024044  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
 4650  024046  010067  155150                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
 4651  024052  012767  000004  155124          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
 4652  024060  000413                          BR      5$
 4653  024062  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
 4654  024070  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
 4655  024076  016746  153674                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
 4656  024102  004767  177340                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
 4657  024106  000000                  4$:     .WORD   0
 4658  024110                          5$:
 4659  024110  105767  000221                  TSTB    $LFLG           ;;SHOULD LOG AN ERROR?
 4660  024114  001422                          BEQ     10$             ;;IF NOT:  BR
 4661  024116  017600  000004                  MOV     @4(SP),R0               ;;GET ERROR #
 4662  024122  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 4663  024130  012701  001344                  MOV     #$ASTAT,R1      ;;POINT TO TABLE START
 4664  024134  005711                  6$:     TST     (R1)            ;;END OF TABLE?
 4665  024136  100404                          BMI     8$              ;;IF SO:  BR
 4666  024140  020021                          CMP     R0,(R1)+        ;;PROPER ENTRY?
 4667  024142  001406                          BEQ     9$              ;;IF SO:  BR
 4668  024144  005721                          TST     (R1)+           ;;MOVE PAST COUNTER WORD
 4669  024146  000772                          BR      6$              ;;KEEP LOOKING
 4670  024150  026701  155336          8$:     CMP     $APTR,R1        ;;TABLE FULL?
 4671  024154  001402                          BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
 4672  024156  010021                          MOV     R0,(R1)+        ;;SET UP NEW ENTRY
 4673  024160  005211                  9$:     INC     (R1)            ;;BUMP ERROR COUNT
 4674  024162  105767  000150          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
 4675  024166  001416                          BEQ     12$             ;;IF NOT:  BR
 4676  024170  005767  155030                  TST     $ENV            ;;RUNNING UNDER APT?
 4677  024174  001413                          BEQ     12$             ;;IF NOT:  BR
 4678  024176  005767  155002          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
```

```
4679  024202  001375                        BNE     11$             ;;IF NOT:  WAIT
4680  024204  017667  000004  154774        MOV     @4(SP),$FATAL   ;;GET ERROR #
4681  024212  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
4682  024220  005267  154760                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
4683  024224  105067  000106          12$:  CLRB    $FFLG           ;;CLEAR FATAL FLAG
4684  024230  105067  000101                CLRB    $LFLG           ;;CLEAR LOG FLAG
4685  024234  105067  000074                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
4686  024240  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4687  024242  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4688  024244  000207                        RTS     PC              ;;RETURN
4689  024246                          $ATY6:
4690  024246  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4691  024250  016700  155236                MOV     $APTR,R0
4692  024254  162700  001344                SUB     #$ASTAT,R0      ;;GET SIZE OF STAT TABLE
4693  024260  005767  154720          1$:   TST     $MSGTY          ;;SEE IF DONE LAST COMMUNICATION
4694  024264  001375                        BNE     1$              ;;IF NOT:  WAIT
4695  024266  010067  154730                MOV     R0,$MSGLG       ;;SET MESSAGE LENGTH
4696  024272  012767  001344  154720        MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
4697  024300  012767  000002  154676        MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
4698  024306  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4699  024310  000207                        RTS     PC              ;;RETURN
4700  024312                          $ATY7:
4701  024312  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4702  024314  012701  001344                MOV     #$ASTAT,R1      ;;GET START OF TABLE
4703  024320  005721                  1$:   TST     (R1)+           ;;END OF TABLE?
4704  024322  100402                        BMI     2$              ;;IF SO:  BR
4705  024324  005021                        CLR     (R1)+           ;;CLEAR ERROR COUNT
4706  024326  000774                        BR      1$              ;;KEEP CLEARING
4707  024330                          2$:
4708  024330  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4709  024332  000207                        RTS     PC              ;;RETURN
4710  024334     000                  $MFLG:  .BYTE   0               ;;MESSG. FLAG
4711  024335     000                  $LFLG:  .BYTE   0               ;;LOG FLAG
4712  024336     000                  $FFLG:  .BYTE   0               ;;FATAL FLAG
4713        024340                          .EVEN
4714        000200                  APTSIZE=200
4715        000001                  APTENV=001
4716        000100                  APTSPOOL=100
4717        000040                  APTCSUP=040
4718                                ;;******************************************************************
4719
4720                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
4721
4722                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
4723                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
4724                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
4725                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
4726                                ;*REPLACED WITH SPACES.
4727                                ;*CALL:
4728                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
4729                                ;*      TYPDS                   ;;GO TO THE ROUTINE
4730
4731  024340                        $TYPDS:
4732  024340  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4733  024342  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4734  024344  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
```

```
4735  024346  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4736  024350  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
4737  024352  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
4738  024356  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
4739  024362  100004                        BPL     1$              ;;BR IF INPUT IS POS.
4740  024364  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
4741  024366  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
4742  024374  016700  154200          1$:   MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
4743  024400  012703  024562                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
4744  024404  060003                        ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
4745  024406  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
4746  024412  005002                  2$:   CLR     R2              ;;CLEAR THE BCD NUMBER
4747  024414  016001  024552                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
4748  024420  160105                  3$:   SUB     R1,R5           ;;FORM THIS BCD DIGIT
4749  024422  002402                        BLT     4$              ;;BR IF DONE
4750  024424  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
4751  024426  000774                        BR      3$
4752  024430  060105                  4$:   ADD     R1,R5           ;;ADD BACK THE CONSTANT
4753  024432  005702                        TST     R2              ;;CHECK IF BCD DIGIT:0
4754  024434  001002                        BNE     5$              ;;FALL THROUGH IF 0
4755  024436  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
4756  024440  100407                        BMI     7$              ;;BR IF YES
4757  024442  106316                  5$:   ASLB    (SP)            ;;MSD?
4758  024444  103003                        BCC     6$              ;;BR IF NO
4759  024446  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
4760  024454  052702  000060          6$:   BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
4761  024460  052702  000040          7$:   BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
4762  024464  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
4763  024466  005720                        TST     (R0)+           ;;JUST INCREMENTING
4764  024470  020067  155140                CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
4765  024474  103746                        BLO     2$              ;GO DO THE NEXT DIGIT
4766  024476  101002                        BHI     8$              ;;GO TO EXIT
4767  024500  010502                        MOV     R5,R2           ;;GET THE LSD
4768  024502  000764                        BR      6$              ;;GO CHANGE TO ASCII
4769  024504  105726                  8$:   TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
4770  024506  100003                        BPL     9$              ;;BR IF NO
4771  024510  116663  177777  177776        MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
4772  024516  105013                  9$:   CLRB    (R3)            ;;SET THE TERMINATOR
4773  024520  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
4774  024522  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
4775  024524  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
4776  024526  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
4777  024530  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
4778  024532  004567  176662                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4779  024536  024562                        .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
4780  024540  016666  000002  000004        MOV     2(SP),4(SP)     ;;ADJUST THE STACK
4781  024546  012616                        MOV     (SP)+,(SP)
4782  024550  000002                        RTI                     ;;RETURN TO USER
4783  024552  023420                  $DTBL:  10000.
4784  024554  001750                        1000.
4785  024556  000144                        100.
4786  024560  000012                        10.
4787  024562  000004                  $DBLK:  .BLKW   4
4788                                  .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
4789
4790                                  ;;**************************************************************
```

```
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
                                ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOS                   ;;CALL FOR TYPEOUT
                                ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
                                ;*      .BYTE   M               ;;M=1 OR 0
                                ;*                                      ;;1=TYPE LEADING ZEROS
                                ;*                                      ;;0=SUPPRESS LEADING ZEROS
                                ;*
                                ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
                                ;*$TYPOS OR $TYPOC
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPON                   ;;CALL FOR TYPEOUT
                                ;*
                                ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
                                ;*      TYPOC                   ;;CALL FOR TYPEOUT
                                
024572  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024576  116667  000001  000213          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024604  112667  000211                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024610  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024614  000406                          BR      $TYPON
024616  112767  000001  000173  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024624  112767  000006  000167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024632  112767  000005  000156  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024640  010346                          MOV     R3,-(SP)        ;;SAVE R3
024642  010446                          MOV     R4,-(SP)        ;;SAVE R4
024644  010546                          MOV     R5,-(SP)        ;;SAVE R5
024646  116704  000147                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024652  005404                          NEG     R4
024654  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024660  110467  000134                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024664  116704  000127                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024670  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024674  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
024676  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
024700  000404                          BR      3$              ;;GO DO MSB
024702  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
024704  006105                          ROL     R5
024706  006105                          ROL     R5
024710  010503                          MOV     R5,R3
024712  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
024714  105367  000100                  DECB    $OMODE          ;;TYPE THIS DIGIT?
024720  100017                          BPL     7$              ;;BR IF NO
024722  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
024726  001002                          BNE     4$              ;;TEST FOR 0
024730  005704                          TST     R4              ;;SUPPRESS THIS 0?
024732  001403                          BEQ     5$              ;;BR IF YES
024734  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
024736  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
024742  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
```

```
4847  024746  110367  000042                  MOVB    R3,8$           ;;SAVE FOR TYPING
4848  024752  004567  176442                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4849  024756  025014                          .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
4850  024760  105?27  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
4851  024764  003346                          BGT     2$              ;;BR IF MORE TO DO
4852  024766  002402                          BLT     6$              ;;BR IF DONE
4853  024770  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
4854  024772  000743                          BR      2$              ;;GO DO THE LAST DIGIT
4855  024774  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
4856  024776  012604                          MOV     (SP)+,R4        ;;RESTORE R4
4857  025000  012603                          MOV     (SP)+,R3        ;;RESTORE R3
4858  025002  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
4859  025010  012616                          MOV     (SP)+,(SP)
4860  025012  000002                          RTI                     ;;RETURN
4861  025014     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
4862  025015     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
4863  025016     000          $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
4864  025017     000          $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
4865  025020  000000          $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
4866                          ;ERROR TRAP SERVICE ROUTINE
4867  025022  005727          ERRTRP: TST     (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
4868  025024  000000          1$:     .WORD   0               ;CONTAINS ERROR REPORTED FLAG
4869  025026  001010                  BNE     2$              ;BRANCH IF NOT REPORTED
4870  025030  005267  177770          INC     1$              ;SET DOUBLE TRAP FLAG.
4871  025034  011667  154126          MOV     (SP),   $TMP3   ;SAVE THE BAD PC FOR TYPOUT.
4872  025040  004767  174502          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
4873  025044  000031                  .WORD   31              ;ERROR TYPE CODE.
4874  025046  000401                  BR      3$              ;SKIP HALT
4875  025050  000000          2$:     HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
4876                                                          ;BEFORE FIRST WAS PRINTED
4877  025052  005067  177746          3$:     CLR     1$
4878  025056  000002                  RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
4879
4880                          .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
4881                          ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4882  025060                  $TYPAD:
4883  025060  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
4884  025062  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
4885  025064  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
4886  025066  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
4887  025070  016602  000012          MOV     12(SP), R2      ;GET BASE ADDRESS
4888  025074  005003                  CLR     R3              ;WORKING & INDEX REGISTER
4889  025076  005767  153504          TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
4890  025102  001430                  BEQ     1$              ;BRANCH IF NO MEM MGMT
4891  025104  032737  000001  177572  BIT     #1,     @#SR0   ;CHECK IF MEM MGMT ENABLED
4892  025112  001424                  BEQ     1$              ;BRANCH IF MEM MGMT NOT ENABLED
4893  025114  010201                  MOV     R2,     R1      ;COPY VIRTUAL ADR
4894  025116  006101                  ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
4895  025120  006101                  ROL     R1
4896  025122  006101                  ROL     R1
4897  025124  006101                  ROL     R1
4898  025126  006101                  ROL     R1
4899  025130  042701  177761          BIC     #177761, R1     ;CLR ALL EXCEPT BITS 1,2,3
4900  025134  062701  172340          ADD     #KIPAR0, R1     ;SET TO APPROPRIATE PAR
4901  025140  011101                  MOV     (R1),   R1      ;GET CONTENTS OF PAR
4902  025142  012700  000006          MOV     #6,     R0      ;SET UP COUNTER
```

```
4903  025146  006301                  4$:     ASL     R1              ;SHIFT PAR
4904  025150  006103                          ROL     R3              ;SAVE OVERFLOW BITS
4905  025152  077003                          SOB     R0,     4$      ;COUNT SIX SHIFTS
4906  025154  042702  160000                BIC     #160000, R2     ;SAVE BANK BITS
4907  025160  060102                          ADD     R1,     R2      ;COMPUTE PHYSICAL ADDRESS
4908  025162  005503                          ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
4909  025164  006302                  1$:     ASL     R2              ;FIRST DIGIT TO R3
4910  025166  006103                          ROL     R3
4911  025170  012700  000006                MOV     #6,R0           ;DIGIT COUNT
4912  025174  000404                          BR      3$              ;PRINT FIRST DIGIT
4913  025176  006302                  2$:     ASL     R2
4914  025200  006103                          ROL     R3
4915  025202  005301                          DEC     R1
4916  025204  001374                          BNE     2$
4917  025206  012701  000003          3$:     MOV     #3,R1           ;DIGIT SHIFT COUNT
4918  025212  062703  000060                ADD     #60,    R3      ;MAKE IT AN ASCII DIGIT
4919  025216  110367  000036                MOVB    R3,     B$      ;LOAD DIGIT INTO MESSAGE
4920  025222  004567  176172                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4921  025226  025260                          .WORD   B$              ;ADDRESS OF MESSAGE TO BE TYPED
4922  025230  005003                          CLR     R3              ;CLEAR INDEX
4923  025232  005300                          DEC     R0              ;DEC DIGIT COUNT
4924  025234  001360                          BNE     2$
4925  025236  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
4926  025240  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
4927  025242  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
4928  025244  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
4929  025246  012616                          MOV     (SP)+,  (SP)    ;ADJUST THE STACK TO CLEAR DATA
4930  025250  004567  176144                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4931  025254  026711                          .WORD   FILL2           ;ADDRESS OF MESSAGE TO BE TYPED
4932  025256  000207                          RTS     PC              ;RETURN
4933  025260     000          B$:             .BYTE   0               ;ONE DIGIT MESSAGE BUFFER
4934  025261     000                          .BYTE   0               ;MESSAGE TERMINATOR
4935
4936                                          .SBTTL  STANDARD PROGRAM MESSAGES
4937                                  ;;**************************************************************
4938                                  ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
4939                                  ;THE PROGRAM
4940                                  ;;**************************************************************
4941  025262  005015  052113  030461  KMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
4942  025270  024040  042515  047515
4943  025276  054522  046440  047101
4944  025304  043501  046505  047105
4945  025312  024524  040440  040526
4946  025320  046111  041101  042514
4947  025326     000
4948  025327     015  046412  046505  MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
4949  025334  051117  020131  040515
4950  025342  035120     000
4951  025345     015  041012  052131  BYTMES: .ASCIZ  <15><12>'BYTE MEMORY MAP:'
4952  025352  020105  042515  047515
4953  025360  054522  046440  050101
4954  025366  000072
4955  025370  005015  040520  044522  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
4956  025376  054524  046440  046505
4957  025404  051117  020131  040515
4958  025412  035120     000
```

```
 4959  025415    015  043012  047522  FROM:   .ASCIZ  <15><12>'FROM '
 4960  025422 020115     000
 4961  025425    040  047524  000040  TO:     .ASCIZ  ' TO '
 4962  025432 005015  047111  052523  INSUFF: .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
 4963  025440 043106  041511  042511
 4964  025446 052116  046440  046505
 4965  025454 051117  027131  027056
 4966  025462 044506  051522  020124
 4967  025470 033061  020113  047516
 4968  025476 020124  046101  020114
 4969  025504 044124  051105  020505
 4970  025512    000
 4971  025513    015  047012  020117  MTR:    .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
 4972  025520 040520  044522  054524
 4973  025526 051040  043505  051511
 4974  025534 042524  051522  043040
 4975  025542 052517  042116     000
 4976  025547    015  051012  051505  PWRMSG: .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
 4977  025554 040524  052122  047111
 4978  025562 020107  043101  042524
 4979  025570 020122  020101  047520
 4980  025576 042527  020122  040506
 4981  025604 046111  051125  006505
 4982  025612 000012
 4983  025614 005015  047516  050040  NOPES:  .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
 4984  025622 051101  052111  020131
 4985  025630 051105  047522  051522
 4986  025636 043040  052517  042116
 4987  025644 047440  020116  042515
 4988  025652 047515  054522  051440
 4989  025660 040503  006516  000012
 4990  025666 005015  051120  043517  PROREL: .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
 4991  025674 040522  020115  047516
 4992  025702 020127  042522  044523
 4993  025710 042504  020123  040502
 4994  025716 045503  040440  020124
 4995  025724 020060  047524  034040
 4996  025732    113
 4997  025733    015  044012  052111          .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
 4998  025740 041440  047117  044524
 4999  025746 052516  020105  047506
 5000  025754 020122  047516  046522
 5001  025762 046101  051040  047125
 5002  025770 044516  043516  005015
 5003  025776    000
 5004  025777    122  043505  051511  MX1:    .ASCIZ  'REGISTER AT '
 5005  026004 042524  020122  052101
 5006  026012 000040
 5007  026014 041440  047117  051124  MX2:    .ASCIZ  ' CONTROLS '
 5008  026022 046117  020123     000
 5009  026027    015  041412  051117  MX3:    .ASCIZ  <15><12>'CORE PARITY '
 5010  026034 020105  040520  044522
 5011  026042 054524  000040
 5012  026046 005015  047515  020123  MX4:    .ASCIZ  <15><12>'MOS PARITY '
 5013  026054 040520  044522  054524
 5014  026062 000040
```

```
5015  026064  005015  051515  030461  MX5:    .ASCIZ  <15><12>'MS11-K CSR '
5016  026072  045455  041440  051123
5017  026100  000040
5018  026102  051515  030461  045455  MX6:    .ASCIZ  'MS11-K MEMORY PRESENT!! TO COMPLETELY TEST RUN DZMML...'
5019  026110  046440  046505  051117
5020  026116  020131  051120  051505
5021  026124  047105  020524  020041
5022  026132  047524  041440  046517
5023  026140  046120  052105  046105
5024  026146  020131  042524  052123
5025  026154  051040  047125  042040
5026  026162  046532  046115  027056
5027  026170  000056
5028  026172  005015  047516  046440  NOMEM:  .ASCIZ  <15><12>'NO MEMORY FOUND.'
5029  026200  046505  051117  020131
5030  026206  047506  047125  027104
5031  026214     000
5032  026215     015  005012  044412  FADMES: .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5033  026222  050116  052125  040440
5034  026230  046114  050040  051101
5035  026236  046501  052105  051105
5036  026244  020123  047111  047440
5037  026252  052103  046101     056
5038  026257     015  043012  051111          .ASCIZ  <15><12>'FIRST ADDRESS:  '
5039  026264  052123  040440  042104
5040  026272  042522  051523  020072
5041  026300  000040
5042  026302  005015  040514  052123  LADMES: .ASCIZ  <15><12>'LAST ADDRESS:   '
5043  026310  040440  042104  042522
5044  026316  051523  020072  020040
5045  026324     000
5046  026325     015  037412  042101  BADADR: .ASCIZ  <15><12>''?ADDRESS IN UNMAPPED BANK?'
5047  026332  051104  051505  020123
5048  026340  047111  052440  046516
5049  026346  050101  042520  020104
5050  026354  040502  045516  000077
5051  026362  005015  042523  042514  CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
5052  026370  052103  041440  047117
5053  026376  052123  047101  035124
5054  026404     000
5055  026405     015  052412  042516  UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
5056  026412  050130  041505  042524
5057  026420  020104  042515  047515
5058  026426  054522  050040  051101
5059  026434  052111  020131  051105
5060  026442  047522  000122
5061  026446  005015  051120  043517  PRELOC: .ASCIZ  <15><12>'PROGRAM RELOCATED TO '
5062  026454  040522  020115  042522
5063  026462  047514  040503  042524
5064  026470  020104  047524  000040
5065  026476  005015  047515  042522  MTOE:   .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
5066  026504  052040  040510  020116
5067  026512  047117  020105  040520
5068  026520  044522  054524  042440
5069  026526  051122  051117  043040
5070  026534  052517  042116  000056
```

```
5071  026542  005015  041523  047101  SCANM:  .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
5072  026550  044516  043516  046440
5073  026556  046505  051117  020131
5074  026564  047506  020122  040502
5075  026572  020104  040520  044522
5076  026600  054524  000056
5077  026604  005015  040520  044522  PEWNC:  .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
5078  026612  054524  042440  051122
5079  026620  051117  053440  046111
5080  026626  020114  047516  020124
5081  026634  046103  040505  027122
5082  026642     000
5083  026643     015  047012  020117  NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
5084  026650  042515  047515  054522
5085  026656  052040  051505  042524
5086  026664  027104     000
5087  026667     015  051412  044513  SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
5088  026674  050120  047111  020107
5089  026702  042524  052123  021440
5090  026710     000
5091  026711     377  000377          FILL2:  .ASCIZ  <377><377>
5092
5093                                  .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
5094                                  ;;*****************************************************************
5095                                  ;; MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
5096                                  ;;*****************************************************************
5097  026714  040520  044522  054524  DM1:    .ASCIZ  'PARITY REGISTER DATA ERROR.'
5098  026722  051040  043505  051511
5099  026730  042524  020122  040504
5100  026736  040524  042440  051122
5101  026744  051117  000056
5102  026750  042101  051104  051505  DM2:    .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
5103  026756  020123  042524  052123
5104  026764  042440  051122  051117
5105  026772  052050  052123  026461
5106  027000  024465  000056
5107  027004  047503  051516  040524  DM4:    .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
5108  027012  052116  042040  052101
5109  027020  020101  051105  047522
5110  027026  024122  051524  033124
5111  027034  030455  024460  000056
5112  027042  047522  040524  044524  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
5113  027050  043516  041040  052111
5114  027056  042440  051122  051117
5115  027064  052050  052123  030461
5116  027072  030455  024462  000056
5117  027100  047515  020123  042522  DM6:    .ASCIZ  'MOS REFRESH TEST ERROR (TST 30-31).'
5118  027106  051106  051505  020110
5119  027114  042524  052123  042440
5120  027122  051122  051117  024040
5121  027130  051524  020124  030063
5122  027136  031455  024461  000056
5123  027144  020063  047530  020122  DM7:    .ASCIZ  '3 XOR 9 PATTERN ERROR(TST13-16).'
5124  027152  020071  040520  052124
5125  027160  051105  020116  051105
5126  027166  047522  024122  051524
```

```
  5127  027174  030524  026463  033061
  5128  027202  027051     000
  5129  027205     115  051101  044103  DM10:   .ASCIZ  "MARCHING 1'S AND 0'S ERROR(TST 27)."
  5130  027212  047111  020107  023461
  5131  027220  020123  047101  020104
  5132  027226  023460  020123  051105
  5133  027234  047522  024122  051524
  5134  027242  020124  033462  027051
  5135  027250     000
  5136  027251     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST17).'
  5137  027256  020131  042515  047515
  5138  027264  054522  040440  042104
  5139  027272  042522  051523  042440
  5140  027300  051122  051117  052050
  5141  027306  052123  033461  027051
  5142  027314     000
  5143  027315     104  052101  050111  DM12:   .ASCIZ  "DATIP WITH WRONG PARITY DIDN'T TRAP(TST17)."
  5144  027322  053440  052111  020110
  5145  027330  051127  047117  020107
  5146  027336  040520  044522  054524
  5147  027344  042040  042111  023516
  5148  027352  020124  051124  050101
  5149  027360  052050  052123  033461
  5150  027366  027051     000
  5151  027371     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
  5152  027376  050040  051101  052111
  5153  027404  020131  051124  050101
  5154  027412  042520  026104  041040
  5155  027420  052125  047040  020117
  5156  027426  042522  044507  052123
  5157  027434  051105  051440  047510
  5158  027442  051527  042440  051122
  5159  027450  051117  043040  040514
  5160  027456  027107     000
  5161  027461     120  051101  052111  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
  5162  027466  020131  042522  044507
  5163  027474  052123  051105  047040
  5164  027502  052117  046440  050101
  5165  027510  042520  020104  051501
  5166  027516  041440  047117  051124
  5167  027524  046117  044514  043516
  5168  027532  052040  044510  020123
  5169  027540  042101  051104  051505
  5170  027546  024123  051524  030524
  5171  027554  024467  000056
  5172  027560  047515  042522  052040  DM16:   .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
  5173  027566  040510  020116  047117
  5174  027574  020105  042522  044507
  5175  027602  052123  051105  044440
  5176  027610  042116  041511  052101
  5177  027616  042105  050040  051101
  5178  027624  052111  020131  051105
  5179  027632  047522  027122     000
  5180  027637     104  052101  020101  DM17:   .ASCIZ  "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17)."
  5181  027644  044123  052517  042114
  5182  027652  023516  020124  040510
```

```
 5183  027660  042526  041440  040510
 5184  027666  043516  042105  053440
 5185  027674  042510  020116  040520
 5186  027702  044522  054524  042440
 5187  027710  051122  051117  052040
 5188  027716  040522  050120  042105
 5189  027724  052050  052123  033461
 5190  027732  027051     000
 5191  027735     122  047101  047504  DM20:   .ASCIZ  'RANDOM DATA ERROR(TST20).'
 5192  027742  020115  040504  040524
 5193  027750  042440  051122  051117
 5194  027756  052050  052123  030062
 5195  027764  027051     000
 5196  027767     111  051516  051124  DM21:   .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST21-26).'
 5197  027774  041525  044524  047117
 5198  030002  042440  042530  052503
 5199  030010  044524  047117  042440
 5200  030016  051122  051117  052050
 5201  030024  052123  030462  031055
 5202  030032  024466  000056
 5203  030036  051120  043517  040522  DM23:   .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
 5204  030044  020115  047503  042504
 5205  030052  041440  040510  043516
 5206  030060  042105  053440  042510
 5207  030066  020116  042522  047514
 5208  030074  040503  042524  027104
 5209  030102     000
 5210  030103     124  040522  050120  DM24:   .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
 5211  030110  042105  020054  052502
 5212  030116  020124  047516  051040
 5213  030124  043505  051511  042524
 5214  030132  020122  040510  020104
 5215  030140  051105  047522  020122
 5216  030146  044502  020124  042523
 5217  030154  027124     000
 5218  030157     124  040522  050120  DM25:   .ASCIZ  'TRAPPED TO 114.'
 5219  030164  042105  052040  020117
 5220  030172  030461  027064     000
 5221  030177     106  044501  042514  DM26:   .ASCIZ  'FAILED TO TRAP.'
 5222  030204  020104  047524  052040
 5223  030212  040522  027120     000
 5224  030217      50  041501  044524  DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
 5225  030224  047117  042440  040516
 5226  030232  046102  020105  040527
 5227  030240  047123  052047  051440
 5228  030246  052105  027051     000
 5229  030253      15  052012  040522  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
 5230  030260  050120  042105  052040
 5231  030266  020117  020064     000
 5232
 5233                                  ;;*****************************************************************
 5234                                  ;DATA COLUMN HEADINGS
 5235                                  ;;*****************************************************************
 5236
 5237  030273     120  004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
 5238  030300  004507  027523  004502
```

```
 5239  030306  040527  000123
 5240  030312  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
 5241  030320  050057  004503  040515
 5242  030326  051411  041057  053411
 5243  030334  051501     000
 5244  030337     126  050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
 5245  030344  027520  041520  046411
 5246  030352  004501  027523  000102
 5247  030360  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
 5248  030366  050057  004503  042522
 5249  030374  004507  040515     000
 5250  030401     126  050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
 5251  030406  027520  041520  046411
 5252  030414  052501  004524  042522
 5253  030422  004507  027523  004502
 5254  030430  040527  000123
 5255  030434  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
 5256  030442  050057  004503  052511
 5257  030450  004524  040515  051411
 5258  030456  041057  053411  051501
 5259  030464     000
 5260  030465     126  050057  004503  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
 5261  030472  027520  041520  051411
 5262  030500  041522  046440  004501
 5263  030506  051504  020124  040515
 5264  030514  051411  041057  053411
 5265  030522  051501     000
 5266  030525     126  050057  004503  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
 5267  030532  027520  041520  052011
 5268  030540  050122  050057  000103
 5269  030546  027526  041520  050011  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
 5270  030554  050057  004503  051124
 5271  030562  027520  041520  051011
 5272  030570  043505  053411  051501
 5273  030576     000
 5274  030577     126  050057  004503  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
 5275  030604  027520  041520  051011
 5276  030612  043505  053411  051501
 5277  030620     000
 5278  030621     122  043505  053411  DH30:   .ASCIZ  'REG    WAS     MA      WAS'
 5279  030626  051501  046411  004501
 5280  030634  040527  000123
 5281
 5282                                 ;;*****************************************************************
 5283                                 ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
 5284                                 ;;*****************************************************************
 5285  030640     000     377     000  DF1:    .BYTE   0,-1,0,0
 5286  030643     000
 5287  030644     000     377     377  DF2:    .BYTE   0,-1,-1,0,0
 5288  030647     000     000
 5289  030651     000     377     377  DF3:    .BYTE   0,-1,-1,-2,-2
 5290  030654     376     376
 5291  030656     000     377     377  DF14:   .BYTE   0,-1,-1,-1,0,0
 5292  030661     377     000     000
 5293  030664     000     377     000  DF21:   .BYTE   0,-1,0,-1,0,0
 5294  030667     377     000     000
```

F15


```
5295  030672    377    000    377  DF30:   .BYTE   -1,0,-1,-2
5296  030675    376
5297                              .EVEN
5298
5299          032110              . = 32110                    ;THE LOADERS ARE SAVE HERE TO END OF 8K
5300
5301          000001              .END
```

```
ABASE = 000000            384     425
ACDW1 = 000000            384     427
ACDW2 = 000000            384     428
ACPUOP= 000000            384     394
ADDW0 = 000000            384     429
ADDW1 = 000000            384     430
ADDW10= 000000            384     439
ADDW11= 000000            384     440
ADDW12= 000000            384     441
ADDW13= 000000            384     442
ADDW14= 000000            384     443
ADDW15= 000000            384     444
ADDW2 = 000000            384     431
ADDW3 = 000000            384     432
ADDW4 = 000000            384     433
ADDW5 = 000000            384     434
ADDW6 = 000000            384     435
ADDW7 = 000000            384     436
ADDW8 = 000000            384     437
ADDW9 = 000000            384     438
ADEVCT= 000000            384     390
ADEVM = 000000            384     426
AE    = 000001            179#   2416    2502    3728
AENV  = 000000            384     395
AENVM = 000000            384     396
AFATAL= 000000            384     387
AMADR1= 000000            384     412
AMADR2= 000000            384     416
AMADR3= 000000            384     419
AMADR4= 000000            384     422
AMAMS1= 000000            384     406
AMAMS2= 000000            384     414
AMAMS3= 000000            384     417
AMAMS4= 000000            384     420
AMSGAD= 000000            384     392
AMSGLG= 000000            384     393
AMSGTY= 000000            384     386
AMTYP1= 000000            384     407
AMTYP2= 000000            384     415
AMTYP3= 000000            384     418
AMTYP4= 000000            384     421
APASS = 000000            384     389
APRIOR= 000000            384
APTCSU= 000040           4576    4717#
APTENV= 000001           4165    4569    4637    4715#
APTS.Z= 000200            873    4714#
APTSPO= 000100           4571    4639    4716#
ASWREG= 000000            384     397
ATESTN= 000000            384     388
AUNIT = 000000            384     391
AUSWR = 000000            384     398
AVECT1= 000000            384     423
AVECT2= 000000            384     424
BADADR  026325           1432    5046#
BANKNO  016036           1615    1624    1650    1660    3375#
BITPT   001544            522#   1122#   1123#   1159    1160    1161    1162    1170#   1171#   1180    1182    1185    1188#
```

```
                        2392    2394    2450    2452    3126*   3127*   3134    3136    3139*   3140*   3143    3145    3150
                        3151    3170    3173*   3176    3177    3191    3192    3193    3195    3204*   3205*   3209*   3214
                        3216    3237*   3238*   3240    3242    3244    3246    3250    3251    3281*   3283    3285    3288
                        3295    3297    3302    3303    3318    3320    3328*   3329*   3331    3333    3338*   3340    3342
                        3374    3378    3379
BIT0   = 000001          123#     235     251    1027    1201    1355    1366    1456    1474    2439    2468    2484    3126
                        3745    3778    3909    4051
BIT00  = 000001          113#     123
BIT01  = 000002          112#     122
BIT02  = 000004          111#     121
BIT03  = 000010          110#     120
BIT04  = 000020          109#     119
BIT05  = 000040          108#     118
BIT06  = 000100          107#     117
BIT07  = 000200          106#     116
BIT08  = 000400          105#     115    4000
BIT09  = 001000          104#     114    4010    4179
BIT1   = 000002          122#
BIT10  = 002000          103#    4151
BIT11  = 004000          102#    4017
BIT12  = 010000          101#
BIT13  = 020000          100#    3152    3179    3252    3289    4159
BIT14  = 040000           99#     217    3986
BIT15  = 100000           98#    2468    3204    3505
BIT2   = 000004          121#
BIT3   = 000010          120#
BIT4   = 000020          119#
BIT5   = 000040          118#
BIT6   = 000100          117#
BIT7   = 000200          116#
BIT8   = 000400          115#    3209    3538
BIT9   = 001000          114#
BLKMSK  001606           547#    3129    3200    3232    4037*   4042*
BPTVEC= 000014           130#
BYTMES  025345          1021    4951#
CKPMER  017544          1738    2400    3309    3737#
CLRPAR  020176          1121    1196    3782    3853#
CONST   026362          1444    5051#
CR     = 000015            38#    4615    4625
CRLF   = 000200            39#     903    4586    4625
CTRLS   005042          1261    1271#
DOISP  = 177570            45#     361     861
DF1     030640           716    5285#
DF14    030656           771     776     781     803     808     813    5291#
DF2     030644           721     731     736     741     746     751     792     818     823    5287#
DF21    030664           797    5293#
DF3     030651           726     756     761     766     787    5289#
DF30    030672           828     833    5295#
DH1     030273           714    5237#
DH12    030337           759     764    5244#
DH14    030360           76?     779    5247#
DH15    030401           774    5250#
DH2     030312           719     724     729     734     739     744     749     754     785     790    5240#
DH21    030434           795    5255#
DH23    030465           801    5260#
DH24    030525           806    5266#
```

```
DH25    030546          811    5269#
DH26    030577          816     821    5274#
DH30    030621          826    5278#
DIDBH   012720         2694#
DIDBL   012626         2645#
DIDO    012536         2596#
DIPDO   013016         2744#
DISPLA  001142          361#    861*    869*    878*   3572*   3625*   4032*   4150*
DISPRE  000174          195#    869
DM1     026714          713     773    5097#
DM10    027205          748    5129#
DM11    027251          753    5136#
DM12    027315          758    5143#
DM13    027371          763    5151#
DM14    027461          768    5161#
DM16    027560          778    5172#
DM17    027637          783    5180#
DM2     026750          718     723    5102#
DM20    027735          789    5191#
DM21    027767          794    5196#
DM23    030036          800    5203#
DM24    030103          805    5210#
DM25    030157          810    5218#
DM26    030177          815    5221#
DM27    030217          820    5224#
DM31    030253          830    5229#
DM4     027004          728    5107#
DM5     027042          733    5112#
DM6     027100          738    5117#
DM7     027144          743    5123#
DONE    013714         3019#
DPDBH   013202         2842#
DPDBL   013110         2793#
DSWR  = 177570           44#    360     860
DT1     001640          570#    715
DT12    001666          574#    760     765
DT14    001700          576#    770     780
DT15    001712          578#    775
DT2     001652          572#    720     725     730     735     740     745     750     755     786     791
DT21    001730          581#    796
DT23    001746          584#    802
DT24    001764          587#    807
DT25    001774          589#    812
DT26    002010          591#    817     822
DT30    002022          593#    827
DT31    002034          595#    832
EMTVEC= 000030          133#
ENDINS  021532         4036    4109    4122#
ERRTRP  025022          206     992    1100    3663   4867#
ERRVEC= 000004          126#    205     858     859*    870*    917*    927*    952*    992*   1015*   1025*   1085*   1100*
                       3566*   3619*   3659*   3663*   3991    3992*   3994*   3997*
FADMAP  001570          540#   1407*   1408*   3193    3195    3318    3320    3342    3344   4051*   4078    4079    4087
                       4089
FADMFS  026215         1360    5032#
FCSMSK  001566          538#   3346    4112*   4114*
FILL2   026711         4931    5091#
```

```
FLAGBK  001557            532#  3185*  3275*  3278*  3296*
FROM    025415           1034   3929   4959#
FSTADR  001562            535#  1343   1441*  3722   4043
GMPR    004042           1001   1081#
GMPRA   004070           1086#  1098
GMPRB   004104           1085   1092#
GMPRC   004112           1090   1094#
GMPRD   004152           1083   1106#
GNS   = ****** U          194    902
HT    = 000011             36#  4584   4625
IMPCHE= 177746            186#   928*
IMPCK   003254            916    917    926#
INITDN  014772           1511   1547   1574   1649   1659   3200#
INITEX  015104           3196   3198   3217   3219#
INITMM  014334           1494   1537   1585   1614   1623   1691   1705   1741   1762   1785   1810   1820   1863
                         1912   1923   1966   2015   2026   2074   2200   2211   2259   2391   2538   2599   2648
                         2697   2747   2796   2845   2889   2958   2971   2991   3004   3126#  3395
INSERT  021032           4034#
INSUFF  025432           1003   4962#
IOTVEC= 000020            131#
KIPAR0= 172340            166#   242    261*  3111*  3532*  3598*  4900
KIPAR1= 172342            167#   267*  3112*  3533*  3599*
KIPAR2= 172344            168#   963    983*  1060*  1169*  1173   2467   3113*  3132*  3138*  3149   3206*  3236*
                         3254   3327*  3359   3514*  3517*  3532   3593*  3787   3788*  3827*  3832*
KIPAR3= 172346            169#  1057   1060   1061*  3114*  3149*  3154*  3254*  3255*  3503*  3506*  3514   3533
                         3594*
KIPAR4= 172350            170#  3115*
KIPAR5= 172352            171#  3116*
KIPAR6= 172354            172#  3117*
KIPAR7= 172356            173#  3118*
KIPDR0= 172300            155#   237   3103*
KIPDR1= 172302            156#  3104*
KIPDR2= 172304            157#  3105*
KIPDR3= 172306            158#  3106*
KIPDR4= 172310            159#  3107*
KIPDR5= 172312            160#  3108*
KIPDR6= 172314            161#  3109*
KIPDR7= 172316            162#  3110*
LADMAP  001602            546#  1435*  1436*  1459*  1460*  3143   3145   3162   3164   3177   3186   3214   3216
                         3228   3230   3244   3246   3263   3265   3285   3295   3297   4085   4086   4087   4089
LADMES  026302           1387   5042#
LADMSK  001600            544#  3147   3166   3188   3248   3267   3287   3299   4117*  4119*
LDDISP  001522            508#   877*
LF    = 000012             37#  4619   4625
LMAD    001520            507#  3637   3648*  3655   3669*
LSTADR  001574            541#  1349*  1399*  1403   1414   1453*  1454*  1455*  1456*  1457   3218   4052   4059
MAMF    017444           1471   2529   3714#
MAMF1   017530           3728#  3730
MAMF2   017542           3715   3717   3723   3732#
MANUAL  005330           1107   1262   1329   1342#  1434
MANUL1  005366           1348   1355#
MANUL2  005770           1350   1451#
MAPMEM  003274            927    940#
MAPRB   004220           1126   1136#
MASK4K= 017777            183#   954    979   1053   1405   1457   3190   3301   3820   4076   4083   4112   4117
MEMMAP  001524            230    509#   941    942   1180   1182   1185   1380   1382   1421   1423   1426   1428
```

```
                        3510    3512    3521    3523    3543    3548    3790    3792
MEMMES  025327           950    4948#
MMAMES  025262           922    4941#
MMAVA   000606           233     290#    914*    920*    960     981    1012*   1018*   1055    1125    1167    1437    1451
                        1472    2464    3130    3202    3233    3325    3356    3501    3591    3640    3649    3785    3825
                        4889
MMDOWN  015602          1523    1559    1581    1655    1671    3212    3318#
MMINIT  014206           919    1017    1128    1476    3102#
MMORE   001554           528#   3210*   3211*   3219*   3304    3347
MMUP    015112          1507    1543    1599    1619    1634    1695    1716    1752    1775    1798    1814    1857    1900
                        1916    1960    2003    2019    2067    2187    2204    2252    2372    2528    2563    2613    2662
                        2712    2761    2810    2860    2940    2964    2984    2997    3017    3228#   3399
MMVEC = 000250           144#
MPAMEM  004160          1102    1121#
MPEND   002066           614#
MPPATS  002042           562     603#
MPRX    002270           560     694#    1081    1097    1101    1164    1236    1249    1260    1276    1320    1328    2384
                        3307    3714    3737
MPR0    002070           559     629#    1084    1138    1200    1271    1314    1323
MPR1    002100           633#
MPR10   002210           669#
MPR11   002220           673#
MPR12   002230           677#
MPR13   002240           681#
MPR14   002250           685#    1175    1241    1243    1244    1250    1253*
MPR15   002260           689#    1177    1245    1247    1248    1254    1257*
MPR2    002110           637#
MPR3    002120           641#
MPR4    002130           645#
MPR5    002140           649#
MPR6    002150           653#
MPR7    002160           657#
MPR8    002170           661#
MPR9    002200           665#
MTMAP   025370          1198    4955#
MTOE    026476          3697    5065#
MTR     025513          1104    4971#
MX1     025777          1222    5004#
MX2     026014          1230    5007#
MX3     026027          1206    5009#
MX4     026046          1212    5012#
MX5     026064          1218    5015#
MX6     026102          1259    5018#
NOMEM   026172          3901    5028#
NOMTST  026643          4055    5083#
NOPES   025614          3839    4983#
OEFLG   001560           533#   2411*   2521*
PARVEC= 000114           180#   2425*   2429*   2436*   3568*   3621*   3724*   3725*
PBTRP   012000           561    2436#
PESRV   017336           563    3677#
PEWNC   026604          3814    5077#
PHYAOR  015762          1495    1512    1538    1548    1575    1586    3354#
PIRQ  = 177772            43#
PIRQVE= 000240           137#
PMEMAP  001540           519#   1136*   1137*   1161*   1162*   1243*   1244*   1247*   1248*   2392    2394
PRELOC  026446          3484    5061#
```

```
PRGMAP  000602          253    254    274    289#   908*   909*  3024  3026  3030  3495  3527  3535*  3553
                       3579*  3588   3628*  3629*  4062   4063
PROREL  025666         4990#
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   017652          3703   3751   3756   3767#
PSW    = 177776           41#    896   1227   1364   1391   1448  3071  3940  3948  3982  4105  4126  4145
                       4177   4211   4244   4251   4264   4310  4409  4495  4541
PWRMSG  025547           326   4976#
PWRVEC= 000024           132#   295*   296*   305*   311*   323*  324*  853*  854*  3567*  3620*
RADTAB  001614           556#  3573   3612
RANTST  012416          2536#
RELOC   016244          3459#  3529   3558   3595   3605
RELOCF  000600           269    288#   910*  1470   3211   3561*  3565*  3574  3582  3603  3627*  3718  4044
                       4215   4236   4238   4538   4742
RELTOP  016366          3036   3495#
RELO    016770           276   3039   3588#
RESCHK  005214          1277   1314#
RESLDR  017176           282   3042   3637#
RESRVD  001516           505#   558   1278*  1282   1287   1292  1300  2469*  2470  2471  2500
RESTAR  000300           201    226#   328    907
RESTOR  000304           202    228#
REST1   000306           227    229#
REST2   000324           231    233#
RESVEC= 000010           127#
ROTATE  016116          1764   1787   3404#
RW     = 000006           176#  3103   3104   3105   3106   3110
SAVLDR  017256           911   3086   3655#
SAVTST  001534           515#   996*   997*  1367*  1368*  1409*  1410*  3024  3026  4060  4061
SCANM   026542          3776   5071#
SELECT  002640           199    843#
SELFLG  001556           531#   841*   843*  1347
SETAE   017510          2399   3719   3721   3724#
SETCON  016076          1761   1784   2390   3394#
SKPMES  026667          4099   5087#
SPRNT   020220          1284   1302   1335   2407   3869#
SPRNTA  020306          3879   3883   3888#
SPRNTB  020312          3871   3889#
SPRNTP  020244          2454   2475   2488   2505   3877#
SPRNTQ  020232          3700   3747   3753   3805   3873#
SPRNTO  020250          1295   1517   1553   1665   2430   2446  2514  3875  3878#
SPRNT1  020256          1628   3881#
SPRNT2  020274          1500   1592   1710   1746   1769   1792  1826  1833  1840  1847  1870  1879  1888
                        1929   1936   1943   1950   1973   1982  1991  2032  2039  2046  2053  2080  2089
                        2098   2105   2114   2123   2130   2139  2148  2155  2164  2173  2217  2224  2231
                        2238   2265   2274   2283   2290   2299  2308  2315  2324  2333  2340  2349  2358
                        2553   2898   2907   2929   2977   3010  3886#
SPRNT3  020270          2606   2655   2705   2754   2803   2853  3885#
SR0    = 177572           148#   235    251*   918*  1016*  1474  3119*  3642*  3651*  4891
```

```
SR1     = 177574          149#
SR2     = 177576          150#
SR3     = 172516          151#
STACK   = 001100           31#   229    270    272    557    851    926    940   1011
START   = 002632          198    841#
STARTA  = 002646          232    284    842    844#
START1  = 006046          281   1458   1468#  1469   3037   3088
STKLMT= 177774             42#
SWR     = 001140          303    316*   360#   849    860*   862    868*   875*   892    915   1013   1082   1720
                         1722   2386   3028   3569   3571*  3622   3624*  3716   3720   3739   3986   4000   4002
                         4010   4017   4046   4151   4159   4172   4179   4291   4336*
SWREG   = 000176          196#   868    832   4291   4307
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   4046
SW06    = 000100           79#    89   1082   2386
SW07    = 000200           78#    88   3028
SW08    = 000400           77#    87   1720
SW09    = 001000           76#    86
SW1     = 000002           94#
SW10    = 002000           75#
SW11    = 004000           74#
SW12    = 010000           73#   915   1013
SW13    = 020000           72#
SW14    = 040000           71#
SW15    = 100000           70#
SW2     = 000004           93#
SW3     = 000010           92#
SW4     = 000020           91#
SW5     = 000040           90#  3720
SW6     = 000100           89#  3716   3739
SW7     = 000200           88#
SW8     = 000400           87#
SW9     = 001000           86#
TBITVE= 000014            128#
TIMOUT  = 003572          203   1011#  1073
TKVEC   = 000060          135#
TMAP    = 004532         1172   1196#
TMPFAD  = 001564          537#  3197   3322   4043*  4048   4050*  4072   4074*  4075*  4076   4091   4113
TMPLAD  = 001576          543#  3148*  3167*  3249*  3268*  3271   4059*  4080   4082*  4083   4091   4118
TMPT    = 001550          525#  3150*  3151*  3155*  3156*  3158   3160   3162   3164   3176*  3181*  3183   3186
                         3191*  3192*  3228   3230   3250*  3251*  3256*  3257*  3259   3261   3263   3265   3288*
                         3291*  3293   3302*  3303*
TO      = 025425         1044   1066   3922   4961#
TPVEC   = 000064          136#
TRAPVE= 000034            134#
TRTVEC= 000014            129#
TSTMAP  = 001530          512#  3134   3136   3158   3160   3170   3183   3240   3242   3259   3261   3283   3293
                         3331   3333   3340   4060*  4061*  4062*  4063*  4066   4068*  4069*  4078*  4079*  4085*
                         4086*  4093   4095
TST1    = 006114         1473   1475   1487#
TST10   = 006766         1721   1725   1734#
```

```
TST11   007050          1740    1757#
TST12   007134          1780#
TST13   007216          1802#
TST14   007542          1806    1904#
TST15   010072          1908    2007#
TST16   010714          2011    2192#
TST17   011536          2196    2381#
TST2    006240          1533#
TST20   012410          2388    2533#
TST21   012524          2590#
TST22   012614          2594    2639#
TST23   012706          2643    2688#
TST24   013004          2692    2738#
TST25   013076          2742    2787#
TST26   013170          2791    2836#
TST27   013264          2840    2886#
TST3    005334          1570#
TST30   013470          2955#
TST31   013602          2988#
TST32   013722           566    1491    3022#
TST4    006440          1610#
TST5    006530          1645#
TST6    006624          1687#   1727
TST6A   006632          1690#   1728
TST7    006654          1701#   1727
TYPMAP  020320           993    1234    3896#
UNEXPT  026405          3679    5055#
UP    = 000000           175#   3103    3104    3105    3106    3110
UUP     001612           549#   1141    1145    1179*   2415    2422
UUPBYT  011600          2392#
UUPB0   011544          2384#
UUPB1   011640          2401#   2527
UUPB2   011704          2413#   2523
UUPB3   012254          2483    2492#
UUPB4   012344          2433    2449    2518#
UUPB5   012370          2398    2404    2525#
U3X9    016164          1811    1913    2016    2201    3427#
$APTHD  001330           456     462#
$APTR   001512           499#   4670    4691
$ASTAT  001344           472#    499    4659    4663    4689    4692    4696    4702
$ASTEN  001510           498#
$ATYC   023756          4630    4632#
$ATY1   023732          4628#
$ATY3   023740          4574    4629#
$ATY4   023750          4168    4631#
$ATY6   024246          4689#
$ATY7   024312          4700#
$AUTOB  001134           357#    899*   4299    4473
$BASE   001260           425#
$BOOR   001122           352#    584     587     589    3475*   3677#
$BOOAT  001126           354#    570     572     578     581     584     593    3473*   3889*
$BELL   001174           376#   4154    4191
$COM1   001264           427#
$COM2   001266           428#
$CHKRC  023726          4591#   4601#   4608    4617*   4622#
$CKSWR  022272          3983    4127    4146    4178    4291#
```

```
$CMTAG  001100          340#   846   847   855
$CM3  = 000000          370#
$CM4  = 000004          370#   371#  372#  373#  374#
$CNTLG  023217         4303   4468#
$CNTLU  023212         4327   4436  4467#
$CPUOP  001232          399#
$CRLF   001201          378#   995  1139  3045  4163  4191  4201  4226  4232  4277  4340  4442  4467
                       4531   4590  4625
$DBLK   024562         4743   4779  4787#
$DDW0   001270          429#
$DDW1   001272          430#
$DDW10  001314          439#
$DDW11  001316          440#
$DDW12  001320          441#
$DDW13  001322          442#
$DDW14  001324          443#
$DDW15  001326          444#
$DDW2   001274          431#
$DDW3   001276          432#
$DDW4   001300          433#
$DDW5   001302          434#
$DDW6   001304          435#
$DDW7   001306          436#
$DDW8   001310          437#
$DDW9   001312          438#
$DEVCT  001214          390#
$DEVM   001262          426#
$DOAGN  014162         3062   3078  3085  3087#
$DTBL   024552         4747   4783#
$ENDAD  014136          215    883  3080#  4186
$ENDCT  014072          855   3064#
$ENDMG  014166         3067   3089#
$ENULL  014203         3074   3092#
$ENV    001224          395#   890  3034  4165  4569  4637  4676
$ENVM   001225          396#   873  4571  4576  4639
$EOP    014042         3027   3029  3055#
$EOPCT  014064          855*  3061#  3065
$ERFLG  001103          343#  3970  4006  4008  4014*  4124  4148*  4191
$ERMAX  001115          349#  4008  4031*  4124
$ERROR  021546         1285   1296  1303  1336  1501  1518  1554  1593  1629  1666  1711  1747  1770
                       1793   1827  1834  1841  1648  1871  1880  1889  1930  1937  1944  1951  1974
                       1983   1992  2033  2040  2047  2054  2081  2090  2099  2106  2115  2124  2131
                       2140   2149  2156  2165  2174  2218  2225  2232  2239  2266  2275  2284  2291
                       2300   2309  2316  2325  2334  2341  2350  2359  2408  2431  2447  2455  2476
                       2489   2506  2515  2554  2607  2656  2706  2755  2804  2854  2899  2908  2930
                       2978   3011  3476  3688  3701  3748  3754  3806  4142#  4872
$ERRPC  001116          350#   570   572   574   576   578   581   584   587   589   591  4156*  4157*
                       4158   4191  4207  4214
$ERRTB  002332          564    711#
$ERRTY  021772         4161   4199#
$ERTTL  001112          347#  3802*  4155*  4191
$ESCAP  001172          375#  4030*  4182  4184  4191
$ETABL  001224          394#
$ETEND  001330          445#   468
$FATAL  001206          387#  4680*
$FFLG   024336         4628*  4631*  4674  4683*  4712#
```

```
$FILLC  001156          368#  4594  4625
$FILLS  001155          367#  4626
$GDAOR  001120          351#   570   572   574   576   578   581   584   593  3474*  3869*  3878*  3881*
                       3882*  3886* 3887*
$GDDAT  001124          353#   570   572   574   578   581   584  3472*  3870*  3888*
$GET42  014126         3075#
$GTSWR  022344          897   4304#
$HD   = 000000           12
$HIBTS  001330          463#
$HIOCT  023416         1371   1384  1395  1398*  1400  1411  4518*  4530#
$ICNT   001104          344#  4021* 4022  4024*  4123
$ILLUP  000756          295    311   330#
$INTAG  001135          358#  4341  4473
$ITEMB  001114          348#  4158* 4167  4191   4204
$LF     001202          379#  4191  4457  4467   4531  4625
$LFLG   024335         4659   4684* 4711#
$LPAOR  001106          345#  1469* 1470* 1727*  4012* 4028* 4033  4111*  4123
$LPERR  001110          346#  4012  4029* 4123   4181
$MADR1  001236          412#
$MADR2  001242          416#
$MADR3  001246          419#
$MADR4  001252          422#
$MAIL   001204          385#   464   468   872    890   4027  4165  4569
$MAMS1  001234          406#
$MAMS2  001240          414#
$MAMS3  001244          417#
$MAMS4  001250          420#
$MBAOR  001332          464#
$MFLG   024334         4629*  4635  4685* 4710#
$MNEW   023235         4313   4471#
$MSGAD  001220          392#  4645* 4648  4696*
$MSGLG  001222          393#  4650* 4695*
$MSGTY  001204          386#  4643  4651* 4678   4682* 4693  4697*
$MSWR   023224         4306   4469#
$MTYP1  001235          407#
$MTYP2  001241          415#
$MTYP3  001245          418#
$MTYP4  001251          421#
$MXCNT  021534         4025   4123#
$NULL   001154          366#  4596  4625
$NWTST= 000001         1478#  1480  1524#  1526  1561#  1563  1601#  1603  1636#  1638  1677#  1679  1680
                       1697#  1699  1730#  1732  1754#  1777# 1799#  1901# 2004#  2189# 2373#  2375  2530#
                       2565#  2567  2614#  2616  2663#  2665  2713#  2715  2762#  2764  2811#  2813  2862#
                       2864   2942# 2944   2985#
$OCNT   025016         4820*  4850* 4863#
$OMODE  025020         4815*  4819* 4824   4827*  4838*  4865#
$OVER   021020         3987   4005  4013   4023   4032#
$PASS   001212          389#   872* 3058*  3059*  3068   3089  4019  4038  4064  4124
$PASTM  001336          466#
$PRINT  023420          325    885   921    949    994   1002  1020  1033  1043  1065  1103  1197  1205
                       1211   1217  1221   1229   1238   1258  1359  1386  1431  1443  3044  3066  3073
                       3483   3678  3696   3775   3813   3838  3900  3921  3928  4054  4098  4153  4162
                       4200   4223  4225   4229   4231   4270  4276  4302  4305  4312  4326  4339  4360
                       4417   4424  4430   4435   4441   4443  4447  4451  4456  4525  4527  4537#  4589
                       4778   4848  4920   4930
$PWRAD  000752          328#
```

```
$PWRDN  000610          295#   323    853
$PWRMG  000746          326#
$PWRUP  000662          305    311#
$QUES   001200          377#  4191   4361   4448   4467   4528   4531   4625
$RDCHR  022576         4374#  4410
$RDLIN  022716         4402#  4496
$RDOCT  023246         1365   1392   1449   4487#
$RDSZ = 000010         4395#
$SAVR6  000762          304*   312    313*   314*   332#
$SCOPE  020536         1488   1534   1571   1611   1646   1688   1702   1735   1758   1781   1803   1905   2008
                       2193   2382   2534   2591   2640   2689   2739   2788   2837   2887   2956   2989   3020
                       3979#
$SETUP= 000130          188#   852    853    855    856    883    887   3057   3980   4143   4175   4186   4286
                       4473
$STUP = 177777          188#
$SVLAD  020764         3995   4026#
$SVPC = 000010          213#   218
$SWR  = 167400            1#    12     17     18     19     20     21     22     23     24    329    374    375
                        376    856   1493   1536   1573   1613   1648   1690   1704   1737   1760   1783   1808
                       1910   2013   2198   2384   2536   2596   2645   2694   2744   2793   2842   2889   2958
                       2991   3052   3057   3076   3088   3089   3971   3972   3973   3974   3975   3986   3998
                       4000   4001   4006   4007   4008   4015   4016   4017   4029   4032   4123   4134   4135
                       4136   4137   4138   4151   4159   4172   4179   4191
$SWREG  001226          397#   875
$SWRMK= 000340            1#    24     25   3975   3976   4002   4003
$TESTN  001210          388#  4027*
$TIMES  001170          279*   374#  3022*  3057*  4015*  4022   4025*  4123
$TKB    001146          363#  4284   4295   4319   4378   4384
$TKS    001144          362#  4284   4293   4316   4343*  4376   4382
$TMP0   001160          370#   576    578    581    589    591    593   3355*  3365*  3367*  3873*  3877*  3885*
$TMP1   001162          371#   589    591    593   3874*
$TMP2   001164          372#   958*   959*   962*   973*  1349   1397
$TMP3   001166          373#   595    948*   964*   970*   972*   974*  1398   4871*
$TN   = 000032            1#    12   1478   1493#  1524   1536#  1561   1573#  1601   1613#  1636   1648#  1677
                       1690#  1697   1699   1704#  1717   1730   1737#  1740   1754   1760#  1777   1783#  1799
                       1802   1808#  1901   1904   1910#  2004   2007   2013#  2189   2192   2198#  2373   2384#
                       2530   2536#  2565   2590   2596#  2614   2639   2645#  2663   2688   2694#  2713   2738
                       2744#  2762   2787   2793#  2811   2836   2842#  2862   2889#  2942   2958#  2985   2991#
$TPB    001152          365#  4614*  4625
$TPFLG  001157          369#  4563   4625
$TPS    001150          364#  4612   4625
$TSTM   001334          465#
$TSTNM  001102          280*   342#  1726*  3023*  3970   4004   4026*  4027   4032   4040   4102   4124   4150
                       4191
$TTYIN  023202         4404   4405   4421   4444   4462   4466#
$TYPAD  025060         1037   1048   1070   3487   4257   4882#
$TYPDS  024340         3072   4252   4731#
$TYPE   023446         4542   4563#  4656
$TYPEC  023662         4345   4593   4600   4607   4612#  4613
$TYPEX  023730         4618   4620   4623#
$TYPOC  024616         1228   4212   4245   4311   4818#
$TYPON  024632         4817   4820#
$TYPOS  024572         3941   3949   4106   4265   4813#
$UNIT   001216          391#
$UNITM  001340          467#
$USWR   001230          398#
```

```
$VECT1  001254            423#
$VECT2  001256            424#
$VERPC  001514            504#   572    574    576    578    581    584    587    589    591   4214#  4215#
$XTSTR  020562           3989#
$OFILL  025017           4814#  4818#  4828   4864#
$40CAT= ****** U         3986   4161
.     = 032110            190#   194#   205#   213    214#   216#   218#   219#   307    331    339#   380    452
                          453#   455#   457#   534#   694#   850    903#   905   3089   3093   4123   4124   4191
                         4284   4466#  4467   4473   4531   4625   4713#  4787#  5299#
.CONST  001610            548#  1450#  1690   1704
.EIGHT  001634            565#  4764
.ERRTB  001632            564#  4220
.MPPAT  001626            562#  1737
.MPRX   001622            560#  2414   2421   2444   2482   3683   3727   3742   3797   3855
.MPRO   001620            559#  2438
.PBTRP  001624            561#  2425
.PESRV  001630            563#  2429   2436   3724
.RESRV  001616            558#
.STACK  001614            557#  1468
.TST32  001636            566#  4057
.$ASTA= ****** U         4629   4632
.$X   = 001330            452#   457
.3X9    007230           1808#
```

F16


```
ABORT      1#   260  3142  3175  3213  3385  3478  3497  3555  3590  3639
CKSWR      1#  3980  4124  4143  4175
CKWD       1#  1293  1498  1515  1590  1626  1663  1708  1744  1767  1790  1824  1831  1838  1845
        1868  1877  1886  1927  1934  1941  1948  1971  1980  1989  2030  2037  2044  2051  2078
        2087  2096  2103  2112  2121  2128  2137  2146  2153  2162  2171  2215  2222  2229  2236
        2263  2272  2281  2288  2297  2306  2313  2322  2331  2338  2347  2356  2473  2503  2551
        2604  2653  2703  2752  2801  2851  2896  2905  2927  2975  3008
CKWD2      1#  1497  1588  1706  1743  1822  1830  1837  1844  1866  1876  1885  1925  1933  1940
        1947  1969  1979  1988  2028  2036  2043  2050  2076  2086  2095  2102  2111  2120  2127
        2136  2145  2152  2161  2170  2213  2221  2228  2235  2261  2271  2280  2287  2296  2305
        2312  2321  2330  2337  2346  2355  2550  2973  3006
COMMEN     1#  138#   835  1462
ENDCOM     1#    12  138#   839  1466
ERROR     32#
ESCAPE   138#
GETPRI   138#
GETSWR   138#  887#
GTSWR      1#  894
LDPDR      1#  3102  3104  3105  3106  3110
MORETA   333#  447
MULT     138#
NEWTST     1#  138#  1478  1524  1561  1601  1636  1677  1697  1730  1754  1777  1799  1901  2004
        2189  2373  2530  2565  2614  2663  2713  2762  2811  2862  2942  2985
POP        1#  138#   316   317  2498  2511  3369  3386  3488  3580  3618  3705  3731  3759  3822
        3832  3840  3860  3952  3959  4519  4686  4687  4698  4707  4773  4925
PRINT      1#   921   949   994  1002  1019  1033  1043  1065  1103  1197  1205  1211  1217  1220
        1229  1359  1385  1431  1442  3043  3483  3678  3696  3775  3813  3838  3900  4054  4097
        4920  4930
PUSH       1#  138#   297   303  2457  2480  3358  3376  3459  3498  3515  3611  3681  3726  3741
        3767  3787  3794  3853  3904  3930  4489  4632  4634  4655  4689  4700  4731  4882
RDCHR      1#  4407
RDDEC      1#
RDLIN      1#  4492
RDOCT      1#  1362  1389  1446
REPORT     1#  138#
RESREG     1#
SAVREG     1#
SCOPE     33#
SCOPEX   3965#  4034
SCOPIN   3965#  3984
SETPRI   138#
SETUP      1#  138#   844
SIMTRP     1#   894  1225  1362  1389  1446  3069  3938  3946  3980  4103  4124  4143  4175  4209
        4242  4249  4262  4308  4407  4493  4539
SKIP     138#  1740
SLASH      1#  138#   618   628
SPACE    138#
STARS      1#  138#   211   220   225   293   309   335   380   383   449   451   458   471   501
         503   551   555   567   569   599   601   930   939  1007  1010  1076  1079  1110  1119
        1130  1134  1191  1194  1265  1269  1309  1312  1352  1354  1478  1486  1524  1532  1561
        1569  1601  1609  1636  1644  1673  1676  1677  1686  1697  1700  1730  1733  1754  1756
        1777  1779  1799  1801  1816  1818  1859  1861  1901  1903  1919  1921  1962  1964  2004
        2006  2021  2023  2069  2071  2189  2191  2206  2208  2254  2256  2373  2380  2530  2532
        2565  2589  2614  2638  2663  2687  2713  2737  2762  2786  2811  2835  2862  2885  2942
        2954  2985  2987  3046  3095  3101  3123  3125  3222  3227  3312  3317  3350  3353  3372
        3374  3391  3393  3401  3403  3424  3426  3456  3458  3492  3494  3585  3587  3632  3636
```

```
            3672    3676    3710    3713    3734    3736    3762    3766    3850    3852    3865    3868    3892    3895    3967
            4130    4191    4283    4286    4366    4395    4475    4532    4536    4548    4627    4718    4790    4937    4940
            5094    5096    5233    5235    5282    5284
$HRSU        138#     856#
$HROR3         1#    2057    2062    2177    2182    2242    2247    2362    2367    3451
$IREGS      1478#    1480    1526    1563    1603    1638    1680
TYPAOR         1#    1036    1047    1069    3486    4256
TYPBIN       138#
TYPBN          1#
TYPBYT         1#    4101    4259
TYPDEC       138#    3068    4248
TYPDS          1#    3069    4249
TYPE           1#     325     885     921     949     994    1002    1020    1033    1043    1065    1103    1197    1205    1211
            1217    1221    1229    1359    1386    1431    1443    3044    3066    3073    3483    3678    3696    3775    3813
            3838    3900    3921    3928    4054    4098    4153    4162    4200    4223    4225    4229    4231    4270    4276
            4302    4304    4312    4326    4338    4359    4417    4424    4430    4435    4441    4443    4446    4451    4456
            4525    4527    4589    4778    4848    4920    4930
TYPNAM         1#     138#    879
TYPNUM       138#
TYPOC          1#    1225    4209    4242    4308
TYPOCS       138#    3936    3944
TYPOCT       138#    1224    4207    4241    4307
TYPON          1#
TYPOS          1#    3938    3946    4103    4262
TYPTXT       138#
$CKWD          1#    1294    1334    1499    1516    1552    1591    1627    1664    1709    1745    1768    1791    1825    1832
            1839    1846    1869    1878    1887    1928    1935    1942    1949    1972    1981    1990    2031    2038    2045
            2052    2079    2088    2097    2104    2113    2122    2129    2138    2147    2154    2163    2172    2216    2223
            2230    2237    2264    2273    2282    2289    2298    2307    2314    2323    2332    2339    2348    2357    2406
            2474    2504    2513    2552    2605    2654    2704    2753    2802    2852    2897    2906    2928    2976    3009
$INDN          1#    1511    1547    1574    1649    1659
$INMM          1#    1494    1537    1585    1614    1623    1691    1705    1741    1762    1785    1810    1820    1863    1912
            1923    1966    2015    2026    2074    2200    2211    2259    2391    2538    2599    2648    2697    2747    2796
            2845    2889    2958    2971    2991    3004    3394
$MMDN          1#    1521    1557    1579    1653    1669
$MMUP          1#    1505    1541    1597    1617    1632    1693    1714    1750    1773    1796    1812    1855    1898    1914
            1958    2001    2017    2065    2185    2202    2250    2370    2560    2611    2660    2710    2759    2808    2858
            2962    2982    2995    3015    3397
$$CKWD         1#    1284    1295    1302    1335    1500    1517    1553    1592    1628    1665    1710    1746    1769    1792
            1826    1833    1840    1847    1870    1879    1888    1929    1936    1943    1950    1973    1982    1991    2032
            2039    2046    2053    2080    2089    2098    2105    2114    2123    2130    2139    2148    2155    2164    2173
            2217    2224    2231    2238    2265    2274    2283    2290    2299    2308    2315    2324    2333    2340    2349
            2358    2407    2430    2446    2454    2475    2488    2505    2514    2553    2606    2655    2705    2754    2803
            2853    2898    2907    2929    2977    3010    3699    3747    3752    3804
$$CMRE       333#
$$CMTM       333#     370     371     372     373
$$ESCA       138#
$$NEWT       138#    1478    1524    1561    1601    1636    1677    1697    1730    1754    1777    1799    1901    2004    2189
            2373    2530    2565    2614    2663    2713    2762    2811    2862    2942    2985
$$SETM       872#
$$SKIP       138#    1740
.CHBHD      2942#    2944
.C11        2373#    2375
.DIDBH      2663#    2665
.DIDBL      2614#    2616
.DIDO       2565#    2567
```

```
.DIPDO   2713#  2715
.DPDBH   2811#  2813
.DPDBL   2762#  2764
.EQUAT      1#    28
.ERROR      1#  1285  1296  1303  1336  1501  1518  1554  1593  1629  1666  1711  1747  1770  1793
          1827  1834  1841  1848  1871  1880  1889  1930  1937  1944  1951  1974  1983  1992  2033
          2040  2047  2054  2081  2090  2099  2106  2115  2124  2131  2140  2149  2156  2165  2174
          2218  2225  2232  2239  2266  2275  2284  2291  2300  2309  2316  2325  2334  2341  2350
          2359  2408  2431  2447  2455  2476  2489  2506  2515  2554  2607  2656  2706  2755  2804
          2854  2899  2908  2930  2978  3011  3476  3688  3701  3748  3754  3806  4872
.HEADE      1#     2
.KT11       1#   140
.MARHD   2 62#  2864
.SCOPE      1#  1487  1533  1570  1610  1645  1687  1701  1734  1757  1780  1802  1904  2007  2192
          2381  2533  2590  2639  2688  2738  2787  2836  2886  2955  2988  3019
.SETUP      1#   188
.SWRHI      1#    13
.SWRLO      1#    25#
.TM7     1697#  1699
.$ACT1      1#   209
.$APTB      1#   381#
.$APTH      1#   447
.$APTY      1#  4625
.$ASTA      1#   469
.$CATC      1#   188
.$CMTA      1#   333
.$EOP       1#  3046
.$ERRO      1#  4128
.$ERRT      1#  4191
.$POWE      1#   291
.$RDOC      1#  4473
.$READ      1#  4281
.$SCOP      1#  3965
.$TYPD      1#  4718
.$TYPE      1#  4546
.$TYPO      1#  4788

. ABS.  032110     000

 ERRORS DETECTED:  0

 DZQMCD,DZQMCD/SOL/CRF/NL:TOC=DZQMCD.P11
 RUN-TIME:  30 19 1 SECONDS
 RUN-TIME RATIO: 552/52=10.5
 CORE USED:  31K  (61 PAGES)
```