

# MS11K

0-124K MEMORY DIAGNOSTIC  
MD-11-DZQMC-D

EP-DZQMC-D-DL-D  
COPYRIGHT © 75-77  
FICHE 1 OF 1

OCT 1977  
**digital**  
MADE IN USA

This image displays a grid of 124 small diagnostic test patterns, arranged in approximately 10 rows and 12 columns. Each pattern is a square frame containing binary data (0s and 1s) and graphical waveforms, likely representing memory addresses and their corresponding data or error status. The patterns are organized into several distinct sections, with some containing more complex waveforms and others showing simpler binary sequences. The overall layout is a dense array of these diagnostic elements, used for testing memory integrity.

801

EOF10200005801

00010000 770920  
=====

IDENTIFICATION11

HDR1DZQMC0SEQ

00010000

770920  
SEQ 0001

PRODUCT CODE: MAINDEC-11-DZQMC-D-D  
PRODUCT NAME: 0-124K MEMORY EXERCISER, 16K VERSION.  
DATE: SEPTEMBER, 1977  
MAINTAINER: DIAGNOSTIC ENGINEERING, 32J

The information in this document is subject to change without notice and should not be construed as a commitment by digital equipment corporation. digital equipment corporation assumes no responsibility for any errors that may appear in this document.

The software in this document is furnished under a license and may only be used or copied in accordance with the terms of such license.

Digital equipment corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by digital.

Copyright (C) 1975, 1977 by Digital Equipment Corporation

REVISION HISTORY  
=====

Revision A:	May 1975
Revision B:	October 1975
Revision C:	October 1976
Revision D:	June 1977

TABLE OF CONTENTS

- 1.0 GENERAL PROGRAM INFORMATION.
  - 1.1 Program Purpose (Abstract)
  - 1.2 System Requirements
  - 1.3 Related Documents and Standards
  - 1.4 Diagnostic Hierarchy Prerequisites
  - 1.5 Assumptions
  
- 2.0 OPERATING INSTRUCTIONS
  - 2.1 Loading and Starting Procedure
  - 2.2 Special Environments
  - 2.3 Program Options
  - 2.4 Execution Times
  
- 3.0 ERROR INFORMATION
  - 3.1 Error Reporting
  - 3.2 Error Halts
  
- 4.0 PERFORMANCE AND PROGRESS REPORTS
  
- 5.0 DEVICE INFORMATION TABLES
  - 5.1 CORE PARITY REGISTER
  - 5.2 MOS PARITY REGISTER
  - 5.3 MSII-K CSR
  
- 6.0 SUB-TEST SUMMARIES
  - 6.1 Section 1: Address Tests
  - 6.2 Section 2: Worst Case Noise Tests
  - 6.3 Section 3: Instruction Execution Tests
  - 6.4 Section 4: MOS Tests
  - 6.5 Special Toggle in Tests
  
- 7.0 PROGRAM FUNCTIONAL FLOW CHARTS
  
- 8.0 PROGRAM LISTING

1.0 GENERAL PROGRAM INFORMATION.

1.1 Program Purpose (Abstract)

This program has the ability to test memory from address 000000 to address 757777. It does so using:

- A. Unique addressing techniques
- B. Worst 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. PDP11-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 SW05=1, then:

Load Address 214,  
Press START.

B. If running without memory management, then:

Load Address <214+relocation factor>  
(Relocation factor is typed when the program is relocated),  
Press START.

C. If running with memory management and the unibus has not been initialized (via reset instruction, start switch, etc.), then:

Load Address 777707 (PC)  
Deposit 214  
Press CONTINUE

D. If running with memory management and the unibus has been initialized:

Load Address 772340 (KIPARD)  
Deposit <(relocation factor)/100>  
(Example: Relocation factor=540000, then deposit 005400)  
Load Address 777572 (SRO)  
Deposit 000001  
Load Address 777707 (PC)  
Deposit 214  
Press Continue

2.1.6 Starting address 220:

Byte address memory map typeout routine. This routine

performs DATI, DATIP, DATO, and DATOB on all possible addresses, and types the ranges of addresses which do not cause a timeout trap.

## 2.2 Special Environments

If the program is run in quick verify mode under ACT11 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 printer 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 BK 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.

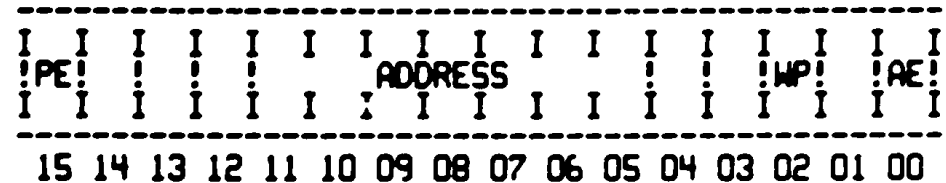
4.0 PERFORMANCE AND PROGRESS REPORTS

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



BIT03	INHIBIT MODE POINTER	<p>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 INHIBITED FROM OPERATING IN THE ECC DISABLE MODE OR DIAGNOSTIC CHECK MODE. THE INHIBIT MODE POINTER INDICATES WHICH 16K IS BEING INHIBITED,,,BIT 3 =1</p> <p>THE SECOND 16K OF MEMORY IS INHIBITED. WHEN BIT 13 IS SET TO A 0, BIT 3 BECOMES INOPERATIVE.</p>
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 01

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	000001	MOV R0,R1	:SAVE IN R1
16	000037	15: CMP R0,#50	:CHECK UPPER LIMIT :(IN SWITCH REGISTER)
20	177570		
22	001403	BEG 25	: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 15	:LOOP UNTIL DONE
32	010004	25: MOV R0,R4	:SAVE UPPER LIMIT
34	020001	35: CMP R0,R1	:CHECK IF AT LOWER LIMIT
* 36	001767	BEG 15	:BRANCH IF DONE
40	024000	CMP -(R0),R0	:CHECK DATA WRITTEN
42	001774	BEG 35	:BRANCH IF OK
44	000000	HALT	:ERROR
46	000772	BR 35	: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 45	:BRANCH TO PROGRAM #2

These are the additions to toggle-in-program #1:

LOCATION	CONTENTS	MNEMONIC	COMMENT
50	010402	45: MOV R4, R2	:GET UPPER LIMIT
52	005142	55: COM -(R2)	:COMPLEMENT ADDRESS
54	020201	CMP R2, R1	:CHECK IF AT LOWER LIMIT
56	001375	BNE 55	:LOOP UNTIL DONE
60	020204	65: CMP R2, R4	:CHECK IF AT UPPER LIMIT
62	001755	BEQ 15	: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 65	:BRANCH IF OK
74	000000	HALT	:ERROR
76	000770	BR 65	: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)

\*\*\*\*\*

COPYRIGHT 1977  
DIGITAL EQUIPMENT CORPORATION



TABLE OF CONTENTS  
\*\*\*\*\*

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

TABLE OF CONTENTS  
\*\*\*\*\*

PAGE 33	TEST 16: 3 XOR 9 (FOR PARITY)
PAGE 35	TEST 17: PARITY BYTE TEST
PAGE 39	TEST 20
PAGE 40	TEST 21: EXICUTE DATI, DATO
PAGE 41	TEST 22: EXICUTE DATI, DATOB (LO BYTE)
PAGE 42	TEST 23: EXICUTE DATI, DATOB (HI BYTE)
PAGE 43	TEST 24: EXICUTE DATIP, DATO
PAGE 44	TEST 25: EXICUTE DATIP, DATOB (LO BYTE)
PAGE 45	TEST 26: EXICUTE DATIP, DATOB (HI BYTE)
PAGE 46	TEST 27: MARCHING 1'S AND 0'S
PAGE 49	TEST 30: MOS REFRESH TEST
PAGE 51	TEST 31: MOS REFRESH TEST
PAGE 53	DONE
PAGE 54	END OF PASS
PAGE 55	MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES
PAGE 57	SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS
PAGE 58	RELOCATION SUBROUTINES
PAGE 60	PARITY ROUTINES
PAGE 62	SPECIAL PRINTOUT ROUTINES

F02

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

DECFL0 VER 00.12 08-SEP-77 10:00 PAGE 01

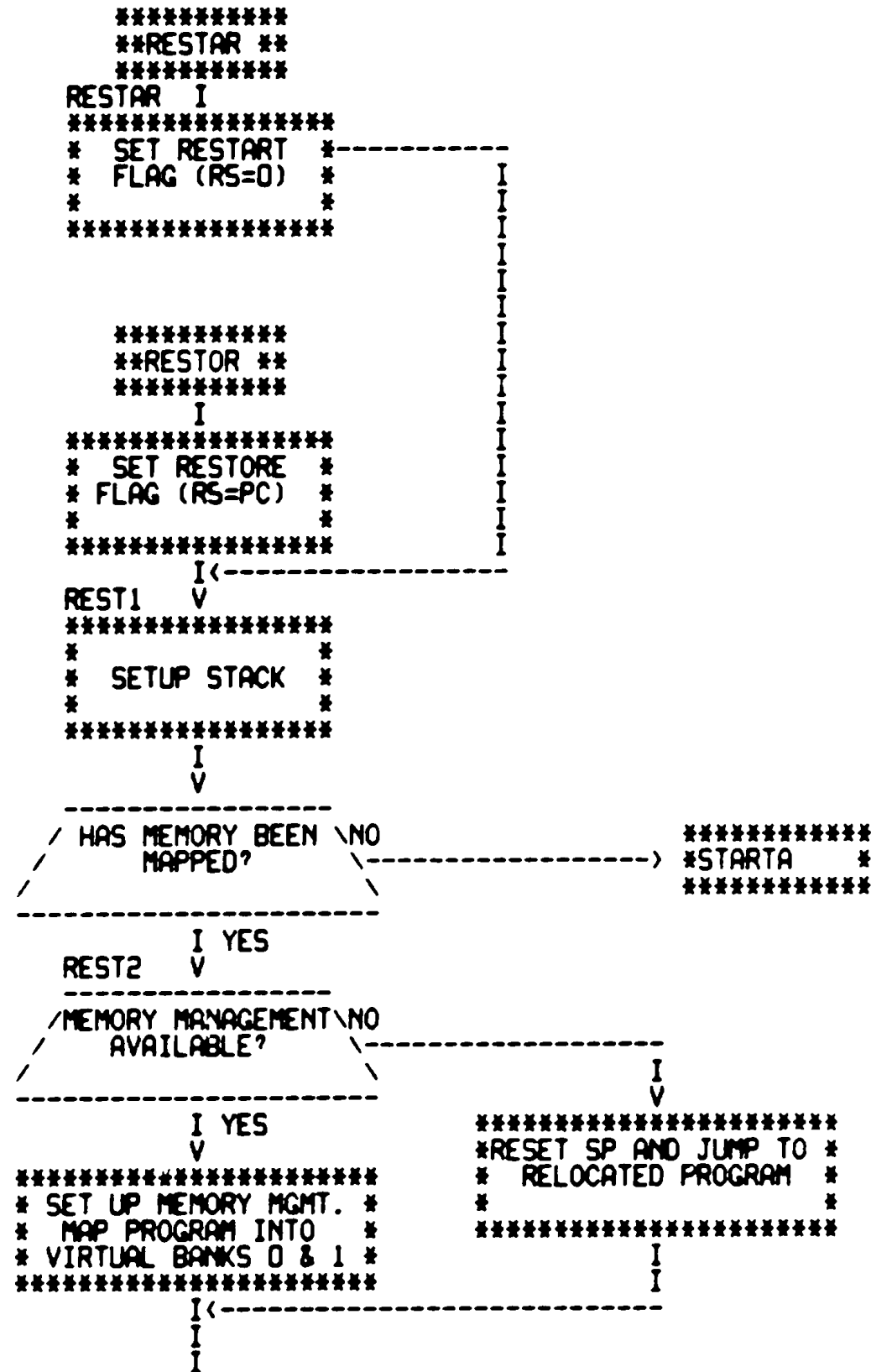
SEQ 0018

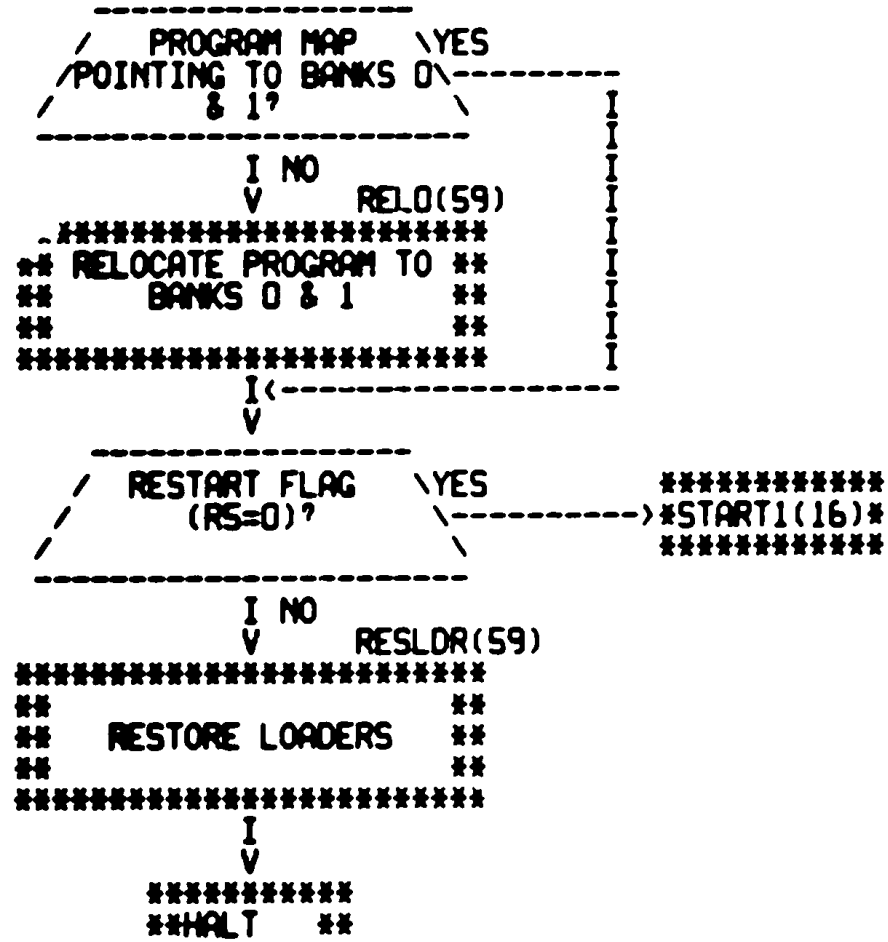
```
*****  
* SWITCH SETTINGS AND *  
* BASIC DEFINITIONS *  
* *  
*****
```

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

SA=210 . =300

SA=214





.S72

41

```

*****
**SPWRDN **
*****
  I
  V
*****
* SILLUP -> VECTOR *
* SAVE REGISTERS *
* SPWRDN -> VECTOR *
*****
  I
  V
*****
**HALT **
*****

```

```

*****
**SPWRUP **
*****
  I
  V
*****
* WAIT LOOP FOR TTY *
* RESTORE REGISTERS *
* SPWRDN -> VECTOR *
*****
  I
  V
*****
TYPE POWER FAIL MESSAGE
*****
  I
  V
*****
**RETURN **

```

```

*****
**SILLUP **
*****
  I
  V
*****
**HALT **
*****

```

.=1100

```
*****  
* STANDARD 'SYSMAC' *  
* 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 *  
*****
```

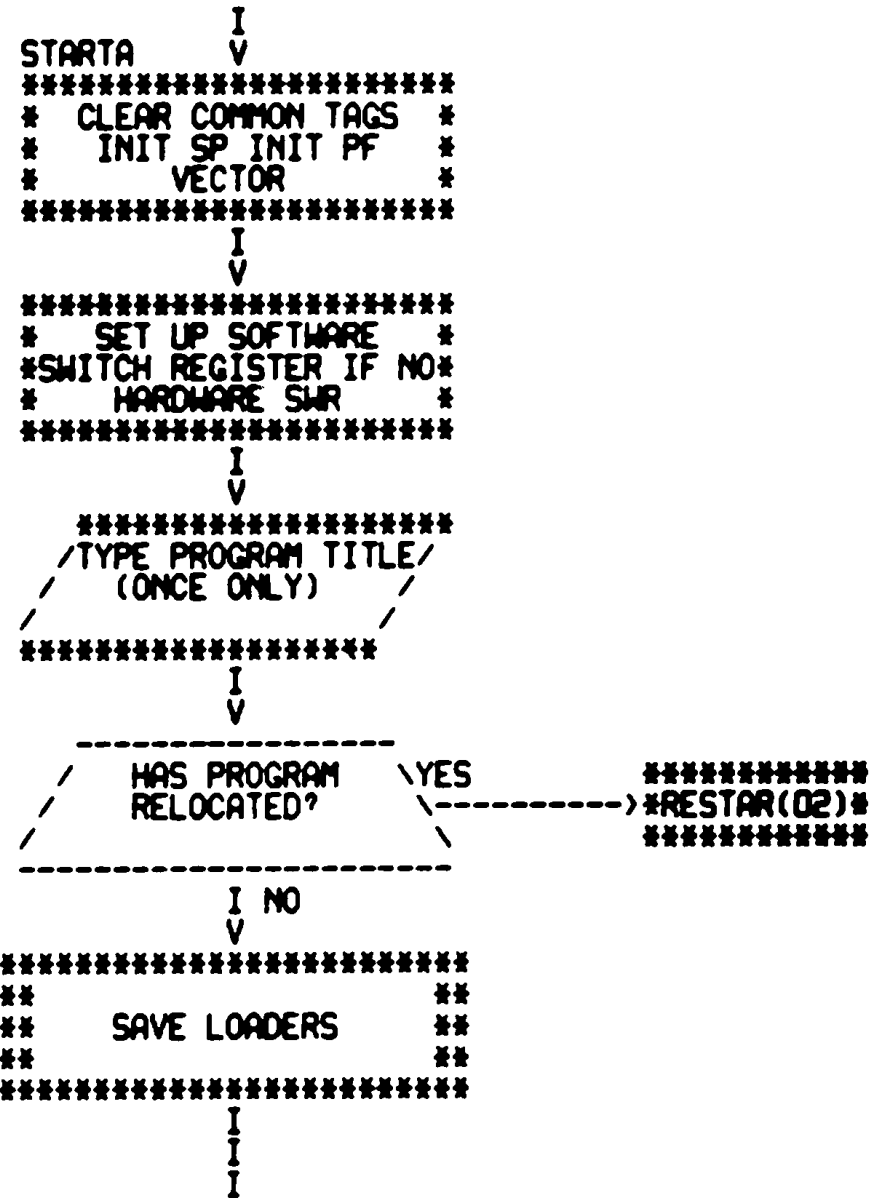
```
*****  
* ERROR MESSAGE POINTER *  
* TABLE *  
* *  
*****
```

SA=204

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

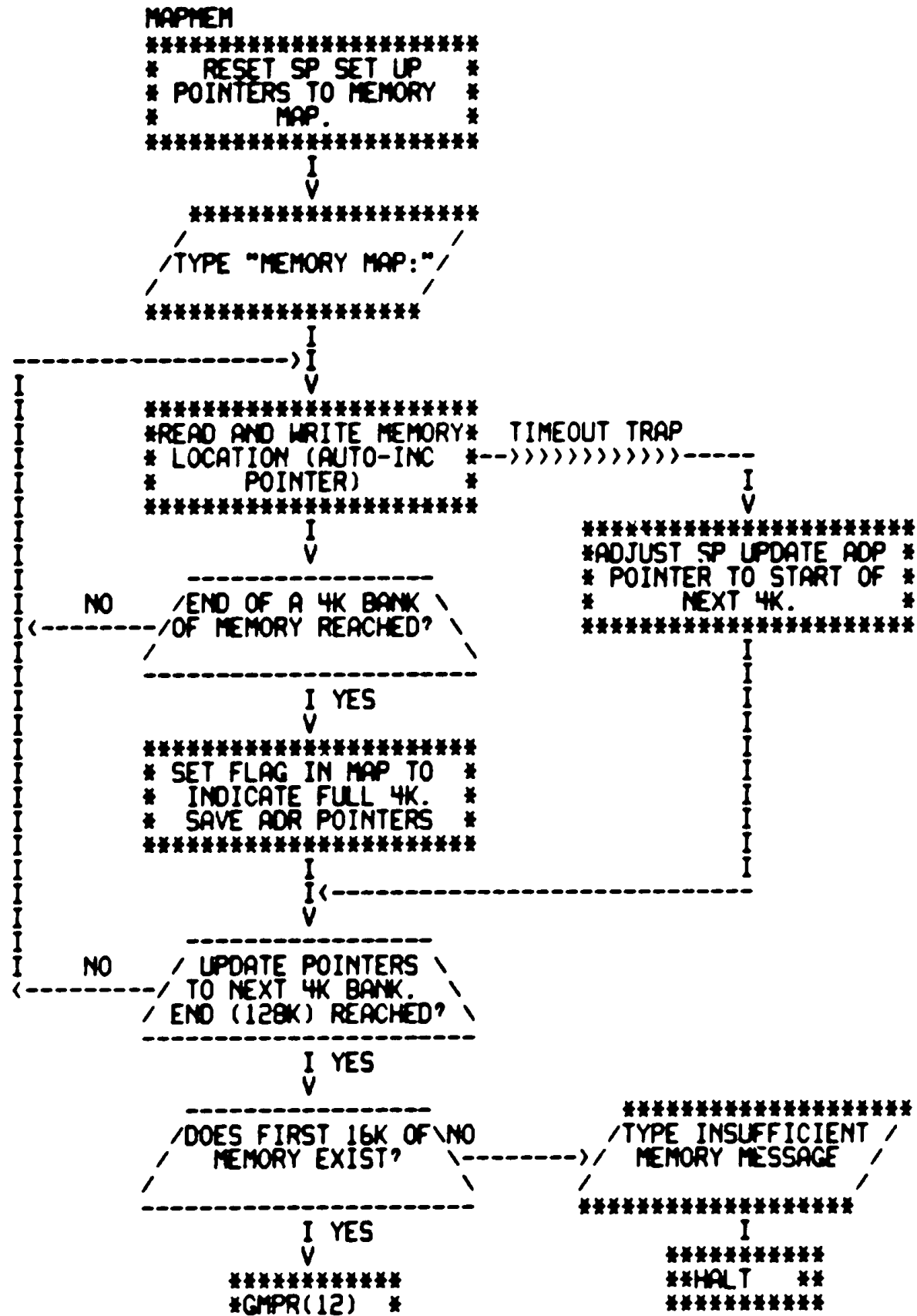
SA=200

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

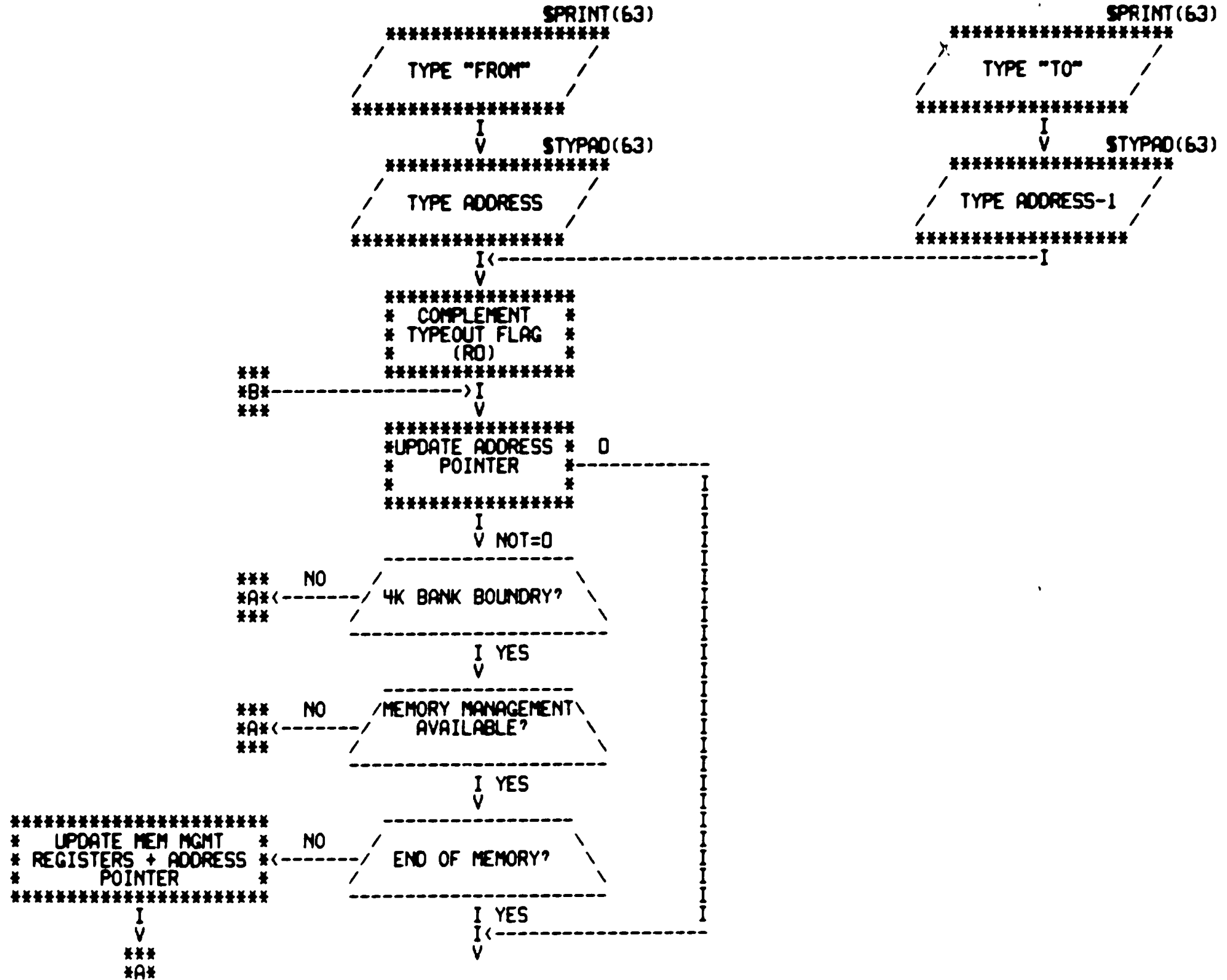


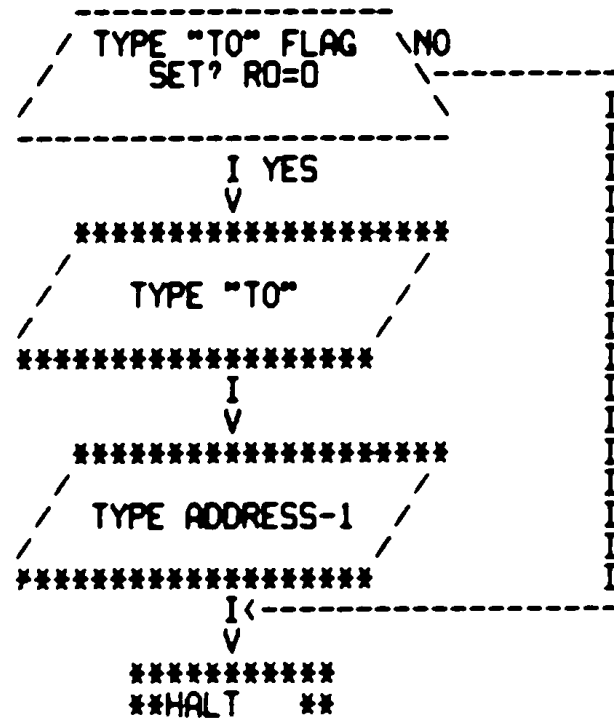






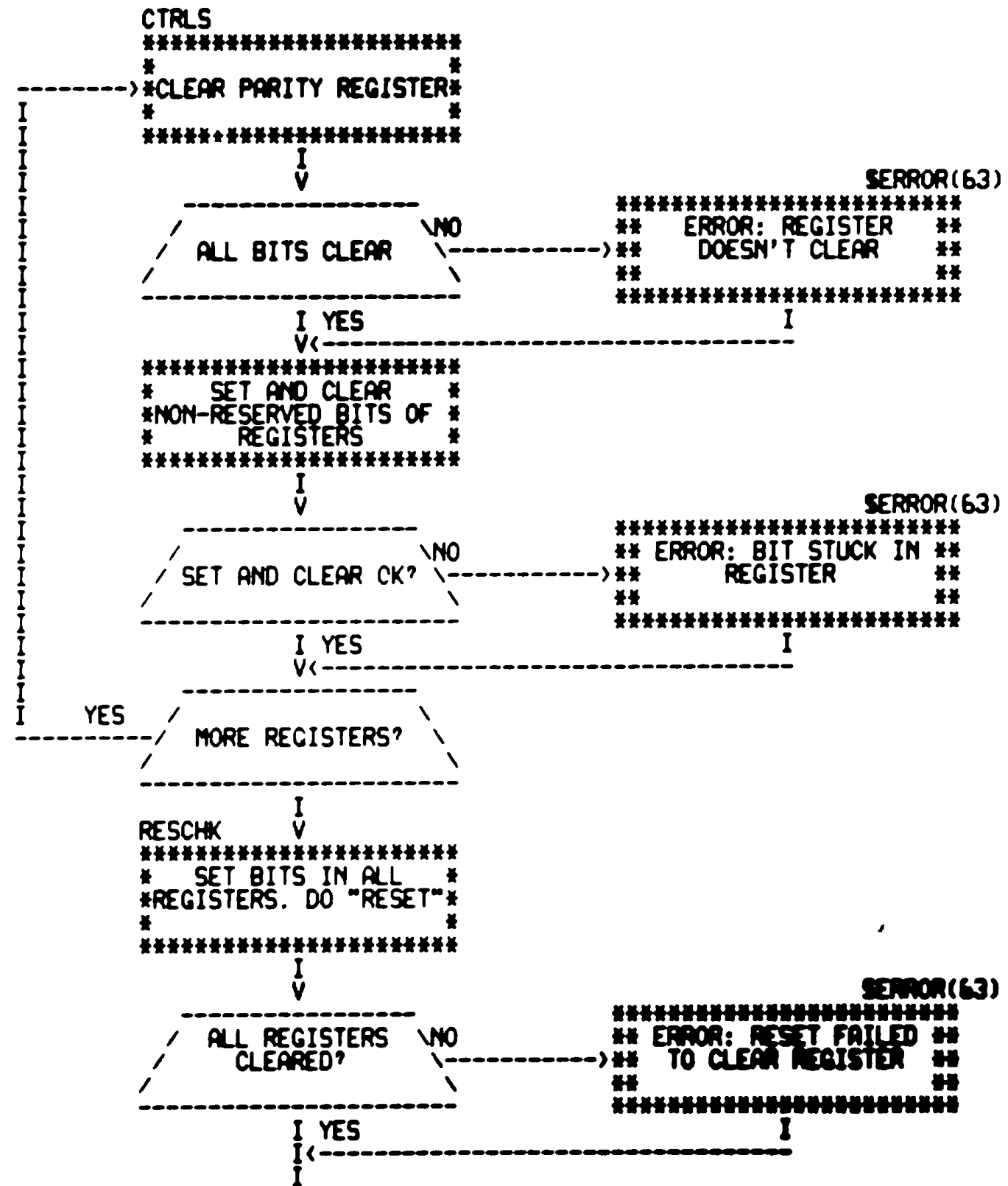




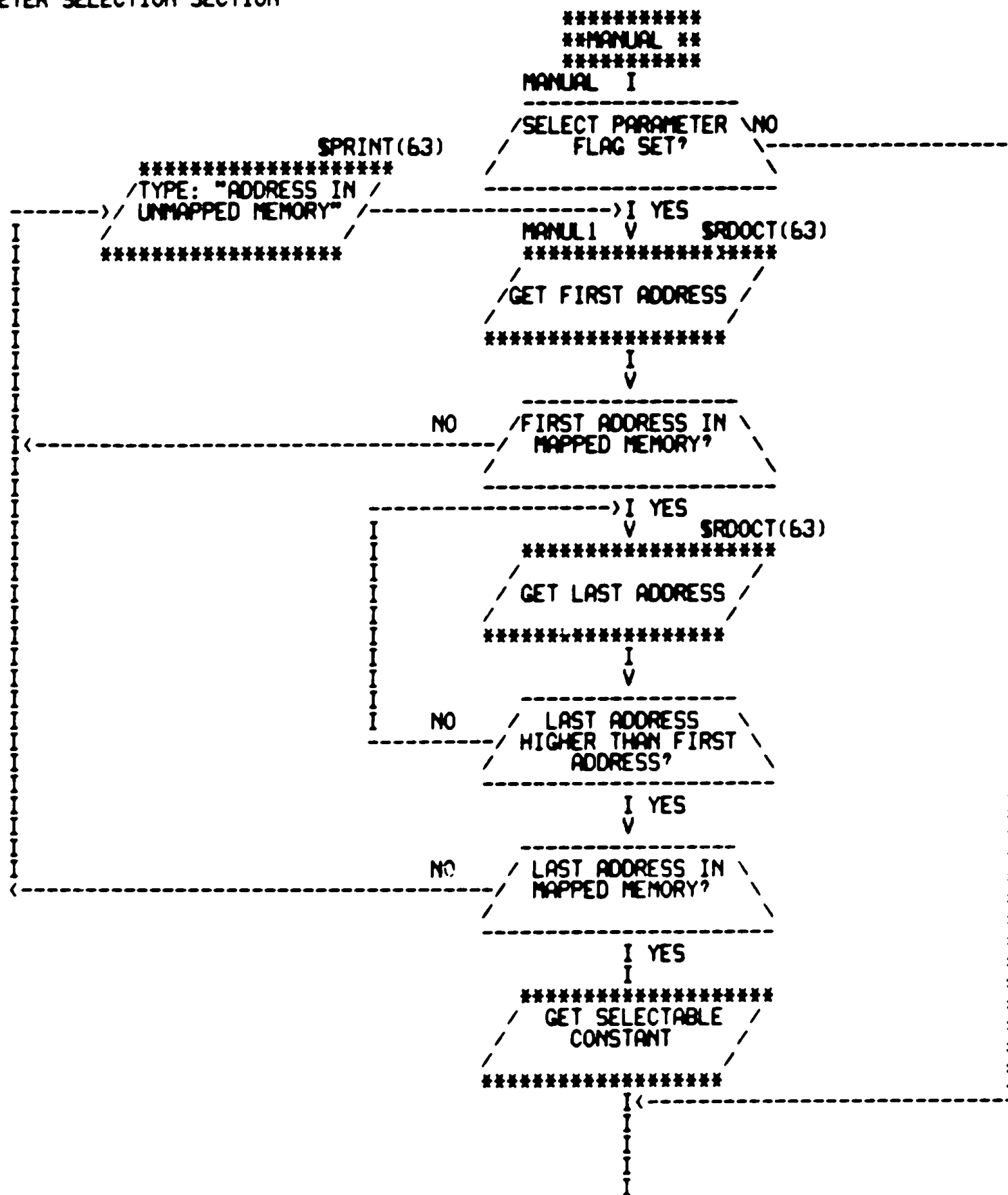












H03

MAINDEC-11-DZQMC-D 0-124K MEMORY EXERCISER. (16K VERSION)  
START1: START OF PASS

DECFL0 VER 00.12 08-SEP-77 10:00 PAGE 16

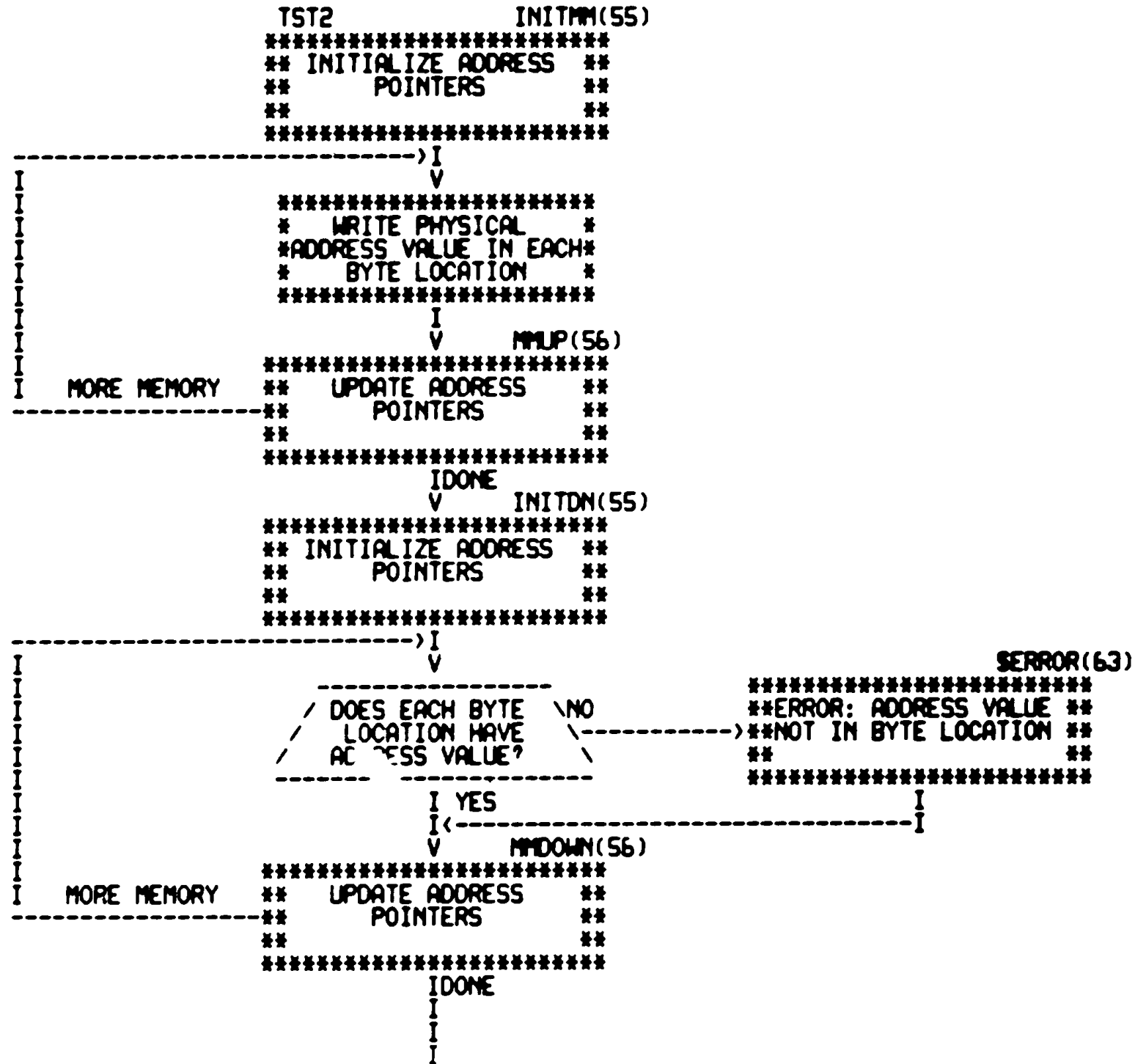
SEQ 0033

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

```

TST1          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V
I             *****
I             * WRITE PHYSICAL *
I             * ADDRESS VALUE IN EACH *
I             *   WORD LOCATION   *
I             *****
I             I
I             V          MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
I             **   POINTERS     **
I             **                   **
I             *****
I             IDONE
I             V          INITDN(55)
I             *****
I             ** INITIALIZE ADDRESS **
I             **   POINTERS         **
I             **                   **
I             *****
I             ----->I
I             V
I             /-----\
I             /  DOES EACH  \ NO
I             / LOCATION HAVE \
I             / ADDRESS VALUE? \
I             \-----/
I             I YES
I             I<-----I
I             V          MMDOWN(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
I             **   POINTERS     **
I             **                   **
I             *****
I             IDONE
I             I
I             I
I             I

```



```

TST3          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
V
*****
*   WRITE ONE'S      *
* COMPLEMENT OF ADR *
* INTO WORD LOCATION *
*****
I
V          MMDOWN(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
IDONE
V          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
V
-----
/ DOES EACH WORD \ NO
/ HAVE COMPLEMENT OF \
/   ADR. VALUE?   \
-----
I YES
I<-----I
V          MMUP(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
IDONE
I
I
I

```

SERROR(63)

```

*****
**ERROR: COMPLEMENT OF **
**ADR. NOT IN WORD LOC.**
**
*****

```

```

TST4          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
*****
** WRITE BANK # VALUE **
** INTO EACH BYTE    **
**   LOCATION        **
*****
          I
          V          MMUP(56)
*****
I MORE MEMORY ** UPDATE ADDRESS **
----->I      **   POINTERS         **
**                   **
*****
          IDONE
          V          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
          / DOES EACH BYTE \ NO
          / HAVE ITS BANK # \----->
          / VALUE?          \
          /                   \
          I YES
          I<----->I
          V          MMUP(56)
*****
I MORE MEMORY ** UPDATE ADDRESS **
----->I      **   POINTERS         **
**                   **
*****
          IDONE
          I
          I
          I

```

```

*****
** ERROR: BANK # VALUE **
** NOT IN LOCATION    **
**                   **
*****

```

```

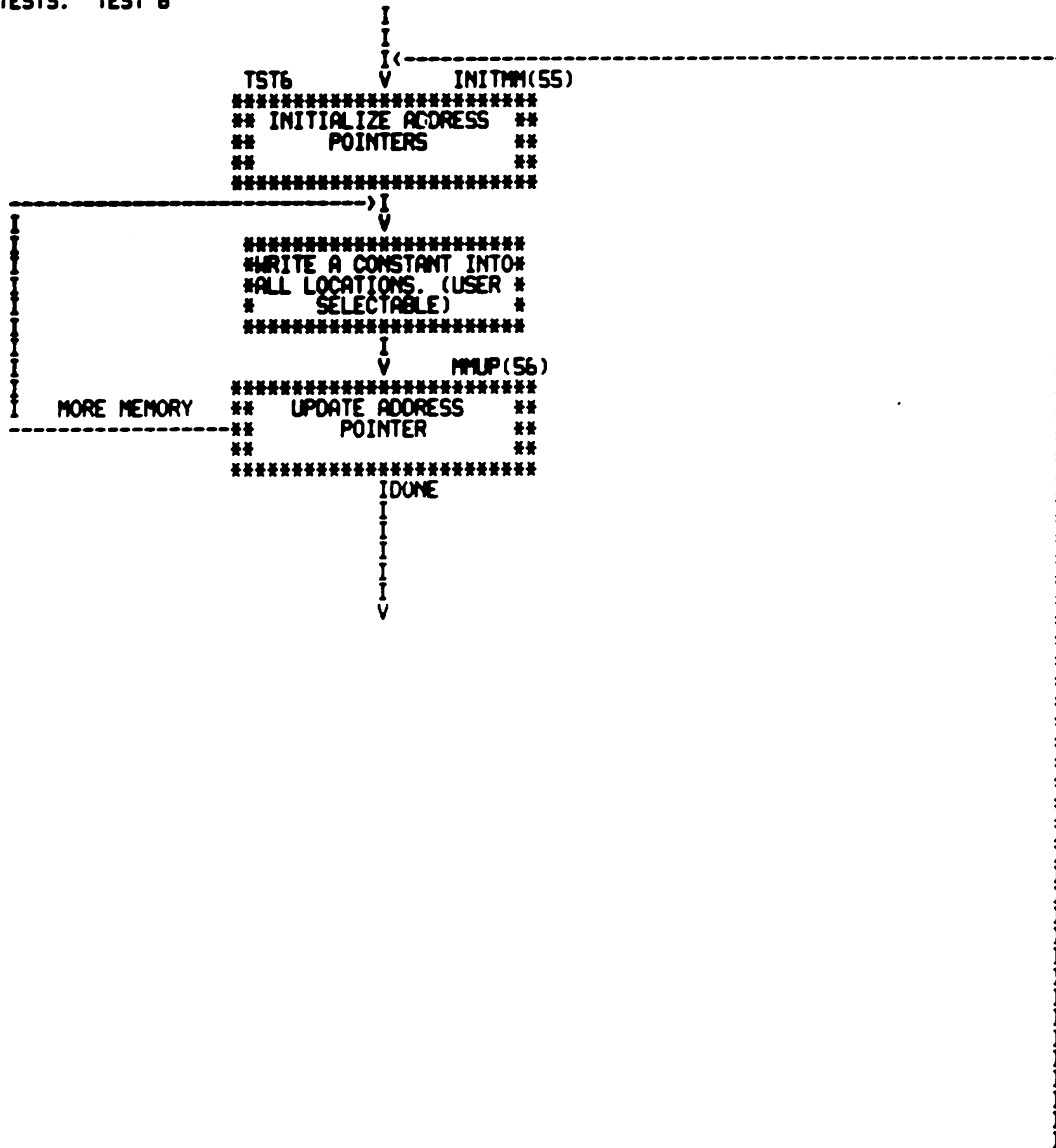
TST5          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
*****
**WRITE 1'S COMPLEMENT *
** OF BANK NUMBER INTO *
**   BYTE LOCATION     *
*****
          I
          V          MNDOWN(56)
MORE MEMORY *****
** UPDATE ADDRESS     **
**   POINTERS         **
**                   **
*****
          IDONE
          V          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
          / DOES EACH BYTE \ NO
          / HAVE COMPLEMENT OF \
          /   BANK VALUE?   \
          ----->I
          I YES
          I<-----I
          V          MNDOWN(56)
MORE MEMORY *****
** UPDATE ADDRESS     **
**   POINTERS         **
**                   **
*****
          IDONE
          I
          I
          I

```

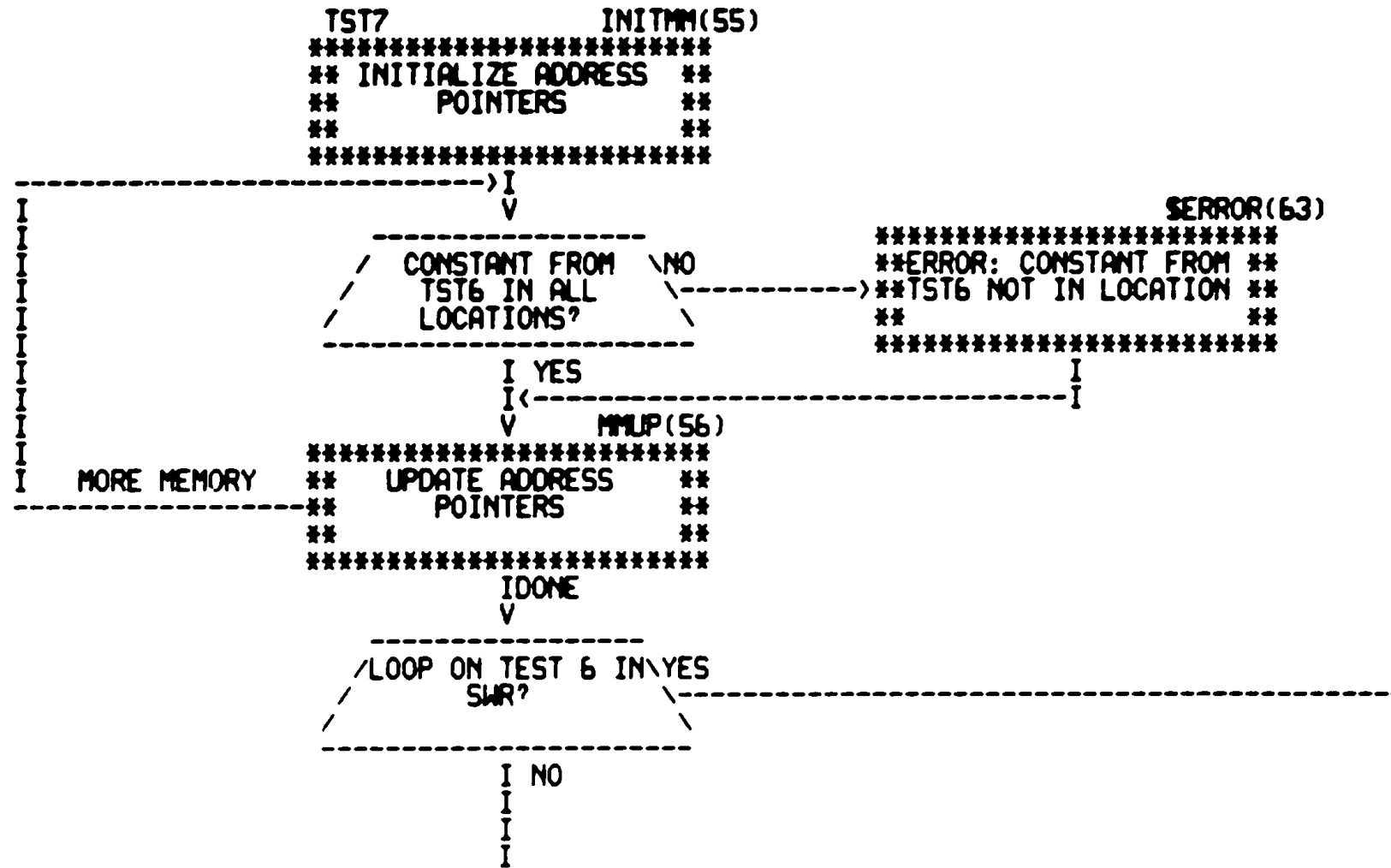
```

*****
**ERROR: COMPLEMENT OF **
** BANK # NOT IN BYTE **
**   LOC.               **
*****

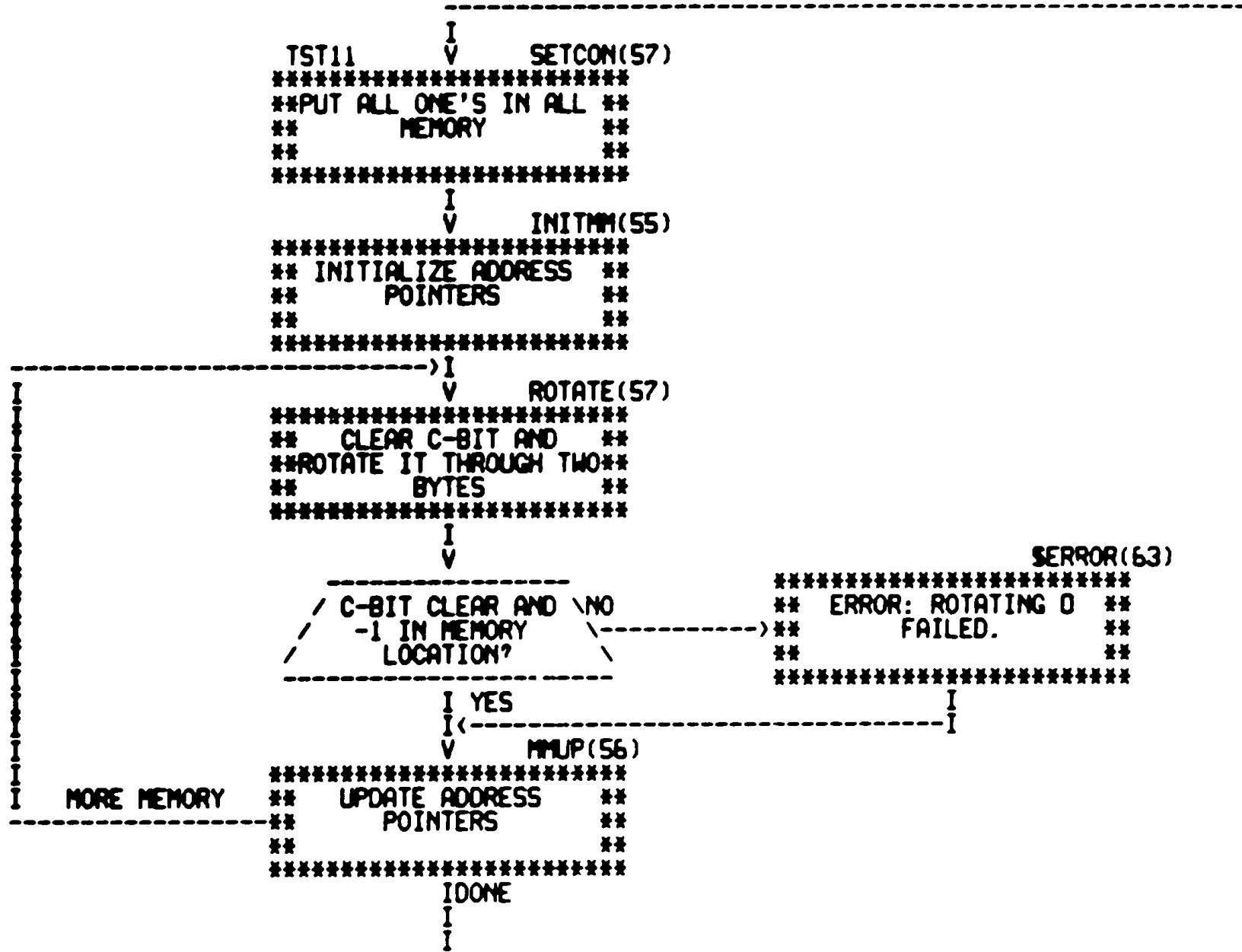
```

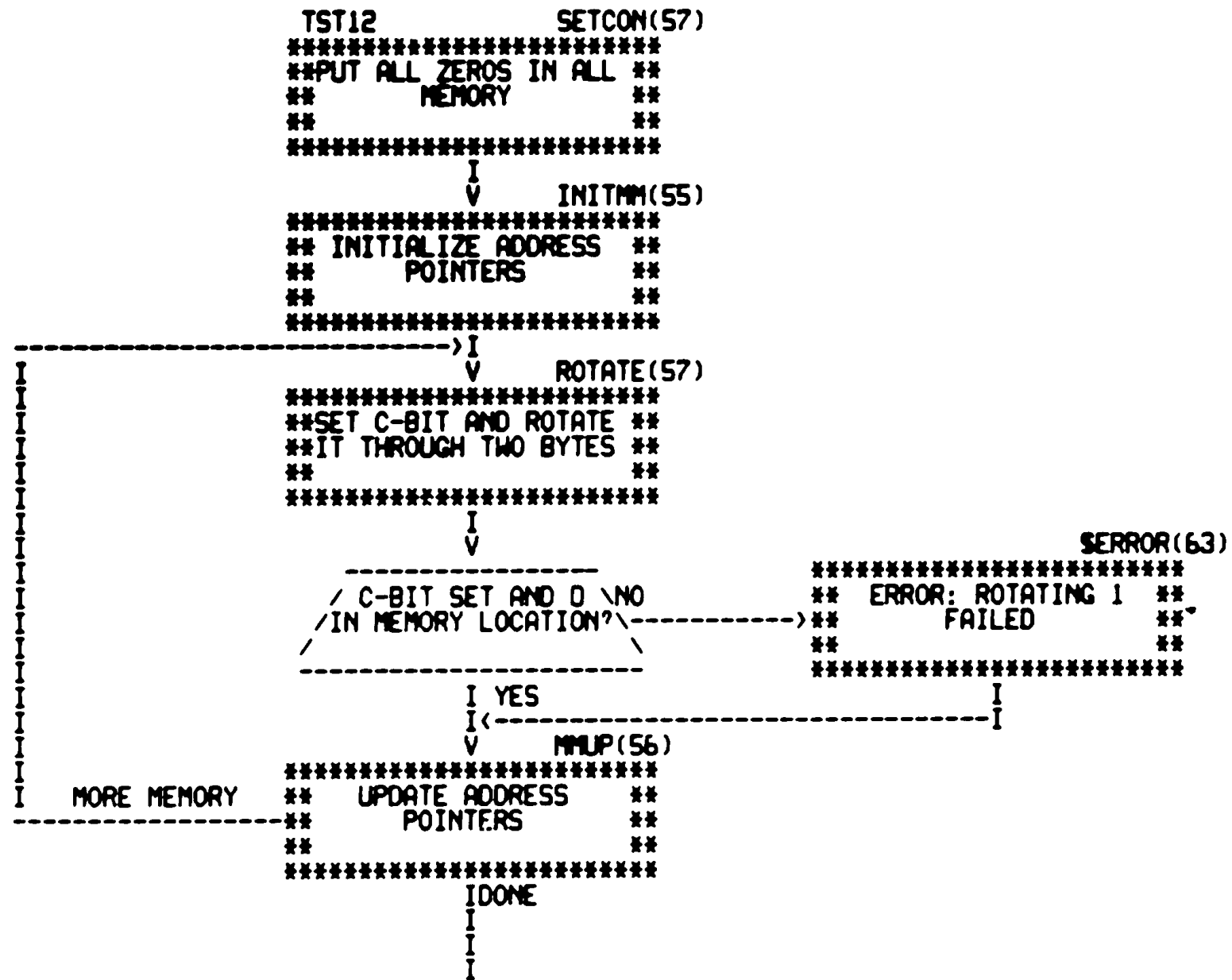












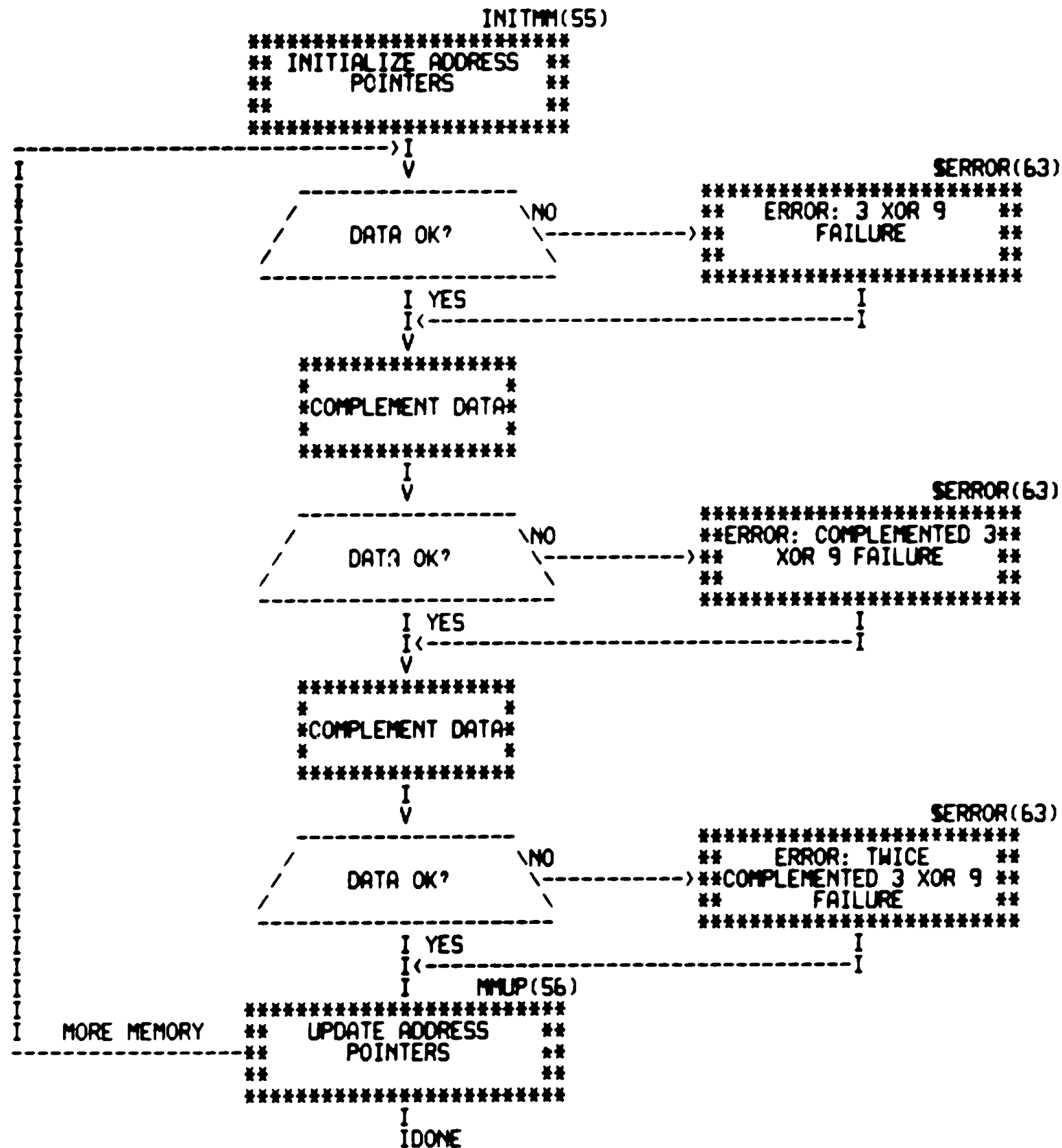
```

TST13          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I             V      W3X9(57)
I             *****
I             ** WRITE 256. WORD **
I             **   BLOCKS WITH   **
I             ** 0,0,0,0,-1,-1,-1,-1 **
I             *****
I             I
I             V      MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
*****
I             IDONE
I             V      INITMM(55)
I             *****
I             ** INITIALIZE ADDRESS **
I             **   POINTERS         **
I             **                   **
I             *****
----->I
I             V
I             /-----\
I             /256. WORD BLOCKS \ NO
I             / WRITTEN WITH   \----->
I             / 0,0,0,0,-1,-1,-1,-1 \
I             \-----/
I             I YES
I             I<-----I
I             V      MMUP(56)
I             *****
I MORE MEMORY ** UPDATE ADDRESS **
----- **   POINTERS         **
**                   **
*****
I             IDONE
I             I
I             I
I             I
I             I

```

ERROR(63)

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



```

TST14          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
I              V      W3X9(57)
I              *****
I              ** WRITE 256. WORD **
I              **   BLOCKS WITH   **
I              ** -1,-1,-1,-1,0,0,0 **
I              *****
I              I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
I              **   POINTERS     **
I              **                   **
I              *****
I              IDONE
I              V      INITMM(55)
I              *****
I              ** INITIALIZE ADDRESS **
I              **   POINTERS         **
I              **                   **
I              *****
I              ----->I
I              V
I              /256. WORD BLOCKS \NO
I              / WRITTEN WITH  \----->I
I              /-1,-1,-1,-1,0,0,0 \
I              -----
I              I YES
I              I<-----I
I              V      MMUP(56)
I              *****
I MORE MEMORY ** UPDATE ADDRESS **
I              **   POINTERS     **
I              **                   **
I              *****
I              IDONE
I              I
I              I
I              I
I              I

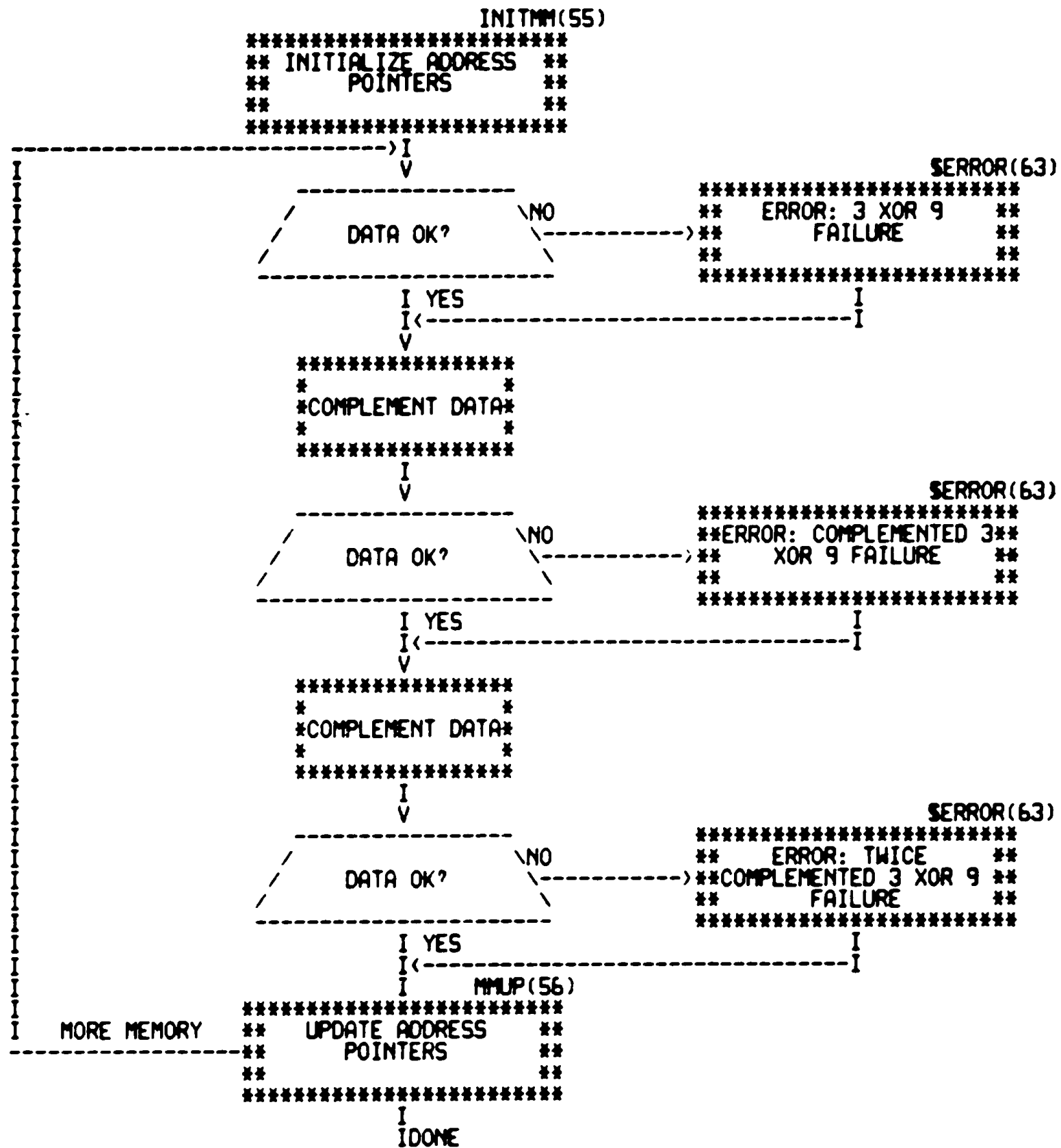
```

\*\*\*\*\* \$ERROR(63) \*\*\*\*\*

```

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

```





```

TST15          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
  V          W3X9(57)
*****
**   WRITE 256. WORD  **
** BLOCKS WITH 401 AND **
**      -1            **
*****
  I
  V          MMUP(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
          IDONE
          V          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
>I
  V
-----
/256. WORD BLOCKS \NO
/ WRITTEN WITH 401 \----->
AND -1?           \
-----
          I YES
          I<-----I
          V          MMUP(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
*****
**   POINTERS         **
**                   **
*****
          IDONE
          I
          I
          I
          I

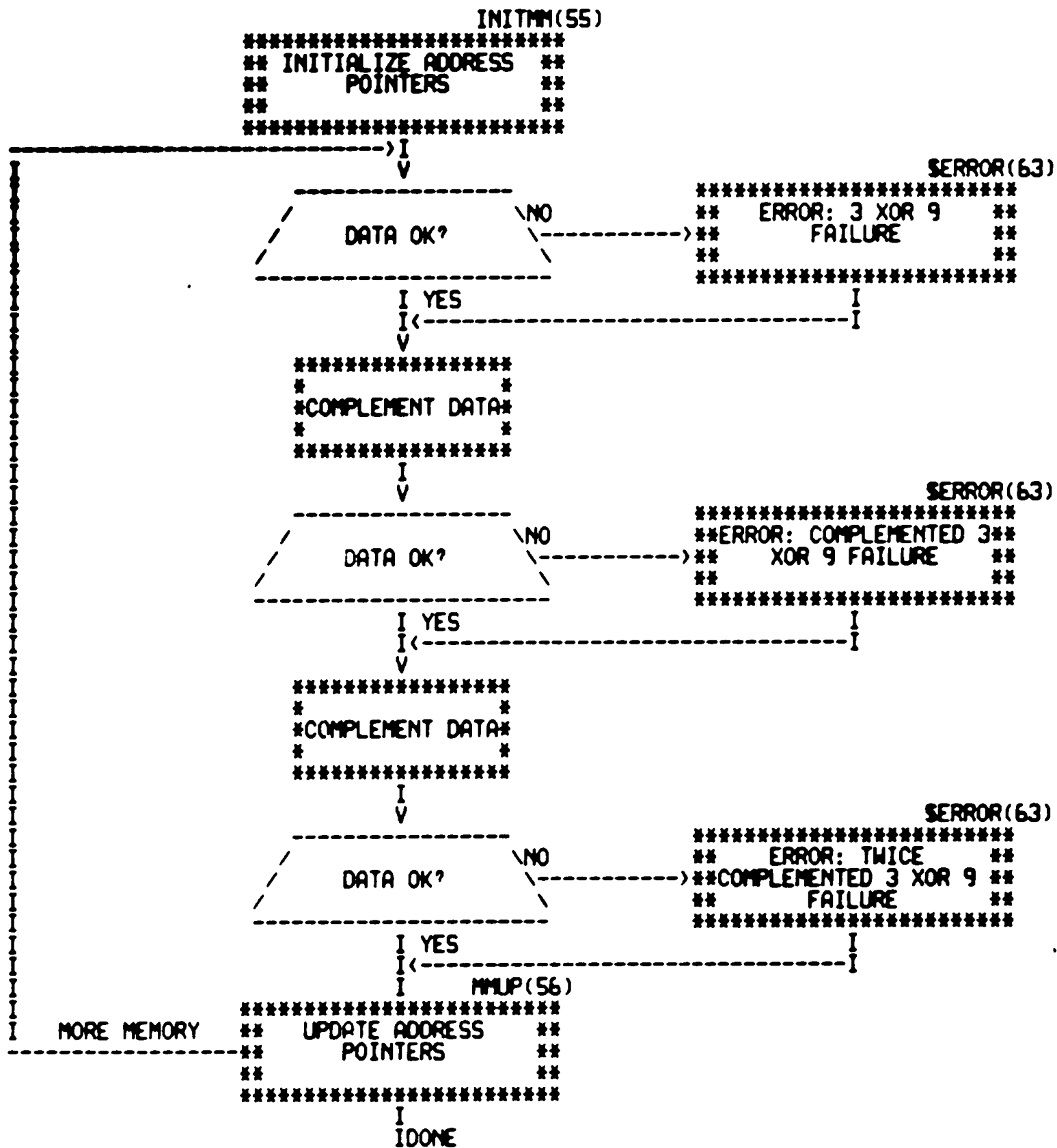
```

```

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

```

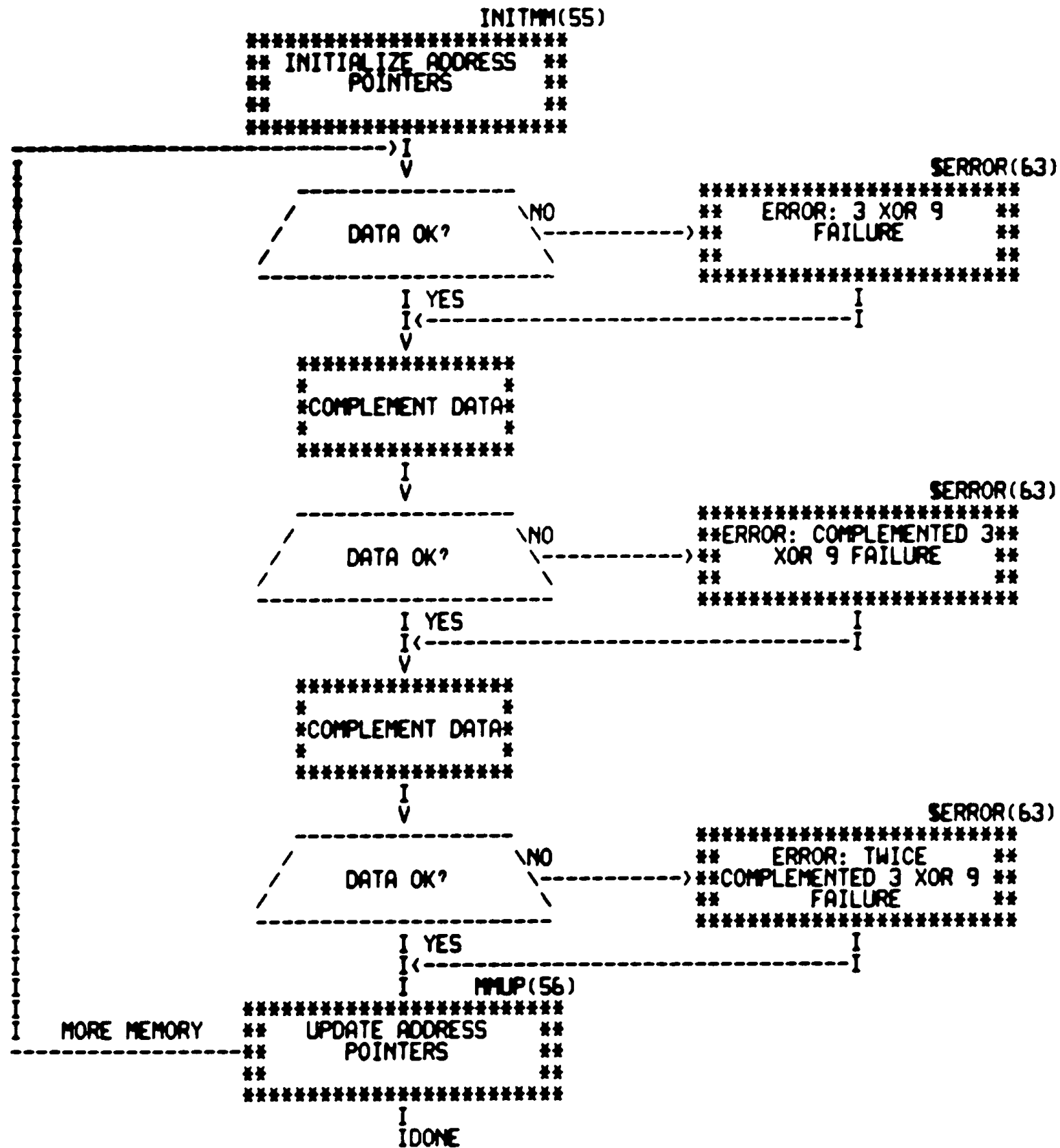
K04

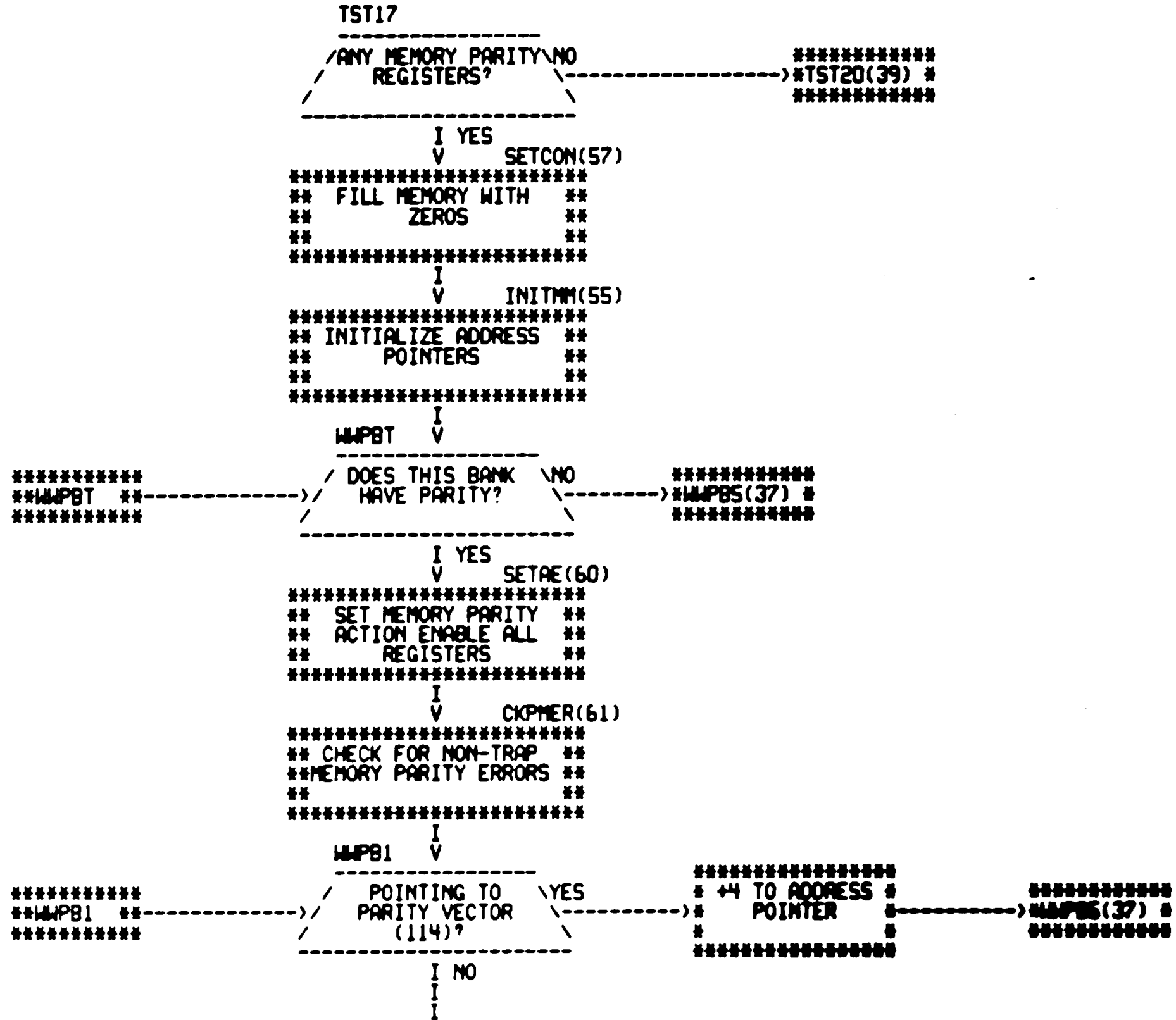


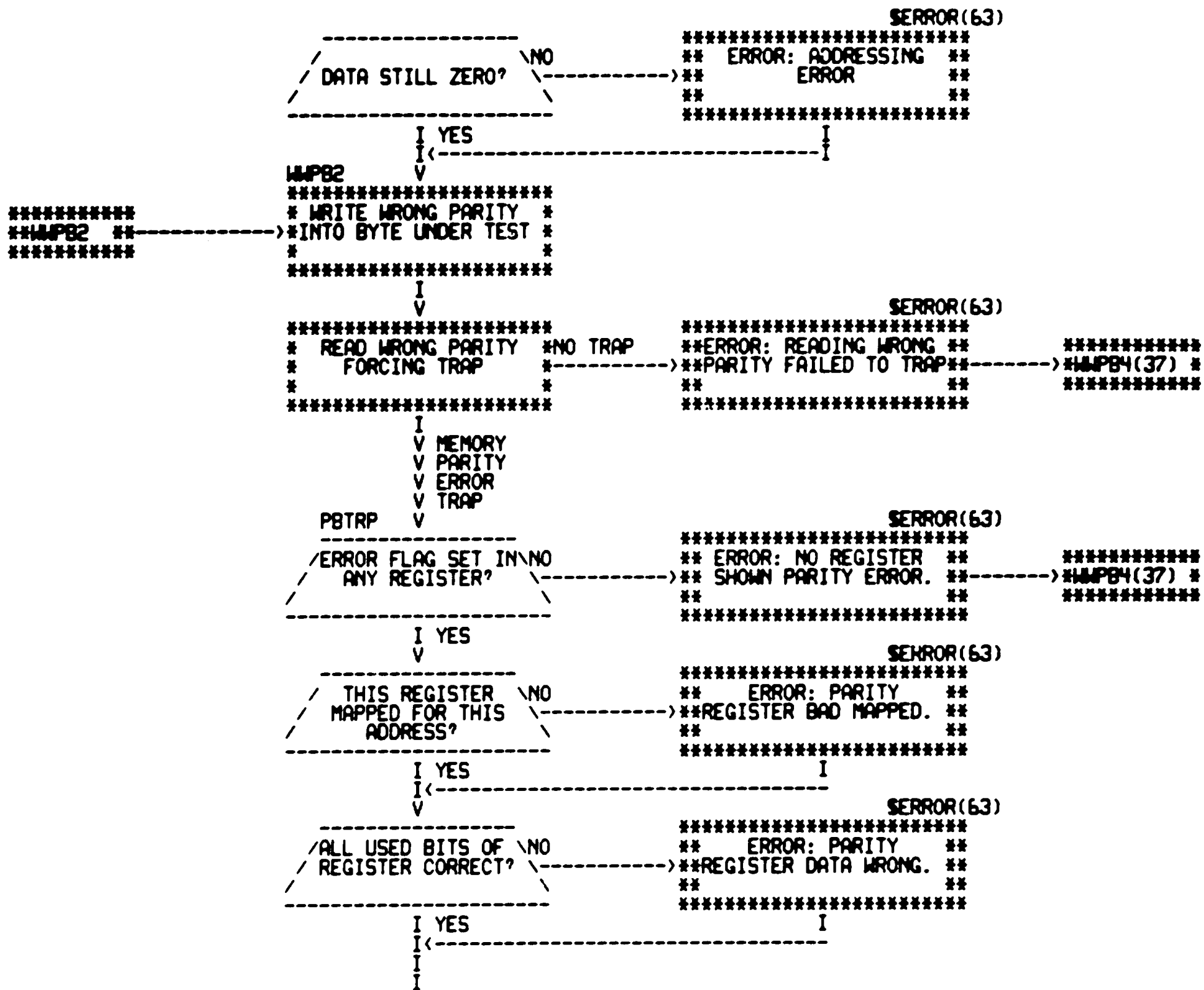
```

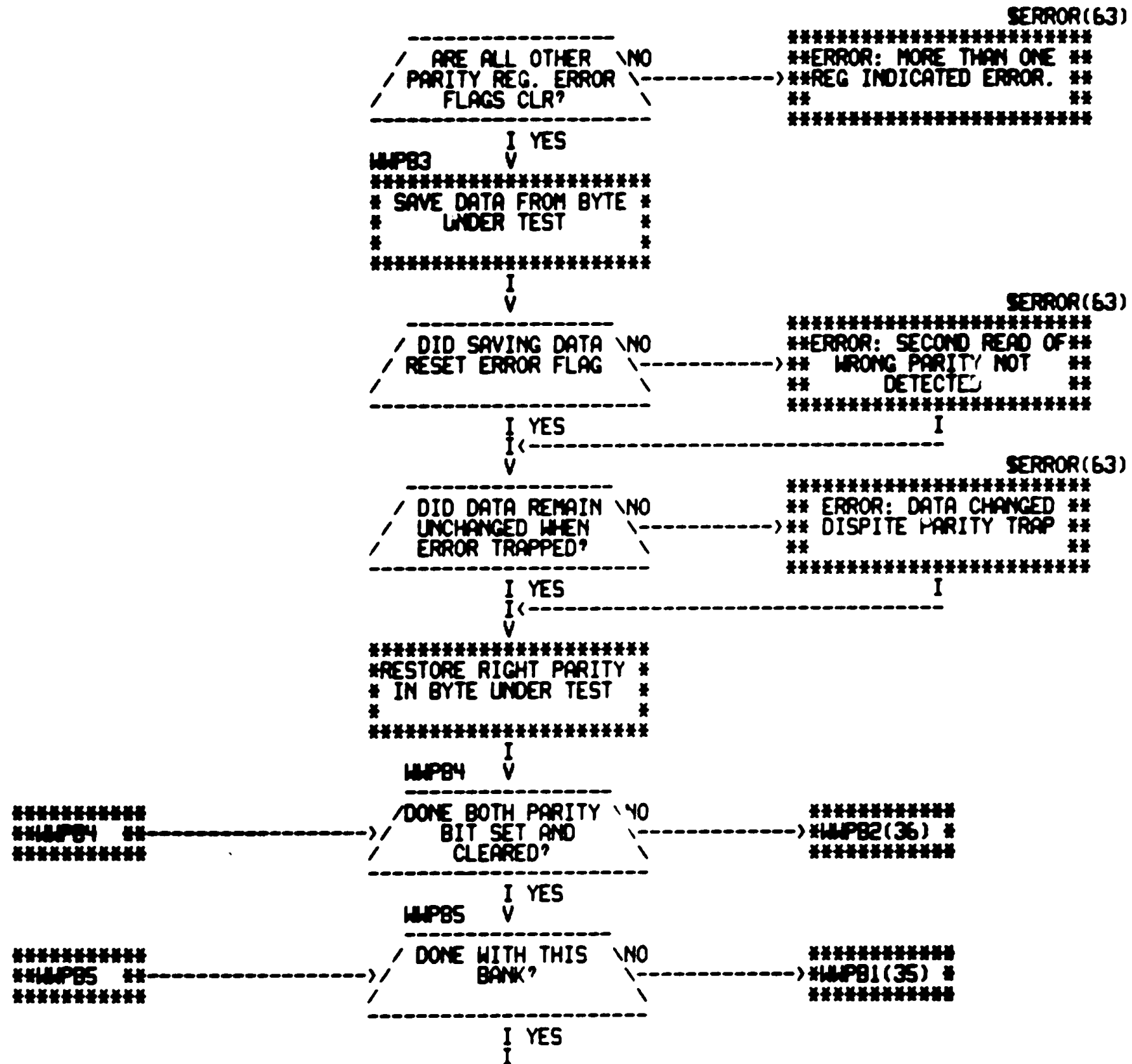
TST16          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V      W3X9(57)
*****
** WRITE 256. WORD   **
** BLOCKS WITH -1 AND **
**       401        **
*****
          I
          V      MMUP(56)
*****
I MORE MEMORY ** UPDATE ADDRESS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

```









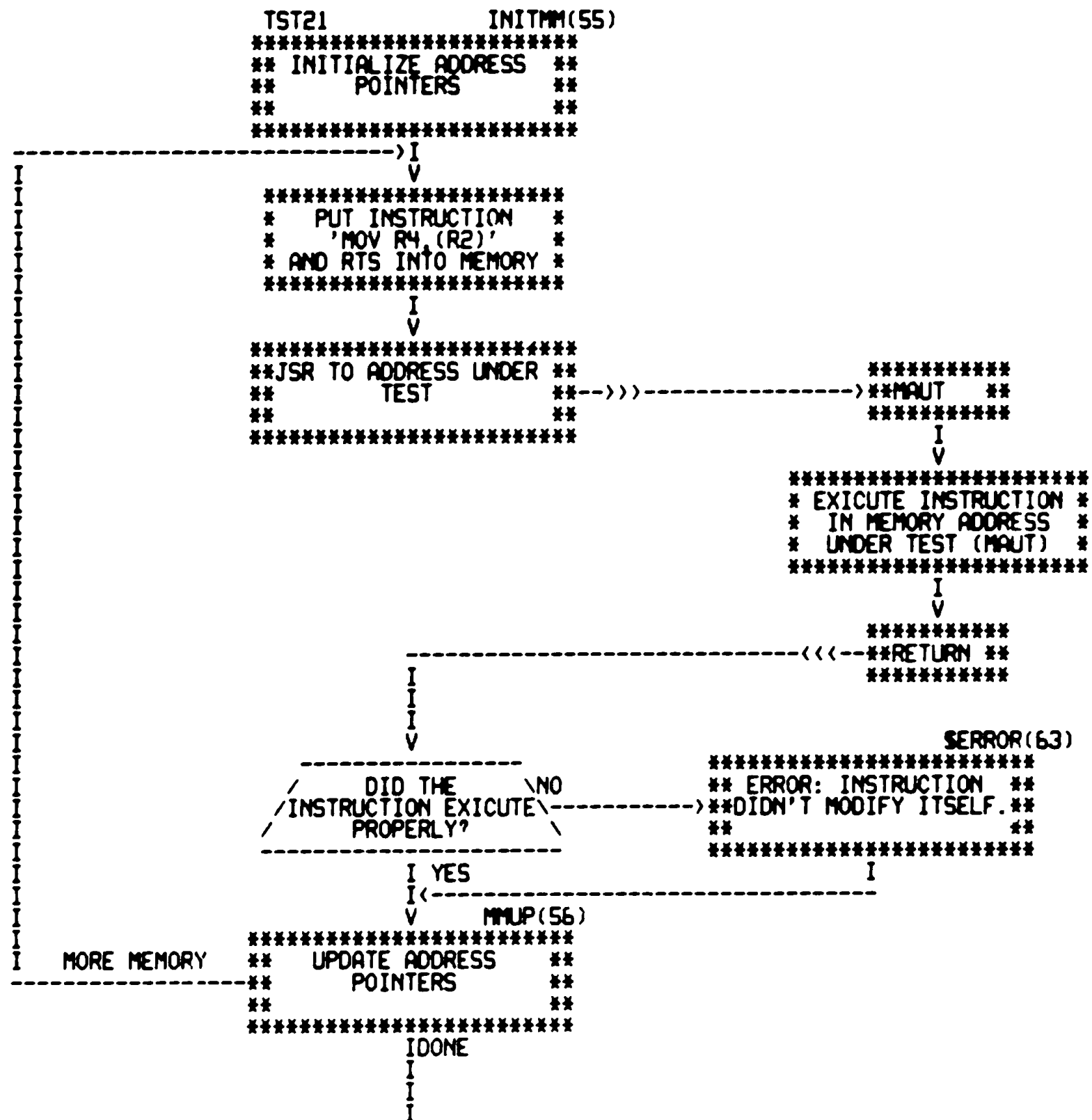
```

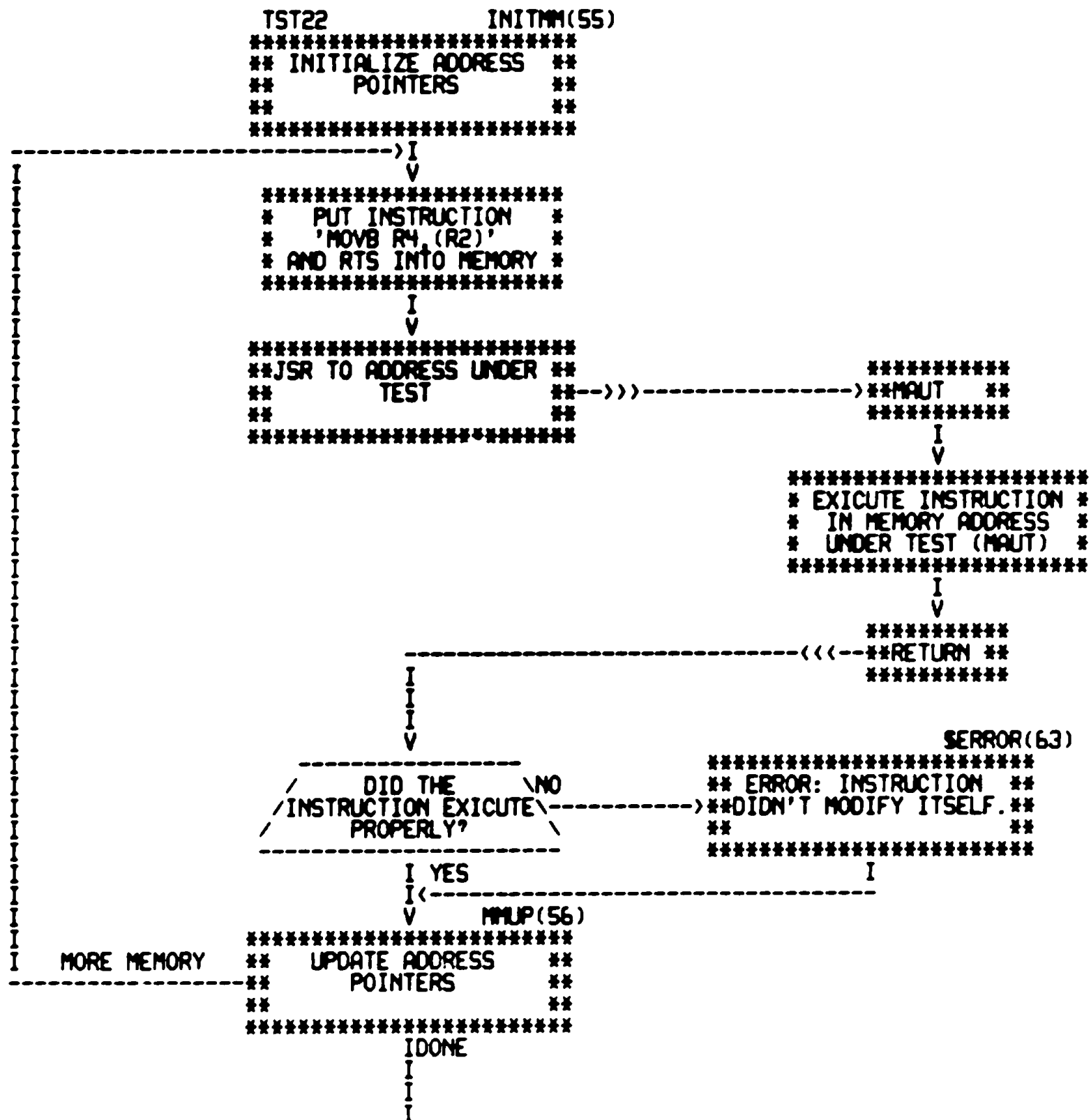
                                     MMAP(56)
*****
***** MORE MEMORY ** UPDATE ADDRESS **
*MAPBT(35) *<-----**   POINTERS   **
*****
                                     IDONE
                                     V
                                     MAMF(60)
*****
** RESET ALL PARITY **
**   REGISTERS   **
**               **
*****
                                     I
                                     I
                                     I

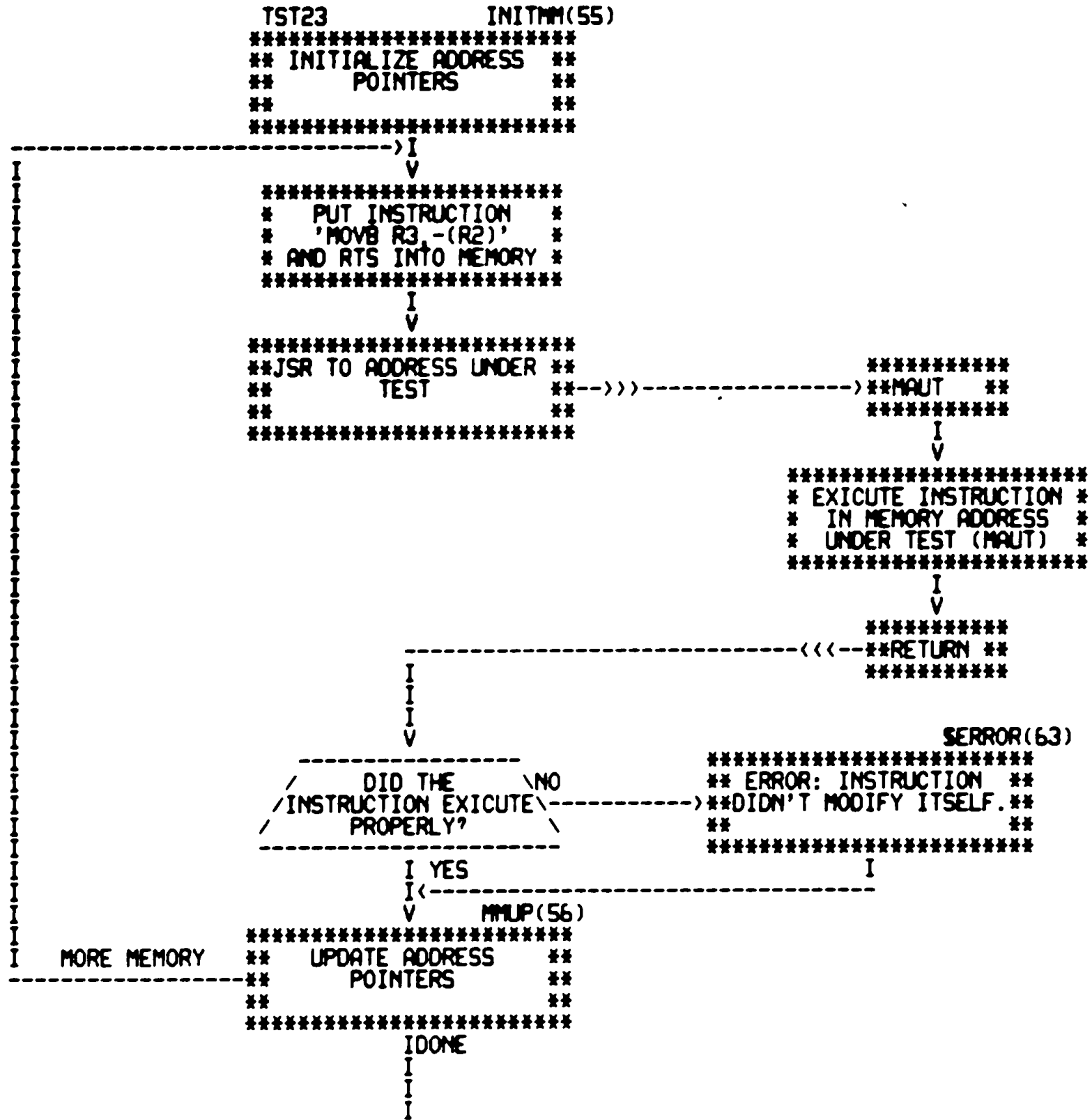
```

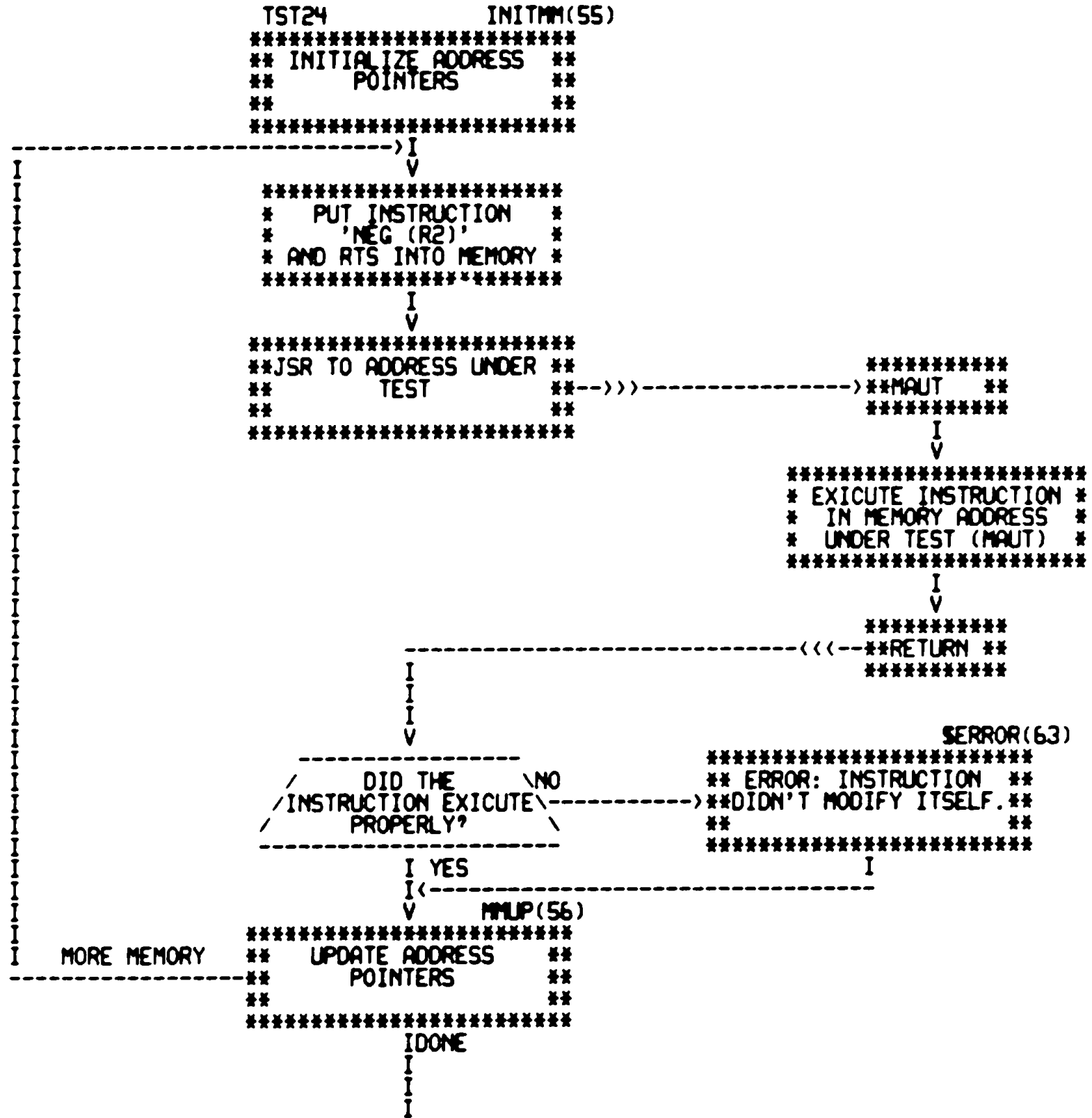


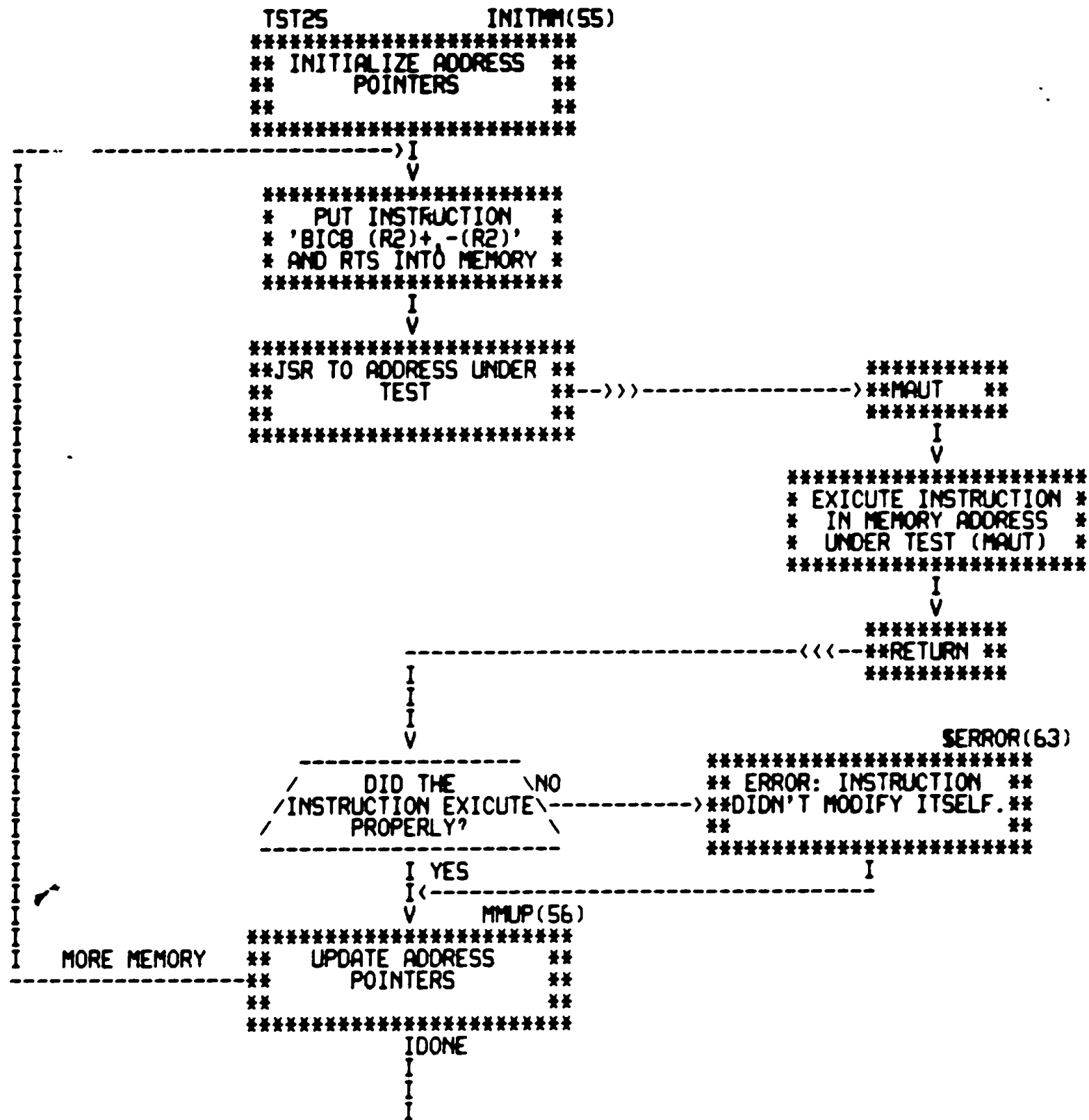
```
*****
**TST20 **
*****
TST20      I      INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS          **
**                     **
*****
----->I
        |
        V
*****
* COPY 2K BLOCK OF *
* PROGRAM CODE INTO *
* MEMORY UNDER TEST *
*****
        |
        V
-----/ DID "RANDOM" DATA\ NO
         / COPY OK? \----->I
        \
        \
        I YES
        I <-----I
        V      MMUP(56)
*****
** UPDATE ADDRESS **
**   POINTERS      **
**                     **
*****
MORE MEMORY
-----
        IDONE
        I
        I
        I
```

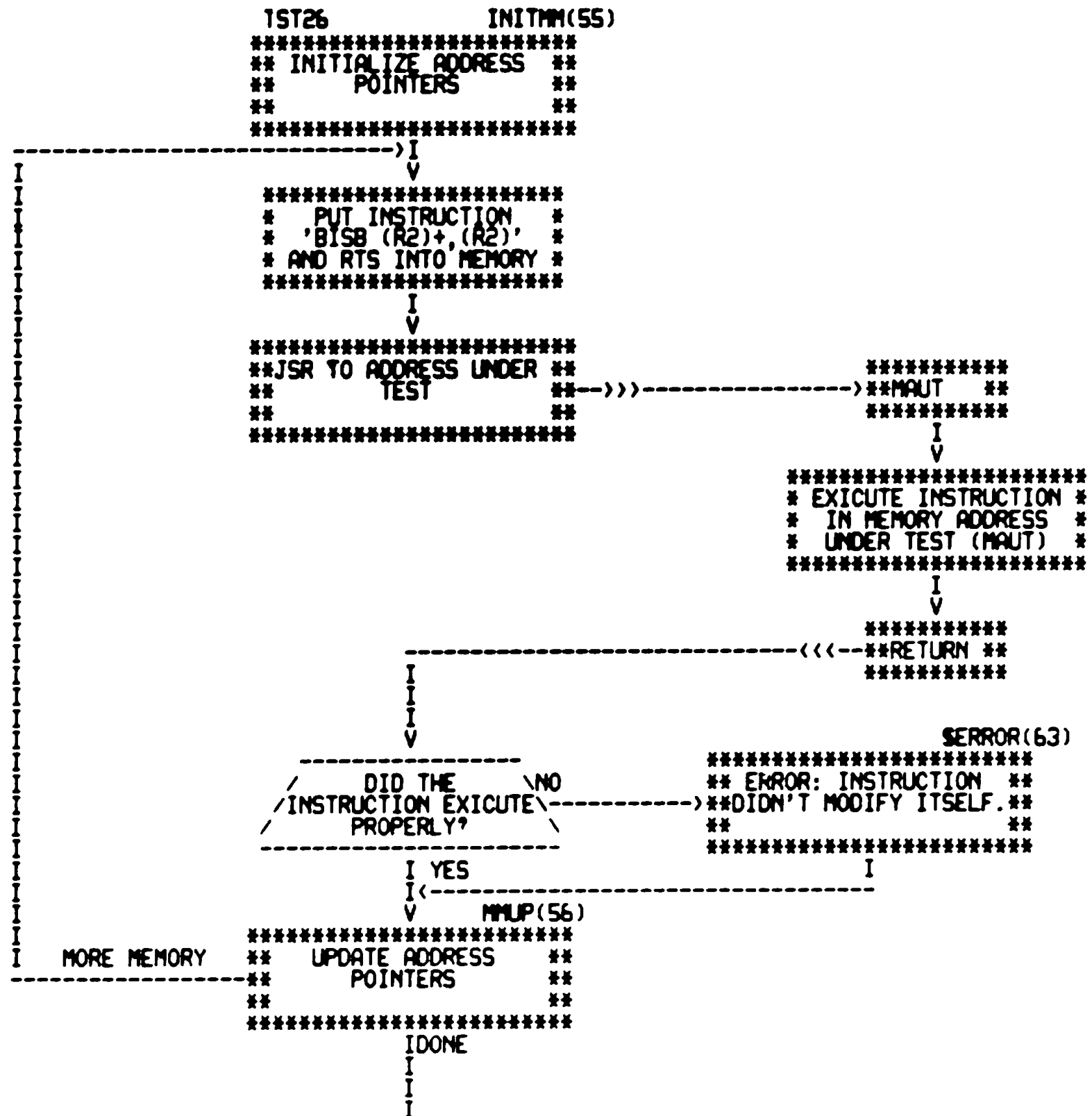


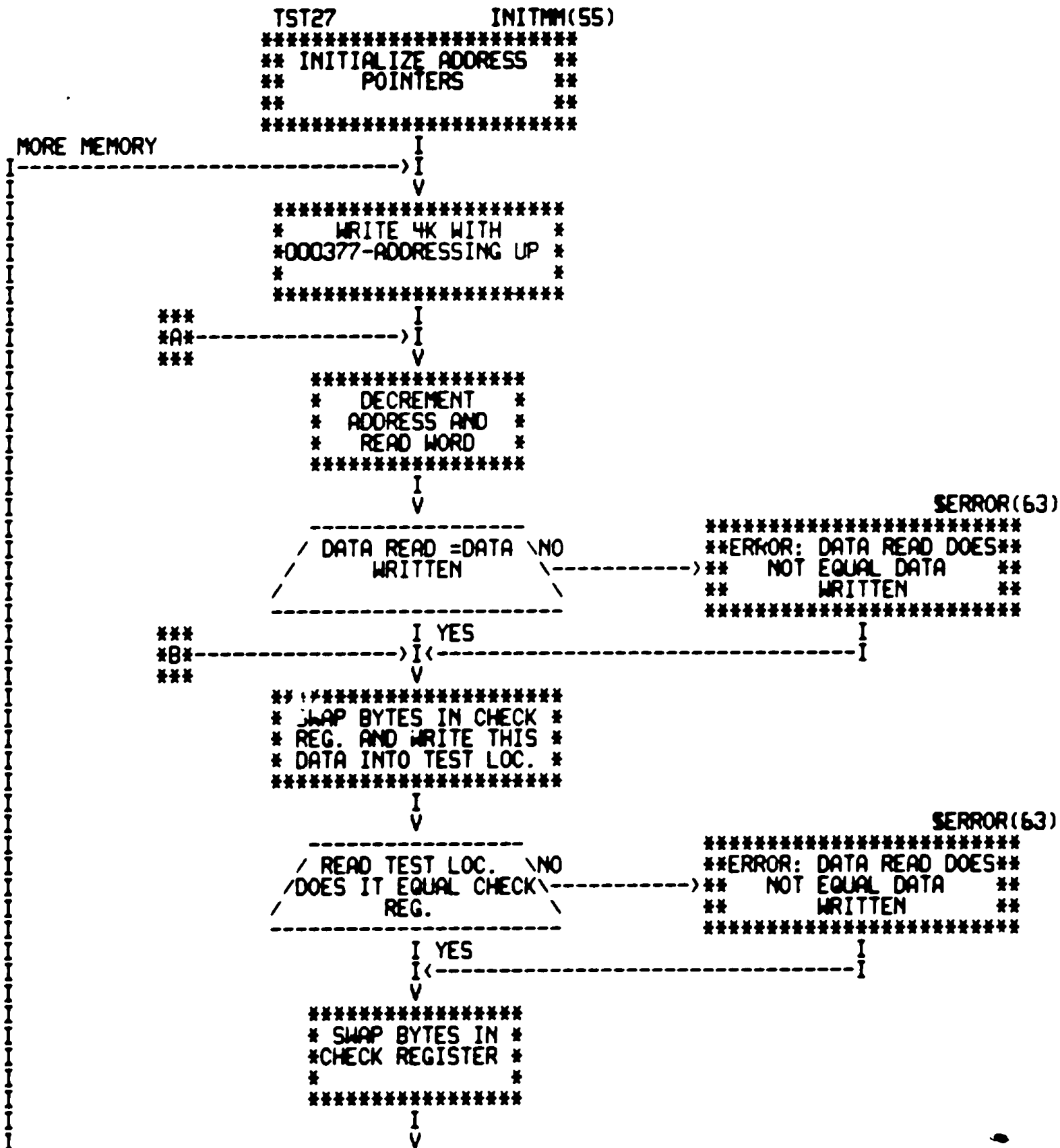




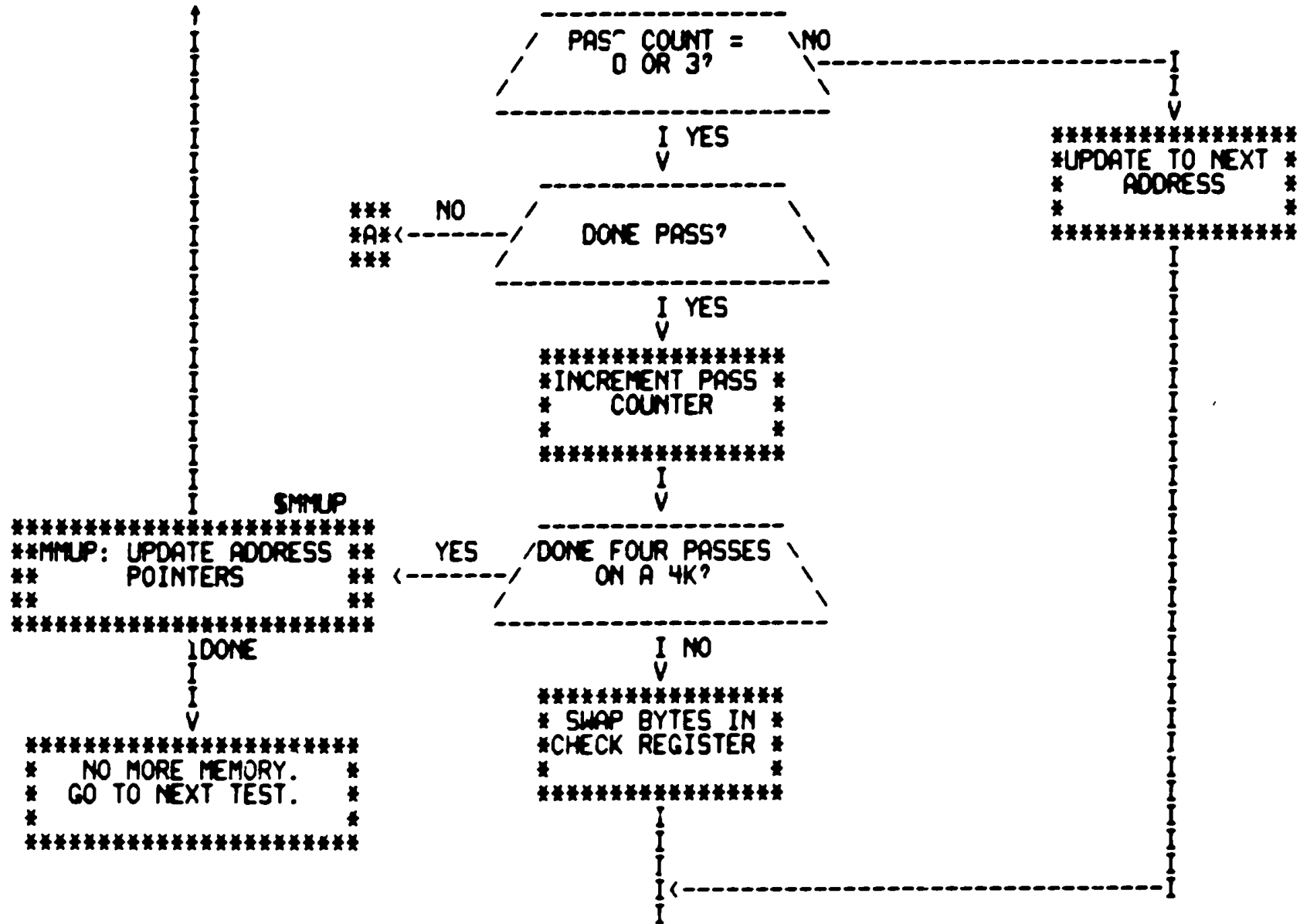


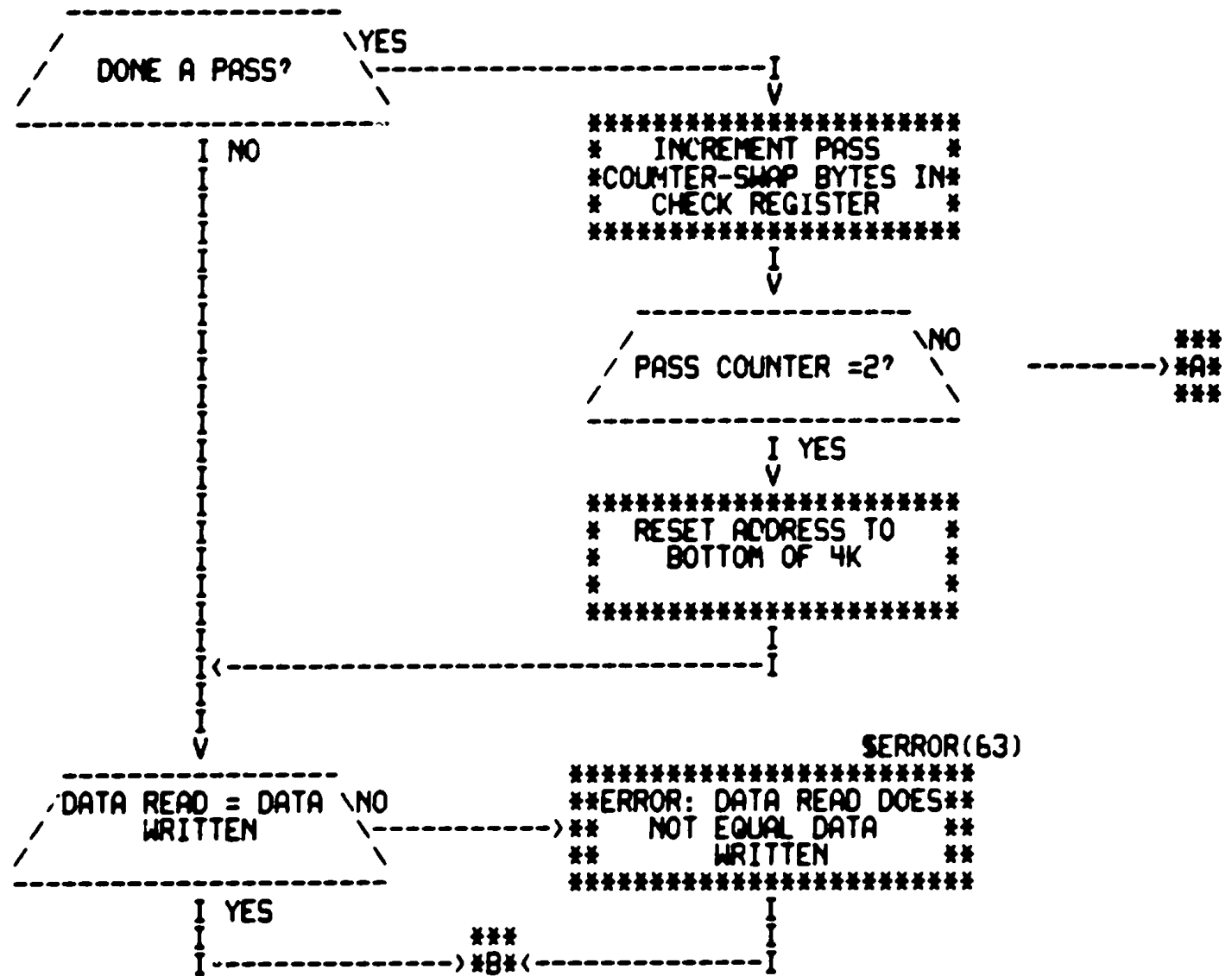


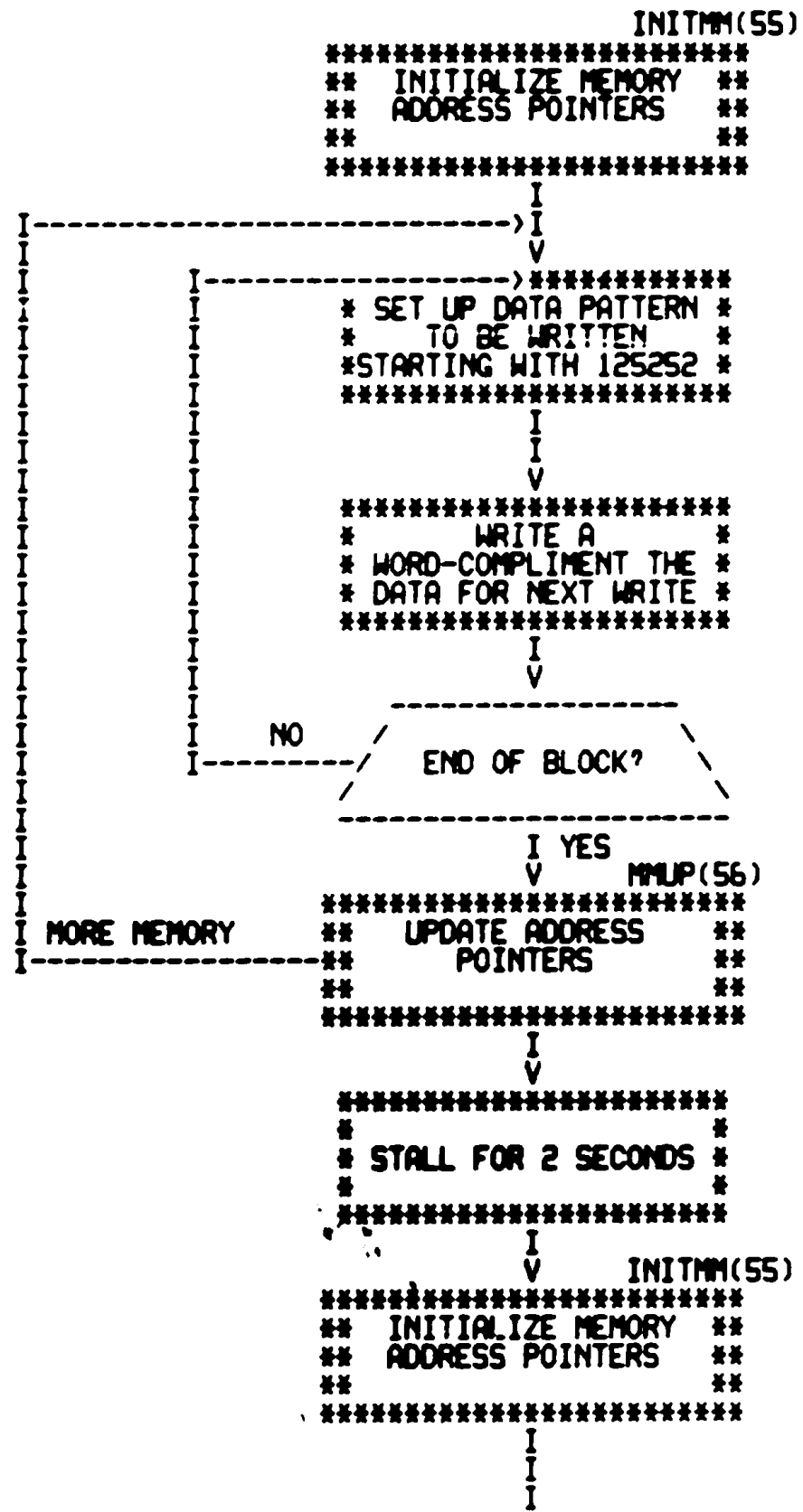


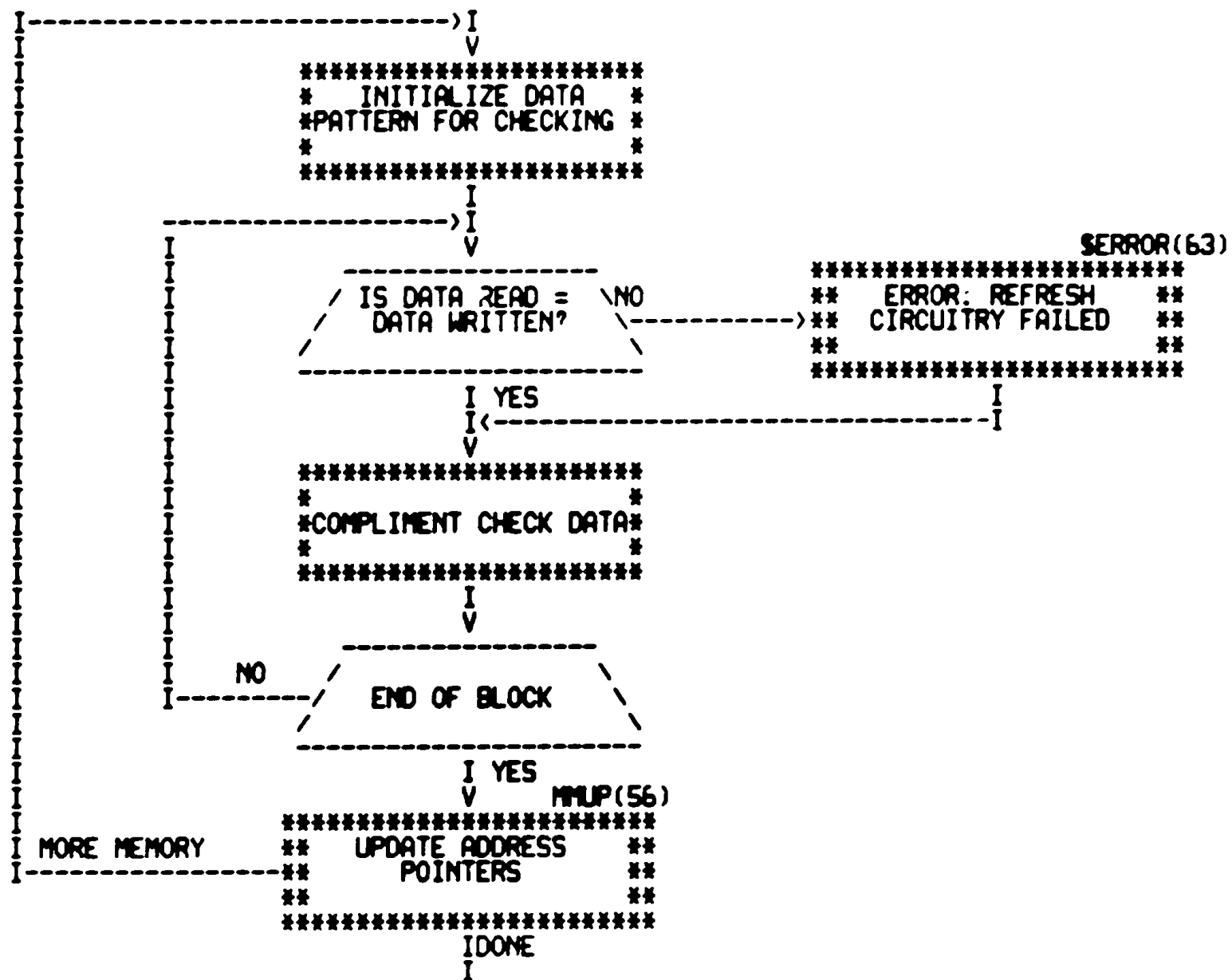


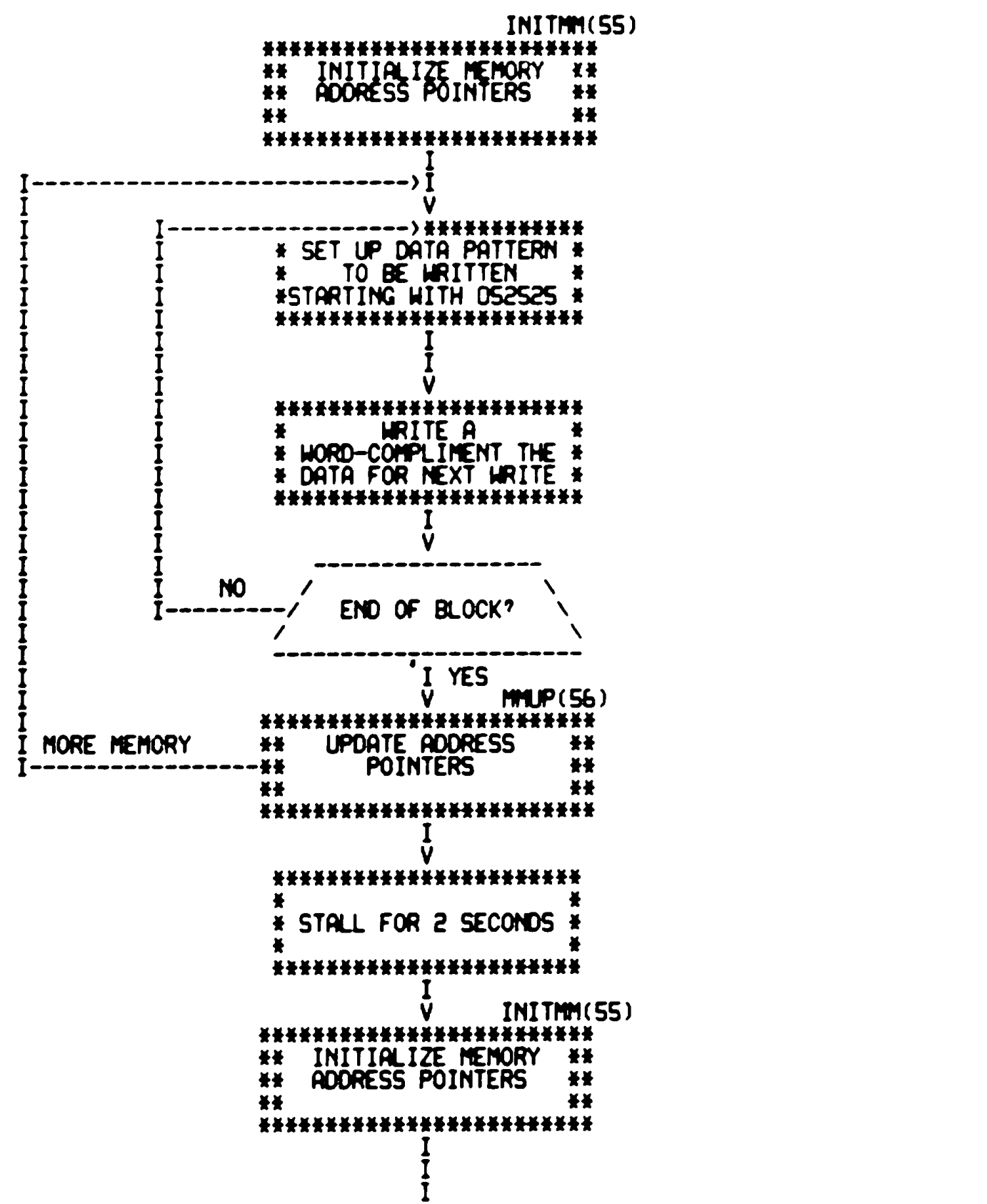


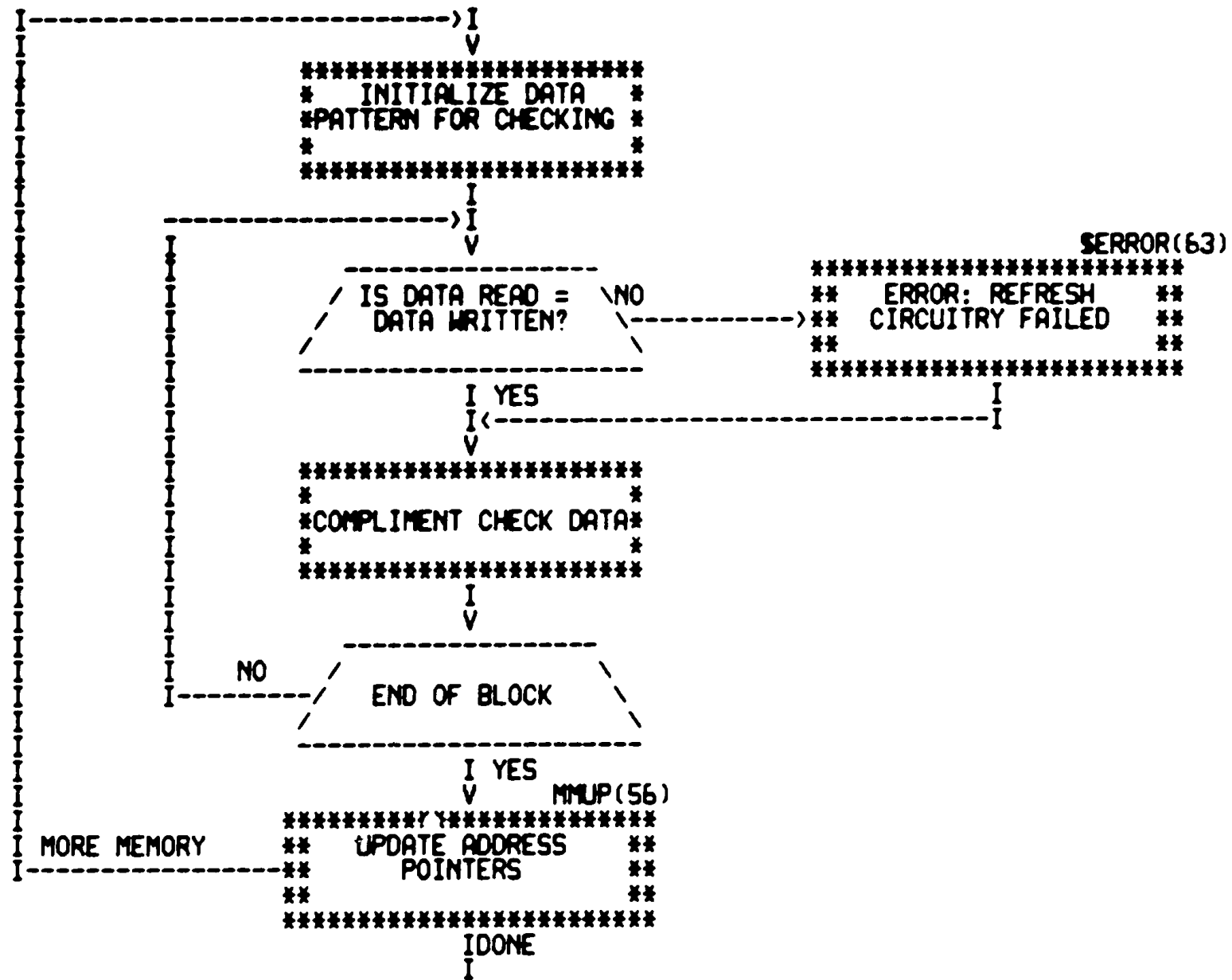


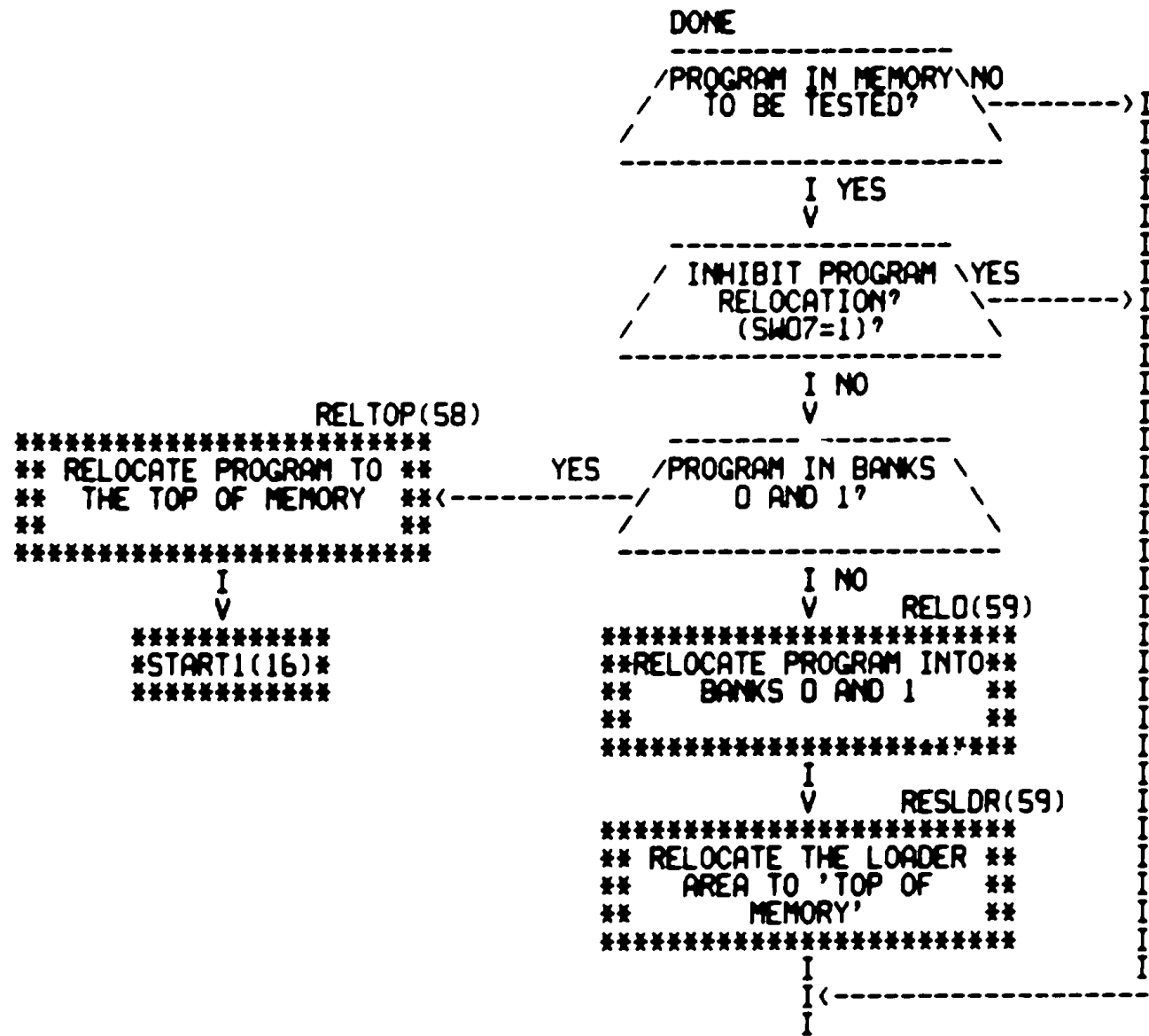


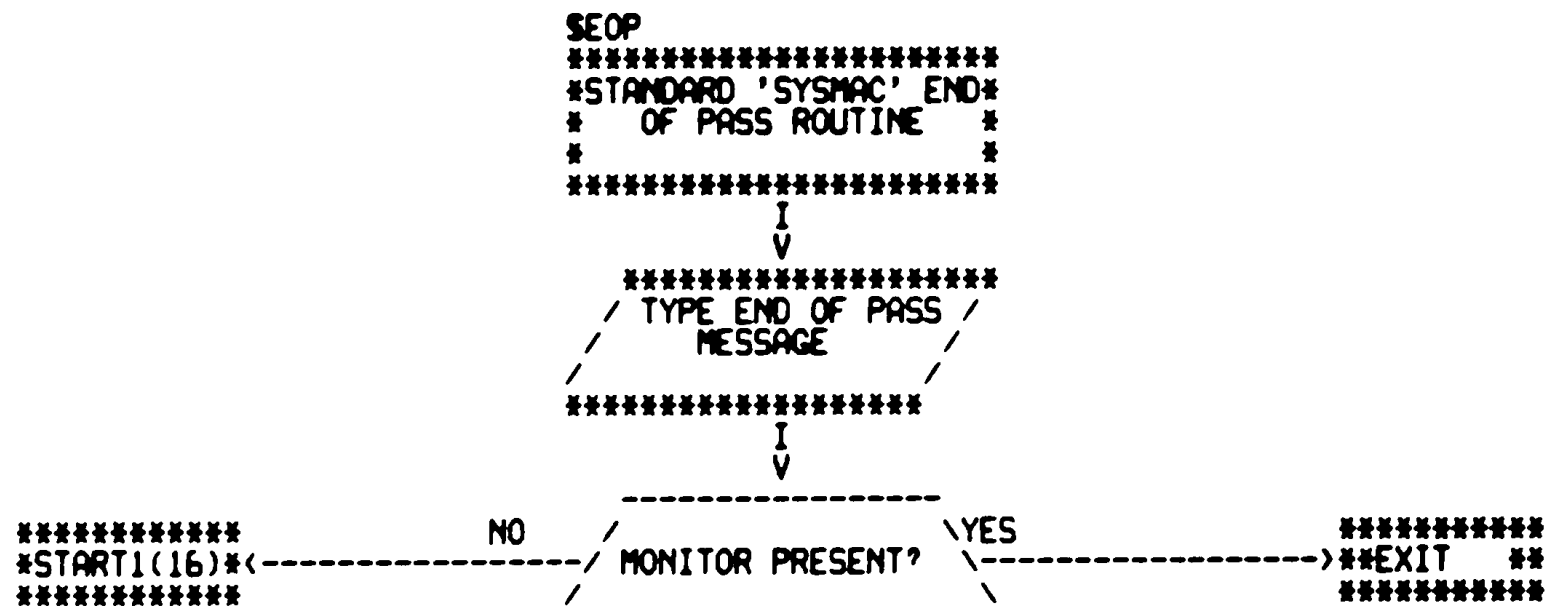










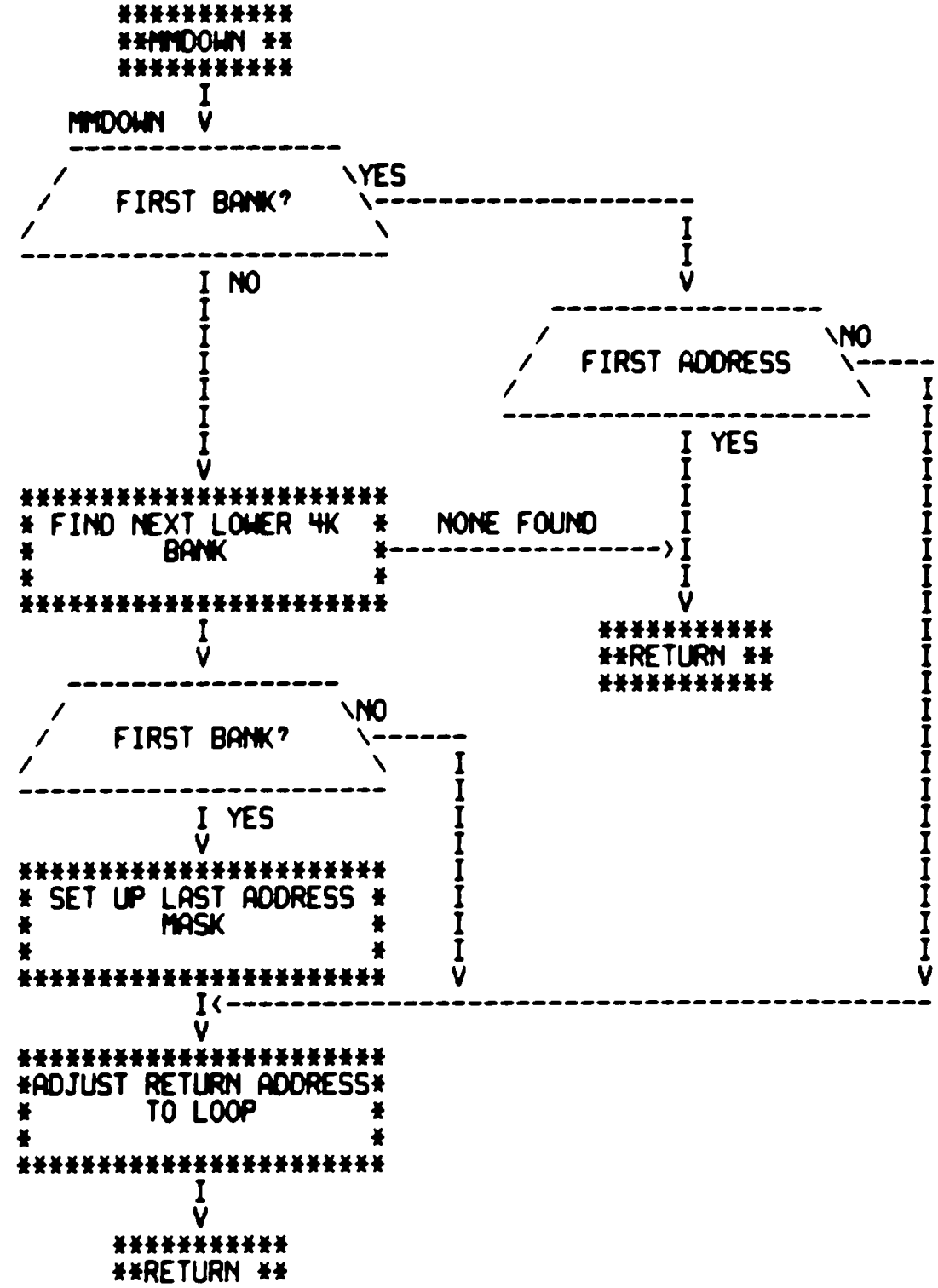
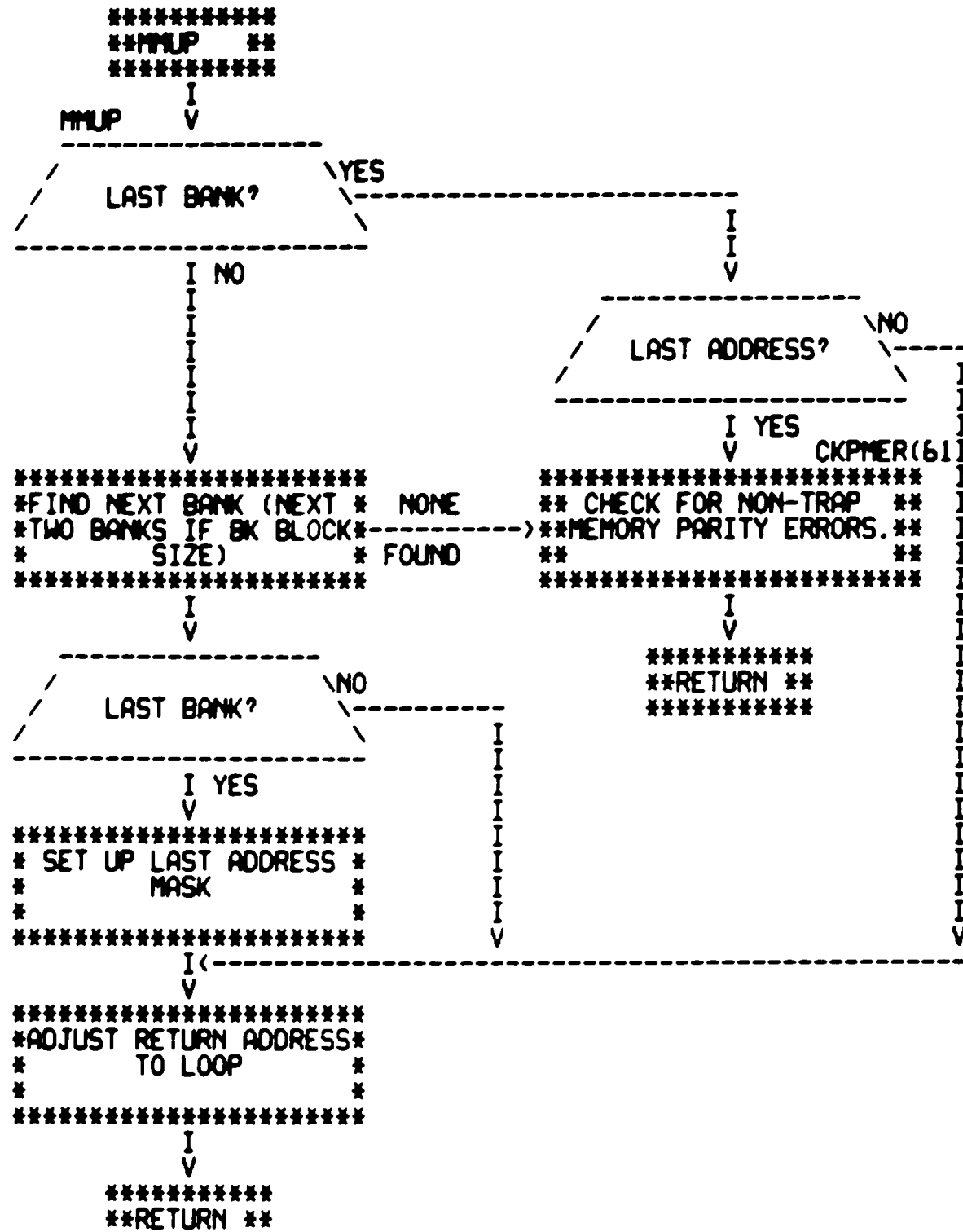




```
*****  
**MMINIT **  
*****  
MMINIT  
*****  
*SET UP ALL THE MEMORY*  
*MANAGEMENT REGISTERS.*  
*  
*****  
*****  
**RETURN **  
*****
```

```
*****  
**INITMM **  
*****  
INITMM  
*****  
*LOCATE FIRST BANK IN *  
* TEST MAP *  
*  
*****  
-----  
/ LAST BANK AS \ NO  
/ WELL? \  
-----  
I YES  
V  
*****  
*  
*SET UP LAST ADR MASK *  
*  
*****  
I<  
V  
-----  
/ BK BLOCK SIZE? \ NO  
-----  
I YES  
V  
*****  
* SET UP SECOND BANK *  
* POINTERS *  
*  
*****  
I<  
V  
-----  
/ FIRST BANK FLAG \ NO  
/ SET? \  
-----  
I YES  
V  
*****  
* PUT FIRST ADDRESS *  
* INTO ADDRESS POINTER *  
*  
*****  
I  
I  
I  
I  
I  
I
```

```
*****  
**INITDN **  
*****  
INITDN  
*****  
* SET UP ADDRESS *  
* POINTERS TO LAST *  
* POSSIBLE ADDRESS *  
*****  
I  
V MPOWN(56)  
*****  
**SEARCH DOWNWARDS FOR **  
**TOP BANK IN TEST MAP **  
**  
*****  
I  
V  
-----  
/ LAST BANK FLAG \ NO  
/ SET? \  
-----  
I YES  
V  
*****  
*PUT LAST ADDRESS INTO*  
* ADDRESS POINTER *  
*  
*****  
I  
I  
I  
I  
I  
I  
I  
I  
I  
I  
I<  
INITEX  
*****  
* SAVE RETURN PC FOR *  
* LOOP ADDRESS *  
*  
*****  
I  
V  
*****  
**RETURN **  
*****  
I  
I  
I  
I  
I  
I
```



```

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

```

```

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

```

```

*****
**SETCON **
*****
      I
SETCON V INITMM(55)
*****
** INITIALIZE ADDRESS **
** POINTERS **
** **
*****
      I
      V
*****
* PUT THE CONTENTS OF *
* R0 INTO MEMORY *
* *
*****
      I
      V MMUP(56)
*****
** UPDATE ADDRESS **
** POINTERS **
** **
*****
      I
      V IDONE
*****
**RETURN **
*****

```

```

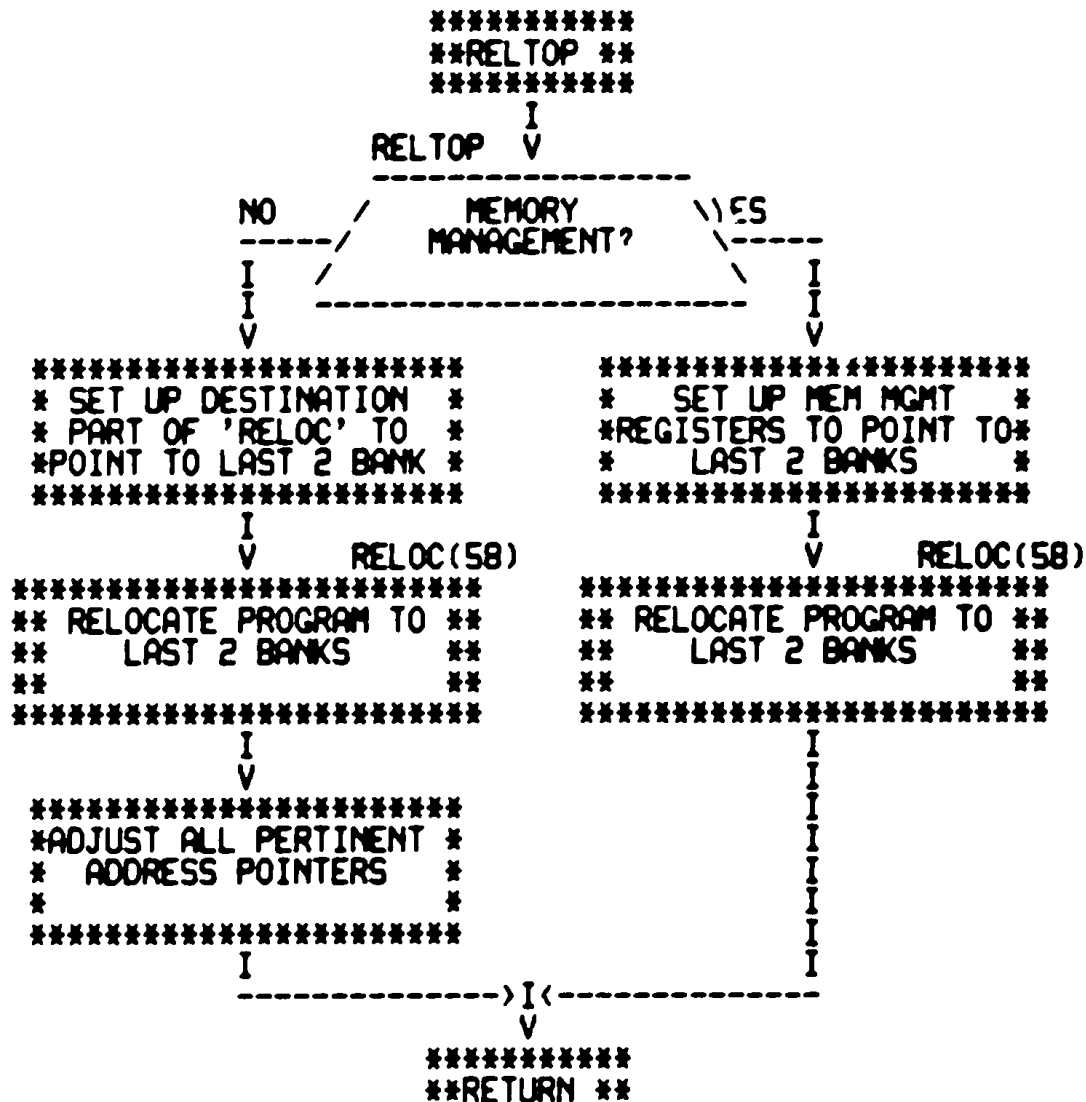
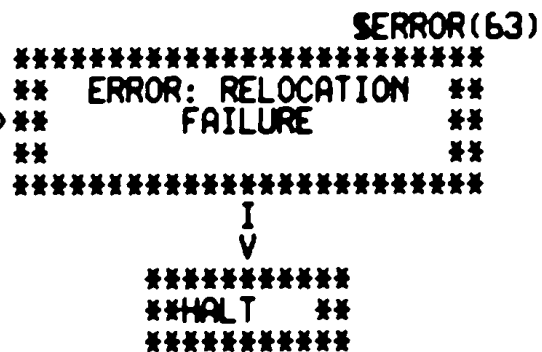
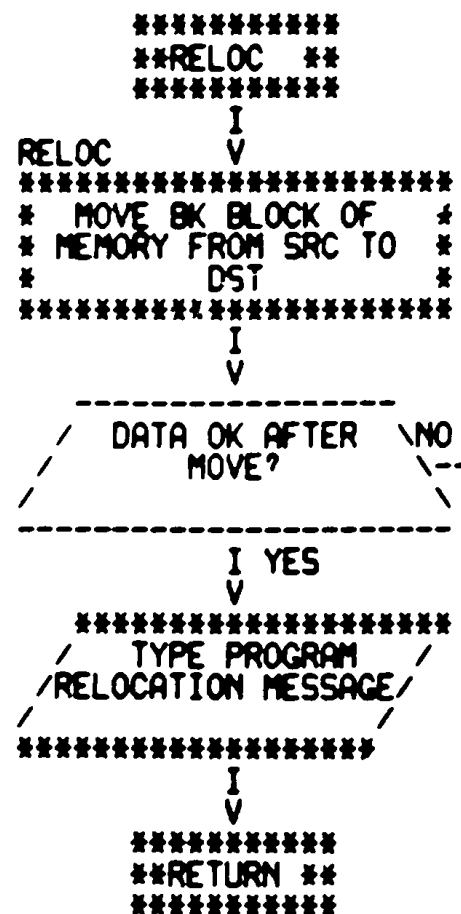
*****
**ROTATE **
*****
      I
ROTATE V
*****
* ROTATE C-BIT THROUGH *
* 16 BIT WORD. *
*****
      I
      V
*****
**RETURN **
*****

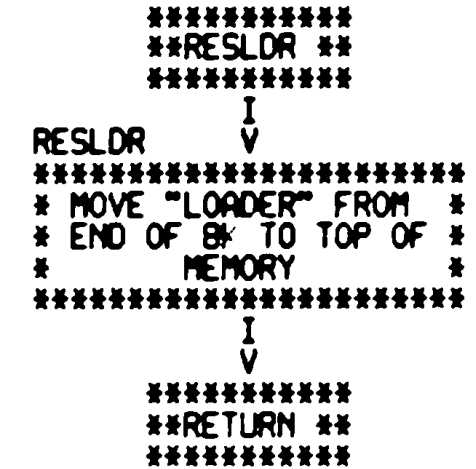
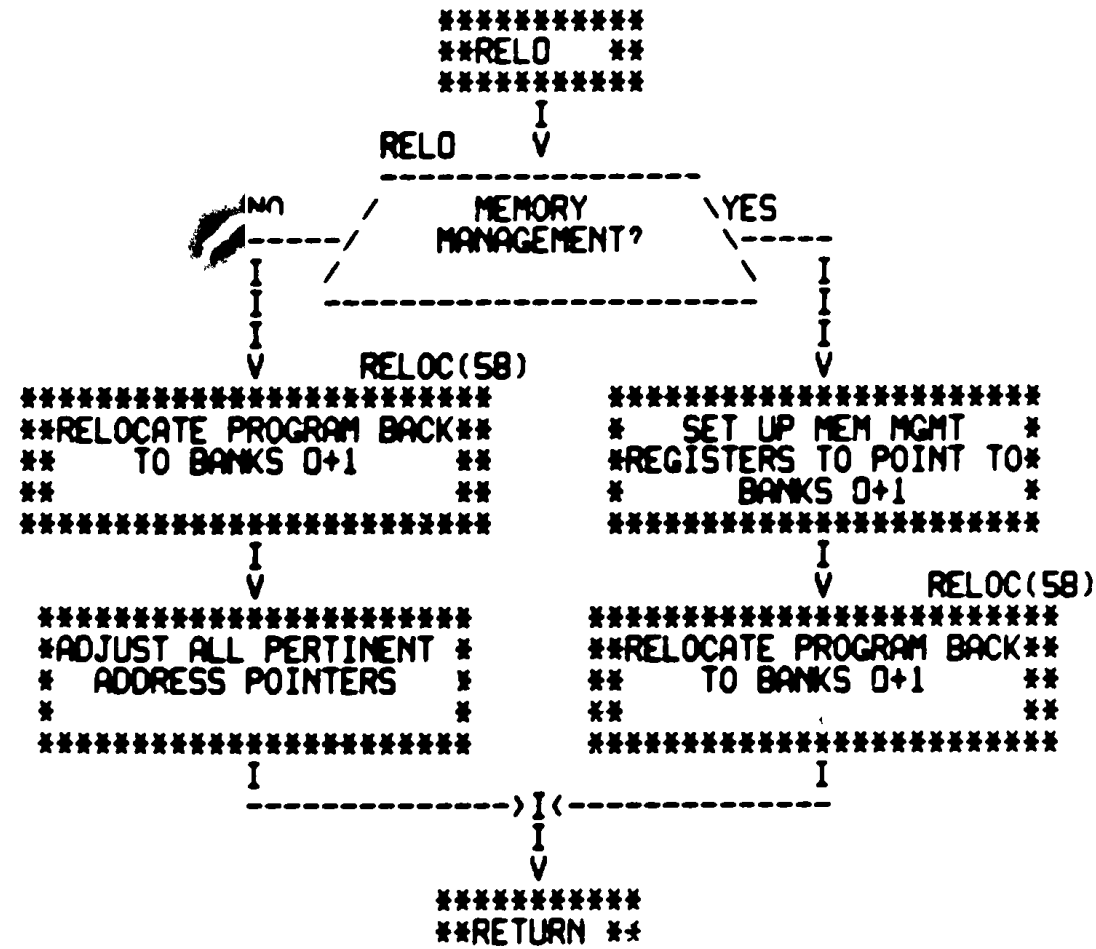
```

```

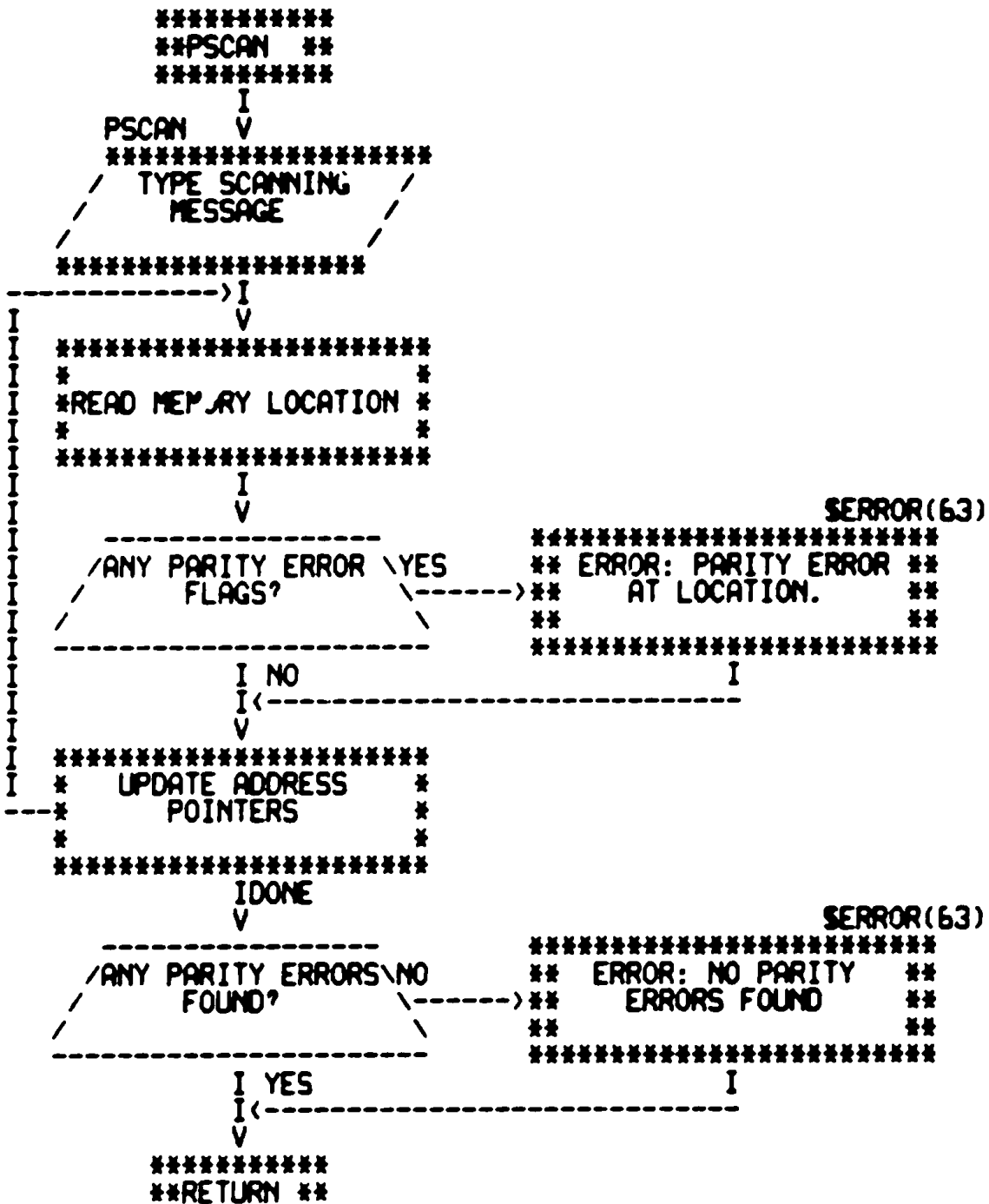
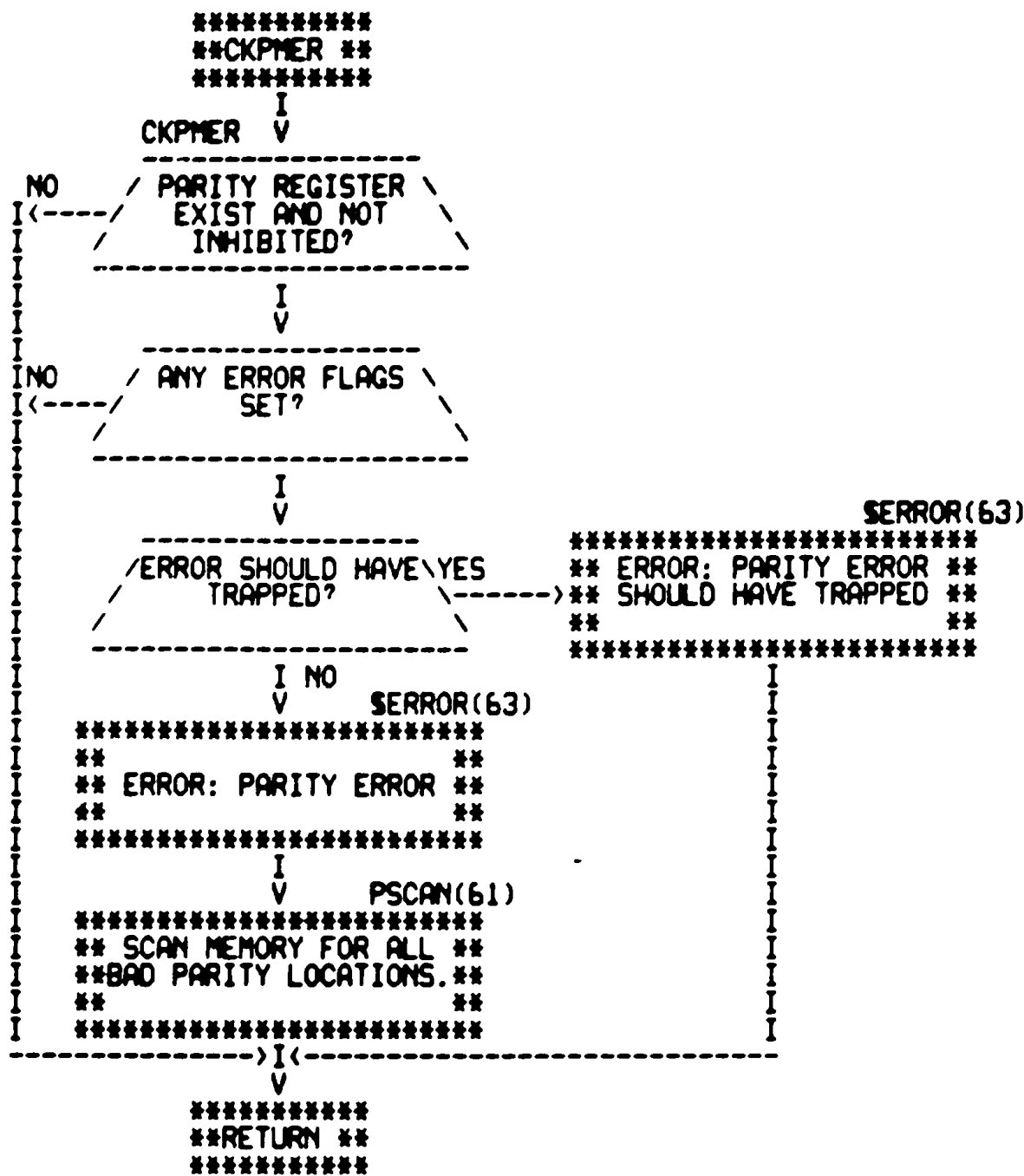
*****
**W3X9 **
*****
      I
W3X9 V
*****
* WRITE 256 WORD WITH 4 *
* OF A PATTERN THEN 4 *
* OF ANOTHER *
*****
      I
      V
*****
**RETURN **
*****

```









```

*****
**SPRNT **-->I
*****

*****
**SPRNT0 **-->I
*****

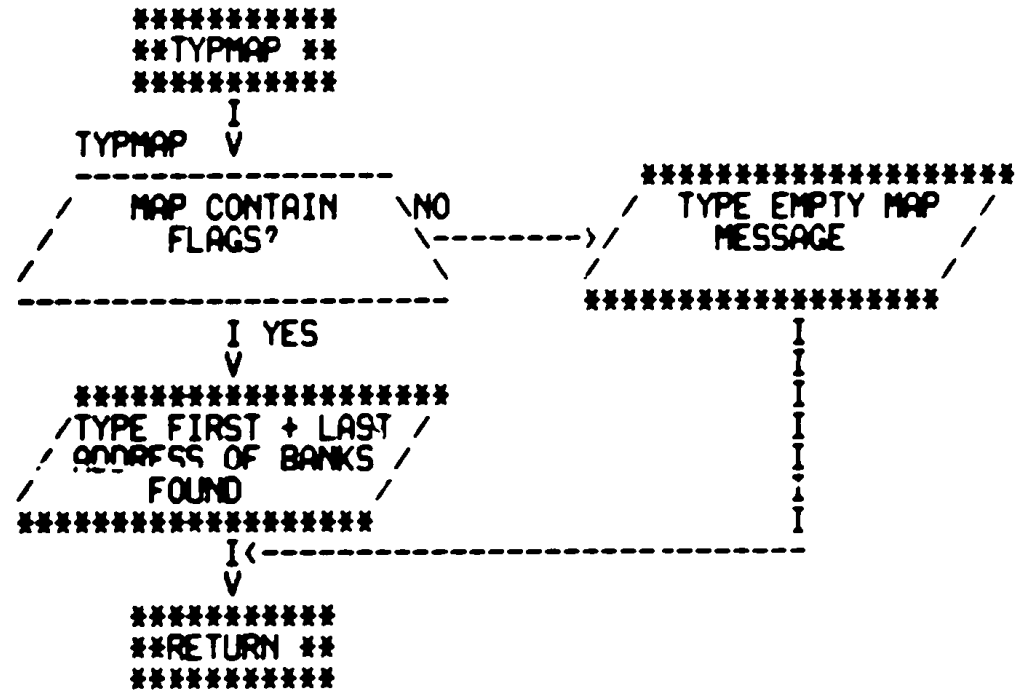
*****
**SPRNT1 **-->I
*****

*****
**SPRNT2 **-->I
*****

*****
**SPRNT3 **-->I
*****

*****
**ROUTINES TO SET UP *
** DATA FOR ERROR *
** TYPE CUTS. *
*****
I
V
*****
**RETURN **

```





```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

*****
** .END **

```





F07

MAINDEC-11-DZQMC-D 0-124K MEMORY EXERCISER. (16K VERSION)  
FLOW CHART CROSS REFERENCE LIST

DECFL0 VER 00.12 08-SEP-77 10:00 PAGE 66

SEQ 0083

SSCOPE	63#			
STYPAD	63	63#		
STYPDS	10	10	63	63#
STYPE	63	63#		
STYPOC	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.
;#

```

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

.SBTTL OPERATIONAL SWITCH SETTINGS

```

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

```

.SBTTL BASIC DEFINITIONS

```

;#INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100 STACK= 1100
;#EQUIV EMT,ERROR ;;BASIC DEFINITION OF ERROR CALL
;#EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL

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

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

```

```
57      000007      PC=      X7      ;;PROGRAM COUNTER
58
59      :#PRIORITY LEVEL DEFINITIONS
60      000000      PR0=      0      ;;PRIORITY LEVEL 0
61      000040      PR1=      40     ;;PRIORITY LEVEL 1
62      000100      PR2=     100     ;;PRIORITY LEVEL 2
63      000140      PR3=     140     ;;PRIORITY LEVEL 3
64      000200      PR4=     200     ;;PRIORITY LEVEL 4
65      000240      PR5=     240     ;;PRIORITY LEVEL 5
66      000300      PR6=     300     ;;PRIORITY LEVEL 6
67      000340      PR7=     340     ;;PRIORITY LEVEL 7
68
69      :#"SWITCH REGISTER" SWITCH DEFINITIONS
70      100000      SW15=   100000
71      040000      SW14=    40000
72      020000      SW13=    20000
73      010000      SW12=    10000
74      004000      SW11=     4000
75      002000      SW10=     2000
76      001000      SW09=     1000
77      000400      SW08=     400
78      000200      SW07=     200
79      000100      SW06=     100
80      000040      SW05=     40
81      000020      SW04=     20
82      000010      SW03=     10
83      000004      SW02=      4
84      000002      SW01=      2
85      000001      SW00=      1
86      .EQUIV      SW09,SW49
87      .EQUIV      SW08,SW48
88      .EQUIV      SW07,SW47
89      .EQUIV      SW06,SW46
90      .EQUIV      SW05,SW45
91      .EQUIV      SW04,SW44
92      .EQUIV      SW03,SW43
93      .EQUIV      SW02,SW42
94      .EQUIV      SW01,SW41
95      .EQUIV      SW00,SW40
96
97      :#DATA BIT DEFINITIONS (BIT00 TO BIT15)
98      100000      BIT15=  100000
99      040000      BIT14=   40000
100     020000      BIT13=   20000
101     010000      BIT12=   10000
102     004000      BIT11=    4000
103     002000      BIT10=    2000
104     001000      BIT09=    1000
105     000400      BIT08=    400
106     000200      BIT07=    200
107     000100      BIT06=    100
108     000040      BIT05=    40
109     000020      BIT04=    20
110     000010      BIT03=    10
111     000004      BIT02=     4
112     000002      BIT01=     2
```

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

```

169      172346      KIPAR3= 172346
170      172350      KIPAR4= 172350
171      172352      KIPAR5= 172352
172      172354      KIPAR6= 172354
173      172356      KIPAR7= 172356
174
175      000000      UP = 0 ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
176      000006      RW = 6 ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
177
178      ;* PARITY MEMORY DEFINITIONS.
179      000001      AE=1 ;PARITY ACTION ENABLE
180      000114      PARVEC=114 ;PARITY TRAP VECTOR
181
182      ;* MISCELLANEOUS ASSIGNMENTS
183      017777      MASK4K= 17777 ;MASK FOR 4K ADDRESS BANK BOUNDRY.
184
185      ;* CACHE REGISTER DEFINITIONS.
186      177746      IMPCHE= 177746
187
188      .SBTTL TRAP CATCHER
189
190      000000      .=0
191      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
192      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
193      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
194      000174      .=174
195      000174      000000      DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
196      000176      000000      SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
197
198      000200      000137      002632      .SBTTL STARTING ADDRESS(ES)
199      000204      000167      002430      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
200      JMP SELECT ;;STARTING ADDRESS TO ALLOW THE OPERATOR TO
201      ;SELECT VARIOUS PARAMETERS.
202      000210      000167      000064      JMP RESTAR ;;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
203      000214      000167      000064      JMP RESTOR ;;RESTORE LOADERS TO END OF MEMORY AND HALT.
204      000220      000167      003346      JMP TIMOUT ;;TYPE OUT MEMORY MAP, BYTE BY BYTE.
205
206      000004      000004      .=ERRVEC
207      000006      025022      .WORD ERRTRP
208      .WORD 0
209
210      .SBTTL ACT11 HOOKS
211
212      ;*****
213      ;HOOKS REQUIRED BY ACT11
214      000010      000010      $SVPC=. ;SAVE PC
215      000046      000046      .=46
216      000046      014136      $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
217      000052      000052      .=52
218      000052      040000      .WORD BIT14 ;;2)SET LOC.52 TO BIT14
219      000010      000010      .=$$VPC ;; RESTORE PC

```



```

219      000300      .-300
220
221
222
223
224
225
226 000300 005005
227 000302 000401
228 000304 010705
229 000306 012706 001100
230 000312 005767 001206
231 000316 001002
232 000320 000167 002322
233 000324 005767 000256
234 000330 001470
235 000332 032737 000001 177572
236 000340 001034
237 000342 012700 172300
238 000346 012701 000010
239 000352 012720 077406
240 000356 005301
241 000360 001374
242 000362 012700 172340
243 000366 005020
244 000370 012720 000200
245 000374 012720 000400
246 000400 012720 000600
247 000404 012720 001000
248 000410 012720 001200
249 000414 012720 001400
250 000420 012720 007600
251 000424 012737 000001 177572
252 000432 005000
253 000434 016701 000142
254 000440 016702 000140
255 000444 006202
256 000446 006001
257 000450 103404
258 000452 062700 000200
259 000456 100372
260 000460 000000
261 000462 010037 172340
262 000466 000137 000472
263 000472 062700 000200
264 000476 006202
265 000500 006001
266 000502 103373
267 000504 010037 172342
268 000510 000410
269 000512 016700 000052
270 000516 062700 001100
271 000522 010006
272 000524 062700 177432
273 000530 000110
274 000532 022767 000003 000042

```

```

*****
* 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".
*****
RESTAR: CLR R5 ; CLEAR FLAG TO INDICATE RESTART.
        BR REST1 ; GO RESTORE PROGRAM BEFORE RESTARTING.
RESTOR: MOV PC, R5 ; PUT DATA INTO FLAG FOR RESTORE.
REST1: MOV #STACK, SP ; SET UP THE STACK POINTER.
        TST MEMMAP ; CHECK IF THE MEMORY HAS BEEN MAPPED.
        BNE REST2 ; BR IF MEMORY MAPPED.
        JMP STARTA ; GO START
REST2: TST MMAPA ; CHECK IF MEM MGMT AVAILABLE.
        BEQ 10$ ; BR IF NO MEM MGMT.
        BIT #BIT0, @#SRO ; CHECK IF MEM MGMT ACTIVE.
        BNE 2$ ; BR IF MEM MGMT ALREADY SET UP.
        MOV #KIP0R0, R0 ; POINT TO FIRST MEM MGMT DATA REG.
        MOV #8, R1 ; SET UP COUNTER.
1$: MOV #077406, (R0)+ ; MAP FIRST 28K 1-FOR-1.
        DEC R1 ; COUNT REGESTERS.
        BNE 1$ ; BR IF MORE REG.
        MOV #KIPAR0, R0 ; POINT TO FIRST MEM MGMT ADDRESS REG.
        CLR (R0)+ ; PAR0 MAPPED INTO BANK0.
        MOV #200, (R0)+ ; PAR1 MAPPED INTO BANK1.
        MOV #400, (R0)+ ; PAR2 MAPPED INTO BANK2.
        MOV #600, (R0)+ ; PAR3 MAPPED INTO BANK3.
        MOV #1000, (R0)+ ; PAR4 MAPPED INTO BANK4.
        MOV #1200, (R0)+ ; PAR5 MAPPED INTO BANK5.
        MOV #1400, (R0)+ ; PAR6 MAPPED INTO BANK6.
        MOV #7600, (R0)+ ; PAR7 MAPPED INTO BANK37.
        MOV #BIT0, @#SRO ; ENABLE MEM MGMT.
        CLR R0 ; INIT TEMP PAR REG.
2$: MOV PRGMAP, R1 ; GET THE PROGRAM MAP...LO 64K.
        MOV PRGMAP+2, R2 ; ...HI 64K.
3$: ASR R2 ; SHIFT THE MAP POINTER...HI
        ROR R1 ; ...LO.
        BCS 4$ ; BR WHEN FIRST BANK FOUND.
        ADD #200, R0 ; UPDATE TMP PAR TO NEXT BANK.
        BPL 3$ ; BR IF MORE.
        HALT ; FATAL ERROR!!! MAP EMPTY?
4$: MOV R0, @#KIPAR0 ; PUT TEMP PAR INTO FIRST PAR.
        JMP @#5$ ; JUMP INTO PROGRAM IF NOT THERE ALREADY.
5$: ADD #200, R0 ; KEEP UPDATING TEMP PAR REG.
        ASR R2 ; SHIFT POINTER...HI
        ROR R1 ; ...LO
        BCC 5$ ; BR IF TOP BANK NOT YET FOUND.
        MOV R0, @#KIPAR1 ; SET UP SECOND PROGRAM ANK POINTER.
        BR 20$ ; BR TO RELOCATE SECTION.
10$: MOV RELOCF, R0 ; GET RELOCATION FACTOR.
        ADD #STACK, R0 ; SET UP STACK POINTER.
        MOV R0, SP ; SET STACK TO RELOCATE PROGRAM.
        ADD #20$-STACK, R0 ; ADJUST R0 TO RELOCATED "20$" ADDRESS.
        JMP (R0) ; GO TO "20$" (RELOCATED).
20$: CMP #3, PRGMAP ; CHECK IF PROGRAM IS IN BANKS 0 AND 1.

```

L07

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMC0.P11 26-JUL-77 15:01 ACT11 HOOKS

MACY11 30(1046) 08-SEP-77 10:19 PAGE 6

SEQ 0089

275	000540	001402	
276	000542	004767	016222
277	000546	005705	
278	000550	001006	
279	000552	005067	000412
280	000556	105067	000320
281	000562	000167	005260
282	000566	004767	016404
283	000572	000000	
284	000574	000167	002046
285			
286			
287			
288	00060C	000000	
289	000602	000000	000000
290	000606	000000	

```

21$: BEQ 21$ ;BR IF IN BANKS 0 AND 1.
      JSR PC, RELO ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
      YST RS ;CHECK RESTART/RESTORE FLAG.
      BNE 22$ ;BR IF RESTORE.
      CLR $TIMES ;CLEAN UP BEFORE STARTING.
      CLR $STNM
      JMP START1
22$: JSR PC, RESLDR ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
      HALT ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
      JMP STARTA ;HALT AFTER RESTORING THE LOADERS.
                    ;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.
RELOC: .WORD 0 ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
PRGMAP: .WORD 0,0 ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
MMAVA: .WORD 0 ;MEMORY MANAGEMENT AVAILABLE FLAG.

```

```

291      .SBTTL  POWER DOWN AND UP ROUTINES
292
293      ;*****
294      :POWER DOWN ROUTINE
295      $PWRDN: MOV    $SILLUP, @#PWRVEC ;:SET FOR FAST UP
296      MOV    #340, @#PWRVEC+2 ;:PRIO:7
297      MOV    R0, -(SP) ;:PUSH R0 ON STACK
298      MOV    R1, -(SP) ;:PUSH R1 ON STACK
299      MOV    R2, -(SP) ;:PUSH R2 ON STACK
300      MOV    R3, -(SP) ;:PUSH R3 ON STACK
301      MOV    R4, -(SP) ;:PUSH R4 ON STACK
302      MOV    R5, -(SP) ;:PUSH R5 ON STACK
303      MOV    @SWR, -(SP) ;:PUSH @SWR ON STACK
304      MOV    SP, $SAVR6 ;:SAVE SP
305      MOV    $SPWRUP, @#PWRVEC ;:SET UP VECTOR
306      HALT
307      BR     .-2 ;:HANG UP
308
309      ;*****
310      :POWER UP ROUTINE
311      $PWRUP: MOV    $SILLUP, @#PWRVEC ;:SET FOR FAST DOWN
312      MOV    $SAVR6, SP ;:GET SP
313      CLR    $SAVR6 ;:WAIT LOOP FOR THE TTY
314      IS:   INC    $SAVR6 ;:WAIT FOR THE INC
315      BNE   IS ;:OF WORD
316      MOV    (SP)+, @SWR ;:POP STACK INTO @SWR
317      MOV    (SP)+, R5 ;:POP STACK INTO R5
318      MOV    (SP)+, R4 ;:POP STACK INTO R4
319      MOV    (SP)+, R3 ;:POP STACK INTO R3
320      MOV    (SP)+, R2 ;:POP STACK INTO R2
321      MOV    (SP)+, R1 ;:POP STACK INTO R1
322      MOV    (SP)+, R0 ;:POP STACK INTO R0
323      MOV    $SPWRDN, @#PWRVEC ;:SET UP THE POWER DOWN VECTOR
324      MOV    #340, @#PWRVEC+2 ;:PRIO:7
325      JSR    R5, $PRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
326      $PWRMG: .WORD PWRMSG ;:POWER FAIL MESSAGE POINTER
327      $PWRAD: .WORD (PC)+, (SP) ;:RESTART AT RESTART
328      RTI ;:RESTART ADDRESS
329      $SILLUP: HALT ;:THE POWER UP SEQUENCE WAS STARTED
330      BR     .-2 ;:BEFORE THE POWER DOWN WAS COMPLETE
331      $SAVR6: 0 ;:PUT THE SP HERE
332

```

.SBTTL COMMON TAGS

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

333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388

SCMTAG: . =1100 ; ; START OF COMMON TAGS  
STSTNM: .WORD 0 ; ; CONTAINS THE TEST NUMBER  
SERFLG: .BYTE 0 ; ; CONTAINS ERROR FLAG  
SICNT: .WORD 0 ; ; CONTAINS SUBTEST ITERATION COUNT  
SLPADR: .WORD 0 ; ; CONTAINS SCOPE LOOP ADDRESS  
SLPERR: .WORD 0 ; ; CONTAINS SCOPE RETURN FOR ERRORS  
SERTTL: .WORD 0 ; ; CONTAINS TOTAL ERRORS DETECTED  
SITEMB: .BYTE 0 ; ; CONTAINS ITEM CONTROL BYTE  
SERMAX: .BYTE 1 ; ; CONTAINS MAX. ERRORS PER TEST  
SERPC: .WORD 0 ; ; CONTAINS PC OF LAST ERROR INSTRUCTION  
SGDADR: .WORD 0 ; ; CONTAINS ADDRESS OF 'GOOD' DATA  
SBDADR: .WORD 0 ; ; CONTAINS ADDRESS OF 'BAD' DATA  
SGDAT: .WORD 0 ; ; CONTAINS 'GOOD' DATA  
SBDAT: .WORD 0 ; ; CONTAINS 'BAD' DATA  
 ; ; RESERVED--NOT TO BE USED  
SAUTOB: .BYTE 0 ; ; AUTOMATIC MODE INDICATOR  
SINTAG: .BYTE 0 ; ; INTERRUPT MODE INDICATOR  
SWR: .WORD DSWR ; ; ADDRESS OF SWITCH REGISTER  
DISPLAY: .WORD DDISP ; ; ADDRESS OF DISPLAY REGISTER  
\$TKS: 177560 ; ; TTY KBD STATUS  
\$TKB: 177562 ; ; TTY KBD BUFFER  
\$TPS: 177564 ; ; TTY PRINTER STATUS REG. ADDRESS  
\$TPB: 177566 ; ; TTY PRINTER BUFFER REG. ADDRESS  
\$NULL: .BYTE 0 ; ; CONTAINS NULL CHARACTER FOR FILLS  
\$FILLS: .BYTE 2 ; ; CONTAINS # OF FILLER CHARACTERS REQUIRED  
\$FILLC: .BYTE 12 ; ; INSERT FILL CHARS. AFTER A "LINE FEED"  
\$STPFLG: .BYTE 0 ; ; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)  
\$STMP0: .WORD 0 ; ; USER DEFINED  
\$STMP1: .WORD 0 ; ; USER DEFINED  
\$STMP2: .WORD 0 ; ; USER DEFINED  
\$STMP3: .WORD 0 ; ; USER DEFINED  
\$TIMES: 0 ; ; MAX. NUMBER OF ITERATIONS  
\$ESCAPE: 0 ; ; ESCAPE ON ERROR ADDRESS  
\$BELL: .ASCIZ <207><377><377> ; ; CODE FOR BELL  
\$QUES: .ASCIZ /?/ ; ; QUESTION MARK  
\$CRLF: .ASCIZ <15> ; ; CARRIAGE RETURN  
\$LF: .ASCIZ <12> ; ; LINE FEED  
\*\*\*\*\*  
.SBTTL APT MAILBOX-ETABLE  
\*\*\*\*\*  
.EVEN  
\$MAIL: ; ; APT MAILBOX  
\$MSGTY: .WORD AMSGTY ; ; MESSAGE TYPE CODE  
\$FATAL: .WORD AFATAL ; ; FATAL ERROR NUMBER  
\$TESTN: .WORD ATESTN ; ; TEST NUMBER

389 001212 000000  
 390 001214 000000  
 391 001216 000000  
 392 001220 000000  
 393 001222 000000  
 394 001224  
 395 001224 000  
 396 001225 000  
 397 001226 000000  
 398 001230 000000  
 399 001232 000000  
 400  
 401  
 402  
 403  
 404  
 405  
 406 001234 000  
 407 001235 000  
 408  
 409  
 410  
 411  
 412 001236 000000  
 413  
 414 001240 000  
 415 001241 000  
 416 001242 000000  
 417 001244 000  
 418 001245 000  
 419 001246 000000  
 420 001250 000  
 421 001251 000  
 422 001252 000000  
 423 001254 000000  
 424 001256 000000  
 425 001260 000000  
 426 001262 000000  
 427 001264 000000  
 428 001266 000000  
 429 001270 000000  
 430 001272 000000  
 431 001274 000000  
 432 001276 000000  
 433 001300 000000  
 434 001302 000000  
 435 001304 000000  
 436 001306 000000  
 437 001310 000000  
 438 001312 000000  
 439 001314 000000  
 440 001316 000000  
 441 001320 000000  
 442 001322 000000  
 443 001324 000000  
 444 001326 000000

\$PASS: .WORD APASS  
 \$DEVCT: .WORD ADEVCT  
 \$UNIT: .WORD AUNIT  
 \$MSGAD: .WORD AMSGAD  
 \$MSGLG: .WORD AMSGLG  
 \$ETABLE:  
 \$ENV: .BYTE AENV  
 \$ENVM: .BYTE AENVM  
 \$SWREG: .WORD ASWREG  
 \$USWR: .WORD AUSWR  
 \$CPUOP: .WORD ACPUOP  
 \*  
 \*  
 \*  
 \*  
 \*  
 \*  
 \*  
 \*  
 \$MAMS1: .BYTE AMAMS1  
 \$MTYP1: .BYTE AMTYP1  
 \*  
 \*  
 \*  
 \*  
 \$MADR1: .WORD AMADR1  
 \*  
 \$MAMS2: .BYTE AMAMS2  
 \$MTYP2: .BYTE AMTYP2  
 \$MADR2: .WORD AMADR2  
 \$MAMS3: .BYTE AMAMS3  
 \$MTYP3: .BYTE AMTYP3  
 \$MADR3: .WORD AMADR3  
 \$MAMS4: .BYTE AMAMS4  
 \$MTYP4: .BYTE AMTYP4  
 \$MADR4: .WORD AMADR4  
 \$VECT1: .WORD AVECT1  
 \$VECT2: .WORD AVECT2  
 \$BASE: .WORD ABASE  
 \$DEVN: .WORD ADEVN  
 \$CDW1: .WORD ACDW1  
 \$CDW2: .WORD ACDW2  
 \$DDW0: .WORD ADDW0  
 \$DDW1: .WORD ADDW1  
 \$DDW2: .WORD ADDW2  
 \$DDW3: .WORD ADDW3  
 \$DDW4: .WORD ADDW4  
 \$DDW5: .WORD ADDW5  
 \$DDW6: .WORD ADDW6  
 \$DDW7: .WORD ADDW7  
 \$DDW8: .WORD ADDW8  
 \$DDW9: .WORD ADDW9  
 \$DDW10: .WORD ADDW10  
 \$DDW11: .WORD ADDW11  
 \$DDW12: .WORD ADDW12  
 \$DDW13: .WORD ADDW13  
 \$DDW14: .WORD ADDW14  
 \$DDW15: .WORD ADDW15

;; PASS COUNT  
 ;; DEVICE COUNT  
 ;; I/O UNIT NUMBER  
 ;; MESSAGE ADDRESS  
 ;; MESSAGE LENGTH  
 ;; APT ENVIRONMENT TABLE  
 ;; ENVIRONMENT BYTE  
 ;; ENVIRONMENT MODE BITS  
 ;; APT SWITCH REGISTER  
 ;; USER SWITCHES  
 ;; 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, PD0=07, 0=10  
 BIT 10=REAL TIME CLOCK  
 BIT 9=FLOATING POINT PROCESSOR  
 BIT 8=MEMORY MANAGEMENT  
 ;; HIGH ADDRESS, M.S. BYTE  
 ;; MEM. TYPE, BLK#1  
 MEM. TYPE, BYTE — (HIGH BYTE)  
 900 NSEC CORE=001  
 300 NSEC BIPOLAR=002  
 500 NSEC MOS=003  
 ;; HIGH ADDRESS, BLK#1  
 MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE  
 ;; HIGH ADDRESS, M.S. BYTE  
 ;; MEM. TYPE, BLK#2  
 MEM. LAST ADDRESS, BLK#2  
 ;; HIGH ADDRESS, M.S. BYTE  
 ;; MEM. TYPE, BLK#3  
 MEM. LAST ADDRESS, BLK#3  
 ;; HIGH ADDRESS, M.S. BYTE  
 ;; MEM. TYPE, BLK#4  
 MEM. LAST ADDRESS, BLK#4  
 INTERRUPT VECTOR#1, BUS PRIORITY#1  
 INTERRUPT VECTOR#2, BUS PRIORITY#2  
 ;; BASE ADDRESS OF EQUIPMENT UNDER TEST  
 ;; DEVICE MAP  
 CONTROLLER DESCRIPTION WORD#1  
 CONTROLLER DESCRIPTION WORD#2  
 DEVICE DESCRIPTOR WORD#0  
 DEVICE DESCRIPTOR WORD#1  
 DEVICE DESCRIPTOR WORD#2  
 DEVICE DESCRIPTOR WORD#3  
 DEVICE DESCRIPTOR WORD#4  
 DEVICE DESCRIPTOR WORD#5  
 DEVICE DESCRIPTOR WORD#6  
 DEVICE DESCRIPTOR WORD#7  
 DEVICE DESCRIPTOR WORD#8  
 DEVICE DESCRIPTOR WORD#9  
 DEVICE DESCRIPTOR WORD#10  
 DEVICE DESCRIPTOR WORD#11  
 DEVICE DESCRIPTOR WORD#12  
 DEVICE DESCRIPTOR WORD#13  
 DEVICE DESCRIPTOR WORD#14  
 DEVICE DESCRIPTOR WORD#15



APT STATISTICS TABLE

Line	Address	Value	Field	Description
501				*****
502				THE FOLLOWING TAGS ARE USER DEFINED
503				*****
504	001514	000000	\$VERPC: .WORD 0	VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE (SERTYP).
505	001516	070032	RESRVD: .WORD 070032	CORE PARITY REG BITS RESERVED FOR FUTURE USE. NOTE: FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
506				LAST CONTIGUOUS MEMORY ADDRESS (+2)
507	001520	000000	LMAD: .WORD 0	CONTAINS DISPLAY REGISTER IMAGE
508	001522	000000	LDDISP: .WORD 0	MEMORY MAP - EACH BIT CORRESPONDS TO 4K
509	001524		MEMMAP: .WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
510	001524	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
511	001526	000000	.WORD 0	TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
512	001530		TSTMAP: .WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
513	001530	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
514	001532	000000	.WORD 0	SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
515	001534		SAVTST: .WORD 0	TEST EACH BANK ONCE.
516			.WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
517	001534	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
518	001536	000000	.WORD 0	PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
519	001540		PMEMAP: .WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
520	001540	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
521	001542	000000	.WORD 0	BITMAP - WHICH BANKS ARE SELECTED FOR TEST.
522	001544		BITPT: .WORD 0	POINTER TO CURRENT 4K BANK OF MEMORY
523	001544	000000	.WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
524	001546	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
525	001550		TMPPT: .WORD 0	TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
526	001550	000000	.WORD 0	FIRST WORD CONTAINS LOW (0-64K) MAP
527	001552	000000	.WORD 0	SECOND WORD CONTAINS HIGH (64-128K) MAP
528	001554	000000	.WORD 0	LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
529			MMORE: .WORD 0	SET UP BY "INITM" AND "INITDN" ROUTINES.
530				USED BY "MMUP" AND "MMDOWN" ROUTINES.
531	001556	000	SELFLG: .BYTE 0	OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
532	001557	000	FLAGBK: .BYTE 0	BK BLOCK INDICATOR. USED IN "INITM" AND "MMUP".
533	001560	000	OEFLG: .BYTE 0	000/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
534		001562	.EVEN	
535	001562	000000	FSTADR: .WORD 0	FIRST VIRTUAL ADDRESS TO BE TESTED.
536				FIRST ADDRESS IS USER SELECTABLE.
537	001564	000000	TMPFAD: .WORD 0	ADJUSTED FIRST ADDRESS.
538	001566	000000	FADMSK: .WORD 0	BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
539				TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
540	001570	000000	FADMAP: .WORD 0,0	MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
541	001574	000000	LSTADR: .WORD 0	LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
542				LAST ADDRESS IS USER SELECTABLE.
543	001576	000000	TMPHAD: .WORD 0	ADJUSTED LAST ADDRESS.
544	001600	000000	LADMSK: .WORD 0	BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
545				TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
546	001602	000000	LADMAP: .WORD 0,0	MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
547	001606	000000	BLKMSK: .WORD 0	BLOCK MASK, DETERMINES THE BLOCK SIZE.
548	001610	000000	.CONST: .WORD 0	USER SELECTABLE CONSTANT DATA.
549	001612	000004	WMP: .WORD 4	WRITE WRONG PARITY COMMAND
550				
551				*****
552				RELATIVE ADDRESSING TABLE.
553				THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
554				RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.
555				*****
556	001614		RADTAB:	

557 001614 001100  
 558 001616 001516  
 559 001620 002070  
 560 001622 002270  
 561 001624 012000  
 562 001626 002042  
 563 001630 017336  
 564 001632 002332  
 565 001634 000010  
 566 001636 013722

.STACK: STACK ;STACK POINTER INITIAL ADDRESS.  
 .RESRV: RESRVD ;PARITY REGISTER RESERVED BIT MASK ADDRESS.  
 .MPRO: MPRO ;MEMORY PARITY REGISTER TABLE ADDRESS.  
 .MPRX: MPRX ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.  
 .PBTRP: PBTRP ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.  
 .MPPAT: MPPATS ;MEMORY PARITY PATTERN TABLE ADDRESS.  
 .PESRV: PESRV ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.  
 .ERRTB: SERRTB ;ERROR TYPEOUT TABLE PONTER.  
 .EIGHT: 8. ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.  
 .TST32: TST32 ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.

\*\*\*\*\*  
 DATA CONTAINERS FOR ERROR PRINTOUT.  
 \*\*\*\*\*

570 001640 001116 001120 001124 DT1: SERRPC,SGDADR,SGDDAT,SBDDAT,0  
 571 001646 001126 000000  
 572 001652 001514 001116 001120 DT2: SVERPC,SERRPC,SGDADR,SGDDAT,SBDDAT,0  
 573 001660 001124 001126 000000  
 574 001666 001514 001116 001120 DT12: SVERPC,SERRPC,SGDADR,SGDDAT,0  
 575 001674 001124 000000  
 576 001700 001514 001116 001160 DT14: SVERPC,SERRPC,STMPO,SGDADR,0  
 577 001706 001120 000000  
 578 001712 001514 001116 001120 DT15: SVERPC,SERRPC,SGDADR,STMPO,SGDDAT,SBDDAT,0  
 579 001720 001160 001124 001126  
 580 001726 000000  
 581 001730 001514 001116 001160 DT21: SVERPC,SERRPC,STMPO,SGDADR,SGDDAT,SBDDAT,0  
 582 001736 001120 001124 001126  
 583 001744 000000  
 584 001746 001514 001116 001120 DT23: SVERPC,SERRPC,SGDADR,SBDADR,SGDDAT,SBDDAT,0  
 585 001754 001122 001124 001126  
 586 001762 000000  
 587 001764 001514 001116 001122 DT24: SVERPC,SERRPC,SBDADR,0  
 588 001772 000000  
 589 001774 001514 001116 001122 DT25: SVERPC,SERRPC,SBDADR,STMPO,STMP1,0  
 590 002002 001160 001162 000000  
 591 002010 001514 001116 001160 DT26: SVERPC,SERRPC,STMPO,STMP1,0  
 592 002016 001162 000000  
 593 002022 001160 001162 001120 DT30: STMPO,STMP1,SGDADR,SBDDAT,0  
 594 002030 001126 000000  
 595 002034 001166 000000 DT31: STMP3,0  
 596 002040 177777 .WORD -1 ;TABLE TERMINATOR.

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

603 002042 125325 MPPATS: 125325 ;EVEN,ODD  
 604 002044 152652 152652 ;ODD,EVEN  
 605 002046 052452 052452 ;EVEN,ODD  
 606 002050 025125 025125 ;ODD,EVEN  
 607 002052 102070 102070 ;EVEN,EVEN  
 608 002054 072527 072527 ;ODD,ODD  
 609 002056 177777 177777 ;EVEN,EVEN  
 610 002060 107030 107030 ;ODD,ODD  
 611 002062 152525 152525 ;ODD,EVEN  
 612 002064 000000 0 ;EXTRA PATTERN HOLDER FOR



613  
614 002066 000000  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629 002070 172101  
630 002072 000000  
631 002074 000000  
632 002076 000000  
633 002100 172103  
634 002102 000000  
635 002104 000000  
636 002106 000000  
637 002110 172105  
638 002112 000000  
639 002114 000000  
640 002116 000000  
641 002120 172107  
642 002122 000000  
643 002124 000000  
644 002126 000000  
645 002130 172111  
646 002132 000000  
647 002134 000000  
648 002136 000000  
649 002140 172113  
650 002142 000000  
651 002144 000000  
652 002146 000000  
653 002150 172115  
654 002152 000000  
655 002154 000000  
656 002156 000000  
657 002160 172117  
658 002162 000000  
659 002164 000000  
660 002166 000000  
661 002170 172121  
662 002172 000000  
663 002174 000000  
664 002176 000000  
665 002200 172123  
666 002202 000000  
667 002204 000000  
668 002206 000000

MPEND: 0

;FUTURE USE  
;TABLE TERMINATOR

.SBTTL MEMORY PARITY REGISTER ADDRESS TABLE

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

MPRO: 172100 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR1: 172102 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR2: 172104 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR3: 172106 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR4: 172110 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR5: 172112 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR6: 172114 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR7: 172116 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR8: 172120 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K  
MPR9: 172122 +1 ; PARITY STATUS REGISTER  
0 ; CONTROL MAP (LOW 64K)  
0 ; CONTROL MAP (HIGH 64K)  
0 ; MASK FOR MOS, CORE, MS11-K

669 002210 172125  
 670 002212 000000  
 671 002214 000000  
 672 002216 000000  
 673 002220 172127  
 674 002222 000000  
 675 002224 000000  
 676 002226 000000  
 677 002230 172131  
 678 002232 000000  
 679 002234 000000  
 680 002236 000000  
 681 002240 172133  
 682 002242 000000  
 683 002244 000000  
 684 002246 000000  
 685 002250 172135  
 686 002252 000000  
 687 002254 000000  
 688 002256 000000  
 689 002260 172137  
 690 002262 000000  
 691 002264 000000  
 692 002266 000000  
 693  
 694 002270 000021  
 695  
 696

MPR10: 172124 +1  
 0  
 0  
 0  
 MPR11: 172126 +1  
 0  
 0  
 0  
 MPR12: 172130 +1  
 0  
 0  
 0  
 MPR13: 172132 +1  
 0  
 0  
 0  
 MPR14: 172134 +1  
 0  
 0  
 0  
 MPR15: 172136 +1  
 0  
 0  
 0

; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K  
 ; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K  
 ; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K  
 ; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K  
 ; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K  
 ; PARITY STATUS REGISTER  
 ; CONTROL MAP (LOW 64K)  
 ; CONTROL MAP (HIGH 64K)  
 ; MASK FOR MOS, CORE, MS11-K

; THIS IS THE END OF THE TABLE !  
 MPRX: .BLKW 17. ; TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.  
 ; (THE EXTRA WORD IS FOR A TERMINATOR.)

697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752

.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

\$ERRTB:

;; \* ITEM 1  
DM1 ;: PARITY REGISTER DATA ERROR.  
DH1 ;: PC REG, S/B WAS  
DT1 ;: \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF1 ;: 16, 18, 16, 16  
;; \* ITEM 2  
DM2 ;: ADDRESS TEST ERROR(TST1-5).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 3  
DM2 ;: ADDRESS TEST ERROR(TST1-5).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF3 ;: 16, 18, 18, 8, 8  
;; \* ITEM 4  
DM4 ;: CONSTANT DATA ERROR(TST6-10).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 5  
DM5 ;: ROTATING BIT ERROR(TST11-12).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 6  
DM6 ;: MOS REFRESH TEST ERROR (TST30-31).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 7  
DM7 ;: 3 XOR 9 PATTERN ERROR(TST13-16).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 10  
DM10 ;: MARCHING 1'S AND 0'S ERROR(TST27).  
DH2 ;: V/PC, P/PC, MA, S/B WAS  
DT2 ;: \$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT  
DF2 ;: 16, 18, 18, 16, 16  
;; \* ITEM 11

002332  
002332 026714  
002334 030273  
002336 001640  
002340 030640  
002342 026750  
002344 030312  
002346 001652  
002350 030644  
002352 026750  
002354 030312  
002356 001652  
002360 030651  
002362 027004  
002364 030312  
002366 001652  
002370 030644  
002372 027042  
002374 030312  
002376 001652  
002400 030644  
002402 027100  
002404 030312  
002406 001652  
002410 030644  
002412 027144  
002414 030312  
002416 001652  
002420 030644  
002422 027205  
002424 030312  
002426 001652  
002430 030644

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, \$TMPD, \$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, \$TMPD, \$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, \$TMPD, \$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 EXECUTION ERROR(TST21-26).
795	002534	030434	DH21	: V/PC P/PC IUT MA S/B WAS
796	002536	001730	DT21	: \$VERPC, \$ERRPC, \$TMPD, \$GDADR, \$GDDAT, \$BDDAT
797	002540	030664	DF21	: 16, 18, 16, 18, 16, 16
798			* ITEM 22	
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			;* ITEM 25	
810	002562	030157	DM25	; TRAPPED TO 114.
811	002564	030546	DH25	; V/PC, P/PC, TRP/PC, REG, WAS
812	002566	001774	DT25	; SVERPC, SERRPC, SBADR, STMP0, STMP1
813	002570	030656	DF14	; 16, 18, 18, 16
814			;* ITEM 26	
815	002572	030177	DM26	; FAILED TO TRAP.
816	002574	030577	DH26	; V/PC, P/PC, REG, WAS
817	002576	002010	DT26	; SVERPC, SERRPC, STMP0, STMP1
818	002600	030644	DF2	; 16, 18, 18, 16
819			;* ITEM 27	
820	002602	030217	DM27	; (ACTION ENABLE WASN'T SET).
821	002604	030577	DH26	; V/PC, P/PC, REG, WAS
822	002606	002010	DT26	; SVERPC, SERRPC, STMP0, SBDDAT
823	002610	030644	DF2	; 16, 18, 18, 16
824			;* ITEM 30	
825	002612	000000	0	; NO MESSAGE.
826	002614	030621	DH30	; REG, WAS, MA, WAS
827	002616	002022	DT30	; STMP0, STMP1, SGDADR, SBDDAT
828	002620	030672	DF30	; 18, 16, 18, 8
829			;* ITEM 31	
830	002622	030253	DM31	; TRAPPED TO 4
831	002624	000000	0	; NO HEADER
832	002626	002034	DT31	; STMP3
833	002630	030672	DF30	; 18

# K08

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
845 .SBTTL INITIALIZE THE COMMON TAGS
846 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
847 002646 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
848 002652 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
849 002654 022706 001140 CMP #SWR,R6 ;;DONE?
850 002660 001374 BNE -6 ;;LOOP BACK IF NO
851 002662 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
852 ;;INITIALIZE A FEW VECTORS
853 002666 012737 000610 000024 MOV #SPWRDN,@#PWVEC ;;POWER FAILURE VECTOR
854 002674 012737 000340 000026 MOV #340,@#PWVEC+2 ;;LEVEL 7
855 002702 016767 011164 011154 MOV SENDCT,SEOPCT ;;SETUP END-OF-PROGRAM COUNTER
856 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
857 ;;EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
858 002710 013746 000004 MOV @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
859 002714 012737 002750 000004 MOV #645,@#ERRVEC ;;SET UP ERROR VECTOR
860 002722 012767 177570 176210 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
861 002730 012767 177570 176204 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
862 002736 022777 177777 176174 CMP #-1,@SWR ;;TRY TO REFERENCE HARDWARE SWR
863 002744 001012 BNE 66$ ;;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 #SSWREG,SWR ;;NO,USE APT SWITCH REGISTER
876 003020
877 003020 005067 176476 67$: CLR LDDISP ;;CLEAR DISPLAY REGISTER STORAGE LOCN
878 003024 005077 176112 CLR @DISPLAY ;;CLEAR DISPLAY REGISTER
879
880 .SBTTL TYPE PROGRAM NAME
881 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
882 003030 005227 177777 INC #-1 ;;FIRST TIME?
883 003034 001046 BNE 68$ ;;BRANCH IF NO
884 003036 022737 014136 000042 CMP #SENDAD,@#42 ;;ACT-11?
885 003044 001442 BEQ 68$ ;;BRANCH IF YES
886 003046 004567 020346 JSR R5,SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
887 003052 003124 .WORD 69$ ;;ADDRESS OF MESSAGE TO BE TYPED
888 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
889 003054 005737 000042 TST @#42 ;;ARE WE RUNNING UNDER XXDP/ACT?
003060 001015 BNE 70$ ;;BRANCH IF YES
  
```

# L08

```

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 SGTSMR ROUTINE
895                                     ;* WIHTOUT 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, SGTSMR   ;; GO TO THE SUBROUTINE
898 003112 000403                    BR    71$
899 003114 112767 000001 176012 70$:  MOVB   #1, SAUTOB  ;; SET AUTO-MODE INDICATOR
900 003122 000413                    71$:
901 003122 000413                    BR    68$           ;; GET OVER THE ASCIZ
902                                     ;; 69$: .ASCIZ <CRLF>'MAINDEC-11-DZQMC-D'<CRLF>
903 003152 68$:
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 RELOCATED 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   #SRO        ;; 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, SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
922 003252 025262                    .WORD  MMAMES     ;; ADDRESS OF MESSAGE TO BE TYPED
923                                     ;; "KT11 AVAILABLE"
924
925                                     ;*
926 003254 012706 001100                    IMPCK: MOV    #STACK, SP  ;; CHECK IF 11/60 CACHE PRESENT, IF SO TURN IT OFF!!!
927 003260 012737 003274 000004          MOV    #MAPMEM, #ERRVEC
928 003266 052767 000014 174452          BIS   #14, IMPCK
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,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
950 003334 025327      .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
951 003336 012737 003452 000004      MOV      #25, @#ERRVEC  ;SET UP TIMEOUT VECTOR
952 003344 011222      MOV      (R2), (R2)+    ;READ+WRITE ALL MEMORY
953 003346 032702 017777      BIT      #MASK4K, R2    ;CHECK FOR 4K BOUNDARY
954 003352 001374      BNE     1$              ;BRANCH IF MORE IN BANK
955 003354 050310      BIS     R3,      (R0)   ;SET FLAG FOR BANK...LO 64K.
956 003356 050411      BIS     R4,      (R1)   ;...HI 64K.
957 003360 010267 175600      MOV      R2,      $TMP2 ;SAVE ADDRESS POINTER.
958 003364 005367 175574      DEC     $TMP2          ;ADJUST TO LAST ADR, LAST BANK.
959 003370 005767 175212      TST     MMHVA          ;CHECK FOR MEM MGMT.
960 003374 001432      BEQ     3$              ;BR IF NO MEM MGMT.
961 003376 042767 160000 175560      BIC     #160000, $TMP2 ;CLEAR BANK BITS ON RELATIVE ADDRESS.
962 003404 013705 172344      MOV     @#KIPAR2, R5    ;SAVE KIPAR2.
963 003410 005067 175552      CLR     $TMP3          ;MAKE SURE HI BITS ARE INIT.
964 003414 006305      ASL     R5              ;SHIFT IT 6 PLACES.
965 003416 006305      ASL     R5
966 003420 006305      ASL     R5
967 003422 006305      ASL     R5
968 003424 006305      ASL     R5
969 003426 006167 175534      ROL     $TMP3
970 003432 006305      ASL     R5
971 003434 006167 175526      ROL     $TMP3
972 003440 060567 175520      ADD     R5,      $TMP2 ;MAKE LAST ADR PHYSICAL.
973 003444 005567 175516      ADC     $TMP3
974 003450 000404      BR      3$              ;GO TO UPDATE POINTERS.
975
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     MMHVA          ;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   $CALF          ;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

```



# N08

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER      MACY11 30(1046) 08-SEP-77 10:19 PAGE 21  
 DZQMC.D.P11      26-JUL-77 15:01      GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0104

```

1002 003562 004567 017632                    JSR    RS,    SPRINT    ;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$:    HALT                   ;FATAL ERROR HALT...
1006                                                                                   ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
1007                                                                                   ;*****
1008                                                                                   ;* SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
1009                                                                                   ;* DATI, DATIP, DATO, AND DATOB.
1010                                                                                   ;*****
1011 003572 012706 001100                    TIMEOUT: MOV    #STACK, SP           ;SET UP THE STACK POINTER.
1012 003576 005067 175004                    CLR    MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG.
1013 003602 032777 010000 175330                    BIT    #SW1?,    JSWR           ;CHECK IF MEM MGMT TO BE INHIBITED.
1014 003610 001011                    BNE    1$           ;BR IF NO MEM MGMT.
1015 003612 012737 003634 000004                    MOV    #1$,    @#ERRVEC    ;SET TIMEOUT FOR MEM MGMT CHECK.
1016 003620 005037 177572                    CLR    @#SRO           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1017 003624 004767 010356                    JSR    PC,    MMINIT           ;INIT ALL MEM MGMT REGISTERS.
1018 003630 005267 174752                    INC    MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
1019 003634                    1$:                                                                                  
1020 003634 004567 017560                    JSR    RS,    SPRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
1021 003640 025345                    .WORD    BYTMS           ;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$,    @#ERRVEC    ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
1026 003654 105712                    TSTB   (R2)           ;DO DATI ONLY.
1027 003656 032702 000001                    10$:    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$:    BISB   (R2),    (R2)           ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
1031 003670 005700                    TST    R0           ;CHECK FOR PREVIOUS TYPED.
1032 003672 001023                    BNE    30$           ;BR IF ALREADY TYPED "FROM".
1033 003674 004567 017520                    JSR    RS,    SPRINT    ;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,    STYPAD           ;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$:    CMP    (SP)+,    (SP)+           ;POP TWO OFF STACK.
1041 003714 005700                    TST    R0           ;CHECK FOR PREVIOUS TYPED.
1042 003716 001411                    BEQ    30$           ;BR IF ALREADY TYPED "TO".
1043 003720 004567 017474                    JSR    RS,    SPRINT    ;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,    STYPAD           ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1049 003736 005202                    INC    R2           ;RESET ADDRESS POINTER.
1050 003740 005100                    29$:    COM    R0           ;RESET PREVIOUS TYPED FLAG.
1051 003742 005202                    30$:    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 BOUNDARY.
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 RS, SPRINT ;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 TIMEOUT ;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 175064 GMPR: MOV #MPRX, R4 ;SET UP POINTER TO PARITY REG EXIST TABLE.
1082 004046 032777 000100 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 GMPRA: MOV #GMPRB, @ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1086 004070 042713 000001 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 RS, SPRINT ;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

```

1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164

004160 004767 014012  
004164 012767 000001 175352  
004172 005067 175350  
004176 012702 014000  
004202 005767 174400  
004206 001404  
004210 012702 054000  
004214 004767 007766  
  
004220 005067 175314  
004224 005067 175312  
004230 012703 002070  
004234 032713 000001  
004240 001052  
004242 013773 00.612 000000  
  
004250 011212  
004252 005712  
004254 043773 001612 000000  
004262 005773 000000  
  
004266 100014  
004270 032773 007740 000000  
004276 001404  
004300 012763 070032 000006  
004306 000413  
004310 012763 077772 000006 55:  
004316 000407  
004320 032773 007740 000000 65:  
004326 001417  
004330 012763 070000 000006  
004336 056763 175202 000002 75:  
004344 056763 175176 000004  
004352 056767 175166 175160  
004360 056767 175162 175154  
004366 062703 000010  
004372 020327 002270

```
.SBTTL MAP PARITY MEMORY
;*****
;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
; - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
; - PARITY GENERATE OR DETECT LOGIC FAILED
; - PARITY ERROR BIT FAILED TO SET
; - PARITY BITS IN MEMORY LOCATION FAILED
; - I.E. BIT STUCK AT GOOD PARITY VALUE
;*****
MPAMEM: JSR PC, CLRPAR ;INITIALIZE ALL PARITY REGISTERS
MOV #1, BITPT ;INITIALIZE 4K POINTER
CLR BITPT+2 ;CLEAR HI 64K POINTER
MOV #14000, R2 ;SET ADR POINTER TO 14000.
TST MMABA ;CHECK FOR MEM MGMT
BEQ MAPRB ;BRANCH IF NO MEM MGMT
MOV #54000, R2 ;SET ADR POINTER TO PAR2
JSR PC, MMINIT ;SET UP ALL MEMORY MGMT REGISTERS.

;*****
;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
;* THEN WRITE TEST LOCATION VIA DAT0 & READ TEST LOCATION VIA DAT1
;* THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
;*****
MAPRB: CLR PMAP ;CLEAR THE PARITY MEMORY MAP
CLR PMAP+2
15: MOV #MPRO, R1 ;INITIALIZE TABLE ADDRESS
25: BIT #1, (R3) ;IS THIS REGISTER PRESENT?
BNE 35 ;NO - GET THE NEXT ONE
MOV @#WWP, @ (R3) ;YES - SET WRITE WRONG PARITY
;AND CLEAR REST OF REGISTER
MOV (R2), (R2) ;WRITE WRONG PARITY
TST (R2) ;READ WRONG PARITY
BIC @#WWP, @ (R3) ;CLEAR WRITE WRONG PARITY
TST @ (R3) ;OTHERWISE, CHECK TO SEE IF THIS
;CONTROL REGISTER GOT A PARITY
;ERROR
BPL 65 ;BRANCH IF IT DIDN'T AND CHECK
BIT #7740, @ (R3) ;IS IT A CORE PAR. REG.
BEQ 55 ;BRANCH IF NOT.
MOV #70032, 6 (R3) ;IF IT IS SET UP MASK
BR 75 ;AND BRANCH TO SET BITS.
55: MOV #77772, 6 (R3) ;IF NOS SET UP MASK
BR ;AND BRANCH TO SET BIT.
65: BIT #7, @ (R3) ;IF ANY BITS ARE SET
BEQ 35 ;THEN CSR IS MS11-K.
MOV #70000, 6 (R3) ;IF MS11-K SET MASK.
75: BIS BITPT, 2 (R3) ;SET FLAG IN MAP FOR THIS PARITY REGISTER
BIS BITPT+2, 4 (R3)
BIS BITPT, PMAP ;SET FLAG IN PARITY MAP
BIS BITPT+2, PMAP+2
35: ADD #10, R3 ;STEP UP TO NEXT REGISTER
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      MMVA      ;CHECK FOR MEM MGMT
1168 004406 001444          BEQ      10$          ;BR IF NO MEM MGMT
1169 004410 062737 000200 172344 4$:  ADD      #200      2#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1170 004416 006367 175122          ASL      BITPT     ;UPDATE BANK POINTER...LO 64K.
1171 004422 006167 175120          ROL      BITPT+2   ;...HI 64K.
1172 004426 100441          BMI      TMAP      ;BR IF ALL DONE.
1173 004430 023727 172344 001000  CMP      2#KIPAR2,#1000 ;THIS CODE TESTS IF MS11-K IS
1174 004436 001013          BNE      12$          ;PRESENT AND IF IT IS I SET
1175 004440 032737 000003 002252  BIT      #3,2#MPR14+2 ;THE BIT TO DISABLE ECC IN
1176 004446 001004          BNE      13$          ;THE LOCATION MAP THAT IS
1177 004450 032737 000003 002262  BIT      #3,2#MPR15+2 ;USED AS THE COMMAND TO
1178 004456 001400          BEQ      13$          ;WRITE WRONG PARITY.
1179 004460 012737 020004 001612 13$:  MOV      #20004,2#MMP
1180 004466 036767 175052 175030 12$:  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,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1198 004542 025370          .WORD    MMAP      ;ADDRESS OF MESSAGE TO BE TYPED
1199          ;"PARITY MEMORY MAP:"
1200 004544 012703 002070          MOV      #MPRO,   R3 ;INITIALIZE TABLE POINTER
1201 004550 032713 000001 1$:  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      SPRINT ;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,     $TYPOC ;GO TO THE SUBROUTINE
1229 004654 004567 016540      JSR   R5,     SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1230 004660 026014              .WORD  MX2      ;ADDRESS OF MESSAGE TO BE TYPED
1231                          ;"CONTROLS"
1232 004662 010300              MOV   R3,     R0   ;SET UP R0 FOR TYPMAP ROUTINE.
1233 004664 005720              TST   (R0)+      ;UPDATE POINTER TO MAP.
1234 004666 004767 013426      JSR   PC,     TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1235 004672 062703 000010      2$:  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,     SPRINT ;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,@#PMEAP ;IF YES THEN CLEAR THE BITS IN
1244 004730 043737 002254 001540  BIC   @#MPR14+4,@#PMEAP ;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,@#PMEAP ;IF YES I AM GOING TO
1248 004754 043737 002264 001542  BIC   @#MPR15+4,@#PMEAP+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              CLP   (R5)
1253 004776 052737 000001 002250  BIS   #1,@#MPR14
1254 005004 027537 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,     SPRINT ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1259 005026 026102              .WORD  MX6
1260 005030 005737 002270      9$:  TST   @#MPRX
1261 005034 001002              BNE   CTRLS
1262 005036 000167 000266      JMP   MANUAL    ;ARE THERE ANY PARITY REGISTERS TO TEST?
1263                          ;IF SO TEST THE BITS IN THE REGISTERS,
1264                          ;IF NO JUMP OVER REGISTER TESTS.
1264                          .SBTTL TEST PARITY REGISTERS
1265                          ;*****
1266                          ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1267                          ;* CAN BE SET AND CLEARED.
1268                          ;* THIS IS A ONCE ONLY TEST.
1269                          ;*****
1270
1271 005042 012703 002070      CTRLS: MOV   #MPR0,  R3   ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1272 005046 011302 1$:  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

```

F09

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMCD.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 26

SEQ 0109

TEST PARITY REGISTERS

```

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, RO         ;LOAD RO 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          JSR  PC, SPRT     ;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  RO, RESRVD ;IS THIS BIT RESERVED?
1288 005132 001025          BNE  3$           ;YES - DON'T TEST IT
1289 005134 010012          MOV  RO, (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  RO, 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          JSR  PC, SPRT   ;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          JSR  PC, SPRT   ;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  RO       ;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, 2(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$: MOV  (R3), R2 ;STORE PARITY REGISTER ADDRESS
1325 005260 052703 000010          ADD  #10, R3
1326 005264 032702 000001          BIT  #1, R2
1327 005270 001372          BNE  2$
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 0047E7 012700 64$: JSR      PC,          SPRINT ;SET UP VALUES FOR ERROR PRINTING.
1336 005320 004767 014222 JSR      PC,          SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1337 005324 000001          .WORD   1           ;ERROR TYPE CODE.
1338 005326          65$:          BR      2$           ;BRANCH BACK TO CHECK NEXT REGISTER
1339 005326 000753          BR      2$           ;BRANCH BACK TO CHECK NEXT REGISTER
1340
1341
1342 005330 012700 000014  MANUAL: MOV     #12,   RO      ;SET COUNTER TO CLEAR 12 WORDS.
1343 005334 012701 001562          MOV     #FSTADR,R1    ;STARTING AT FSTADR.
1344 005340 005021 1$:      CLR     (R1)+    ;CLEAR THE LOCATIONS.
1345 005342 005300          DEC     RO           ;COUNT.
1346 005344 001375          BNE     1$           ;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.  
\*\*\*\*\*

1351  
1352  
1353  
1354  
1355 005366 012700 000001  
1356 005372 005001  
1357 005374 005002  
1358 005376 005003  
1359 005400 004567 016014  
1360 005404 026215  
1361  
1362  
1363  
1364 005406 013746 177776  
1365 005412 004767 015630  
1366 005416 042716 000001  
1367 005422 005067 174106  
1368 005426 005067 174104  
1369 005432 062702 020000  
1370 005436 005503  
1371 005440 020367 015752  
1372 005444 103403  
1373 005446 101006  
1374 005450 020216  
1375 005452 101004  
1376 005454 006300  
1377 005456 006101  
1378 005460 100364  
1379 005462 000507  
1380 005464 030067 174034  
1381 005470 001003  
1382 005472 030167 174030  
1383 005476 001501  
1384 005500 016704 015712  
1385 005504  
1386 005504 004567 015710  
1387 005510 026302  
1388  
1389  
1390  
1391 005512 013746 177776  
1392 005516 004767 015524  
1393 005522 005716  
1394 005524 001010  
1395 005526 005767 015664  
1396 005532 001005  
1397 005534 016716 173424  
1398 005540 016767 173422 015650  
1399 005546 012667 174022  
1400 005552 020467 015640  
1401 005556 101352  
1402 005560 103403  
1403 005562 021667 174006  
1404 005566 101346  
1405 005570 032716 017777  
1406 005574 001404

MANUL1: MOV #BIT0, R0 ;SET UP BANK POINTER.  
CLR R1 ;...HI 64K.  
CLR R2 ;CLEAR ADDRESS POINTER.  
CLR R3 ;...HI ADDRESS BITS.  
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
.WORD FADMES ;ADDRESS OF MESSAGE TO BE TYPED  
"FIRST ADDRESS:"  
;\* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$RDOCT ROUTINE  
;\* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY \*\*SYSMAC\*\*.  
MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK  
JSR PC, \$RDOCT ;GO TO THE SUBROUTINE  
BIC #BIT0, (SP) ;MAKE SURE ADDRESS IS ON A WORD BOUNDARY.  
CLR SAVTST ;INIT TEST MAP...LO 64K.  
CLR SAVTST+2 ;...HI 64K.  
15: ADD #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.  
ADC R3  
CMP R3, \$HIOCT ;CHECK HI ADDRESS BITS.  
BLO 25 ;BR IF NOT HI ENOUGH YET.  
BHI 35 ;BR IF PAST SELECTED ADDRESS.  
CMP R2, (SP) ;CHECK THE LO ADDRESS BITS.  
BHI 35 ;BR IF PAST SELECTED ADDRESS.  
25: ASL R0 ;UPDATE POINTER...LO 64K.  
ROL R1 ;...HI 64K.  
BPL 15 ;BR BACK TO CHECK NEXT BANK.  
BR 175 ;BR IF OVERFLOW.  
35: BIT R0, MEMMAP ;CHECK IF BANK EXISTS.  
BNE 45 ;BR IF BANK EXISTS.  
BIT R1, MEMMAP+2 ;CHECK HI 64K.  
BEQ 175 ;BR IF ADDRESS IN UN-MAPPED BANK.  
45: MOV \$HIOCT, R4 ;SAVE FIRST ADR HI BITS.  
105: JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.  
.WORD LAOMES ;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\*\*.  
MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK  
JSR PC, \$RDOCT ;GO TO THE SUBROUTINE  
TST (SP) ;CHECK IF ADR 0 SELECTED (DEFAULT).  
BNE 115 ;BR IF NOT 0 (DEFAULT)  
TST \$HIOCT ;CHECK HI BITS.  
BNE 115 ;BR IF NOT 0 (DEFAULT).  
MOV \$TMP2, (SP) ;SET UP DEFAULT LAST ADR.  
MOV \$TMP3, \$HIOCT  
115: MOV (SP)+, LSTADR ;GET THE DATA.  
CMP R4, \$HIOCT ;CHECK FOR LAST ADR BELOW FIRST ADR.  
BHI 105 ;BR IF LAST BELOW FIRST.  
BLO 125 ;BR IF LAST ABOVE FIRST.  
CMP (SP), LSTADR ;CHECK FOR LAST BELOW FIRST.  
BHI 105 ;BR IF LAST BELOW FIRST.  
125: BIT #MASK4K, (SP) ;CHECK IF FIRST ADR ON BANK BOUNDARY.  
BEQ 135 ;BR IF ON BOUNDARY.



```

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 005676 001005          BNE    20$ ;BR IF IT EXISTS.
1430 005702 005726    17$:  TST    (SP)+ ;ADJUST THE STACK.
1431 005704 004567 015510          JSR    R5,    $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1432 005710 026325          .WORD  BA0ADR ;ADDRESS OF MESSAGE TO BE TYPED
1433          ;"?ADDRESS IN UNMAPPED BANK?"
1434          ;LOOP BACK TO THE BEGINNING.
1435 005712 000606          BR     MANUAL ;SET UP MAP FOR LAST ADDRESS.
1436 005714 010067 173662    20$:  MOV    R0,    LADMAP
1437 005720 010167 173660          MOV    R1,    LADMAP+2
1438 005724 005767 172656    21$:  TST    MAVA ;CHECK FOR MEMORY MANAGEMENT.
1439 005730 001404          BEQ    22$ ;BR IF NO MEM MGMT.
1440 005732 042716 160000          BIC    #160000, (SP) ;ADJUST FSTADR TO VIRTUAL BANK 0.
1441 005736 062716 040000          ADD    #40000, (SP) ;...TO VIRTUAL BANK 2.
1442 005742 012667 173614    22$:  MOV    (SP)+, FSTADR ;SAVE FIRST ADDRESS OFF THE STACK.
1443 005746          30$:  JSR    R5,    $SPRINT ;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          ;* WITHOUT 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,    $RDOCT ;GO TO THE SUBROUTINE
1450 005764 012667 173620          MOV    (SP)+, .CONST ;SAVE THE CONSTANT
1451 005770 005767 172612    MANUL2: TST    MAVA ;CHECK IF MEM MGMT IS AVAILABLE.
1452 005774 001406          BEQ    31$ ;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    31$:  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 BOUNDARY.
1458 006034 001004          BNE    START1 ;BR IF NOT ON BOUNDARY.
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,	MMMF		;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,	@#SRO		;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
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487 006114
1488 006114 004567 012416
1489 006120 000001
1490
1491 006122 000167 005574
1492
1493
1494 006126 004467 006202
1495 006132 004767 007624
1496 006136 010012
1497 006140 012201
1498 006142 02C001
1499 006144 011405
1500 006146 004767 012122
1501 006152 004767 013370
1502 006156 000002
1503 006160
1504 006160 062700 000002
1505 006164 030502
1506 006166 001363
1507 006170 004767 006716
1508
1509
1510
1511 006174 004467 006572
1512 006200 004767 007556
1513 006204 162700 000002
1514 006210 014201
1515 006212 020001
1516 006214 001405
1517 006216 004767 012026
1518 006222 004767 013320
1519 006226 000002
1520 006230
1521 006230 030502
1522 006232 001364
1523 006234 004767 007342

```

```

.SBTTL SECTION 1: MEMORY ADDRESS TESTS
*****
*TEST 1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST1:
      JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 1        ;MINIMUM BLOCK SIZE OF 1 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP    TST32    ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
; * UPWARDS WORD ADDRESSING.
      JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR    PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOV    R0,    (R2)   ;WRITE VALUE OF ADDRESS INTO ADDRESS
      MOV    (R2)+, R1    ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR    PC,    SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 2        ;ERROR TYPE CODE.
65$:  ADD    #2,    R0    ;ADD #2 TO PHYSICAL ADDRESS
      BIT    R5,    R2    ;CHECK FOR END OF A BLOCK.
      BNE    2$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,    MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
; * CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
; * DOWNWARDS WORD ADDRESSING.
      JSR    R4,    INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR    PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$:   SUB    #2,    R0    ;DEC DATA BY 2
      MOV    -(R2), R1    ;GET THE DATA FROM MEMORY
      CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    67$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:  JSR    PC,    SPRINT0 ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 2        ;ERROR TYPE CODE.
67$:  BIT    R5,    R2    ;CHECK FOR END OF A BLOCK.
      BNE    4$      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,    MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

```

1524
1525
1526
1527
1528
1529
1530
1531
1532
1533 006240
1534 006240 004567 012272
1535 006244 000000
1536
1537 006246 004467 006062
1538 006252 004767 007504
1539 006256 110022
1540 006260 005200
1541 006262 030502
1542 006264 001374
1543 006266 004767 006620
1544
1545
1546
1547 006272 004467 006474
1548 006276 004767 007460
1549 006302 005300
1550 006304 114201
1551 006306 120001
1552 006310 001405
1553 006312 004767 011732
1554 006316 004767 013224
1555 006322 000003
1556 006324
1557 006324 030502
1558 006326 001365
1559 006330 004767 007246
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570 006334
1571 006334 004567 012176
1572 006340 000000
1573
1574 006342 004467 006424
1575 006346 004767 007410
1576 006352 005100
1577 006354 062700 000002
1578 006360 010042
1579 006362 030502

```

```

*****
;TEST 2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
;*
;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
*****
↑ST2:
      JSR     R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD  0        ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
;* UPWARDS BYTE ADDRESSING.
      JSR     R4,    INITM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR     PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOVB   RO,    (R2)+  ;WRITE VALUE OF ADDRESS INTO ADDRESS
      INC    RO      ;ADD ONE TO PHYSICAL ADDRESS
      BIT   R5,    R2    ;CHECK FOR END OF A BLOCK.
      BNE   2$,    ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR   PC,    MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.

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

*****
;TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
;*
;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
;* R1 = DATA READ FROM MEMORY (WAS)
;* R2 = VIRTUAL ADDRESS
;* R3 = NOT USED
;* R4 = NOT USED
;* R5 = BLOCK BOUNDARY BIT MASK.
*****
↑ST3:
      JSR     R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD  0        ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
;* DOWNWARDS WORD ADDRESSING.
      JSR     R4,    INITM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR     PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
      COM   RO      ;COMPLEMENT THE ADR
2$:   ADD    #2,    RO     ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
      MOV   RO,    -(R2)  ;PUT DATA INTO MEMORY
      BIT   R5,    R2    ;CHECK FOR END OF A BLOCK.

```

M09

```

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
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$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
1593 006416 004767 013124 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1594 006422 000002          .WORD 2      ;ERROR TYPE CODE.
1595 006424
1596 006424 162700 000002 65$: 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$.
1600
1601 ;*****
1602 ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1603 ;*
1604 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1605 ;* R1 = DATA READ FROM MEMORY (WAS)
1606 ;* R2 = VIRTUAL ADDRESS
1607 ;* R3 = NOT USED
1608 ;* R4 = NOT USED
1609 ;* R5 = BLOCK BOUNDARY BIT MASK.
1610 ;*****
1610 006440
1611 006440 004567 012072 1$T4: 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$: JSR PC, SPRINT1 ;SET UP VALUES FOR ERROR PRINTING.
1629 006512 004767 013030 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1630 006516 000003          .WORD 3      ;ERROR TYPE CODE.
1631 006520
1632 006520 030502          65$: 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
1638
1639
1640
1641
1642
1643
1644
1645 006530
1646 006530 004567 012002
1647 006534 000000
1648
1649 006536 004467 006230
1650 006542 004767 007270
1651 006546 005100
1652 006550 110042
1653 006552 030502
1654 006554 001375
1655 006556 004767 007020
1656
1657
1658
1659 006562 004467 006204
1660 006566 004767 007244
1661 006572 005100
1662 006574 114201
1663 006576 020001
1664 006600 001405
1665 006602 004767 011442
1666 006606 004767 012734
1667 006612 000003
1668 006614
1669 006614 030502
1670 006616 001366
1671 006620 004767 006756

```

```

*****
;TEST 5 WRITE 1'S COMPLEMENT OF BANK #
; R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
; R1 = DATA READ FROM MEMORY (WAS)
; R2 = VIRTUAL ADDRESS
; R3 = NOT USED
; R4 = NOT USED
; R5 = BLOCK BOUNDARY BIT MASK.
*****
TS:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * DOWNWARDS BYTE ADDRESSING.
; * DOWNWARDS BYTE ADDRESSING.
JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
COM R0 ;1'S COMPLEMENT OF BANK #
2$: MOVB R0, -(R2) ;PUT 1'S COM OF BANK # INTO MEMORY
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.

; * CHECK THAT DATA WRITTEN CAN BE READ.
; * DOWNWARDS BYTE ADDRESSING.
3$: JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
JSR PC, BANKNO ;GET THE BANK # INTO R0
COM R0 ;SET 1'S COMPLEMENT OF BANK #
4$: MOVB -(R2), R1 ;READ DATA OUT OF MEMORY
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRTD ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
;WORD 3 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

```

.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 = BINARY BOUNDARY BIT MASK.  
\*\*\*\*\*

```
TST6: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
TST6A: MOV .CONST, R0 ;GET USER CONSTANT
      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R0, (R2)+ ;WRITE CONSTANT INTO MEMORY.
      BIT R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

\*\*\*\*\*  
\*TEST 7 READ MEMORY AND COMPARE TO CONSTANT.  
\* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST 6.  
\*\*\*\*\*

```
TST7: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      MOV .CONST, R0 ;GET USER CONSTANT
      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 4 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

\* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.  
\* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7  
\* BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.

```
BIT #SW08, JSWR ;CHECK THAT LOOP ON TEST BIT SET
BEQ TST10 ;BRANCH IF NOT LOOP ON TEST
MOV JSWR, -(SP) ;GET SWITCH REGISTER DATA.
BIC #177740, (SP) ;CLEAR NON-TEST-NUMBER SWITCHES.
CMP #6, (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
BNE TST10 ;BRANCH IF NOT TEST 6
SUB #1, $TSTNM ;RESET TEST NUM
SUB #TST7-TST6, $LPADR ;RESET LOOP ADR
```

1672				
1673				
1674				
1675				
1676				
1677				
1678				
1679				
1680				
1681				
1682				
1683				
1684				
1685				
1686				
1687	006624			
1688	006624	004567	011706	
1689	006630	000000		
1690	006632	016700	172752	
1691	006636	004467	005472	
1692	006642	010022		
1693	006644	030502		
1694	006646	001375		
1695	006650	004767	006236	
1696				
1697				
1698				
1699				
1700				
1701	006654			
1702	006654	004567	011656	
1703	006660	000000		
1704	006662	016700	172722	
1705	006666	004467	005442	
1706	006672			
1707	006672	012201		
1708	006674	020001		
1709	006676	001405		
1710	006700	004767	011370	
1711	006704	004767	012636	
1712	006710	000004		
1713	006712			
1714	006712	030502		
1715	006714	001366		
1716	006716	004767	006170	
1717				
1718				
1719				
1720	006722	032777	000400	172210
1721	006730	001416		
1722	006732	017746	172202	
1723	006736	042716	177740	
1724	006742	022726	000006	
1725	006746	001007		
1726	006750	162767	000001	172124
1727	006756	162767	000030	172122

C10

```

1728 006764 000722          BR      TST6A          ;GO TO TEST 6
1729
1730
1731 ;*****
1732 ;*TEST 10      MORSE CASE NOISE (PARITY) WORD TESTING
1733 ;* CHECK MEMORY WITH A SERIES OF PATTERNS
1734 ;*****
1734 006766          †ST10:
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$:     JSR      PC,      SPANT2 ;SET UP VALUES FOR ERROR PRINTING.
1747 007030 004767 012512      JSR      PC,      SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1748 007034 000004          .WORD    4             ;ERROR TYPE CODE.
1749 007036
1750 007036 030502          65$:     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

```



D10

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMC.D.P11 26-JUL-77 15:01

T11 ROTATE A "0" BIT THROUGH A FIELD OF ONES. MACY11 30(1046) 08-SEP-77 10:19 PAGE 37

SEQ 0120

```

1754
1755
1756
1757 007050
1758 007050 004567 011462
1759 007054 000000
1760 007056 012700 177777
1761 007062 004767 007010
1762 007066 004467 005242
1763 007072 000241
1764 007074 004767 007016
1765 007100 016201 177776
1766 007104 103402
1767 007106 020001
1768 007110 001405
1769 007112 004767 011156
1770 007116 004767 012424
1771 007122 000005
1772 007124
1773 007124 030502
1774 007126 001361
1775 007130 004767 005756
1776
1777
1778
1779
1780 007134
1781 007134 004567 011376
1782 007140 000000
1783 007142 005000
1784 007144 004767 006726
1785 007150 004467 005160
1786 007154 000261
1787 007156 004767 006734
1788 007162 016201 177776
1789 007166 103002
1790 007170 020001
1791 007172 001405
1792 007174 004767 011074
1793 007200 004767 012342
1794 007204 000005
1795 007206
1796 007206 030502
1797 007210 001361
1798 007212 004767 005674

```

```

*****
*TEST 11 ROTATE A "0" BIT THROUGH A FIELD OF ONES.
*****
†ST11:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV #-1, R0 ;SET CHECK WORD
JSR PC, SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: CLC ;CLEAR CARRY BIT IN PSW
JSR PC, ROTATE
MOV -2(R2), R1 ;GET RESULT
BCS 63$ ;BRANCH IF 'C' BIT WAS SET
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
63$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 5 ;ERROR TYPE CODE.
64$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

```

*****
*TEST 12 ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
*****
†ST12:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
CLR R0 ;SET CHECK WORD
JSR PC, SETCON ;PUT THE CONTENTS OF R0 IN ALL MEMORY
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: SEC ;SET 'C' BIT IN PSW
JSR PC, ROTATE ;GO ROTATE '1' BIT
MOV -2(R2), R1 ;GET RESULT
BCC 63$ ;BRANCH IF 'C' IS CLEAR
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 64$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
63$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 5 ;ERROR TYPE CODE.
64$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

E10

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMC0.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 38

SEQ 0121

```

1799                                     ;*****
1800                                     ;*TEST 13      3 XOR 9 TEST PATTERN.
1801                                     ;*****
1802 007216                               TST13:
1803 007216 004567 011314                JSR   R5,   $SCOPE ; GO TO SCOPE ROUTINE.
1804 007222 000777                        .WORD  777       ; MINIMUM BLOCK SIZE OF 256. WORDS
1805                                     ; REQUIRED FOR THIS TEST.
1806 007224 000167 000312                JMP   TST14      ; SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1807                                     ; AVAILABLE FOR TEST.
1808 007230 005000                          .3X9: CLR   R0       ; SET UP TEST DATA
1809 007232 012703 177777                MOV   #-1,   R3     ; SET COM DATA REG
1810 007236 004467 005072                JSR   R4,   INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
1811 007242 004767 006716                15:   JSR   PC,   W3X9 ; WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
1812 007246 030502                        BIT   R5,   R2      ; CHECK FOR END OF A BLOCK.
1813 007250 001374                        BNE   15,     ; BRANCH IF MORE IN CURRENT BLOCK.
1814 007252 004767 005634                JSR   PC,   MMUP   ; FIND NEXT BLOCK AND LOOP TO 15.
1815
1816                                     ;*****
1817                                     ;* CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
1818                                     ;*****
1819 007256 005000                          11$: CLR   R0       ; SET CHECK WORD
1820 007260 004467 005050                JSR   R4,   INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
1821 007264 012704 000100                12$: MOV   #64.,  R4   ; SET 256. WORD COUNTER
1822 007270
1823 007270 012201                          11$: MOV   (R2)+, R1   ; GET THE DATA FROM MEMORY UNDER TEST.
1824 007272 020001                          CMP   R0,   R1      ; COMPARE THE CHECK WORD WITH THE DATA READ.
1825 007274 001405                          BEQ   65$,     ; BRANCH OVER ERROR CALL IF GOOD DATA.
1826 007276 004767 010772                64$: JSR   PC,   SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
1827 007302 004767 012240                JSR   PC,   $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1828 007306 000007                          .WORD  7           ; ERROR TYPE CODE.
1829 007310
1830 007310 012201                          65$: MOV   (R2)+, R1   ; GET THE DATA FROM MEMORY UNDER TEST.
1831 007312 020001                          CMP   R0,   R1      ; COMPARE THE CHECK WORD WITH THE DATA READ.
1832 007314 001405                          BEQ   67$,     ; BRANCH OVER ERROR CALL IF GOOD DATA.
1833 007316 004767 010752                66$: JSR   PC,   SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
1834 007322 004767 012220                JSR   PC,   $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1835 007326 000007                          .WORD  7           ; ERROR TYPE CODE.
1836 007330
1837 007330 012201                          67$: MOV   (R2)+, R1   ; GET THE DATA FROM MEMORY UNDER TEST.
1838 007332 020001                          CMP   R0,   R1      ; COMPARE THE CHECK WORD WITH THE DATA READ.
1839 007334 001405                          BEQ   69$,     ; BRANCH OVER ERROR CALL IF GOOD DATA.
1840 007336 004767 010732                68$: JSR   PC,   SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
1841 007342 004767 012200                JSR   PC,   $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1842 007346 000007                          .WORD  7           ; ERROR TYPE CODE.
1843 007350
1844 007350 012201                          69$: MOV   (R2)+, R1   ; GET THE DATA FROM MEMORY UNDER TEST.
1845 007352 020001                          CMP   R0,   R1      ; COMPARE THE CHECK WORD WITH THE DATA READ.
1846 007354 001405                          BEQ   71$,     ; BRANCH OVER ERROR CALL IF GOOD DATA.
1847 007356 004767 010712                70$: JSR   PC,   SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
1848 007362 004767 012160                JSR   PC,   $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1849 007366 000007                          .WORD  7           ; ERROR TYPE CODE.
1850 007370
1851 007370 005100                          71$: COM   R0       ; COMPLEMENT CHECK WORD
1852 007372 005304                          DEC   R4       ; DECREMENT 256. WORD COUNTER
1853 007374 001335
1854 007376 005100                          COM   R0       ; COMPLEMENT CHECK WORD

```

# F10

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC0.P11 26-JUL-77 15:01

T13 3 XOR 9 TEST PATTERN.

MACY11 30(1046) 08-SEP-77 10:19 PAGE 39

SEQ 0122

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	0C0100	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,	SPRINT2	;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,	SPRINT2	;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,	SPRINT2	;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\$.

G10

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC0.P11 26-JUL-77 15:01

T14 COMPLEMENT 3 XOR 9 TEST PATTERN

MACY11 30(1046) 08-SEP-77 10:19 PAGE 40

SEQ 0123

```

1901
1902
1903
1904 007542
1905 007542 004567 010770
1906 007546 000777
1907
1908 007550 000167 000316
1909
1910 007554 012700 177777
1911 007560 005003
1912 007562 004467 004546
1913 007566 004767 006372
1914 007572 030502
1915 007574 001374
1916 007576 004767 005310
1917
1918
1919
1920
1921
1922 007602 012700 177777
1923 007606 004467 004522
1924 007612 012704 000100
1925 007616
1926 007616 012201
1927 007620 020001
1928 007622 001405
1929 007624 004767 010444
1930 007630 004767 011712
1931 007634 000007
1932 007636
1933 007636 012201
1934 007640 020001
1935 007642 001405
1936 007644 004767 010424
1937 007650 004767 011672
1938 007654 000007
1939 007656
1940 007656 012201
1941 007660 020001
1942 007662 001405
1943 007664 004767 010404
1944 007670 004767 011652
1945 007674 000007
1946 007676
1947 007676 012201
1948 007700 020001
1949 007702 001405
1950 007704 004767 010364
1951 007710 004767 011632
1952 007714 000007
1953 007716
1954 007716 005100
1955 007720 005304
1956 007722 001335

```

```

*****
*TEST 14 COMPLEMENT 3 XOR 9 TEST PATTERN
*****
TST14:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS
;REQUIRED FOR THIS TEST.
JMP TST15 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
;AVAILABLE FOR TEST.
MOV #-1, R0 ;SET UP TEST DATA
CLR R3 ;SET COM DATA REG
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
* CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
*****
MOV #-1, R0 ;SET CHECK WORD
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$: MOV #64., R4 ;SET 256. WORD COUNTER
12$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
65$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
67$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
69$:
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$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERRR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
71$:
COM R0 ;COMPLEMENT CHECK WORD
DEC R4 ;DECREMENT 256. WORD COUNTER
BNE 12$

```

# H10

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 30(1046) 08-SEP-77 10:19 PAGE 41  
 DZQMC.D.P11 26-JUL-77 15:01 T14 COMPLEMENT 3 XOR 9 TEST PATTERN

SEQ 0124

```

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$.
1961
1962
1963
1964
1965 007736 012700 177777      MOV     8-1,    R0      ;SET UP CHECK WORD.
1966 007742 004467 004366      JSR     R4,    INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1967 007746 012704 000100 21$:   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$:   JSR     PC,    SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
1974 007770 004767 011552      JSR     PC,    SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1975 007774 000007      .WORD   7           ;ERROR TYPE CODE.
1976 007776 73$:
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$:   JSR     PC,    SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
1983 010014 004767 011526      JSR     PC,    SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1984 010020 000007      .WORD   7           ;ERROR TYPE CODE.
1985 010022 75$:
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$:   JSR     PC,    SPRT2   ;SET UP VALUES FOR ERROR PRINTING.
1992 010040 004767 011502      JSR     PC,    SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1993 010044 000007      .WORD   7           ;ERROR TYPE CODE.
1994 010046 77$:
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$.

```

# I10

```

2004      ;*****
2005      ;*TEST 15      MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2006      ;*****
2007      ;TST15:
2008      010072 004567 010440      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
2009      010076 000777              .WORD      777              ;MINIMUM BLOCK SIZE OF 256. WORDS
2010                                      ;REQUIRED FOR THIS TEST.
2011      010100 000167 000610      JMP      TST16              ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2012                                      ;AVAILABLE FOR TEST.
2013      010104 012700 000401      MOV      #401,      R0      ;SET UP PARITY "ALL ZEROS" PATTERN
2014      010110 012703 177777      MOV      #-1,      R3      ;SET COM DATA REG
2015      010114 004467 004214      JSR      R4,      INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2016      010120 004767 006040      15:     JSR      PC,      W3X9      ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2017      010124 030502              BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
2018      010126 001374              BNE     15          ;BRANCH IF MORE IN CURRENT BLOCK.
2019      010130 004767 004756      JSR      PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO 15.
2020
2021      ;*****
2022      ;* CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2023      ;*****
2024      010134 012700 000401      MOV      #401,      R0      ;RESET PARITY "ALL ZEROS" PATTERN.
2025      010140 012703 177777      MOV      #-1,      R3      ;RESET PARITY ALL ONES PATTERN.
2026      010144 004467 004164      JSR      R4,      INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2027      010150 012704 000100      115:    MOV      #64.,      R4      ;SET 256. WORD COUNTER
2028      010154
2029      010154 012201              MOV      (R2)+,      R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2030      010156 020001              CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2031      010160 001405              BEQ     65$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2032      010162 004767 010106      645:    JSR      PC,      SPRT2     ;SET UP VALUES FOR ERROR PRINTING.
2033      010166 004767 011354      JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2034      010172 000007              .WORD      7              ;ERROR TYPE CODE.
2035      010174
2036      010174 012201              MOV      (R2)+,      R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2037      010176 020001              CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2038      010200 001405              BEQ     67$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2039      010202 004767 010066      665:    JSR      PC,      SPRT2     ;SET UP VALUES FOR ERROR PRINTING.
2040      010206 004767 011334      JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2041      010212 000007              .WORD      7              ;ERROR TYPE CODE.
2042      010214
2043      010214 012201              MOV      (R2)+,      R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2044      010216 020001              CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2045      010220 001405              BEQ     69$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2046      010222 004767 010046      685:    JSR      PC,      SPRT2     ;SET UP VALUES FOR ERROR PRINTING.
2047      010226 004767 011314      JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2048      010232 000007              .WORD      7              ;ERROR TYPE CODE.
2049      010234
2050      010234 012201              MOV      (R2)+,      R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2051      010236 020001              CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2052      010240 001405              BEQ     71$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
2053      010242 004767 010026      705:    JSR      PC,      SPRT2     ;SET UP VALUES FOR ERROR PRINTING.
2054      010246 004767 011274      JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2055      010252 000007              .WORD      7              ;ERROR TYPE CODE.
2056      010254
2057      010254 010046              MOV      R0,      -(SP)    ;SAVE R0
2058      010256 010300              MOV      R3,      R0      ;PUT R3 INTO R0
2059      010260 012603              MOV      (SP)+,      R3    ;PUT SAVED R0 INTO R3
    
```

# J10

```

2060 010262 005304          DEC      R4          ;COUNT 256. WORDS
2061 010264 001333          BNE     12$          ;BRANCH IF MORE
2062 010266 010000          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$.
2068
2069
2070
2071
2072 010304 012700 000401      MOV     #401,      R0   ;SET UP PARITY "ALL ZEROS" PATTERN.
2073 010310 012703 177777      MOV     #-1,      R3   ;SET UP ALL ONES PATTERN.
2074 010314 004467 004014      JSR     R4,         INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2075 010320 012704 000100      MCV     #64.,R4     ;SET 256. WORD COUNTER
2076 010324
2077 010324 012201          MOV     (R2)+,     R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2078 010326 020001          CMP     R0,         R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2079 010330 001405          BEQ     73$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2080 010332 004767 007736      JSR     PC,         SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2081 010336 004767 011204      JSR     PC,         $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2082 010342 000007          .WORD   7             ;ERROR TYPE CODE.
2083 010344
2084 010344 005100          COM     R0          ;COMPLEMENT CHECK WORD
2085 010346 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2086 010350 012201          MOV     (R2)+,     R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2087 010352 020001          CMP     R0,         R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2088 010354 001405          BEQ     75$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2089 010356 004767 007712      JSR     PC,         SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2090 010362 004767 011160      JSR     PC,         $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2091 010366 000007          .WORD   7             ;ERROR TYPE CODE.
2092 010370
2093 010370 005100          COM     R0          ;COMPLEMENT CHECK WORD
2094 010372 005142          COM     -(R2)       ;RESTORE DATA
2095 010374 012201          MOV     (R2)+,     R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2096 010376 020001          CMP     R0,         R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2097 010400 001405          BEQ     77$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2098 010402 004767 007666      JSR     PC,         SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2099 010406 004767 011134      JSR     PC,         $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2100 010412 000007          .WORD   7             ;ERROR TYPE CODE.
2101 010414
2102 010414 012201          MOV     (R2)+,     R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2103 010416 020001          CMP     R0,         R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2104 010420 001405          BEQ     79$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2105 010422 004767 007646      JSR     PC,         SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2106 010426 004767 011114      JSR     PC,         $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2107 010432 000007          .WORD   7             ;ERROR TYPE CODE.
2108 010434
2109 010434 005100          COM     R0          ;COMPLEMENT CHECK WORD
2110 010436 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2111 010440 012201          MOV     (R2)+,     R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2112 010442 020001          CMP     R0,         R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2113 010444 001405          BEQ     81$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2114 010446 004767 007622      JSR     PC,         SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
2115 010452 004767 011170      JSR     PC,         $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
    
```

# K10

2116	010456	000007		.WORD	7		;ERROR TYPE CODE.
2117	010460		81\$:				
2118	010460	005100		COM	RO		;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	RO,	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	JSR	PC,	SPRINT2	;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	RO,	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	JSR	PC,	SPRINT2	;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	RO		;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	RO,	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	JSR	PC,	SPRINT2	;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	RO		;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	RO,	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	JSR	PC,	SPRINT2	;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	RO,	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	JSR	PC,	SPRINT2	;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	RO		;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	RO,	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	JSR	PC,	SPRINT2	;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	RO		;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	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.



L10

```

2172 010650 001405          BEQ      95$      ; BRANCH OVER ERROR CALL IF GOOD DATA.
2173 010652 004767 007416 94$: JSR      PC,      SPRINT2 ; 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
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
010714          ; *****
010714 004567 007616 JSR      R5,      $SCOPE  ; GO TO SCOPE ROUTINE.
010720 000777          .WORD    777            ; MINIMUM BLOCK SIZE OF 256. WORDS
2192 010714 000167 000610 JMP      TST17          ; REQUIRED FOR THIS TEST.
2198 010726 012700 177777 MOV      #-1,     R0      ; SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2199 010732 012703 000401 MOV      #401,    R3      ; AVAILABLE FOR TEST.
2200 010736 004467 003372 JSR      R4,      INITMM  ; SET UP ALL ONES PATTERN
2201 010742 004767 005216 JSR      PC,      W3X9    ; SET UP PARITY "ALL ZEROS" PATTERN
2202 010746 030502          BIT      R5,      R2      ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2203 010750 001374          BNE     1$              ; WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2204 010752 004767 004134 JSR      PC,      MMUP     ; CHECK FOR END OF A BLOCK.
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
010756 012700 177777 MOV      #-1,     R0      ; BRANCH IF MORE IN CURRENT BLOCK.
010762 012703 000401 MOV      #401,    R3      ; FIND NEXT BLOCK AND LOOP TO 1$.
010766 004467 003342 JSR      R4,      INITMM  ; CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
010772 012704 000100 JSR      R4,      #64.,   R4 ; *****
010776          11$:          ; SET UP ALL ONES PATTERN
010776          12$:          ; SET UP PARITY "ALL ZEROS" PATTERN
010776          15$:          ; INITIALIZE THE MEMORY ADDRESS POINTERS.
010776          18$:          ; SET 256. WORD COUNTER
010776          21$:          ; GET THE DATA FROM MEMORY UNDER TEST.
011000          22$:          ; COMPARE THE CHECK WORD WITH THE DATA READ.
011002 001405          BEQ     65$            ; BRANCH OVER ERROR CALL IF GOOD DATA.
011004 004767 007264 64$: JSR      PC,      SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
011010 004767 010532 JSR      PC,      $ERROR  ; *** ERROR *** (GO TYPE A MESSAGE)
011014 000007          .WORD    7              ; ERROR TYPE CODE.
011016          65$:          ;
011016          66$:          ; GET THE DATA FROM MEMORY UNDER TEST.
011020          67$:          ; COMPARE THE CHECK WORD WITH THE DATA READ.
011022 001405          BEQ     67$            ; BRANCH OVER ERROR CALL IF GOOD DATA.
011024 004767 007244 66$: JSR      PC,      SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
011030 004767 010512 JSR      PC,      $ERROR  ; *** ERROR *** (GO TYPE A MESSAGE)
011034 000007          .WORD    7              ; ERROR TYPE CODE.
011036          67$:          ;

```

M10

```

2228 011036 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2229 011040 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2230 011042 001405      BEQ     69$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2231 011044 004767 007224    68$:    JSR     PC, SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
2232 011050 004767 010472    JSR     PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2233 011054 000007      .WORD   7             ;ERROR TYPE CODE.
2234 011056
2235 011056 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2236 011060 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2237 011062 001405      BEQ     71$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2238 011064 004767 007204    70$:    JSR     PC, SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
2239 011070 004767 010452    JSR     PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2240 011074 000007      .WORD   7             ;ERROR TYPE CODE.
2241 011076
2242 011076 010046      MOV      RO, -(SP)     ;SAVE RO
2243 011100 010300      MOV      R3, RO        ;PUT R3 INTO RO
2244 011102 012603      MOV      (SP), R3      ;PUT SAVED RO INTO R3
2245 011104 005304      DEC     R4             ;COUNT 256. WORDS
2246 011106 001333      BNE     12$,           ;BRANCH IF MORE
2247 011110 010046      MOV      RO, -(SP)     ;SAVE RO
2248 011112 010300      MOV      R3, RO        ;PUT R3 INTO RO
2249 011114 012603      MOV      (SP)+, R3     ;PUT SAVED RO INTO R3
2250 011116 030502      BIT     R5, R2         ;CHECK FOR END OF A BLOCK.
2251 011120 001324      BNE     11$,           ;BRANCH IF MORE IN CURRENT BLOCK.
2252 011122 004767 003764    JSR     PC, MMUP      ;FIND NEXT BLOCK AND LOOP TO 11$.
2253
2254 ;*****
2255 ;* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
2256 ;*****
2257 011126 012700 177777    MOV     #-1, RO        ;SET UP ALL ONE'S PATTERN
2258 011132 012703 000401    MOV     #401, R3       ;SET UP PARITY "ALL ZEROS" PATTERN
2259 011136 004467 003172    JSR     R4, INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2260 011142 012704 000100    21$:    MOV     #64., R4      ;SET 256. WORD COUNTER
2261 011146
2262 011146 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2263 011150 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2264 011152 001405      BEQ     73$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2265 011154 004767 007114    72$:    JSR     PC, SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
2266 011160 004767 010362    JSR     PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2267 011164 000007      .WORD   7             ;ERROR TYPE CODE.
2268 011166
2269 011166 005100      COM     RO             ;COMPLEMENT CHECK WORD
2270 011170 005142      COM     -(R2)          ;COMPLEMENT TEST DATA
2271 011172 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2272 011174 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2273 011176 001405      BEQ     75$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2274 011200 004767 007070    74$:    JSR     PC, SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
2275 011204 004767 010336    JSR     PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2276 011210 000007      .WORD   7             ;ERROR TYPE CODE.
2277 011212
2278 011212 005100      COM     RO             ;COMPLEMENT CHECK WORD
2279 011214 005142      COM     -(R2)          ;RESTORE DATA
2280 011216 012201      MOV      (R2)+, R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2281 011220 020001      CMP      RO, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2282 011222 001405      BEQ     77$,           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2283 011224 004767 007044    76$:    JSR     PC, SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.

```

# N10

MAINDEC-11-DZQMC-D-0: 0-124K MEMORY EXERCISER, 16K VER MACY11 30(1046) 08-SEP-77 10:19 PAGE 47  
 DZQMC0.P11 26-JUL-77 15:01 T16 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0130

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	RO,	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,	SPRINT2	;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	RO		;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	RO,	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,	SPRINT2	;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	RO		;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	RO,	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,	SPRINT2	;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	RO,	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,	SPRINT2	;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	RO		;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	RO,	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,	SPRINT2	;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	RO		;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	RO,	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,	SPRINT2	;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	RO,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2339	011422	001405			BEQ	91\$		;BRANCH OVER ERROR CALL IF GOOD DATA.

# B11

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 30(1046) 08-SEP-77 10:19 PAGE 48  
 DZQMC0.P11 26-JUL-77 15:01 T16 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0131

2340	011424	004767	006644	90\$:	JSR	PC,	SPRINT2	;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	RO		;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	RO,	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,	SPRINT2	;SET UP VALUES FOR ERROR PRINTING.
2350	011454	004767	010066		JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
2351	011450	000007			.WORD	7		;ERROR TYPE CODE.
2352	011462			93\$:				
2353	011462	005100			COM	RO		;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	RO,	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,	SPRINT2	;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	RO,	-(SP)	;SAVE RO
2363	011510	010300			MOV	R3,	RO	;PUT R3 INTO RO
2364	011512	012603			MOV	(SP)+,	R3	;PUT SAVED RO INTO R3
2365	011514	005304			DEC	R4		;DECREMENT 256. WORD COUNTER
2366	011516	001213			BNE	22\$		;BRANCH IF MORE.
2367	011520	010046			MOV	RO,	-(SP)	;SAVE RO
2368	011522	010300			MOV	R3,	RO	;PUT R3 INTO RO
2369	011524	012603			MOV	(SP)+,	R3	;PUT SAVED RO 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\$.

```

2373
2374
2375
2376
2377
2378
2379
2380
2381 011536
2382 011536 004567 006774
2383 011542 000000
2384 011544 005767 170520
2385 011550 001404
2386 011552 032777 000100 167360
2387 011560 001402
2388 011562 000167 000622
2389 011566 005000
2390 011570 004767 004302
2391 011574 004467 002534
2392 011600 036767 167740 167732
2393 011606 001010
2394 011610 036767 167732 167724
2395 011616 001004
2396 011620 050502
2397 011622 005202
2398 011624 000167 000540
2399 011630 004767 005654
2400 011634 004767 005704
2401 011640 020227 000114
2402 011644 001004
2403 011646 062702 000004
2404 011652 000167 000512
2405 011656 111201
2406 011660 001405
2407 011662 004767 006332
2408 011666 004767 007654
2409 011672 000011
2410 011674
2411 011674 105067 167660
2412 011700 112700 000252
2413 011704 110012
2414 011706 016703 167710
2415 011712 053773 001612 000700
2416 011720 052733 000001
2417 011724 005713
2418 011726 001371
2419
2420 011730 110012
2421 011732 016703 167664
2422 011736 043733 001612
2423 011742 005713
2424 011744 001374
2425 011746 016737 167652 000114
2426
2427 011754 105412
2428

```

```

*****
*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
*****
TST17:
JSR RS, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
WWPB0: TST MPRX ;CHECK FOR ANY PARITY MEMORY.
BEQ 1$ ;BR IF NO PARITY MEMORY.
BIT #SW06, 2$WR ;CHECK FOR INHIBIT PARITY SWITCH.
BEQ 2$ ;BR IF NOT SET.
1$: JMP TST2U ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2$: CLR R0 ;ZERO TO BE PUT IN ALL MEMORY.
JSR PC, SETCO ;ROUTINE TO LOAD ALL MEMORY.
JSR R4, INITA ;INITIALIZE THE MEMORY ADDRESS POINTERS.
WWPBYT: BIT BITPT, PMEMAP ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
BNE 2$ ;BR IF PARITY MEM.
BIT BITPT+2, PMEMAP+2 ;...HI 64K.
BNE 2$ ;BR IF PARITY MEM.
BIS RS, R2 ;POINT TO END OF BLOCK.
INC R2 ;FIRST ADDR OF NEXT BLOCK.
JMP WWPB5 ;BR TO FIND NEXT BLOCK.
2$: JSR PC, SETAE ;SET ACTION ENABLE (EVEN IF BANK0.)
JSR PC, CKPME ;CHECK FOR ANY NON TRAP PARITY ERRORS.
WWPB1: CMP R2, #114 ;CHECK IF POINTING TO PARITY ERROR VECTOR.
BNE 3$ ;BR IF NOT AT VECTOR.
ADD #4, R2 ;SKIP PARITY VECTOR.
JMP WWPB5 ;CHECK FOR BLOCK END.
3$: MOVB (R2), R1 ;CHECK IF BYTE STILL CLEARED.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRINT ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SEERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 11 ;ERROR TYPE CODE.
65$: CLRB OEFLG ;CLEAR ODD/EVEN FLAG.
MOVB #252, R0 ;SET UP DATA...EVEN, SETS PARITY BIT.
WWPB2: MOVB R0, (R2) ;MOV DATA INTO TEST LOCATION.
MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
10$: BIS 2$WWP, 2(R3) ;SET WRITE WRONG PARITY.
BIS #AE, 2(R3)+
TST (R3) ;CHECK FOR TABLE TERMINATOR.
BNE 10$ ;BR IF MORE REGS IN TABLE.
;* SET WRONG PARITY IN LOCATION UNDER TEST.
MOVB R0, (R2) ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
11$: BIC 2$WWP, 2(R3)+ ;CLEAR WRITE WRONG PARITY.
TST (R3) ;CHECK FOR TABLE TERMINATOR.
BNE 11$ ;BR IF MORE PARITY REGISTERS.
MOV .PBTRP, 2$PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
;* DETECT WRONG PARITY VIA DATIP: DATOB SHOULDN'T EXECUTE.
NEGB (R2) ;DATIP (DATOB AND COM PARITY BIT.)
;* SHOULD HAVE TRAPPED TO PBTRP.

```

D11

```

2429 011756 016737 167646 000114      MOV    .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
2430 011764 004767 006260      JSR    PC,          SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2431 011770 004767 007552      JSR    PC,          $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2432 011774 000012      .WORD 12              ;ERROR TYPE CODE.
2433 011776 000562      BR     WWPB4          ;SKIP TRAP SERVICE.
2434
2435
2436 012000 016737 167624 000114      ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
2437 012006 022626      PBTRP: MOV    .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2438 012010 016703 167604      CMP    (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
2439 012014 032713 000001      MOV    .MPRO, R3      ;GET PARITY REG AND MAP TABLE POINTER.
2440 012020 001003      21$:  BIT    #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
2441 012022 017301 000000      BNE    22$           ;BR IF IT DOESN'T EXIST.
2442 012026 100413      MOV    @R3, R1        ;GET THE CONTENTS.
2443 012030 062703 000010      BMI    23$           ;BR IF ERROR FLAG SET.
2444 012034 020367 167562      ADD    #10, R3        ;MOVE POINTER TO NEXT REG.
2445 012040 103765      CMP    R3, .MPRX     ;CHECK FOR END OF TABLE.
2446 012042 004767 006202      BLO    21$           ;BR IF MORE REGISTERS.
2447 012046 004767 007474      JSR    PC,          SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
2448 012052 000013      JSR    PC,          $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2449 012054 000533      .WORD 13              ;ERROR TYPE CODE.
2450 012056 036763 167462 000002      BR     WWPB4          ;EXIT AFTER ERROR.
2451 012064 001011      23$:  BIT    BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
2452 012066 036763 167454 000004      BNE    24$           ;BR IF THIS REGISTER CONTROLS THIS BANK.
2453 012074 001005      BIT    BITPT+2, 4(R3) ;CHECK THE HI 64K.
2454 012076 004767 006142      BNE    24$           ;BR IF THIS REGISTER CONTROLS THIS BANK.
2455 012102 004767 007440      65$:  JSR    PC,          SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2456 012106 000014      JSR    PC,          $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2457 012110      .WORD 14              ;ERROR TYPE CODE.
2458 012110 010046      24$:  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    @SKIPAR2, R0  ;ADD MEM MGMT OFFSET.
2468 012144 052700 100001      25$:  BIS    #BIT15+BIT0, R0 ;SET ERROR AND AE BIT IN CHECK WORD.
2469 012150 016367 000006      MOV    6(R3), RESRVD ;GET APPROPRIATE MASK.
2470 012156 046700 167334      BIC    RESRVD, R0    ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2471 012162 046701 167330      BIC    RESRVD, R1    ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2472
2473 012166 020001      ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) COULD BE NOP'ED FOR UNMIXED MEMORY TYPES.
2474 012170 001405      CMP    R0, R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
2475 012172 004767 006046      BEQ    67$           ;BRANCH OVER ERROR CALL IF GOOD DATA.
2476 012176 004767 007344      66$:  JSR    PC,          SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2477 012202 000015      JSR    PC,          $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2478 012204      .WORD 15              ;ERROR TYPE CODE.
2479 012204 005073 000000      67$:  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.
    
```

E11

2485	012230	001370		BNE	26\$			;BR IF THIS PARITY REG DOESN'T EXIST.
2486	012232	017301	000000	MOV	2(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,	SPRNT		;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: MOV	(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 CONTROLLING REG, SO NO TRAP SHOULD OCCURE.				
2495	012256	111212		MOV	(R2), (R2)			;RESTORE RIGHT PARITY
2496				;NOTE: THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS				
2497				WHICH DO ONLY DATOB TO DESTINATION OF MOVW INSTRUCTIONS.				
2498	012260	012603		MOV	(SP)+,R3			;POP STACK INTO R3
2499	012262	017301	000000	MOV	2(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,	SPRNT		;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, 2(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		CMP	R0, R1			;CHECK THE DATA.
2513	012330	001405		BEQ	67\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
2514	012332	004767	005712	66\$: JSR	PC,	SPRNT		;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: MOV	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 WWPB5
2529	012404	004767	005034	JSR	PC,	MAMF		;GO RESET PARITY REGISTERS.

# F11

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC0.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 52  
 T20 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.

SEQ 0135

```

2530 ;*****
2531 ;*TEST 20 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.*
2532 ;*****
2533 TST20:
2534 JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
2535 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2536 RANTST: MOV PC, R3 ;GET CURRENT PROGRAM COUNTER.
2537 BIC #7777, R3 ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
2538 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2539 1$: MOV R2, -(SP) ;SAVE MEMORY POINTER.
2540 MOV R3, -(SP) ;SAVE "DATA" POINTER.
2541 2$: MOV (R3)+, (R2)+ ;MOV CODE INTO TEST MEMORY.
2542 BIT #7777, R3 ;CHECK FOR END OF "DATA TABLE"
2543 BNE 3$ ;BRANCH IF MORE
2544 SUB #10000, R3 ;RESET POINTER TO START OF "RANDOM DATA"
2545 3$: BIT R5, R2 ;CHECK FOR END OF BLOCK
2546 BNE 2$ ;BRANCH IF MORE
2547 MOV (SP)+, R3 ;RESET "DATA" POINTER.
2548 MOV (SP)+, R2 ;RESET MEMORY POINTER.
2549 4$: MOV (R3)+, R0 ;GET S/B DATA.
2550 MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
2551 CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2552 BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2553 64$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2554 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2555 .WORD 20 ;ERROR TYPE CODE.
2556 65$:
2557 BIT #7777, R3 ;CHECK FOR END OF "DATA TABLE"
2558 BNE 5$ ;BR IF MORE.
2559 SUB #10000, R3 ;RESET POINTER TO TOP OF "DATA TABLE".
2560 5$:
2561 BIT R5, R2 ;CHECK FOR END OF A BLOCK.
2562 BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
2563 JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```



2564  
2565  
2566  
2567  
2568  
2569  
2570  
2571  
2572  
2573  
2574  
2575  
2576  
2577  
2578  
2579  
2580  
2581  
2582  
2583  
2584  
2585  
2586  
2587  
2588  
2589  
2590  
2591  
2592  
2593  
2594  
2595  
2596  
2597  
2598  
2599  
2600  
2601  
2602  
2603  
2604  
2605  
2606  
2607  
2608  
2609  
2610  
2611  
2612  
2613

012524  
012524 004567 006006  
012530 000003  
012532 000167 000056  
012536 012703 010412  
012542 012704 000205  
012546 010400  
012550 004467 001560  
012554 010322  
012556 010412  
012560 004542  
012562 012201  
012564 020001  
012566 001405  
012570 004767 005474  
012574 004767 006746  
012600 000021  
012602  
012602 010322  
012604 030502  
012606 001363  
012610 004767 002276

```

.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 BOUNDARY BIT MASK.
*****
TST21:
      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST22      ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
DIDO:  MOV      #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
      MOV      #205,  R4      ;GET 'RTS R5'
      MOV      R4,      R0      ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      1INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    MOV      R3,      (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:    MOV      R4,      (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2)   ;GO EXECUTE THE IUT.
      MOV      (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC,      SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:   MOV      R3,      (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
      BNE      2$          ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      NMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

```

# H11

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC.D.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 54  
 T22 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.

SEQ 0137

2614  
 2615  
 2616  
 2617  
 2618  
 2619  
 2620  
 2621  
 2622  
 2623  
 2624  
 2625  
 2626  
 2627  
 2628  
 2629  
 2630  
 2631  
 2632  
 2633  
 2634  
 2635  
 2636  
 2637  
 2638  
 2639  
 2640  
 2641  
 2642  
 2643  
 2644  
 2645  
 2646  
 2647  
 2648  
 2649  
 2650  
 2651  
 2652  
 2653  
 2654  
 2655  
 2656  
 2657  
 2658  
 2659  
 2660  
 2661  
 2662

012614			
012614	004567	005716	
012620	000003		
012622	000167	000060	
012626	012703	110412	
012632	012704	000205	
012636	012700	110605	
012642	004467	001466	
012646	010322		
012650	010412		
012652	004542		
012654	012201		
012656	020001		
012660	001405		
012662	004767	005402	
012666	004767	006654	
012672	000021		
012674			
012674	010322		
012676	030502		
012700	001363		
012702	004767	002204	

```

*****
*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 LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
*          LOCATION              PLACED THERE      AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000              110412           110605
* THRU TEST / 40002            000205           000205
*
* 2ND PASS / 40002              110412           110605
* THRU TEST / 40004            000205           000205
*
*          ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†S122:
      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST23        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
DIDBL·  MOV      #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
      MOV      #205, R4        ;GET 'RTS R5'
      MOV      #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    MOV      R3,      (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:    MOV      R4,      (R2)   ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2)   ;GO EXECUTE THE IUT.
      MOV      (R2)+, R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC,      SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:   MOV      R3,      (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
      BNE      2$          ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

2663  
2664  
2665  
2666  
2667  
2668  
2669  
2670  
2671  
2672  
2673  
2674  
2675  
2676  
2677  
2678  
2679  
2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699  
2700  
2701  
2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712

012706  
012706 004567 005624  
012712 000003  
012714 000167 000064  
012720 012703 110342  
012724 012704 000205  
012730 012700 161342  
012734 004467 001374  
012740 010322  
012742 010412  
012744 004562 177776  
012750 005302  
012752 012201  
012754 020001  
012756 001405  
012760 004767 005304  
012764 004767 006556  
012770 000021  
012772  
012772 010322  
012774 030502  
012776 001361  
013000 004767 002106

```

*****
*TEST 23 EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOV B R3, -(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV B' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
*          LOCATION              PLACED THERE      AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000              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 BOUNDARY BIT MASK.
*****
†ST23:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3          ;MINIMUM BLOCK SIZE OF 2 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST24     ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
DIOBH: MOV      #110342, R3 ;GET 'MOV B R3, -(R2)' INSTRUCTION (IUT).
      MOV      #205,    R4  ;GET 'RTS R5'
      MOV      #161342, R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:    MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:    MOV      R4,      (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -2(R2) ;GO EXECUTE THE IUT.
      DEC      R2          ;ADJUST R2 TO POINT TO MAUT.
      MOV      (R2)+,   R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR      PC,      SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:   MOV      R3,      (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2    ;CHECK FOR END OF A BLOCK.
      BNE      2$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729  
2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761

013004  
013004 004567 005526  
013010 000003  
013012 000167 000060  
013016 012703 005412  
013022 012704 000205  
013026 012700 172366  
013032 004467 001276  
013036 010322  
013040 010412  
013042 004542  
013044 012201  
013046 020001  
013050 001405  
013052 004767 005212  
013056 004767 006464  
013062 000021  
013064  
013064 010322  
013066 030502  
013070 001363  
013072 004767 002014

\*\*\*\*\*  
\*TEST 24 EXECUTE DATI, DATIP, DATO THRU MEMORY.  
\* EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.  
\* AN 'RTS RS' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN  
\* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.  
\* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:  
\*  
\* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION  
\* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION  
\*  
\* 1ST PASS / 40000 005412 172366  
\* THRU TEST / 40002 000205 000205  
\*  
\* 2ND PASS / 40002 005412 172366  
\* THRU TEST / 40004 000205 000205  
\*  
\* ETC., ETC., ETC.  
\*  
\* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).  
\* R1 = DATA READ FROM MEMORY (WAS).  
\* R2 = ADDRESS OF IUT/DATA.  
\* R3 = INSTRUCTION UNDER TEST (IUT).  
\* R4 = RTS RS (CODE 205).  
\* R5 = BLOCK BOUNDARY BIT MASK.  
\*\*\*\*\*

TST24:  
JSR R5, \$SCOPE ;GO TO SCOPE ROUTINE.  
.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS  
;REQUIRED FOR THIS TEST.  
JMP TST25 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK  
;AVAILABLE FOR TEST.  
DIPDO: MOV #005412,R3 ;GET 'NEG (R2)' INSTRUCTION (IUT).  
MOV #205,R4 ;GET 'RTS RS'  
MOV #172366,R0 ;SET UP S/B DATA AFTER EXECUTION.  
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1\$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.  
2\$: MOV R4, (R2) ;PUT 'RTS RS' FOLLOWING IUT.  
JSR R5, -(R2) ;GO EXECUTE THE IUT.  
MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.  
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 65\$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
64\$: JSR PC, SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, \$ERROR ;\*\*\* ERROR \*\*\* (GO TYPE A MESSAGE)  
.WORD 21 ;ERROR TYPE CODE.  
65\$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.  
BIT R5, R2 ;CHECK FOR END OF A BLOCK.  
BNE 2\$ ;BRANCH IF MORE IN CURRENT BLOCK.  
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1\$.

# K11

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC0.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 57  
 T25 EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.

SEQ 0140

```

2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787 013076
2788 013076 004567 005434
2789 013102 000003
2790
2791 013104 000167 000060
2792
2793 013110 012703 142242
2794 013114 012704 000205
2795 013120 012700 142000
2796 013124 004467 001204
2797 013130 010322
2798 013132 010412
2799 013134 004542
2800 013136 012201
2801 013140 020001
2802 013142 001405
2803 013144 004767 005120
2804 013150 004767 006372
2805 013154 000021
2806 013156
2807 013156 010322
2808 013160 030502
2809 013162 001363
2810 013164 004767 001722
  
```

```

*****
*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 LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
*          LOCATION              PLACED THERE      AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000              142242          142000
* THRU TEST / 40002              000205          000205
*
* 2ND PASS / 40002              142242          142000
* THRU TEST / 40004              000205          000205
*
*          ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†T25:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3        ;MINIMUM BLOCK SIZE OF 2 WORDS
                          ;REQUIRED FOR THIS TEST.
      JMP      TST26    ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                          ;AVAILABLE FOR TEST.
DPDBL: MOV      #142242,R3 ;GET BICB (R2)+, -(R2)' INSTRUCTION (IUT).
      MOV      #205, R4   ;GET 'RTS R5'
      MOV      #142000,R0 ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV      R4,      (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2) ;GO EXECUTE THE IUT.
      MOV      (R2)+,   R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ      65$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRINT3 ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   21          ;ERROR TYPE CODE.
65$:  MOV      R3,      (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
      BIT      R5,      R2    ;CHECK FOR END OF A BLOCK.
      BNE     2$         ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820  
2821  
2822  
2823  
2824  
2825  
2826  
2827  
2828  
2829  
2830  
2831  
2832  
2833  
2834  
2835  
2836  
2837  
2838  
2839  
2840  
2841  
2842  
2843  
2844  
2845  
2846  
2847  
2848  
2849  
2850  
2851  
2852  
2853  
2854  
2855  
2856  
2857  
2858  
2859  
2860

013170  
013170 004567 005342  
013174 000003  
013176 000167 000062  
013202 012703 152212  
013206 012704 000205  
013212 012700 157212  
013216 004467 001112  
013222 010322  
013224 010412  
013226 004542  
013230 005302  
013232 012201  
013234 020001  
013236 001405  
013240 004767 005024  
013244 004767 006276  
013250 000021  
013252  
013254 030502  
013256 001362  
013260 004767 001626

```

*****
*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 LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 152212 157212
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 152212 157212
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†ST26: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST27 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DPDBH: MOV #152212, R3 ;GET 'BISB (R2)+, (R2)' INSTRUCTION (IUT).
        MOV #205, R4 ;GET 'RTS R5'
        MOV #157212, R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) ;GO EXECUTE THE IUT.
        DEC R2 ;RESET R2 TO POINT TO IUT.
        MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21 ;ERROR TYPE CODE.
65$: MOV R3, (R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT R5, R2 ;CHECK FOR END OF A BLOCK.
        BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

2861  
2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869  
2870  
2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2894  
2895  
2896 013264  
2897 013264 004567 005246  
2898 013270 000000  
2899 013272 004467 001036  
2900 013276 005003  
2901 013300 012700 000377  
2902 013304 010022  
2903 013306 030502  
2904 013310 001375  
2905 013312 014201  
2906 013314 020001  
2907 013316 001405  
2908 013320 004767 004750  
2909 013324 004767 006216  
2910 013330 000010  
2911 013332  
2912 013332 000300  
2913 013334 010012  
2914 013336 011201  
2915 013340 020001  
2916 013342 001405  
2917 013344 004767 004724  
2918 013350 004767 006172  
2919 013354 000010  
2920 013356  
2921 013356 000300  
2922 013360 005703  
2923 013362 001403  
2924 013364 020327 000003  
2925 013370 001010  
2926 013372 030502

.SBTTL SECTION 4:MOS TESTS

\*\*\*\*\*  
\*TEST 27 MARCHING 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.  
\*\*\*\*\*

TST27:

JSR R5, \$SCOPE ;GO TO SCOPE ROUTINE.  
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.  
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.  
1\$: CLR R3 ;CLEAR PASS COUNTER  
MOV #000377, R0 ;SETUP TO WRITE PATTERN  
2\$: MOV R0, (R2) ;WRITE PATTERN  
BIT R5, R2 ;END OF 4K?  
BNE 2\$ ;CONTINUE WRITING IF NO.  
3\$: MOV -(R2), R1 ;GET DATA WRITTEN  
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 65\$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
64\$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, \$ERROR ;\*\*\* ERROR \*\*\* (GO TYPE A MESSAGE)  
.WORD 10 ;ERROR TYPE CODE.  
65\$: SWAB R0 ;SWAP BYTES OF DATA  
4\$: MOV R0, (R2) ;WRITE SWAPPED WORD  
MOV (R2), R1 ;GET DATA WRITTEN  
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.  
BEQ 67\$ ;BRANCH OVER ERROR CALL IF GOOD DATA.  
66\$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.  
JSR PC, \$ERROR ;\*\*\* ERROR \*\*\* (GO TYPE A MESSAGE)  
.WORD 10 ;ERROR TYPE CODE.  
67\$: SWAB R0 ;PUT DATA BACK TO ORIGINAL  
TST R3 ;IF ON PASS 0 OR PASS 3  
BEQ 5\$ ;WE ARE ADDRESSING DOWN  
CMP R3, #3 ;IF ON PASS 1 OR 2 GO TO  
BNE 6\$ ;UPWARD  
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      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      JSR      PC,    SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
2930 013434 004767 006106      JSR      PC,    $ERROR  ; *** ERROR *** (GO TYPE A MESSAGE)
2931 013440 000010      .WORD   10      ; ERROR TYPE CODE.
2932 013442
2933 013442 000733      BR      4$
2934 013444 005203      8$:      INC      R3      ; INCREMENT PASS COUNTER
2935 013446 000300      SWAB    R0      ; SET UP NEW READ WORD
2936 013450 020327 000002      CMP      R3,#2   ; ADDRESSING UP?
2937 013454 001716      BNE      3$      ; 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
2944
2945
2946
2947
2948
2949
2950
2951
2952
2953
2954
*****
* TEST 30 WRITE CHECKERBOARD STARTING WITH '125252' DATA.
* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY, STALL
* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
* DETERIORATE BETWEEN REFRESH CYCLES.
*
* R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
* R1=DATA READ FROM MEMORY(WAS)
* R2=VIRTUAL ADDRESS
* R3=SMALL LOOP COUNTER FOR STALL
* R4=NUMBER OF TIMES SMALL LOOP DONE
* R5=BLOCK BOUNDARY BIT MASK.
*****

```

```

2955 013470
2956 013470 004567 005042      †ST30: JSR      R5,    $SCOPE ; GO TO SCOPE ROUTINE.
2957 013474 000000      .WORD   0        ; NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2958 013476 004467 000632      JSR      R4,    INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2959 013478 012700 125252      MOV      #125252,R0 ; SETUP DATA PATTERN
2960 013476 010022      1$:      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$.
2965 013512 005003      CLR      R3      ; SET UP COUNTER FOR STALL
2966 013514 012704 000046      MOV      #46,   R4   ; DO LOOP 46 TIMES OR 2 SEC. TOTAL.
2967 013530 005303      2$:      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
2974 013550 012201
2975 013552 020001
2976 013554 001405
2977 013556 004767 004512
2978 013562 004767 005760
2979 013570 000006
2980 013570
2981 013570 005100
2982 013572 030502
2983 013574 001365
2984 013576 004767 001310
2985
2986
2987
2988 013602
2989 013602 004567 004730
2990 013606 000000
2991 013610 004467 000520
2992 013614 012700 052525
2993 013620 010022
2994 013622 005100
2995 013624 030502
2996 013626 001374
2997 013630 004767 001256
2998 013634 005003
2999 013636 012704 000046
3000 013642 005303
3001 013644 001376
3002 013646 005304
3003 013650 001374
3004 013652 004467 000456
3005 013656 012700 052525
3006 013662
3007 013662 012201
3008 013664 020001
3009 013666 001405
3010 013670 004767 004400
3011 013674 004767 005646
3012 013700 000006
3013 013702
3014 013702 005100
3015 013704 030502
3016 013706 001365
3017 013710 004767 001176

3$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRN2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERFOR ;*** ERROR *** (WU TYPE A MESSAGE)
.WORD 6 ;ERROR TYPE CODE.

65$:
COM RO
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 3$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
;*****
;TEST 31 WRITE CHECKERBOARD STARTING WITH 052525 DATA
;*****
TST31:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
MOV #052525, RO ;SETUP DATA PATTERN
1$: MOV RO, (R2)+ ;WRITE A WORD
COM RO
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
CLR R3 ;SET COUNTER FOR LOOP
MOV #46, R4 ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
25: DEC R3
BNE 2$
DEC R4
BNE 2$
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
MOV #052525, RO ;INIT PATTERN FOR CHECKING

3$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRN2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERFOR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 6 ;ERROR TYPE CODE.

65$:
COM RO
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 3$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

C12

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMCD.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 62  
 DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.

SEQ 0145

3018						.SBTTL	DONE:	RELOCATE PROGRAM AND REPEAT ALL TESTS.
3019	013714					DONE:		
3020	013714	004567	004616			JSR	RS,	\$SCOPE ;GO TO SCOPE ROUTINE.
3021	013720	000000				.WORD	0	;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3022	013722	005067	165242		TST32:	CLR	\$TIMES	;RESET ITERATION COUNTER FOR RESTARTING TEST.
3023	013726	105067	165150			CLRB	\$TSTNM	;RESET TEST NUMBER.
3024	013732	036767	164644	165574	1\$:	BIT	PRGMAP, SAVTST	;CHECK IF PROGRAM IS IN TEST AREA.
3025	013740	001004				BNE	2\$	;BR IF IT PROG IN MEM TO BE TESTED.
3026	013742	036767	164636	165566		BIT	PRGMAP+2, SAVTST+2	;CHECK HI 64K
3027	013750	001434				BEQ	\$EOP	;BR IF PROG NOT IN MEM TO BE TESTED.
3028	013752	032777	000200	165160	2\$:	BIT	\$SW07, \$SWR	;CHECK FOR INHIBIT RELOCATION SWITCH.
3029	013760	001030				BNE	\$EOP	;SKIP RELOCATION IF SWITCH SET.
3030	013762	022767	000003	164612		CMP	#3, PRGMAP	;CHECK IF PROGRAM IN FIRST BK.
3031	013770	001012				BNE	4\$	;BR IF NOT IN FIRST BK.
3032	013772	005737	000042			TST	2#42	;CHECK FOR A ACT11.
3033	013776	001014				BNE	5\$	;BR IF A ACT11.
3034	014000	105737	001224			TSTB	2#ENV	;CHECK FOR APT
3035	014004	001011				BNE	5\$	;IF APT DO NOT RELOCATE
3036	014006	004767	002354			JSR	PC	RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.
3037	014012	000167	172030		3\$:	JMP	START1	;LOOP BACK AND RUN ALL TESTS AGAIN.
3038								
3039	014016	004767	002746		4\$:	JSR	PC	RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
3040	014022	005737	000042			TST	2#42	;TEST FOR XXDP
3041	014026	001402				BEQ	6\$	;IF NOT RUNNING UNDER MON. DONT
3042	014030	004767	003142		5\$:	JSR	PC,	RESLDR ;RESTORE LOADERS.
3043	014034				6\$:			
3044	014034	004567	007360			JSR	RS	\$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3045	014040	001201				.WORD	\$CRLF	;ADDRESS OF MESSAGE TO BE TYPED

```

3046
3047
3048
3049
3050
3051
3052
3053
3054
3055 014042
3056 014042 000240
3057 014044 005067 165120
3058 014050 005267 165136
3059 014054 042767 100000 165130
3060 014062 005327
3061 014064 000001
3062 014066 003035
3063 014070 012737
3064 014072 000001
3065 014074 014064
3066 014076 004567 007316
3067 014102 014166
3068 014104 016746 165102
3069
3070
3071 014110 013746 177776
3072 014114 004767 010220
3073 014120 004567 007274
3074 014124 014203
3075 014126
3076
3077 014126 016700 163710
3078 014132 001413
3079 014134 000005
3080 014136 004710
3081 014140 000240
3082 014142 000240
3083 014144 000240
3084 014146 023737 000042 000046
3085 014154 001402
3086 014156 004767 003074
3087 014162
3088 014162 000167 171660
3089 014166 005015 047105 020104
3090 014174 040520 051523 021440
3091 014202 000
3092 014203 377 377 000
3093
3094
3095
3096
3097
3098
3099
3100
3101

```

```

;*****
.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
SEOP:
NOP
CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
INC $PASS ;;INCREMENT THE PASS NUMBER
BIC #10000,$PASS ;;DON'T ALLOW A NEG. NUMBER
DEC (PC)+ ;;LOOP?
SEOPCT: .WORD 1
BGT $DOAGN ;;YES
MOV (PC)+,(PC)+ ;;RESTORE COUNTER
SENDCT: .WORD 1
SEOPCT
JSR RS, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;.WORD $SENDMG ;ADDRESS OF MESSAGE TO BE TYPED
MOV $PASS,-($SP) ;SAVE $PASS FOR TYPEOUT
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPDS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
MOV @#PSW,-($SP) ;PUT THE PROCESSOR STATUS ON THE STACK
JSR PC,$STYPDS ;GO TO THE SUBROUTINE
JSR RS,$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
;.WORD $ENULL ;ADDRESS OF MESSAGE TO BE TYPED
SGET42:
MOV 42,RO ;;GET MONITOR ADDRESS
BEQ $DOAGN ;;BRANCH IF NO MONITOR
RESET ;;CLEAR THE WORLD
SENDAD: JSR PC,(RO) ;GO TO MONITOR
NOP ;SAVE ROOM
NOP ;FOR
NOP ;ACT11
CMP @#42,@#46 ;ARE WE UNDER ACT11 OR XXDP
BEQ $DOAGN ;IF ACT11 THEN RESTART
JSR PC,$SAVLDL ;IF XXDP FIRST SAVE MONITOR
$DOAGN: JMP START1 ;RETURN*****
SENDMG: .ASCIZ '<15><12>/END PASS #/'
$ENULL: .BYTE -1,-1,0 ;NULL CHARACTER STRING
.SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
.SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
;*****
;* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
;* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
;* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
;* PARS 4, 5, AND 6 ARE UNUSED.
;*****

```

```

3102 014206
3103 014206 012737 077406 172300
3104 014214 012737 077406 172302
3105 014222 012737 077406 172304
3106 014230 012737 077406 172306
3107 014236 005037 172310
3108 014242 005037 172312
3109 014246 005037 172314
3110 014252 012737 077406 172316
3111 014260 005037 172340
3112 014264 012737 000200 172342
3113 014272 005037 172344
3114 014276 005037 172346
3115 014372 005037 172350
3116 014306 005037 172352
3117 014312 005037 172354
3118 014316 012737 007600 172356
3119 014324 012737 000001 177572
3120 014332 000207
3121
3122
3123
3124
3125
3126 014334 012767 000001 165202
3127 014342 005067 165200
3128 014346 005002
3129 014350 016705 165232
3130 014354 005767 164226
3131 014360 001514
3132 014362 005037 172344
3133 014366 012702 040000
3134 014372 036767 165146 165130 1$:
3135 014400 001015
3136 014402 036767 165140 165122
3137 014410 001011
3138 014412 062737 000200 172344
3139 014420 006367 165120
3140 014424 006167 165116
3141 014430 100360
3142 014432 000000
3143 014434 036767 165104 165140 2$:
3144 014442 001004
3145 014444 036767 165076 165132
3146 014452 001405
3147 014454 016705 165120 3$:
3148 014460 042767 020000 165110
3149 014466 013737 172344 172346 4$:
3150 014474 016767 165044 165046
3151 014502 016767 165040 165042
3152 014510 032705 020000
3153 014514 001505
3154 014516 062737 000200 172346 5$:
3155 014524 006367 165020
3156 014530 006167 165016
3157 014534 100473

```

```

MMINIT:
MOV #200-1#400+UP+RW, @#KIPDR0 ;SET KIPDR0 = RW UP 200 BLOCKS
MOV #200-1#400+UP+RW, @#KIPDR1 ;SET KIPDR1 = RW UP 200 BLOCKS
MOV #200-1#400+UP+RW, @#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
MOV #200-1#400+UP+RW, @#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
CLR @#KIPDR4
CLR @#KIPDR5
CLR @#KIPDR6
MOV #200-1#400+UP+RW, @#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
CLR @#KIPAR0 ;MAP PAR0 INTO BANK0
MOV #200, @#KIPAR1 ;MAP PAR1 INTO BANK1
CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
CLR @#KIPAR3
CLR @#KIPAR4
CLR @#KIPAR5
CLR @#KIPAR6
MOV #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
MOV #1, @#SRO ;ENABLE MEMORY MANAGEMENT
RTS PC ;RETURN

```

```

*****
* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
*****

```

```

INITMM: MOV #BIT0, BITPT ;SET POINTER TO BANK0
CLR BITPT+2 ;CLEAR HI 64K BANK POINTERS
CLR R2 ;SET ADDRESS POINTER TO 0
MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
BEQ 10$ ;BRANCH IF NO MEM MGMT
CLR @#KIPAR2 ;SET UP 3RD PAR TO BANK0
MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
BIT BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED
BNE 2$ ;BRANCH IF MATCH
BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
BNE 2$ ;BRANCH IF MATCH
ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
ROL BITPT+2 ;...HI POINTER.
BPL 1$ ;BR IF MORE.
HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
BIT BITPT, LADMAP ;CHECK IF LAST BANK.
BNE 3$ ;BR IF LAST BANK.
BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
BEQ 4$ ;BR IF NOT LAST BANK.
MOV LADMSK, R5 ;SET MASK TO FIND LAST ADR.
BIC #20000, TMAPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
MOV BITPT+2, TMPPT+2 ;...HI 64K.
BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
BEQ 21$ ;BRANCH IF NOT 8K.
ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
ROL TMPPT+2 ;...HI POINTER.
BMI 20$ ;BR IF NO MORE.

```

F12

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	014554	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, TMAPLAD	;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	014622	100367				BPL	10\$	;BR IF MORE BANKS.
3175	014624	000000				HALT		;FATAL ERROR!!! NO 4K BANK FOUND?
3176	014626	016767	164702	164704	11\$:	MOV	BITPT, TMPPT	;COPY BANK POINTER.
3177	014624	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	162530	164502		ADD	RELOC, 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$: BIT BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADDR.
3215 015066 001004 164460 164514 BNE 34$ ;BR IF LAST BANK FLAG FOUND.
3216 015070 036767 164452 164506 BIT BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADDR.
3217 015076 001402 164452 164506 BEQ INITEX ;BR IF NO LAD FLG FOUND.
3218 015100 016702 164470 34$: MOV LSTADR, R2 ;SET UP R2.
3219 015104 010467 164444 INITEX: MOV R4, MMORE ;PUT RETURN PC INTO "MMORE"
3220 015110 000204 RTS R4 ;RETURN
3221
3222 ;*****
3223 ;* COMMON UPWARDS ADDRESSING ROUTINE
3224 ;* FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3225 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3226 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3227 ;*****
3228 015112 036767 164432 164462 MMUP: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3229 015120 001122 164432 164462 BNE 10$ ;BR IF LAST BANK.
3230 015122 036767 164424 164454 BIT TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3231 015130 001116 164424 164454 BNE 10$ ;BR IF LAST BANK.
3232 015132 016705 164450 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3233 015136 005767 163444 TST MMVA ;CHECK FOR MEM MGMT AVAILABLE
3234 015142 001515 163444 BEQ 20$ ;BRANCH IF NO MEM MGMT
3235 015144 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3236 015150 062737 000200 172344 1$: ADD #200, #KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3237 015156 006367 164362 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3238 015162 006167 164360 ROL BITPT+2 ;...HI POINTER.
3239 015166 100577 164360 BMI 32$ ;BR IF ALL DONE.
3240 015170 036767 164350 164332 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS
3241 015176 001004 164350 164332 BNE 2$ ;BRANCH IF MATCH
3242 015200 036767 164342 164324 BIT BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3243 015206 001760 164342 164324 BEQ 1$ ;BRANCH IF NO MATCH
3244 015210 036767 164330 164364 2$: BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3245 015216 001004 164330 164364 BNE 3$ ;BRANCH IF LAST BANK FLAG.
3246 015220 036767 164322 164356 BIT BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3247 015226 001405 164322 164356 BEQ 4$ ;BR IF NOT LAST BANK.
3248 015230 016705 164344 164334 3$: MOV LADMSK, R5 ;RESET MASK.
3249 015234 042767 020000 164334 BIC #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3250 015242 016767 164276 164300 4$: 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 172344 172346 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$: ADD #200, #KIPAR3 ;UP DATE FORTH PAR.
3256 015300 006367 164244 ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3257 015304 006167 164242 ROL TMPPT+2 ;...HI POINTER.
3258 015310 100513 164232 164210 BMI 30$ ;BR IF NO MORE.
3259 015312 036767 164232 164210 6$: BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3260 015320 001004 164224 164202 BNE 7$ ;BRANCH IF A MATCH.
3261 015322 036767 164224 164202 BIT TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3262 015330 001760 164212 164242 BEQ 5$ ;BRANCH IF NO MEMORY
3263 015332 036767 164212 164242 7$: BIT TMPPT,LADMAP ;CHECK FOR LAST BANK FLAG.
3264 015340 001004 164204 164234 BNE 8$ ;BRANCH IF A MATCH
3265 015342 036767 164204 164234 BIT TMPPT+2,LADMAP+2 ;CHECK HI 64K
3266 015350 001475 164222 164212 BEQ 31$ ;BR IF NO LAST BANK FLAG.
3267 015352 016705 164222 164212 8$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.
3268 015356 052767 020000 164212 BIS #20000, TEMPLAD ;SET VIRTUAL ADR TO BANK 3.
3269 015364 000467 BR 31$

```

# H12

```

3270
3271 015366 026702 164204      10$:  CMP      TMAPLAD, R2      ;CHECK IF LAST ADR REACHED.
3272 015372 001064              BNE      31$              ;BR IF MORE.
3273 015374 000474              BR       32$              ;BR IF ALL DONE.
3274
3275 015376 106267 164155      20$:  ASRB     FLAGBK              ;SHIFT BK FLAG
3276 015402 001407              BEQ     22$              ;BR IF NOT BK BLOCK.
3277 015404 103455              BCS     30$              ;BR IF ANOTHER 4K.
3278 015406 105067 164145      CLR     FLAGBK              ;CLEAR OUT ALL FLAGS.
3279 015412 162702 040000      SUB     #40000, R2         ;BACK UP BK.
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$:  MOV     BITPT, TMPPT      ;SET UP TMP POINTER.
3289 015462 032705 020000      BIT     #BIT13, R5         ;CHECK FOR BK BLOCK SIZE.
3290 015466 001426              BEQ     31$              ;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  MOV     #11, FLAGBK        ;SET BK 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 TO FIND 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
3307 015566 005767 164476      ;* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3308 015572 001402              32$:  TST     MPRX              ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3309 015574 004767 001744              BEQ     33$              ;BR IF NONE.
3310 015600 000207      33$:  JSR     PC, CKPME         ;CHECK FOR PARITY MEMORY ERRORS.
3311
3312
3313
3314
3315
3316
3317
3318 015602 036767 163736 163760  MNDOWN: BIT     BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
3319 015610 001004              BNE     1$                ;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
  
```

```

;*****
;* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
;* FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
;*****
  
```

3326	015636	001425				BEQ	6\$		; BRANCH IF NOT
3327	015640	162737	000200	172344	3\$:	SUB	#200, 2#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	MORE, (SP)		; RESET RETURN ADDRESS
3348	015760	000207			10\$:	RTS	PC		; RETURN



SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.

3349  
3350  
3351  
3352  
3353  
3354 015762 010200  
3355 015764 005067 163170  
3356 015770 005767 162612  
3357 015774 001417  
3358 015776 010146  
3359 016000 013701 172344  
3360 016004 006301  
3361 016006 006301  
3362 016010 006301  
3363 016012 006301  
3364 016014 006301  
3365 016016 006167 163136  
3366 016022 006301  
3367 016024 006167 163130  
3368 016030 060100  
3369 016032 012601  
3370 016034 000207  
3371  
3372  
3373  
3374  
3375 016036 005000  
3376 016040 010146  
3377 016042 010246  
3378 016044 016701 163474  
3379 016050 016702 163472  
3380 016054 006202  
3381 016056 006001  
3382 016060 103403  
3383 016062 105200  
3384 016064 100373  
3385 016066 000000  
3386 016070  
3387 016070 012602  
3388 016072 012601  
3389 016074 000207  
3390  
3391  
3392  
3393  
3394 016076  
3395 016076 004467 176232  
3396 016102 010022  
3397 016104 030502  
3398 016106 001375  
3399 016110 004767 176776  
3400 016114 000207

```

.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 STMP0.
*****
PHYADR: MOV     R2,    R0      ;VIRTUAL INTO R0
        CLR     STMP0      ;CLEAR TEMP SAVE OF HIGH BITS
        TST    MMVA      ;CHECK FOR MEM MGMT AVAILABLE
        BEQ    1$        ;BRANCH IF NO MEM MGMT
        MOV    R1, -(SP)    ;PUSH R1 ON STACK
        MOV    2*IPAR2, R1 ;GET PAR TO BE ADDED TO VIRTUAL
        ASL   R1          ;SHIFT IT 6 TIMES
        ASL   R1
        ASL   R1
        ASL   R1
        ASL   R1
        ROL   STMP0      ;SAVE EXTRA BITS
        ASL   R1
        ROL   STMP0
        ADD   R1,    R0      ;ADD SHIFTED PAR TO VIRTUAL
        MOV   (SP)+,R1     ;POP STACK INTO R1
1$:     RTS    PC        ;RETURN
*****
* SUBROUTINE TO PUT BANK NUMBER INTO R0.
*****
BANKNO: CLR     R0      ;INIT R0
        MOV    R1, -(SP) ;PUSH R1 ON STACK
        MOV    R2, -(SP) ;PUSH R2 ON STACK
        MOV    BITPT, R1 ;GET BANK MAP POINTER...LO 64K.
        MOV    BITPT+2, R2 ;...HI 64K.
1$:     ASR   R2      ;SHIFT POINTER...HI
        ROR   R1      ;...LO
        BCS   2$      ;BR WHEN POINTER FOUND.
        INCB R0      ;COUNT BANKS.
        BPL   1$      ;BR IF NOT OVERFLOW.
        HALT          ;FATAL ERROR!!! NO POINTER FOUND.
2$:     MOV   (SP)+,R2 ;POP STACK INTO R2
        MOV   (SP)+,R1 ;POP STACK INTO R1
        RTS   PC      ;RETURN
*****
* SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
*****
SETCON: JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2$:     MOV    R0,    (R2)+ ;MOV CONSTANT INTO MEMORY
        BIT   R5,    R2     ;CHECK FOR END OF A BLOCK.
        BNE   2$        ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR   PC,    MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
        RTS   PC        ;RETURN

```

K12

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMC.D.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 70  
SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.

SEQ 0153

```

3401
3402
3403
3404 016116 106112
3405 016120 106112
3406 016122 106112
3407 016124 106112
3408 016126 106112
3409 016130 106112
3410 016132 106112
3411 016134 106112
3412 016136 106122
3413 016140 106112
3414 016142 106112
3415 016144 106112
3416 016146 106112
3417 016150 106112
3418 016152 106112
3419 016154 106112
3420 016156 106112
3421 016160 106122
3422 016162 000207
3423
3424
3425
3426
3427 016164 012704 000020
3428
3429 016170 010022
3430 016172 010022
3431 016174 010022
3432 016176 010022
3433
3434 016200 010322
3435 016202 010322
3436 016204 010322
3437 016206 010322
3438
3439 016210 010022
3440 016212 010022
3441 016214 010022
3442 016216 010022
3443
3444 016220 010322
3445 016222 010322
3446 016224 010322
3447 016226 010322
3448
3449 016230 005304
3450 016232 001356
3451 016234 010046
3452 016236 010300
3453 016240 012603
3454 016242 000207

```

```

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

```

```

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

```

```

3455 .SBTTL RELOCATION SUBROUTINES.
3456 *****
3457 * ROUTINE TO RELOCATE PROGRAM CODE
3458 *****
3459 RELOC:
3460 MOV R2,-(SP) ;;PUSH R2 ON STACK
3461 MOV R3,-(SP) ;;PUSH R3 ON STACK
3462 MOV R4,-(SP) ;;PUSH R4 ON STACK
3463 4$: MOV (R5)+, R2 ;;GET FIRST LOCATION.
3464 MOV (R5)+, R3 ;;GET FIRST LOCATION OF DESTINATION.
3465 MOV #20000, R4 ;;SET UP BK COUNTER.
3466 1$: MOV (R2)+, (R3)+ ;;MOV THE DATA.
3467 DEC R4 ;;COUNT THE WORDS.
3468 BNE 1$ ;;BR IF MORE.
3469 MOV #20000, R4 ;;RESET THE COUNTER.
3470 2$: CMP -(R2), -(R3) ;;CHECK THE DATA JUST MOVED.
3471 BEQ 3$ ;;BR IF DATA OK.
3472 MOV (R2), $GDDAT ;;GET SOURCE DATA.
3473 MOV (R3), $BDDAT ;;GET DESTINATION DATA.
3474 MOV R2, $GDADR ;;GET SOURCE ADDRESS.
3475 MOV R3, $BDADR ;;GET DESTINATION ADDRESS.
3476 JSR PC, $ERRR ;;** ERROR ** (GO TYPE A MESSAGE)
3477 .WORD 23 ;;ERROR TYPE CODE.
3478 HALT ;;FATAL ERROR!!! RELOCATION FAILED.
3479 SUB #4, R5 ;;ADJUST RETURN POINTER.
3480 BR 4$ ;;GO BACK AND TRY AGAIN.
3481 3$: DEC R4 ;;COUNT WORDS.
3482 BNE 2$ ;;BR IF MORE.
3483 JSR R5, $PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
3484 .WORD $PRELOC ;;ADDRESS OF MESSAGE TO BE TYPED
3485 ;;"PROGRAM RELOCATED TO "
3486 MOV R3, -(SP) ;;PUT THE DATA ON THE STACK.
3487 JSR PC, $STYPAD ;;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
3488 MOV (SP)+, R4 ;;POP STACK INTO R4
3489 MOV (SP)+, R3 ;;POP STACK INTO R3
3490 MOV (SP)+, R2 ;;POP STACK INTO R2
3491 RTS R5 ;;RETURN
3492 *****
3493 * SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
3494 *****
3495 162206 RELOP: CMP #3, $PRGMAP ;;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
3496 BEQ 1$ ;;BR IF OK
3497 HALT ;;FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
3498 1$: MOV R0,-(SP) ;;PUSH R0 ON STACK
3499 MOV R1,-(SP) ;;PUSH R1 ON STACK
3500 TST MMVA
3501 BEQ 10$
3502 MOV #760C, @#KIPAR3 ;;SET PAR TO TOP OF MEM
3503 CLR R0 ;;INIT BANK POINTER...LO 64K
3504 MOV #BIT15, R1 ;;...HI 64K.
3505 2$: SUB #200, @#KIPAR3 ;;BACK DOWN ONE BANK.
3506 ROR R1 ;;MOVE POINTER...HI 64K.
3507 ROR R0 ;;...LO 64K.
3508 BCS 90$
3509 BIT R1, MEMMAP+2 ;;CHECK FOR BANK EXISTS.
3510

```

```

3511 016446 001003      BNE      3$      ;BR IF AVAILABLE
3512 016450 030067 163050  BIT      RO,    MEMMAP ;CHECK FOR BANK EXISTS.
3513 016454 001764      BEQ      2$      ;BR IF NO BANK FOUND.
3514 016456 013737 172346 172344 3$:  MOV      2*#KIPAR3,2*#KIPAR2 ;COPY PAR
3515 016464 010046      MOV      RO,-(SP) ;PUSH RO ON STACK
3516 016466 010146      MOV      R1,-(SP) ;PUSH R1 ON STACK
3517 016470 162737 000200 172344 4$:  SUB      #200, 2*#KIPAR2 ;BACK DOWN WITH LOW PAR.
3518 016476 006001      ROR      R1      ;SHIFT POINTER.
3519 016500 006000      ROR      RO      ;...LO 64K.
3520 016502 103457      BCS      90$     ;BR IF OVERFLOW.
3521 016504 030167 163016      BIT      R1,    MEMMAP+2 ;CHECK IF BANK EXISTS...HI 64K.
3522 016510 001003      BNE      6$      ;BR IF BANK EXISTS.
3523 016512 030067 163006      BIT      RO,    MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
3524 016516 001764      BEQ      4$      ;BR IF BANK DOESN'T EXIST.
3525 016520 052601      BIS      (SP)+, R1    ;GET SECOND BANK POINTER.
3526 016522 052600      BIS      (SP)+, RO    ;...LO 64K.
3527 016524 030067 162052      BIT      RO,    PRGMAP ;CHECK FOR CONFLICT.
3528 016530 001044      BNE      90$     ;ABORT IF DESTINATION OVERLAYS SOURCE.
3529 016532 004567 177506      JSR      RS,    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      2*#KIPAR2,2*#KIPAR0 ;RELOCATE LO BANK
3533 016550 013737 172346 172342  MOV      2*#KIPAR3,2*#KIPAR1 ;RELOCATE HI BANK.
3534      ;* PROGRAM SHOULD NOW BE EXECUTING 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$:  MOV      #BIT8, RO    ;SET BANK POINTER TO TOP OF MEM.
3539 016570 005001      CLR      R1      ;SET ADDRESS POINTER TO TOP.
3540 016572 162701 020000 11$:  SUB      #20000, R1  ;BACK DOWN ONE BANK.
3541 016576 006200      ASR      RO      ;MOVE POINTER DOWN ONE BANK.
3542 016600 103420      BCS      90$     ;BR IF OVERFLOW.
3543 016602 030067 162716      BIT      RO,    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      RO      ;MOV POINTER DOWN ONE BANK.
3547 016616 103411      BCS      90$     ;BR IF OVERFLOW.
3548 016620 030067 162700      BIT      RO,    MEMMAP ;CHECK IF THIS BANK EXISTS.
3549 016624 001762      BEQ      11$     ;BR TO START OVER IF NO LOWER BANK.
3550 016626 010046      MOV      RO,    -(SP) ;SAVE THE POINTER.
3551 016630 006300      ASL      RO      ;RESET POINTER TO HI BANK.
3552 016632 052600      BIS      (SP)+, RO    ;SET BIT FOR LO BANK.
3553 016634 030067 161742      BIT      RO,    PRGMAP ;CHECK FOR A PROGRAM CONFLICT.
3554 016640 001401      BEQ      12$     ;BR IF NO CONFLICT.
3555
3556 016642 000000 90$:  HALT      ;FATAL ERROR!!! NOT ENOUGH MEMORY??
3557 016644 010167 000006 12$:  MOV      R1,    13$  ;SET DATA FOR RELOCATION SUBROUTINE.
3558 016650 004567 177370      JSR      RS,    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,    RELOC ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3562 016664 060107      ADD      R1,    PC  ;JUMP TO RELOCATED PROGRAM
3563      ;* PROGRAM NOW EXECUTING 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,    RELOC ;SET THE RELOCATION FACTOR.
3566 016674 060137 000004      ADD      R1,    2*#ERRVEC ;ADJUST ERROR VECTOR.

```

RELOCATION SUBROUTINES.

```

3567 016700 060137 000024      ADD    R1,    @#PWVVEC ;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              .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),  @#PWVVEC ;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          SUB    (SP)+,    ;ADJUST RETURN ADDRESS.
3627 017156 005067 161416   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
3634
3635
3636
3637
3638
3639
3640
3641
3642
3643
3644
3645
3646
3647
3648
3649
3650
3651
3652
3653
3654

```

```

*****
* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
* WHENCE IT CAME. THE LOADER AREA IS SAVED AT THE END OF THE BK OF
* PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
*****
RESLDR: MOV    LMAD,   RO      ;CHECK IF THE LOADERS WERE SAVED.
          BNE    1$        ;BR IF LOADER AREA WAS SAVED.
          HALT           ;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
1$:      TST    MMVA      ;CHECK FOR MEM MGMT.
          BEQ    2$        ;SKIP IF NO MEM MGMT.
          CLR    @#SRO     ;DISABLE MEM MGMT.
2$:      MOV    #40000, R1   ;GET END OF BK, ASSUME PROG NOT RELOCATED.
          MOV    #1500., R2 ;GET COUNTER.
3$:      MOV    -(R1), -(R0) ;MOVE THE LOADER AREA.
          DEC    R2        ;COUNT HOW LONG THE AREA IS.
          BNE    3$        ;BR IF NOT MORE TO MOVE.
          CLR    LMAD      ;CLEAR MONITOR SAVED FLAG
          TST    MMVA      ;CHECK FOR MEM MGMT.
          BEQ    4$        ;BR IF NO MEM MGMT.
          INC    @#SRO     ;ENABLE MEM MGMT.
4$:      RTS     PC        ;RETURN.

```

```

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, RO     ;GET END OF BK
3658 017270 010001          MOV    RO,   R1      ;GET END OF BK
3659 017272 012737 017304 000004  MOV    #2$, @#ERRVEC ;SET UP TIMEOUT VECTOR
3660 017300 011020          MOV    (RO), (RO)+  ;SEARCH FOR END OF MEMORY
3661 017302 000776          BR     1$        ;KEEP SEARCHING
3662 017304 022626          CMP    (SP)+, (SP)+ ;RESTORE STACK POINTER
3663 017306 012737 025022 000004  MOV    #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR.
3664 017314 010046          MOV    RO,   -(SP)  ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3665 017316 012702 002734   MOV    #1500., R2    ;SET UP WORD COUNTER
3666 017322 014041          MOV    -(RO), -(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          RTS     PC        ;RETURN

```

```

3671
3672
3673
3674
3675
3676
3677 017336 011667 161560
3678 017342 004567 004052
3679 017346 026405
3680
3681 017350 010146
3682 017352 010346
3683 017354 016703 162242
3684 017360 005733
3685 017362 100415
3686 017364 005713
3687 017366 001374
3688 017370 004767 002152
3689
3690 017374 000024
3691 017376 000417
3692 017400 005713
3693 017402 001415
3694 017404 005733
3695 017406 100374
3696 017410 004567 004004
3697 017414 026476
3698
3699 017416
3700 017416 004767 000610
3701 017422 004767 002.20
3702 017426 000025
3703 017430 004767 000216
3704 017434 000761
3705 017436
3706 017436 012603
3707 017440 012601
3708 017442 000002
3709
3710
3711
3712
3713
3714 017444 005767 162620
3715 017450 071434
3716 017452 032777 000100 161460
3717 017460 001030
3718 017462 005767 161112
3719 017466 001410
3720 017470 032777 000040 161442
3721 017476 001004
3722 017500 026727 162056 001000
3723 017506 103415
3724 017510 016737 162114 000114
3725 017516 005037 000116
3726 017522 010346

```

```

.SBTTL PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
*****
* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
* FIND OUT WHICH REGISTER DETECTED THE ERROR.
* THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
*****
PESRV: MOV (SP), SBOADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
; "UNEXPECTED MEMORY PARITY TRAP."
MOV R1, -(SP) ;PUSH R1 ON STACK
MOV R3, -(SP) ;PUSH R3 ON STACK
.MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
1$: TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
TST (R3) ;CHECK FOR TABLE TERMINATOR
BNE 1$ ;BR IF MORE REGISTERS.
JSR PC, SERROR ;** ERROR ** (GO TYPE A MESSAGE)
;***ERROR*** NO REGISTER INDICATED ERROR
.WORD 24 ;ERROR TYPE CODE.
BR 4$ ;EXIT
2$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
BPL 2$ ;BR IF NO ERROR FLAG.
JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD MTOE ;ADDRESS OF MESSAGE TO BE TYPED
; "MORE THAN ONE ERROR FOUND."
3$: JSR PC, SPRINTQ ;SET UP VALUES FOR ERROR PRINTING.
64$: JSR PC, SERROR ;** ERROR ** (GO TYPE A MESSAGE)
.WORD 25 ;ERROR TYPE CODE.
JSR PC, PSCAN ;GO SCAN MEMORY FOR BAD PARITY.
BR 2$ ;GO LOOK FOR MORE ERRORS.
4$: MOV (SP)+, R3 ;POP STACK INTO R3
MOV (SP)+, R1 ;POP STACK INTO R1
RTI ;RETURN.
*****
ROUTINE TO ENABLE PARIY ERROR ACTION ON MA/MF PARITY MEMORIES
THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
*****
MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
BIT #SW6, 2SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
BEQ SETAE ;BR IF PROG IN BANK 0.
BIT #SW5, 2SWR ;CHECK IF VECTORS PROTECTED.
BNE SETAE ;BR IF VECTOR AREA PROTECTED.
CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
BLO MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.
SETAE: MOV .PESRV, 2#PARVEC ;SET PARITY ERROR TRAP VECTOR
CLR 2#PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
MOV R3, -(SP) ;PUSH R3 ON STACK

```

```

3727 017524 016703 162072
3728 017530 052733 000001
3729 017534 005713
3730 017536 001374
3731 017540 012603
3732 017542 000207
3733
3734
3735
3736
3737 017544 005767 162520
3738 017550 001437
3739 017552 032777 000100 161360
3740 017560 001033
3741 017562 010346
3742 017564 016703 162032
3743 017566 005733
3744 017568 100023
3745 017570 032773 000001 177776
3746 017602 001010
3747 017604 004767 000422
3748 017610 004767 001732
3749 017614 000026
3750 017616 000411
3751 017620 004767 000026
3752 017624
3753 017624 004767 000402
3754 017630 004767 001712
3755 017634 000027
3756 017636 004767 000010
3757 017642 005713
3758 017644 001351
3759 017646 012603
3760 017650 000207
3761
3762
3763
3764
3765
3766
3767 017652
3768 017652 010046
3769 017654 010146
3770 017656 010246
3771 017660 010346
3772 017662 010446
3773 017664 013746 000114
3774 017670 013746 000116
3775 017674 004567 003520
3776 017700 026542
3777
3778 017702 012700 000001
3779 017706 005001
3780 017710 005002
3781 017712 005004
3782 017714 004767 000256

```

```

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

*****
* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
*****
CKPMER: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
        BEQ 4$ ;BR IF NO PARITY REGISTERS.
        BIT #SW6, 2SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
        BNE 4$ ;BR IF PARITY ERROR CHECKING INHIBITED.
        MOV R3, -(SP) ;PUSH R3 ON STACK
        MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
1$: TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
        BPL 3$ ;BR IF NO ERROR
        BIT #BIT0, 2-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
        BNE 2$ ;BR IF NO ACTION ENABLE.
64$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 26 ;ERROR TYPE CODE.
        BR 3$
        JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
2$:
65$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 27 ;ERROR TYPE CODE.
        JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
        BNE 1$ ;BR IF MORE.
4$: MOV (SP)+,R3 ;POP STACK INTO R3
        RTS PC ;RETURN.

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

```



3783	017720	012737	000116	000114	MOV	#116,	2#114.	;HALT, IF ANOTHER PARITY TRAP.
3784	017726	005037	000116		CLR	2#116		
3785	017732	005767	160650		TST	MMAVA		;CHECK FOR MEMORY MANAGEMENT.
3786	017736	001406			B	1\$		;BR IF NO MEM MGMT.
3787	017740	013746	172344		2#KIPAR2,-(SP)			;PUSH 2#KIPAR2 ON STACK
3788	017744	005037	172344		2#KIPAR2			;INIT MEM MGMT TO POINT TO BANK 0.
3789	017750	012702	040000		#40000, R2			;SET ADR POINTER TO PAR2.
3790	017754	030067	161544	1\$:	R0,	MEMMAP		;CHECK IF THIS BANK OF MEM EXISTS.
3791	017760	001003			BNE	2\$		;BR IF THIS BANK EXISTS.
3792	017762	030167	161544		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	2(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	SERIAL		;ADJUST ERROR COUNT.
3803	020014	005204			INC	R4		;SET FLAG TO INDICATE ERROR FOUND.
3804	020016			5\$:				
3805	020016	004767	000210	6\$:	JSR	PC,	TO	;SET UP VALUES FOR ERROR PRINTING.
3806	020022	004767	001520		JSR	PC,	SENROR	*** 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	2-(R3)		;CLEAR THE ERROR FLAG.
3810	020034	105712			TSTB	(R2)		;CHECK IF THE PARITY ERROR WAS CLEARED.
3811	020036	005733			TST	2(R3)+		;CHECK FOR THE ERROR FLAG.
3812	020040	100005			BPL	6\$		;BR IF IT IS OK.
3813	020042	004567	003352		JSR	R5,	SPRINT	;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	2-2(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,	2#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)+,2#KIPAR2		;POP STACK INTO 2#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,	SPRINT	;GO PRINT OUT THE FOLLOWING MESSAGE.

# F13

```

3839 020150 025614
3840 020152
3841 020152 012637 000116
3842 020156 012637 000114
3843 020162 012604
3844 020164 012603
3845 020166 012602
3846 020170 012601
3847 020172 012600
3848 020174 000207
3849
3850
3851
3852
3853 020176
3854 020176 010346 161416
3855 020200 016703
3856 020204 005713
3857 020206 001402
3858 020210 005033
3859 020212 000774
3860 020214
3861 020214 012603
3862 020216 000207
3863
3864
3865
3866
3867
3868
3869 020220 010267 160674
3870 020224 005067 160674
3871 020230 000430
3872
3873 020232 014367 160722
3874 020236 013367 160720
3875 020242 000402
3876
3877 020244 011367 160710
3878 020250 010267 160644
3879 020254 000414
3880
3881 020256 010267 160636
3882 020262 005367 160632
3883 020266 000407
3884
3885 020270 010367 160664
3886 020274 010267 160620
3887 020300 162767 000002 160612
3888 020306 010067 160612
3889 020312 010167 160610
3890 020316 000207
3891
3892
3893
3894

          .WORD  NOPE$          ;ADDRESS OF MESSAGE TO BE TYPED
21$:      MOV      (SP)+, @#116      ;; POP STACK INTO @#116
          MOV      (SP)+, @#114      ;; POP STACK INTO @#114
          MOV      (SP)+, R4         ;; POP STACK INTO R4
          MOV      (SP)+, R3         ;; POP STACK INTO R3
          MOV      (SP)+, R2         ;; POP STACK INTO R2
          MOV      (SP)+, R1         ;; POP STACK INTO R1
          MOV      (SP)+, R0         ;; POP STACK INTO R0
          RTS      PC                ;RETURN.

;*****
;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
;*****
CLRPAR:
          MOV      R3, -(SP)         ;; PUSH R3 ON STACK
          MOV      .MPRX, R3        ;; GET PARITY REGISTER TABLE POINTER.
1$:      TST      (R3)              ;; CHECK FOR THE TABLE TERMINATOR.
          BEQ      2$                ;; BR IF DONE ALL PARITY REGISTERS.
          CLR      @ (R3)+          ;; CLEAR THE PARITY REGISTER.
          BR       1$                ;; BR FOR MORE
2$:      MOV      (SP)+, R3         ;; PC STACK INTO R3
          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 (.SCMTAG)
; * FOR ERROR PRINTOUT BY .SERROR & .SERRTYP ROUTINES FROM **SYSMAC**.
;*****
SPRNT:   MOV      R2, $GDAOR        ;SAVE THE ADDRESS UNDER TEST.
          CLR      $G0DAT          ;SHOULD BE DATA IS "0".
          BR       SPRNTB

SPRNTQ:  MOV      -(R3), $TMP0      ;GET THE PARITY REGISTER ADDRESS.
          MOV      @ (R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
          BR       SPRNTD

SPRNTP:  MOV      (R3), $TMP0      ;GET THE PARITY REGISTER ADDRESS.
SPRNTD:  MOV      R2, $GDAOR        ;GET THE MEMORY ADDRESS BEING TESTED
          BR       SPRNTA          ;BR TO COMMON SECTION.

SPRNT1:  MOV      R2, $GDAOR        ;GET THE MEMORY ADDRESS BEING TESTED
          DEC      $GDAOR          ;ADJUST IT FOR PRINTOUT.
          BR       SPRNTA          ;BR TO COMMON SECTION.

SPRNT3:  MOV      R3, $TMP0        ;GET THE DATA IN R3.
SPRNT2:  MOV      R2, $GDAOR        ;GET THE MEMORY ADDRESS BEING TESTED
          SUB      #2, $GDAOR      ;ADJUST IT FOR PRINTOUT.
SPRNTA:  MOV      R0, $G0DAT        ;GET WHAT THE DATA SHOULD BE
SPRNTB:  MOV      R1, $B0DAT        ;GET WHAT THE DATA WAS
          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
3898 020324 005760 000002
3899 020330 001004
3900 020332 004567 003062
3901 020336 026172
3902
3903 020340 006475
3904 020342
3905 020342 010146
3906 020344 010246
3907 020346 010346
3908 020350 010446
3909 020352 012701 000001
3910 020356 005002
3911 020360 012703 177777
3912 020364 010304
3913 020366 030110
3914 020370 001014
3915 020372 030260 000002
3916 020376 001011
3917 020400 105703
3918 020402 001042
3919 020404 162703 000001
3920 020410 005604
3921 020412 004567 003002
3922 020416 025425
3923 020420 000410
3924 020422 105703
3925 020424 001431
3926 020426 062703 000001
3927 020432 005504
3928 020434 004567 002760
3929 020440 025415
3930 020442
3931 020442 010346
3932 020444 010446
3933 020446 006303
3934 020450 006104
3935 020452 006003
3936 020454 010446
3937
3938
3939
3940 020456 013746 177776
3941 020462 004767 004104
3942 020466 003
3943 020467 000
3944 020470 010346
3945
3946
3947
3948 020472 013746 177776
3949 020476 004767 004070
3950 020502 005

```

```

*****
TYPMAP: TST (R0) ;CHECK IF ANY MEMORY IN MAP...LO 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        TST 2(R0) ;...HI 64K.
        BNE 1$ ;BR IF MEMORY IN MAP.
        JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD NOMEM ;ADDRESS OF MESSAGE TO BE TYPED
        ;"NO MEMORY FOUND."
        BR 6$ ;EXIT

1$: MOV R1,-(SP) ;PUSH R1 ON STACK
    MOV R2,-(SP) ;PUSH R2 ON STACK
    MOV R3,-(SP) ;PUSH R3 ON STACK
    MOV R4,-(SP) ;PUSH R4 ON STACK
    MOV #BIT0, R1 ;SET UP BANK POINTER...LO 64K.
    CLR R2 ;...HI 64K.
    MOV #-1, R3 ;SET UP ADDRESS POINTER TO -1.
    MOV R3, R4 ;HI BITS OF ADDRESS AS WILL.
    BIT R1, (R0) ;CHECK THE MAP FOR THIS BANK.
    BNE 3$ ;BR IF THIS BANK PRESENT.
    BIT R2, 2(R0) ;CHECK HI 64K MAP.
    BNE 3$ ;BR IF THIS BANK PRESENT.
    TSTB R3 ;CHECK FOR PREVIOUS PRINTOUT.
    BNE 5$ ;BR IF ALREADY TYPED "TO"
    SUB #1, R3 ;BACK UP TO LAST ADR OF PREVIOUS BANK.
    SBC R4 ;...HI ADDRESS BITS.
    JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
    BR 4$ ;GO TO TYPE THE ADDRESS.
    TSTB R3 ;CHECK FOR PREVIOUS TYPEOUT.
    BEQ 5$ ;BR IF ALREADY TYPE "FROM".
    ADD #1, R3 ;POINT TO FIRST ADDRESS OF THIS BANK.
    ADC R4 ;...HI BITS OF ADDRESS.
    JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED

4$: MOV R3,-(SP) ;PUSH R3 ON STACK
    MOV R4,-(SP) ;PUSH R4 ON STACK
    ASL R3 ;BIT 15 INTO C-BIT
    ROL R4 ;BIT 15 INTO R4.
    ROR R3 ;RESTORE BITS 14-0.
    MOV R4,-(SP) ;SAVE R4 FOR TYPEOUT
    ;TYPE ADDRESS BITS 21-15
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
    MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
    JSR PC, $TYPOS ;GO TO THE SUBROUTINE
    .BYTE 3 ;TYPE 3 DIGIT(S)
    .BYTE 0 ;SUPPRESS LEADING ZEROS
    MOV R3,-(SP) ;SAVE R3 FOR TYPEOUT
    ;TYPE ADDRESS BITS 14-0
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
    MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
    JSR PC, $TYPOS ;GO TO THE SUBROUTINE
    .BYTE 5 ;TYPE 5 DIGIT(S)

```

# H13

```

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 ;:$
3978 ;:SCOPE ;:SCOPE=IOT
3979 020536 $SCOPE:
3980 ;:$ THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
3981 ;:$ WIHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
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$, @#ERFVEC ;: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 #$$MASK, (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	SERMAX, SERFLG		:: MAX. ERRORS FOR THIS TEST OCCURRED?
4009	020662	101015			BHI	3\$		:: BR IF NO
4010	020664	032777	001000	160246	BIT	#BIT09, 2SWR		:: LOOP ON ERROR?
4011	020672	001404			BEQ	4\$		:: BR IF NO
4012	020674	016767	160210	160204	7\$: MOV	SLPERR, SLPADR		:: SET LOOP ADDRESS TO LAST SCOPE
4013	020702	000446			BR	SOVER		
4014	020704	105067	160173		4\$: CLR	SERFLG		:: ZERO THE ERROR FLAG
4015	020710	005067	160254		CLR	STIMES		:: 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, 2SWR		:: 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	STIMES, \$ICNT		:: CHECK THE NUMBER OF ITERATIONS MADE
4023	020746	002024			BGE	SOVER		:: 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, STIMES		:: SET NUMBER OF ITERATIONS TO DO
4026	020764	105267	160112		5\$VLAD: INCB	\$STNM		:: COUNT TEST NUMBERS
4027	020770	116767	160106	160212	MOV	\$STNM, \$STNM		:: SET TEST NUMBER IN APT MAILBOX
4028	020776	011667	160104		MOV	(SP), SLPADR		:: SAVE SCOPE LOOP ADDRESS
4029	021002	011667	160102		MOV	(SP), SLPERR		:: SAVE ERROR LOOP ADDRESS
4030	021006	005067	160160		CLR	\$ESCAPE		:: CLEAR THE ESCAPE FROM ERROR ADDRESS
4031	021012	112767	000001	160075	MOV	#1, SERMAX		:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4032	021020	016777	160056	160114	SOVER: MOV	\$STNM, 2DISPLAY		:: DISPLAY TEST NUMBER
4033	021026	016716	160054		MOV	SLPADR, (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 BK BOUNDARY 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	\$STNM, #21		:: CHECK IF IN SECTION 3.
4041	021064	103002			BHIS	3\$		:: BR IF IN SECTION 3.
4042	021066	006267	160514		2\$: ASR	BLKMSK		:: RESET BOUNDARY 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, 2SWR		:: 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, TMAPLAD		:: 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.

SCOPE HANDLER ROUTINE

4063	021216	046767	157362	160306	BIC	PRGMAP+2, TSTMAP+2	
4064	021224	005767	157762		TST	SPASS	: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 BOUNDARY.
4073	021264	001416			BEQ	11\$	:BR IF FIRST ADR ON BLOCK BOUNDARY.
4074	021266	050467	160272		BIS	R4, TMPFAD	:ADJUST FIRST ADR TO END OF BLOCK.
4075	021272	005267	160266		INC	TMPLAD	:FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4076	021276	032767	017777	160260	BIT	#MASK4K, TMPFAD	:CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4077	021304	001006			BNE	11\$	:BR IF NOT ON 4K BOUNDARY.
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 BOUNDARY.
4081	021326	001414			BEQ	12\$	:BR IF ON BLOCK BOUNDARY.
4082	021330	040467	160242		BIC	R4, TMPLAD	:ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4083	021334	032767	017777	160234	BIT	#MASK4K, TMPLAD	:CHECK IF ADJUSTED TO 4K BOUNDARY.
4084	021342	001006			BNE	12\$	:BR IF NOT ON 4K BOUNDARY.
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, SPRINT	: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		MOV	\$STNM, (SP)	:PUT THE DATA ON THE STACK.
4103					;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$TYPOS ROUTINE		
4104					;* WITHOUT 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	TMPLAD, 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.

K13

MAINDEC-11-DZQMC-D-0: 0-124K MEMORY EXERCISER, 16K VER  
 DZQMC0.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 83

SEQ 0166

SCOPE HANDLER ROUTINE

```

4119 021522 040567 160052 22$: BIC RS, LADMSK ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4120 021526 006305 ;ASL RS ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4121 021530 001374 ;BNE 22$ ;LOOP UNTIL OVERFLOW.
4122 021532 000207 ;ENDINS: RTS PC ;EXIT SCOPE ROUTINE BACK TO TEST.
4123 021534 000004 ;SMXCNT: 4 ;MAX. NUMBER OF ITERATIONS
4124 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SCKSWR ROUTINE
4125 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4126 021536 013746 177776 ;MOV 2#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4127 021542 004767 000524 ;JSR PC, SCKSWR ;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 SERRTYP 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 ;SERROR:
4143 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE SCKSWR ROUTINE
4144 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4145 021546 013746 177776 ;MOV 2#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4146 021552 004767 000514 ;JSR PC, SCKSWR ;GO TO THE SUBROUTINE
4147 021556 062716 000002 ;ADD #2, (SP) ;ADJUST POINTER PAST CODE WORD.
4148 021562 105267 157315 7$: INCB SERRFLG ;SET THE ERROR FLAG
4149 021566 001775 ;BEQ 7$ ;DON'T LET THE FLAG GO TO ZERO
4150 021570 016777 157306 157344 ;MOV $STNM, 2DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
4151 021576 032777 002000 157334 ;BIT #BIT10, 2SWR ;BELL ON ERROR?
4152 021604 001403 ;BEQ 1$ ;NO - SKIP
4153 021606 004567 001606 ;JSR RS, $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, $ERRPC
4158 021632 117767 157260 157254 ;MOV 2$ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
4159 021640 032777 020000 157272 ;BIT #BIT13, 2SWR ;SKIP TYPEOUT IF SET
4160 021646 001005 ;BNE 20$ ;SKIP TYPEOUTS
4161 021650 004767 000116 ;JSR PC, $SERRTYP ;GO TO USER ERROR ROUTINE
4162 021654 004567 001540 ;JSR RS, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4163 021660 001201 ;.WORD $CALF ;ADDRESS OF MESSAGE TO BE TYPED
4164 021662
4165 021662 122767 000001 157334 20$: CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
4166 021670 001007 ;BNE 2$ ;NO SKIP APT ERROR REPORT
4167 021672 116767 157216 000004 ;MOV 2$ITEMB, 21$ ;SET ITEM NUMBER AS ERROR NUMBER
4168 021700 004767 002044 ;JSR PC, $ATY4 ;; REPORT FATAL ERROR TO APT
4169 021704 000 ;.BYTE 0
4170 021705 000 ;.BYTE 0
4171 021706 000777 22$: BR 22$ ;; APT ERROR LOOP
4172 021710 005777 157224 2$: TST 2SWR ;; HALT ON ERROR
4173 021714 100005 ;BPL 3$ ;SKIP IF CONTINUE
4174 021716 000000 ;HALT ;; HALT ON ERROR!
    
```

```

4175 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4176 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4177 021720 013746 177776 MOV @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4178 021724 004767 000342 JSR PC, $CKSWR ;GO TO THE SUBROUTINE
4179 021730 032777 001000 157202 3$: BIT @BIT09, @SWR ;LOOP ON ERROR SWITCH SET?
4180 021736 001402 4$ BEQ 4$ ;BR IF NO
4181 021740 016716 157144 MOV $LPERR, (SP) ;FUDGE RETURN FOR LOOPING
4182 021744 005767 157222 4$: TST $ESCAPE ;CHECK FOR AN ESCAPE ADDRESS
4183 021750 001402 5$ BEQ 5$ ;BR IF NONE
4184 021752 016716 157214 MOV $ESCAPE, (SP) ;FUDGE RETURN ADDRESS FOR ESCAPE
4185 021756 5$:
4186 021756 022737 014136 000042 CMP @SENDAD, @#42 ;:ACT-11 AUTO-ACCEPT?
4187 021764 001001 BNE 6$ ;:BRANCH IF NO
4188 021766 000000 HALT ;:YES
4189 021770 6$:
4190 021770 000207 RTS PC
4191 ;:*****
4192
4193 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
4194
4195 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4196 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4197 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4198
4199 $ERRTYP:
4200 021772 004567 001422 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4201 021776 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4202 022000 010046 MOV RO, -(SP) ;SAVE RO
4203 022002 005000 CLR RO ;PICKUP THE ITEM INDEX
4204 022004 156700 157104 BISB $ITEMB, RO
4205 022010 001007 BNE 1$ ;IF ITEM NUMBER IS ZERO, JUST
4206 ;TYPE THE PC OF THE ERROR
4207 022012 016746 157100 MOV $ERRPC, -(SP) ;:SAVE $ERRPC FOR TYPEOUT
4208 ;:ERROR ADDRESS
4209
4210 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4211 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4212 022022 004767 002570 MOV @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4213 022026 000513 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4214 022030 016767 157062 157456 1$: BR 10$ ;GET OUT
4215 022036 166767 156536 157450 MOV $ERRPC, $VERPC ;SET UP VIRTUAL PC FOR TYPEOUT.
4216 022044 005300 SUB RELOC, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
4217 022046 006300 DEC RO ;ADJUST THE INDEX SO THAT IT WILL
4218 022050 006300 ASL RO ; WORK FOR THE ERROR TABLE
4219 022052 006300 ASL RO
4220 022054 066700 157552 ADD .ERRTB, RO ;FORM TABLE POINTER
4221 022060 012067 000006 MOV (RO)+, 2$ ;PICKUP "ERROR MESSAGE" POINTER
4222 022064 001406 BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
4223 022066 004567 001326 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4224 022072 000000 .WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
4225 022074 004567 001320 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4226 022100 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4227 022102 012067 000006 MOV (RO)+, 4$ ;PICKUP "DATA HEADER" POINTER
4228 022106 001406 BEQ 5$ ;SKIP TYPEOUT IF 0
4229 022110 004567 001304 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4230 022114 000000 .WORD 0 ;"DATA HEADER" POINTER GOES HERE

```



M13

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMCD.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 85

SEQ 0168

ERROR MESSAGE TYPEOUT ROUTINE

```

4231 022116 004567 001276      JSR      R5,      SPRINT ;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      RELOC, R1 ;ADJUST POINTER
4237 022136 012000      MOV      (R0)+,R0 ;PICKUP "DATA FORMAT" POINTER
4238 022140 066700 156434      ADD      RELOC, 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 STYPOC ROUTINE
4243 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4244 022152 013746 177776      MOV     @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4245 022156 004767 002434      JSR     PC, STYPOC ;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 STYPOS ROUTINE
4250 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4251 022170 013746 177776      MOV     @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4252 022174 004767 002140      JSR     PC, STYPOS ;GO TO THE SUBROUTINE
4253 022200 000417      BR     8$ ;SKIP
4254 022202 122760 177777 177777 17$:  CMPB   B-1, -1(R0) ;CHECK FOR 18-BIT ADDRESS FORMAT.
4255 022210 001004      BNE     18$ ;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, STYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4258 022220 000407      BR     8$ ;SKIP
4259 022222      18$:  CLR     -(SP) ;CLEAR THE STACK.
4260 022222 005046      MOV     @ (R1)+, (SP) ;DATA ON THE STACK.
4261 022224 113116 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOS ROUTINE
4262 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4263 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOS ROUTINE
4264 022226 013746 177776      MOV     @PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4265 022232 004767 002334      JSR     PC, STYPOS ;GO TO THE SUBROUTINE
4266 022236 003 ;TYPE 3
4267 022237 001 ;TYPE LENGTH IN BYTES.
4268 022240 005711      8$:  TST     (R1) ;IS THERE ANOTHER NUMBER?
4269 022242 001404      BEQ     9$ ;BR IF NO
4270 022244 004567 001150      JSR     R5, SPRINT ;GO PRINT OUT THE 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, SPRINT ;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
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 #STKS ;; CHAR THERE?
4294 022306 100101 BPL 15$ ;; IF NO, DON'T WAIT AROUND
4295 022310 117746 156632 MOVB #STKB, -(SP) ;; SAVE THE CHAR
4296 022314 042716 177600 BIC #1C177, (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 SAUTOB, #1 ;; ARE WE RUNNING IN AUTO-MODE?
4300 022334 001466 BEQ 15$ ;; BRANCH IF YES
4301
4302 022336 004567 001056 JSR RS, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4303 022342 023217 .WORD SCNTLG ;ADDRESS OF MESSAGE TO BE TYPED
4304 022344
4305 022344 004567 001050 SGTSWR: JSR RS, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4306 022350 023224 .WORD SMSWR ;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 ;* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSM**.
4310 022356 013746 177776 MOV #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4311 022362 004767 002230 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4312 022366 004567 001026 JSR RS, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4313 022372 023235 .WORD SMNEW ;ADDRESS OF MESSAGE TO BE TYPED
4314 022374 005046 19$: CLR -(SP) ;; CLEAR COUNTER
4315 022376 005046 CLR -(SP) ;; THE NEW SWR
4316 022400 105777 156540 7$: TSTB #STKS ;; CHAR THERE?
4317 022404 100375 BPL 7$ ;; IF NOT TRY AGAIN
4318
4319 022406 117746 156534 MOVB #STKB, -(SP) ;; PICK UP CHAR
4320 022412 042716 177600 BIC #1C177, (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 RS, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4327 022430 023212 .WORD SCNTLU ;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 RS, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4340 022472 001201 .WORD SCRLF ;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      #100,2$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
4344 022512 000002      RTI                    ;;RETURN
4345 022514 004767 001142      15$:   JSR      PC,$TYPEC  ;;ECHO CHAR
4346 022520 021627 000060      16$:   CMP      (SP),#60   ;;CHAR < 0?
4347 022524 002420      BLT                    ;;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      ;;      R0CHR      ;;INPUT A SINGLE CHARACTER FROM THE TTY
4370      ;;      RETURN HERE ;;CHARACTER IS ON THE STACK
4371      ;;
4372      ;;
4373      ;;
4374 022576 011646      $R0CHR: MOV      (SP),-(SP) ;;PUSH DOWN THE PC
4375 022600 016666 000004 000002      MOV      4(SP),2(SP) ;;SAVE THE PS
4376 022606 105777 156332      1$:   TSTB     2$TKS    ;;WAIT FOR
4377 022612 100375      BPL      1$         ;;A CHARACTER
4378 022614 117766 156326 000004      MOVB     2$TKB,4(SP) ;;READ THE TTY
4379 022622 042766 177600 000004      BIC      #1C<177>,4(SP) ;;GET RID OF JUNK IF ANY
4380 022630 026627 000004 000023      CMP      4(SP),#23  ;;IS IT A CONTROL-S?
4381 022636 001013      BNE      3$         ;;BRANCH IF NO
4382 022640 105777 156300      2$:   TSTB     2$TKS    ;;WAIT FOR A CHARACTER
4383 022644 100375      BPL      2$         ;;LOOP UNTIL ITS THERE
4384 022646 117746 156274      MOVB     2$TKB,-(SP) ;;GET CHARACTER
4385 022652 042716 177600      BIC      #1C177,(SP) ;;MAKE IT 7-BIT ASCII
4386 022656 022627 000021      CMP      (SP)+,#21  ;;IS IT A CONTROL-Q?
4387 022662 001366      BNE      2$         ;;IF NOT DISCARD IT
4388 022664 000750      BR       1$         ;;YES, RESUME
4389 022666 026627 000004 000140      3$:   CMP      4(SP),#140 ;;IS IT UPPER CASE?
4390 022674 002407      BLT                    ;;BRANCH IF YES
4391 022676 026627 000004 000175      CMP      4(SP),#175 ;;IS IT A SPECIAL CHAR?
4392 022704 003003      BGT      4$         ;;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 #STTYIN, R3          ;; GET ADDRESS
4405 022726 022703 023212 2$: CMP #STTYIN+8., R3          ;; BUFFER FULL?
4406 022732 101467      BLOS 4$          ;; BR IF YES
4407
4408          ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4409          ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4409 022734 013746 177776      MOV 2$PSW, -(SP)          ;; PUT THE PROCESSOR STATUS ON THE STACK
4410 022740 004767 177632      JSR PC, $RDCHR          ;; GO TO THE SUBROUTINE
4411 022744 112613          MOV# (SP)+, (R3)          ;; GET CHARACTER
4412 022746 122713 000177 10$: 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      MOV# #' \, 9$          ;; TYPE A BACK SLASH
4417 022766 004567 000426          JSR R5, SPRINT          ;; 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$: DEC R3          ;; BACKUP BY ONE
4421 023002 020327 023202          CMP R3, #STTYIN          ;; STACK EMPTY?
4422 023006 103441          BLO 4$          ;; BR IF YES
4423 023010 111367 000164          MOV# (R3), 9$          ;; SETUP TO TYPEOUT THE DELETED CHAR.
4424 023014 004567 000400          JSR R5, SPRINT          ;; 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      MOV# #' \, 9$          ;; TYPE A BACK SLASH
4430 023036 004567 000356          JSR R5, SPRINT          ;; 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, SPRINT          ;; 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, SPRINT          ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4442 023100 001201          .WORD $CRLF          ;; ADDRESS OF MESSAGE TO BE TYPED
4443 023102 004567 000312          JSR R5, SPRINT          ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4444 023106 023202          .WORD STTYIN          ;; ADDRESS OF MESSAGE TO BE TYPED
4445 023110 000706          BR 2$          ;; GO PICKUP ANOTHER CHACTER
4446
4447 023112 004567 000302 4$: JSR R5, SPRINT          ;; 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$: MOV# (R3), 9$          ;; ECHO THE CHARACTER
4451 023126 004567 000266          JSR R5, SPRINT          ;; 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

```

```

4455 023142 105063 177777          CLRB   -1(R3)          ;; CLEAR RETURN (THE 15)
4456 023146 004567 000246          JSR    R5, SPRINT    ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4457 023152 001202                    .WORD  SLF           ;; ADDRESS OF MESSAGE TO BE TYPED
4458 023154 005726                    TST    (SP)+         ;; CLEAN RUBOUT KEY FROM THE STACK
4459 023156 012603                    MOV    (SP)+,R3     ;; RESTORE R3
4460 023160 011646                    MOV    (SP)-,(SP)   ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4461 023162 016666 000004 000002    MOV    4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
4462 023170 012766 023202 000004    MOV    $TTYIN,4(SP)
4463 023176 000002                    RTI                    ;; RETURN
4464 023200 000          9S:      .BYTE  0          ;; STORAGE FOR ASCII CHAR. TO TYPE
4465 023201 000          .BYTE  0          ;; TERMINATOR
4466 023202 000010    $TTYIN: .BLKB  8.    ;; RESERVE 8 BYTES FOR TTY INPUT
4467 023212 052536 005015 000    $CNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
4468 023217 136 006507 000012    $CNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
4469 023224 005015 053523 020122    $MSWR:  .ASCIZ <15><12>/SWR = /
4470 023232 020075 000
4471 023235 040 047040 053505    $MNEW:  .ASCIZ / NEW = /
4472 023242 036440 000040
4473
4474          .SBTTL READ AN OCTAL NUMBER FROM THE TTY
4475          ;; *****
4476          ;; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4477          ;; *CHANGE IT TO BINARY.
4478          ;; *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
4479          ;; *OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
4480          ;; *FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
4481          ;; *THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
4482          ;; *CALL:
4483          ;; *      RDOCT          ;; READ AN OCTAL NUMBER
4484          ;; *      RETURN HERE    ;; LOW ORDER BITS ARE ON TOP OF THE STACK
4485          ;; *                    ;; HIGH ORDER BITS ARE IN $HI OCT
4486
4487 023246 011646 000004 000002    $RDOCT: MOV    (SP)-,(SP)  ;; PROVIDE SPACE FOR THE
4488 023250 016666                    MOV    4(SP),2(SP)      ;; INPUT NUMBER
4489 023256 010046                    MOV    R0,-(SP)        ;; PUSH R0 ON STACK
4490 023260 010146                    MOV    R1,-(SP)        ;; PUSH R1 ON STACK
4491 023262 010246                    MOV    R2,-(SP)        ;; PUSH R2 ON STACK
4492 023264
4493          1S:
4494          ;; * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOLIN ROUTINE
4495          ;; * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4496 023264 013746 177776          MOV    2#PSW, -(SP)    ;; PUT THE PROCESSOR STATUS ON THE STACK
4497 023270 004767 177422          JSR    PC, $RDOLIN    ;; GO TO THE SUBROUTINE
4498 023274 012600                    MOV    (SP)+,R0       ;; GET ADDRESS OF 1ST CHARACTER
4499 023276 010067 000102          MOV    R0,5$          ;; AND SAVE IT
4500 023302 005001                    CLR    R1             ;; CLEAR DATA WORD
4501 023304 005002                    CLR    R2
4502 023306 112046          2S:      MOVB   (R0)+,-(SP)    ;; PICKUP THIS CHARACTER
4503 023310 001420                    BEQ    3$             ;; IF ZERO GET OUT
4504 023312 122716 000060          CMPB   #'0,(SP)      ;; MAKE SURE THIS CHARACTER
4505 023316 003026                    BGT    4$             ;; IS AN OCTAL DIGIT
4506 023320 122716 000067          CMPB   #'7,(SP)
4507 023324 002423                    BLT    4$
4508 023326 006301                    ASL    R1             ;; *2
4509 023330 006102                    ROL    R2
4510 023332 006301                    ASL    R1             ;; *4
4510 023334 006102                    ROL    R2

```

E14

```

4511 023336 006301          ASL    R1          ;;#8
4512 023340 006102          ROL    R2
4513 023342 042716 177770    BIC    #1C7,(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$:   TST    (SP)+  ;;CLEAN TERMINATOR FROM STACK
4517 023354 010166 000012    MOV    R1,12(SP)  ;;SAVE THE RESULT
4518 023360 010267 000032    MOV    R2,$HI OCT
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
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          $HI OCT, .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    RELOC#, 1$ ;;MAKE IT PHYSICAL.
4539          ;; THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4540          ;; WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4541 023432 013746 177776    MOV    2#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          .SBITL 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
4566 023456 000430          BR     3$         ;; LEAVE
    
```

4567	023460	010046		15:	MOV	RO, -(SP)	:: SAVE RO
4568	023462	017600	000002		MOV	22(SP), RO	:: 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	RO, 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	#APTC SUP, \$ENVM
4577	023526	001003			BNE	60\$	:: APT CONSOLE SUPPRESSED
4578	023530	112046		2\$:	MOVB	(RO)+, -(SP)	:: YES, SKIP TYPE OUT
4579	023532	001005			BNE	4\$	:: PUSH CHARACTER TO BE TYPED ONTO STACK
4580	023534	005726			TST	(SP)+	:: BR IF IT ISN'T THE TERMINATOR
4581	023536	012600		60\$:	MOV	(SP)+, RO	:: IF TERMINATOR POP IT OFF THE STACK
4582	023540	062716	000002	3\$:	ADD	#2, (SP)	:: RESTORE RO
4583	023544	000002			RTI		:: ADJUST RETURN PC
4584	023546	122716	000011	4\$:	CMPB	#HT, (SP)	:: RETURN
4585	023552	001431			BEQ	8\$	:: BRANCH IF <HT>
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	RS, \$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							
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	2\$TPS	:: WAIT UNTIL PRINTER IS READY
4613	023666	100375			BPL	\$TYPEC	
4614	023670	116677	000002	155254	MOVB	2(SP), 2\$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)
4620	023722	001402			BEQ	\$TYPEX	:: IS CHARACTER A LINE FEED?
4621	023724	105227			INCB	(PC)+	:: BRANCH IF YES
4622	023726	000000			\$CHARCNT:	.WORD	0

```

4623 023730 000207          STYPEX: RTS      PC
4624                                     .SBTTL  APT COMMUNICATIONS ROUTINE
4625                                     ..*****
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     RO, -(SP)      ;; PUSH RO 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   #APTPOOL, $ENVM  ;; SHOULD SPOOL MESSAGES?
4640 024006 001425          BEQ    3$            ;; IF NOT: BR
4641 024010 017600 000004          MOV     #4(SP), RO      ;; 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     RO, $MSGAD      ;; PUT ADDR IN MAILBOX
4646 024034 105720          2$:   TSTB   (RO)+        ;; FIND END OF MESSAGE
4647 024036 001376          BNE    2$            ;;
4648 024040 166700 155154          SUB     $MSGAD, RO      ;; SUB START OF MESSAGE
4649 024044 006200          ASR    RO              ;; GET MESSAGE LNTH IN WORDS
4650 024046 010067 155150          MOV     RO, $MSGGLT     ;; 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), RO      ;; GET ERROR #
4662 024122 062766 000002 000004  ADD     #2, 4(SP)        ;; BUMP RETURN ADDR.
4663 024130 012701 001344          MOV     #SASTAT, 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     RO, (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     RO, (R1)+       ;; SET UP NEW ENTRY
4673 024160 005211          9$:   INC     (R1)          ;; BUMP ERROR COUNT
4674 024162 105767 000150          10$:  TSTB   $FFLG          ;; SHOULD REPORT FATAL ERPA?
4675 024166 001416          BEQ    12$           ;; 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     24(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          CLR    $FFLG         ;; CLEAR FATAL FLAG
4684 024230 105067 000101          CLR    $LFLG        ;; CLEAR LOG FLAG
4685 024234 105067 000074          CLR    $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          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, $MSGADR ;; 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     # $ASTAT, R1 ;; GET START OF TABLE
4703 024320 005721          TST    (R1)+        ;; END OF TABLE?
4704 024322 100402          BMI    2$          ;; 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          $MFLG: .BYTE 0      ;; MESSG. FLAG
4711 024335          $LFLG: .BYTE 0      ;; LOG FLAG
4712 024336          $FFLG: .BYTE 0      ;; FATAL FLAG
4713          .EVEN
4714          APTSIZE=200
4715          APTENV=001
4716          APTSPool=100
4717          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          ; *      TYPOS          ;; GO TO THE ROUTINE
4730
4731          $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	RELOC, R0	::	GET RELOCATION FACTOR.
4743	024400	012703	024562			MOV	#SDBLK,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									

::\*\*\*\*\*

```

4791 ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
4792 ;*OCTAL (ASCII) NUMBER AND TYPE IT.
4793 ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
4794 ;*CALL:
4795 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4796 ;*      TYPOS    ;;CALL FOR TYPEOUT
4797 ;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
4798 ;*      .BYTE   M              ;;M=1 OR 0
4799 ;*                                  ;;1=TYPE LEADING ZEROS
4800 ;*                                  ;;0=SUPPRESS LEADING ZEROS
4801 ;*
4802 ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
4803 ;*$TYPOS OR $TYPOC
4804 ;*CALL:
4805 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4806 ;*      TYPON    ;;CALL FOR TYPEOUT
4807 ;*
4808 ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
4809 ;*CALL:
4810 ;*      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
4811 ;*      TYPOC    ;;CALL FOR TYPEOUT
4812 ;*
4813 024572 017646 000000      $TYPOS: MOV      2(SP),-(SP)      ;; PICKUP THE MODE
4814 024576 116667 000001 000213  MOVB     1(SP),SOFILL ;; LOAD ZERO FILL SWITCH
4815 024604 112667 000211      MOVB     (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
4816 024610 062716 000002      ADD      #2,(SP)      ;; ADJUST RETURN ADDRESS
4817 024614 000406      BR       $TYPON
4818 024616 112767 000001 000173 $TYPOC: MOVB     #1,SOFILL ;; SET THE ZERO FILL SWITCH
4819 024624 112767 000006 000167  MOVB     #6,SOMODE+1 ;; SET FOR SIX(6) DIGITS
4820 024632 112767 000005 000156 $TYPON: MOVB     #5,SOCNT  ;; SET THE ITERATION COUNT
4821 024640 010346      MOV      R3,-(SP)      ;; SAVE R3
4822 024642 010446      MOV      R4,-(SP)      ;; SAVE R4
4823 024644 010546      MOV      R5,-(SP)      ;; SAVE R5
4824 024646 116704 000147      MOVB     SOMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
4825 024652 005404      NEG      R4
4826 024654 062704 000006      ADD      #6,R4      ;; SUBTRACT IT FOR MAX. ALLOWED
4827 024660 110467 000134      MOVB     R4,SOMODE  ;; SAVE IT FOR USE
4828 024664 116704 000127      MOVB     SOFILL,R4  ;; GET THE ZERO FILL SWITCH
4829 024670 016605 000012      MOV      12(SP),R5  ;; PICKUP THE INPUT NUMBER
4830 024674 005003      CLR      R3      ;; CLEAR THE OUTPUT WORD
4831 024676 006105      1$: ROL     R5      ;; ROTATE MSB INTO "C"
4832 024700 000404      BR       3$      ;; GO DO MSB
4833 024702 006105      2$: ROL     R5      ;; FORM THIS DIGIT
4834 024704 006105      ROL     R5
4835 024706 006105      ROL     R5
4836 024710 010503      MOV      R5,R3
4837 024712 006103      3$: ROL     R3      ;; GET LSB OF THIS DIGIT
4838 024714 105367 000100      DECB     SOMODE    ;; TYPE THIS DIGIT?
4839 024720 100017      BPL     7$      ;; BR IF NO
4840 024722 042703 177770      BIC     #177770,R3 ;; GET RID OF JUNK
4841 024726 001002      BNE     4$      ;; TEST FOR 0
4842 024730 005704      TST     R4      ;; SUPPRESS THIS 0?
4843 024732 001403      BEQ     5$      ;; BR IF YES
4844 024734 005204      4$: INC     R4      ;; DON'T SUPPRESS ANYMORE 0'S
4845 024736 052703 000060      BIS     #'0,R3    ;; MAKE THIS DIGIT ASCII
4846 024742 052703 000040      5$: BIS     #' ,R3  ;; MAKE ASCII IF NOT ALREADY

```

K14

```

4847 024746 110367 000042 MCVB R3,8$ ;:SAVE FOR TYPING
4848 024752 004567 176442 JSR R5, SPRINT ;:GO PRINT OUT THE FOLLOWING MESSAGE.
4849 024756 025014 .WORD 8$ ;:ADDRESS OF MESSAGE TO BE TYPED
4850 024760 105327 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 TYPING.
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$ ;:RETURN TO PROGRAM AND TRY TO RECOVER
4878 025056 000002 RTI
4879
4880 .SBTTL PHYSICAL ADDRESS TYPE ROUTINE
4881 ;* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4882 $TYPAD:
4883 025060 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
4884 025062 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
4885 025064 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
4886 025066 010346 MOV R3,-(SP) ;:PUSH R3 ON STACK
4887 025070 016602 000012 MOV 12(SP),R2 ;:GET BASE ADDRESS
4888 025074 005003 CLR R3 ;:WORKING & INDEX REGISTER
4889 025076 005767 153504 TST MMAVA ;:CHECK FOR MEM MGMT AVAILABLE
4890 025102 001430 BEQ 1$ ;:BRANCH IF NO MEM MGMT
4891 025104 032737 000001 177572 BIT #1, $#SRO ;:CHECK IF MEM MGMT ENABLED
4892 025112 001424 BEQ 1$ ;:BRANCH IF MEM MGMT NOT ENABLED
4893 025114 010201 MOV R2,R1 ;:COPY VIRTUAL ADDR
4894 025116 006101 ROL R1,R1 ;:SHUFFLE BITS 13,14,15 INTO 1,2,3
4895 025120 006101 ROL R1,R1
4896 025122 006101 ROL R1,R1
4897 025124 006101 ROL R1,R1
4898 025126 006101 ROL R1,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 4S: ASL R1 ;SHIFT PAR
4904 025150 006103 ROL R3 ;SAVE OVERFLOW BITS
4905 025152 077003 SOB R0 ;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 1S: ASL R2 ;FIRST DIGIT TO R3
4910 025166 006103 ROL R3
4911 025170 012700 000006 MOV #6, R0 ;DIGIT COUNT
4912 025174 000404 BR 3S ;PRINT FIRST DIGIT
4913 025176 006302 2S: ASL R2
4914 025180 006103 ROL R3
4915 025182 005301 DEC R1
4916 025184 001374 BNE 2S
4917 025186 012701 000003 3S: MOV #3, R1 ;DIGIT SHIFT COUNT
4918 025212 062703 000060 ADD #60, R3 ;MAKE IT AN ASCII DIGIT
4919 025216 110367 000036 MOV R3, B5 ;LOAD DIGIT INTO MESSAGE
4920 025220 004567 176172 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4921 025226 025260 .WORD B5 ;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 2S
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 R5 ;ADDRESS OF MESSAGE TO BE TYPED
4932 025256 000207 RTS PC ;RETURN
4933 025260 000 .BYTE 0 ;ONE DIGIT MESSAGE BUFFER
4934 025261 000 .BYTE 0 ;MESSAGE TERMINATOR

```

.SBTTL STANDARD PROGRAM MESSAGES

```

*****
; VARIOUS MESSAGE PRINTOUTS USED THRUOUT
; THE PROGRAM
*****

```

```

4941 025262 005015 052113 030461 NAMES: .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

```

STANDARD PROGRAM MESSAGES

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	025484	044124	051105	020505	
4970	025492	000			
4971	025500	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	025584	046111	051125	006505	
4982	025592	000112			
4983	025614	005015	047516	050040	NOPE: .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	MXS: .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	020103	041440	047117	
5053	026378	052123	047101	035124	
5054	026404	000			
5055	026412	015	052412	042516	UNEXPT: .ASCIZ <15><12>'UNEXPECTED MEMORY PARITY ERROR'
5056	026420	050130	041505	042524	
5057	026428	020104	042515	047515	
5058	026436	054522	050040	051101	
5059	026444	052111	020131	051405	
5060	026452	047524	000122		
5061	026456	005015	051120	043517	PRELOC: .ASCIZ <15><12>'PROGRAM RELOCATED TO '
5062	026464	040523	020115	042522	
5063	026472	047514	040503	042524	
5064	026478	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	

STANDARD PROGRAM MESSAGES

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	051101	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
5192	027742	020115	040504	040524
5193	027750	042440	051122	051117
5194	027756	052050	052123	030062
5195	027764	027051	000	
5196	027767	111	051516	051124
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
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
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
5219	030164	042105	052040	020117
5220	030172	030461	027064	000
5221	030177	106	044501	042514
5222	030204	020104	047524	052040
5223	030212	040522	027120	000
5224	030217	050	041501	044524
5225	030224	047117	042440	040516
5226	030232	046102	020105	040527
5227	030240	047123	052047	051440
5228	030246	052105	027051	000
5229	030253	015	052012	040522
5230	030260	050120	042105	052040
5231	030266	020117	020064	000
5232				
5233				
5234				
5235				
5236				
5237	030273	120	004503	042522
5238	030300	004507	027523	004502

DM20: .ASCIZ 'RANDOM DATA ERROR(TST20).'

DM21: .ASCIZ 'INSTRUCTION EXECUTION ERROR(TST21-26).'

DM23: .ASCIZ 'PROGRAM CODE CHANGED WHEN RELOCATED.'

DM24: .ASCIZ 'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'

DM25: .ASCIZ 'TRAPPED TO 114.'

DM26: .ASCIZ 'FAILED TO TRAP.'

DM27: .ASCIZ '(ACTION ENABLE WASN'T SET).'

DM31: .ASCIZ '<15><12>'TRAPPED TO 4 '

\*\*\*\*\*  
; DATA COLUMN HEADINGS  
;\*\*\*\*\*

DM1: .ASCIZ 'PC REG S/B WAS'



F15

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER MACY11 30(1046) 08-SEP-77 10:19 PAGE 104  
DZQMC0.P11 26-JUL-77 15:01 ERROR REPORTING MESSAGES AND TABLES.

SEQ 0187

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 BK
5300								
5301	000001					.END		

ABASE = 000000	384	425																		
ACDW1 = 000000	384	427																		
ACDW2 = 000000	384	428																		
ACPUOP = 000000	384	395																		
ADWD0 = 000000	384	429																		
ADWD1 = 000000	384	430																		
ADWD10 = 000000	384	439																		
ADWD11 = 000000	384	440																		
ADWD12 = 000000	384	441																		
ADWD13 = 000000	384	442																		
ADWD14 = 000000	384	443																		
ADWD15 = 000000	384	444																		
ADWD2 = 000000	384	431																		
ADWD3 = 000000	384	432																		
ADWD4 = 000000	384	433																		
ADWD5 = 000000	384	434																		
ADWD6 = 000000	384	435																		
ADWD7 = 000000	384	436																		
ADWD8 = 000000	384	437																		
ADWD9 = 000000	384	438																		
ADEVCT = 000000	384	390																		
ADEVN = 000000	384	426																		
AE = 000001	179#	2416	2502	3728																
AENV = 000000	384	395																		
AENVN = 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																		
AMSGAO = 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#																		
APTEMV = 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																		
BADAOR = 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*							





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										
IMPCH	= 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*											
L30MAP	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
LADNES	026302	1387	5042#											
LAOMSK	001600	544#	3147	3166	3188	3248	3267	3287	3299	4117*	4119*			
LDDISP	001522	508#	877*											
LF	= 000012	37#	4619	4625										
LMAO	001520	507#	3637	3648*	3655	3669*								
LSTADR	001574	541#	1349*	1399*	1403	1414	1453*	1454*	1455*	1456*	1457	3218	4052	4059
M3MF	017444	1471	2529	3714#										
M3MF1	017530	3728#	3730											
M3MF2	017542	3715	3717	3723	3732#									
MANUAL	005330	1107	1262	1329	1342#	1434								
MANUL1	005366	1348	1355#											
MANUL2	005770	1350	1451#											
M3PHEM	003274	927	940#											
M3PR8	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





PRGMAP	000602	253	254	274	289*	908*	909*	3024	3026	3030	3495	3527	3535*	3553
		3579*	3588	3628*	3629*	4062	4063							
PROREL	025666	4990#												
PRO	= 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#									
PSM	= 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*		
RAOTAB	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#										
RESAVD	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#										
RM	= 000006	176#	3103	3104	3105	3106	3110							
SAVLDL	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#											
SPRNTC	020244	2454	2475	2488	2505	3877#								
SPRNTD	020232	3700	3747	3753	3805	3873#								
SPRNT0	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#						
SRO	= 177572	148#	235	251*	918*	1016*	1474	3119*	3642*	3651*	4891			















# F16

MAINDEC-11-DZQMC-D-D: 0-1.24K MEMORY EXERCISER, 16K VER MACY11 30(1046) 08-SEP-77 10:19 PAGE 119  
 DZQMC.D.P11 26-JUL-77 15:01 CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0200

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	3731	3759	3822
POP	1#	138#	316	317	2498	2511	3369	3386	3488	3580	3618	3705			
	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	
ROCHR	1#	4407													
RODEC	1#														
RDLIN	1#	4492													
RODOCT	1#	1362	1389	1446											
REPORT	1#	138#													
RESREG	1#														
SAVREG	1#														
SCOPE	33#														
SCOPEX	3965#	4034													
SCOPI	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



# H16

MAINDEC-11-DZQMC-D-D: 0-124K MEMORY EXERCISER, 16K VER  
DZQMCD.P11 26-JUL-77 15:01

MACY11 30(1046) 08-SEP-77 10:19 PAGE 121  
CROSS REFERENCE TABLE -- MACRO NAMES

SEQ 0202

.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													
.SACT1	1#	209													
.SAPT8	1#	381#													
.SAPTH	1#	447													
.SAPTY	1#	4625													
.SASTA	1#	469													
.SCATC	1#	188													
.SCHTA	1#	333													
.SEOP	1#	3046													
.SERRO	1#	4128													
.SERRT	1#	4191													
.SPOWE	1#	291													
.SRDOC	1#	4473													
.SREAD	1#	4281													
.SSCOP	1#	3965													
.STYPD	1#	4718													
.STYPE	1#	4546													
.STYPO	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)

I16

3