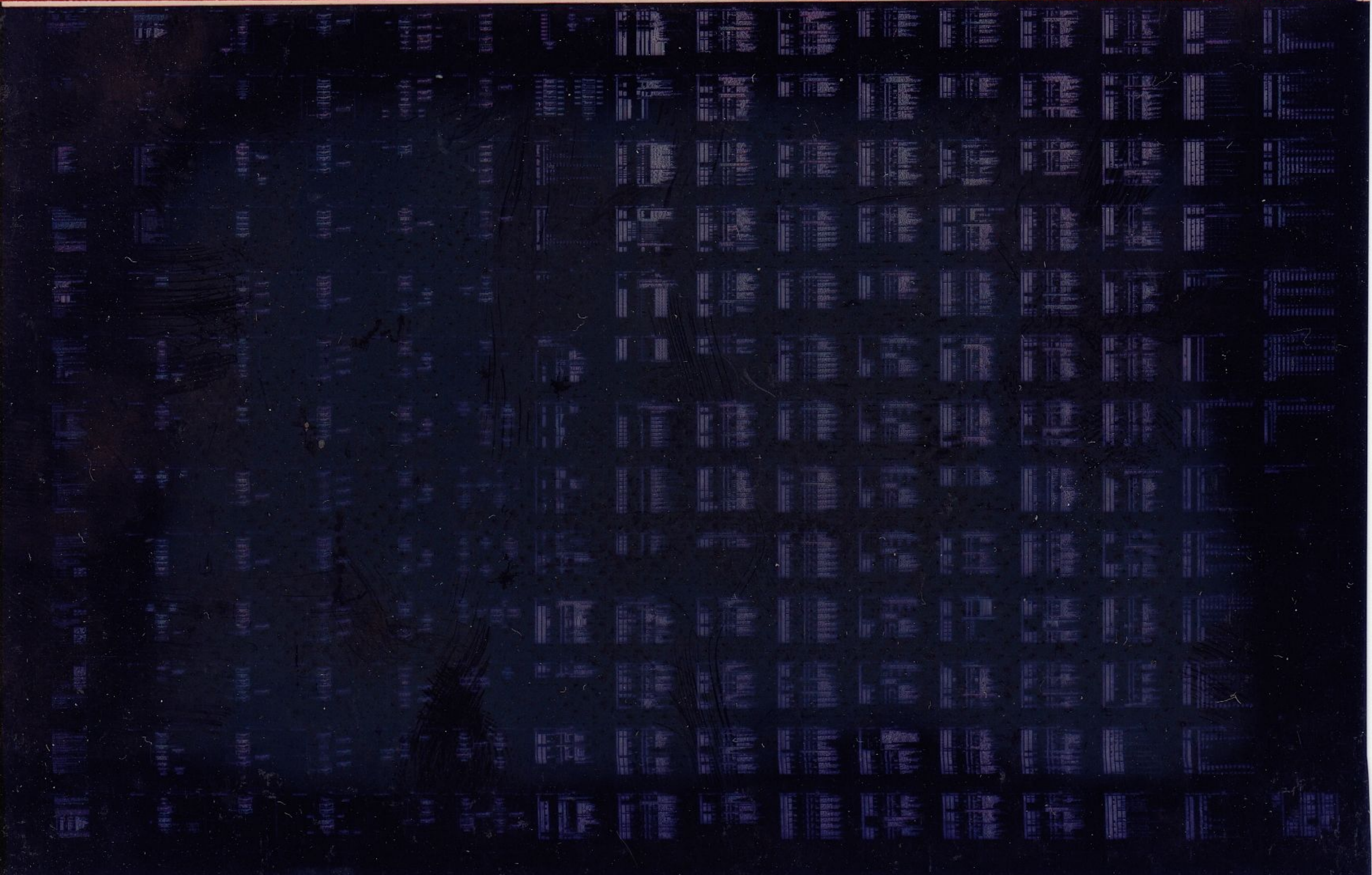


# MS11-K

0-124K MEMORY EXERCISER  
CZQMCE0

AH-9047E-MC  
COPYRIGHT © 75-78  
FICHE 1 OF 1

MAR 1978  
**digital**  
MADE IN USA



EOF1CZD00000411

00010000 780223  
=====

IDENTIFICATION

.PHDR1CZQMCESEG

00010000

780223  
SEQ 0001

PRODUCT CODE: AC-9045E-MC  
 PRODUCT NAME: CZQMCE0 0-124K MEM EXER 16K  
 PRODUCT DATE: FEB 1978  
 MAINTAINER: DIAGNOSTIC ENGINEERING

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.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital or its affiliated companies.

Copyright (c) 1975, 1978 by Digital Equipment Corporation

The following are trademarks of Digital Equipment Corporation:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

REVISION HISTORY

Revision A:	May 1975
Revision B:	October 1975
Revision C:	October 1976
Revision D:	June 1977
Revision E:	December 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. Worse case noise patterns, and
- C. Instruction execution thruout memory.

There is also a special routine to type out all unibus address ranges which do not timeout, as well as two(2) toggle in address tests provided in section 6.1 of this document.

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

## 1.2 System Requirements

## A. Hardware Requirements

PDP11 family processor with a minimum of 16K of memory.  
optional...  
Any parity memory control module.  
KT11 memory management.

## B. Software Requirements

The smallest unit of memory this program will recognize is 4K. If any address in a 4K bank causes a time out trap, that entire bank of memory is ignored by the program.

The program is designed to exercise the vector portion of memory (locations 0-776) in exactly the same manner as the rest of memory. To make this possible, without requiring memory management, no software traps are used in the program. This means that if memory management is not available or is disabled (SW12=1), if the program is relocated out of bank 0, if location 0-776 are selected for test, and if an unexpected hardware trap occurs, the results will be unpredictable.

The program has the proper interface code to allow running under the automated manufacturing test line system - ACT11 and APT.

### 1.3 Related Documents and Standards

- A. Programming practices - Document No. 175-003-009-01
- B. PDP-11 MAINDEC SYSMAC Package - MAINDEC-11-DZQAC-C2-D
- C. MF11-U/UP Core Memory System Maintenance Manual  
Document No. DEC-11-HMFMA-B-D
- D. 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-DFKTH  
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 (KIPARO)  
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 printout on a parity error does not type the good data. Thus a bit drop or pickup will not be typed as such. It is best to run the program for 1 pass with parity disabled, then, restart the program with parity enabled.

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

## 2.4 EXECUTION TIMES

Execution time is dependent on type of memory, and amount of memory. Worse case run times with 900ns memorys are:

- a. For Non-Parity Memory
 

First Pass:	65 seconds for first 16k + 15 seconds for each additional 16k.
Full Pass:	3 minutes 40 seconds for first 16k + 3 minutes for each additional 16k.



Iteration Inhibited: same as first pass

- b. For Parity Memory  
First Pass: 1 minute 40 seconds per 16k.  
Full Pass: 8 minutes per 16K  
Iteration Inhibited: same as first pass

### 3.0 ERROR INFORMATION

#### 3.1 Error Reporting

There are a total of 31(8) types of error reports generated by the program. Some of the key column heading mnemonics are described below for clarity:

PC = Program Counter of error detection code.  
(V/PC=P/PC)

V/PC = Virtual Program Counter. This is where the error detection code can be found in the program listing.

P/PC = Physical Program Counter. This is where the error detection code is actually located in memory.

TRP/PC = Physical Program Counter of the code which caused a trap.

MA = Memory Address

REG = Parity REGISTER address.

PS = Processor Status word.

IUT = Instruction Under Test.

S/B = What contents Should Be.

WAS = What contents WAS.

#### 3.2 Error Halts

With the 'HALT ON ERROR' switch (SW15) not set there are several programmed 'HALTS' in the program:

- A. In the error trap service routine for unexpected traps to vector 4. This one will occur if a 2nd trap to 4 occurs before the error report for the first has had a chance to be printed out.

- B. In the relocation routine if the program is being relocated back to the first 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.

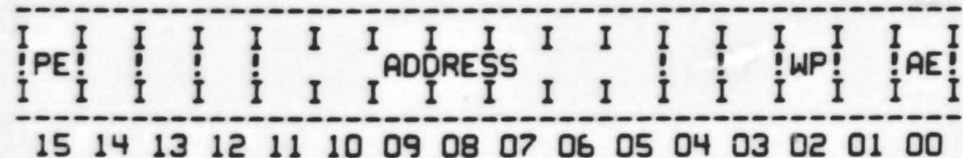
4.0 PERFORMANCE AND PROGRESS REPORTS

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:

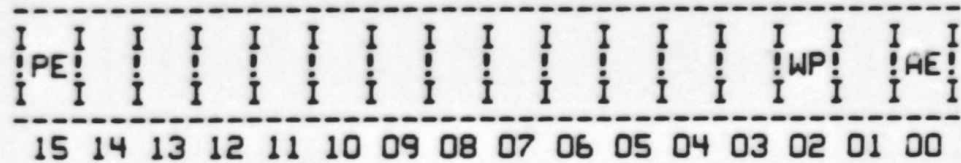
5.1 CORE PARITY REGISTER



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

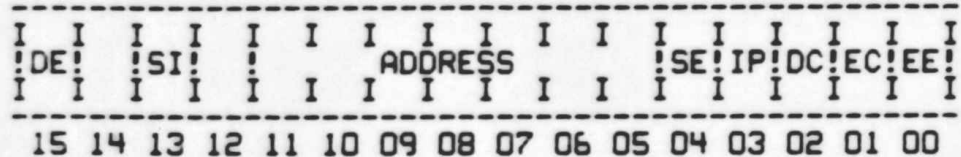
5.2 MOS PARITY REGISTER



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	PARITY ERROR	
BIT02	WRITE WRONG PARITY	NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET
BIT00	ACTION ENABLE	NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

5.3 MS11-K CSR



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15	DOUBLE ERROR	
BIT 13	SET INHIBIT MODE	WHEN THIS BIT IS SET TO A 1, IT ENABLES THE INH MODE POINTER TO INHIBIT EITHER THE FIRST OR SECOND 16K FROM EVER GOING INTO THE DIAG. CHECK OR ECC DISABLE MODE.
BITS 11-5	ERROR ADDRESS	WHEN BIT02 CLEARED CONTAINS HIGH ORDER BITS OF ADDRESS OF PARITY ERROR (BITS 17-11); WHEN BIT02 SET CONTAINS CHECK BITS FOR ECC.
BIT04	SINGLE ERROR	SET WHENEVER SINGLE ERROR OCCURS
BIT03	INHIBIT MODE POINTER	THE INHIBIT MODE POINTER WORKS IN

CONJUNCTION WITH THE SET INHIBIT MODE BIT. WHEN BIT 13 IS SET TO A 1, A 16K PORTION OF MEMORY IS 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

THE SECOND 16K OF MEMORY IS INHIBITED. WHEN BIT 13 IS SET TO A 0, BIT 3 BECOMES INOPERATIVE.

BIT02	DIAGNOSTIC CHECK A	WHEN SET ENABLES READ-WRITE OF CHECK BITS(SEE BITS 11-5)
BIT01	DISABLE ERROR CORRECTION	WHEN SET NO ERROR CORRECTION TAKES PLACE
BIT00	DOUBLE ERROR ENABLE	WHEN SET ENABLES TRAP TO VECTOR 114 ON DOUBLE ERROR.

## 6.0 SUB-TEST SUMMARIES

### 6.1 Section 1: Address Tests.

These tests verify the uniqueness of every memory address.

TEST 1 Writes and reads the value of each memory Word Address into that Memory location. After all memory has been written, all locations are checked again.

TEST 2 Writes the byte value of each address into that byte location and checks it.

TEST 3 Writes the complement of each word address into that location and checks it.

TEST 4 Writes the 4K bank number into each byte of that bank and checks it.

TEST 5 Writes the complement of the bank number into each byte of that bank and checks it.

## 6.2 Section 2: Worst Case Noise Tests.

These are intended to apply maximum stress to the various types of PDP-11 core memories.

TEST 6 and TEST 7 Are supplied to allow the operator to select a single word data pattern (SA=204) and SCOPE on either the writing (DATO) in TEST 6 or the reading (DATI) in TEST 7 of that data.

TEST 10 Writes and then checks a series of single word patterns which are designed to stress parity memory.

TEST 11 Writes all memory with 1's in every bit and then "Ripples" a "0" through it.

TEST 12 Writes all memory with 0's in every bit and then "Ripples" a "1" through it.

TEST 13,14,15, AND 16 Write a pattern which complements when address BIT 3 XOR BIT 9 complements.

TEST 17 Writes wrong parity in each byte of memory and checks that the parity detection logic works. This test is skipped for non-parity memory.

TEST 20 Write "random" program code through memory and checks it.

## 6.3 Section 3: Instruction Execution Tests.

This group of tests place instructions in the memory under test, then executes the instructions, and finally, checks that they executed correctly.

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

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

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

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

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

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

## 6.4 Section 4: Mos Tests

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

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

## 6.5 Special Toggle In Tests

## 6.5.1 Toggle-in-program #1

The following is a toggle in memory address test. This test is useful when an address selection failure is suspected involving the first BK of memory. This program writes the value of each address into itself starting with the lower limit and continuing to the upper limit. After all addresses have been written each address is checked for the correct contents starting with the upper limit and continuing to the lower limit.

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

After toggling the program LA=10\*\*set upper limit\*\*, start

NOTES: The upper limit address obtained from the switch register may be changed during program operation. However occasionally the program may halt because of 'SWITCH BOUNCE'. (The best procedure when changing limits is to stop the program make the change and continue.) The lower limit address (12) may be patched to any desired address.

## 6.5.2 Toggle-in-Program #2

The following is also a toggle in program to be used with toggle-in-program #1 for more complete address testing. This program writes the complement value of each address into itself starting with the upper limit and continuing to the lower limit. After all addresses have been written each address is checked for the correct contents starting with the lower limit address and continuing to the upper limit. Toggle in the following patches to the program above.

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

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

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

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

7.D PROGRAM FUNCTIONAL FLOW CHARTS  
Attached

8.D PROGRAM LISTING  
Attached

FLOW CHART  
\*\*\*\*\*

CZQMCE0 0-124K MEM EXER 16K

\*\*\*\*\*

COPYRIGHT 1978  
DIGITAL EQUIPMENT CORPORATION  
MAYNARD, MASS. 01754



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
PAGE 31	TEST 15: 3 XOR 9 (FOR PARITY)

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: EXECUTE DATI, DATO
PAGE 41	TEST 22: EXECUTE DATI, DATOB (LO BYTE)
PAGE 42	TEST 23: EXECUTE DATI, DATOB (HI BYTE)
PAGE 43	TEST 24: EXECUTE DATIP, DATO
PAGE 44	TEST 25: EXECUTE DATIP, DATOB (LO BYTE)
PAGE 45	TEST 26: EXECUTE 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
PAGE 63	SYSMAC AND STANDARD UTILITY ROUTINES

CZQMCE0 0-124K MEM EXER 16K  
DEFINITIONS, TRAP CATCHER, STARTING ADDRESSES.

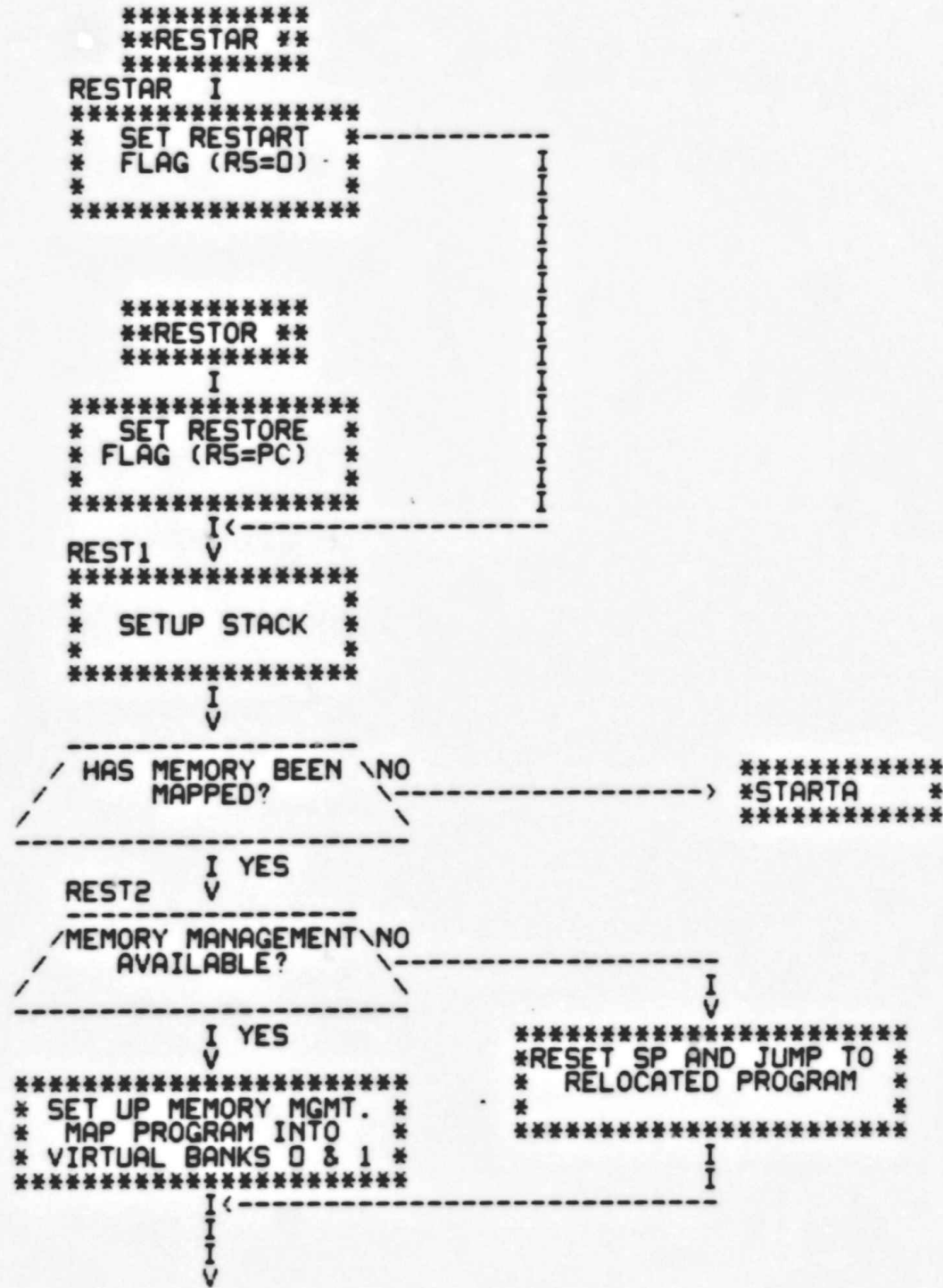
```
*****  
* SWITCH SETTINGS AND *  
* BASIC DEFINITIONS *  
* *  
*****
```

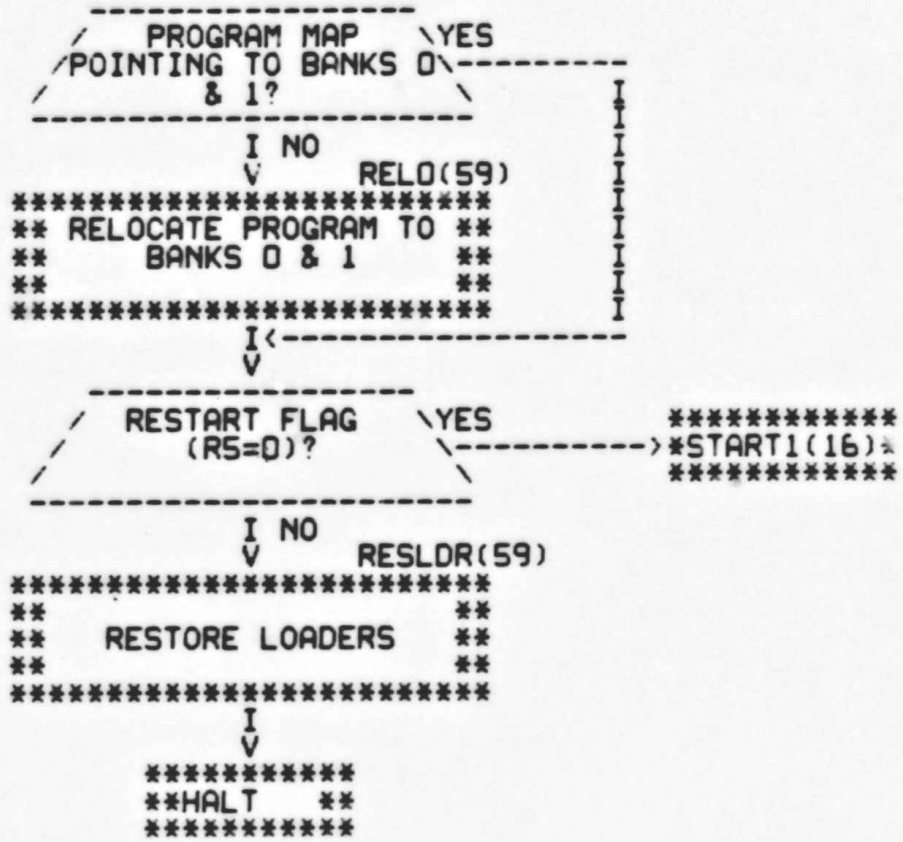
.=0

```
*****  
* TRAP CATCHER AND *  
* STARTING ADDRESSES *  
* *  
*****
```

SA=210 . =300

SA=214





. =572

CZQMCE0 0-124K MEM EXER 16K  
POWER FAIL ROUTINES

```

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

```

```

*****
**$PWRUP **
*****
  I
  V
*****
* WAIT LOOP FOR TTY *
* RESTORE REGISTERS *
* $PWRDN -> VECTOR *
*****
  I
  V
*****
* SPRINT(63)
* / TYPE POWER FAIL \
* / MESSAGE           \
*****
  I
  V
*****
**RETURN **
*****

```

```

*****
**$ILLUP **
*****
  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

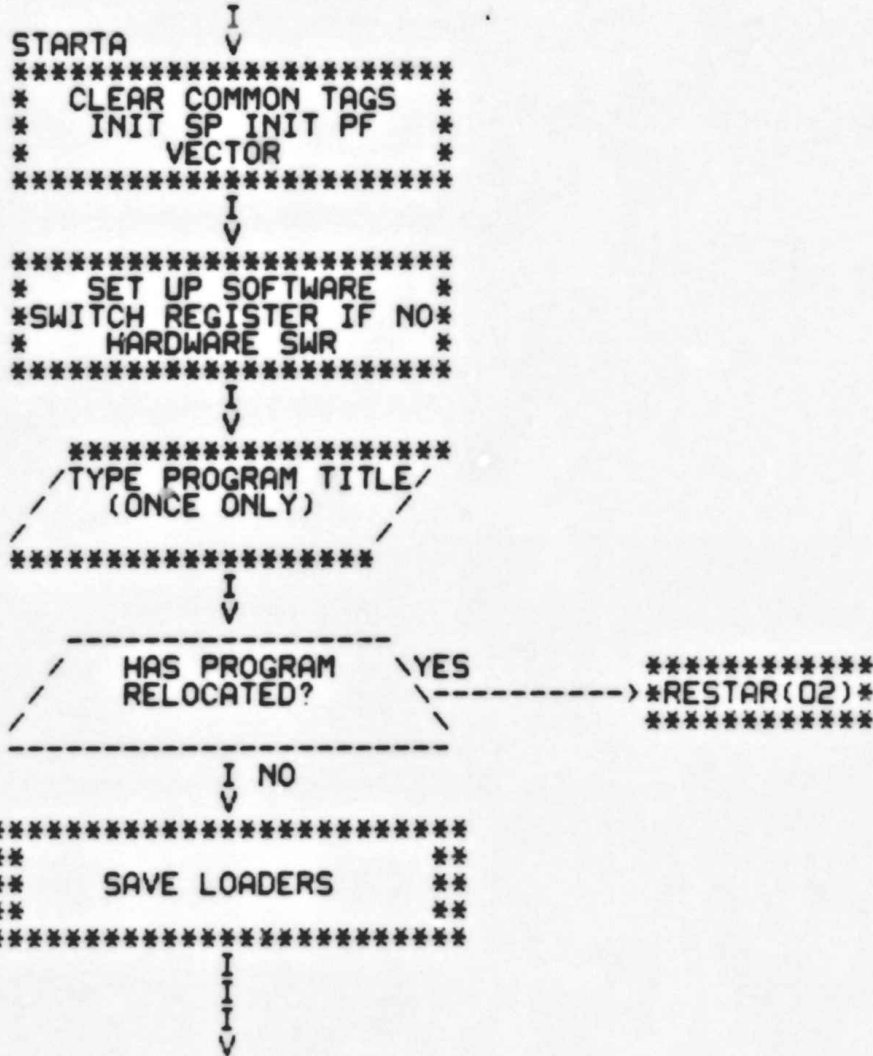
```

SA=200

```

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

```



```

*****
**RESTAR(02)**
*****

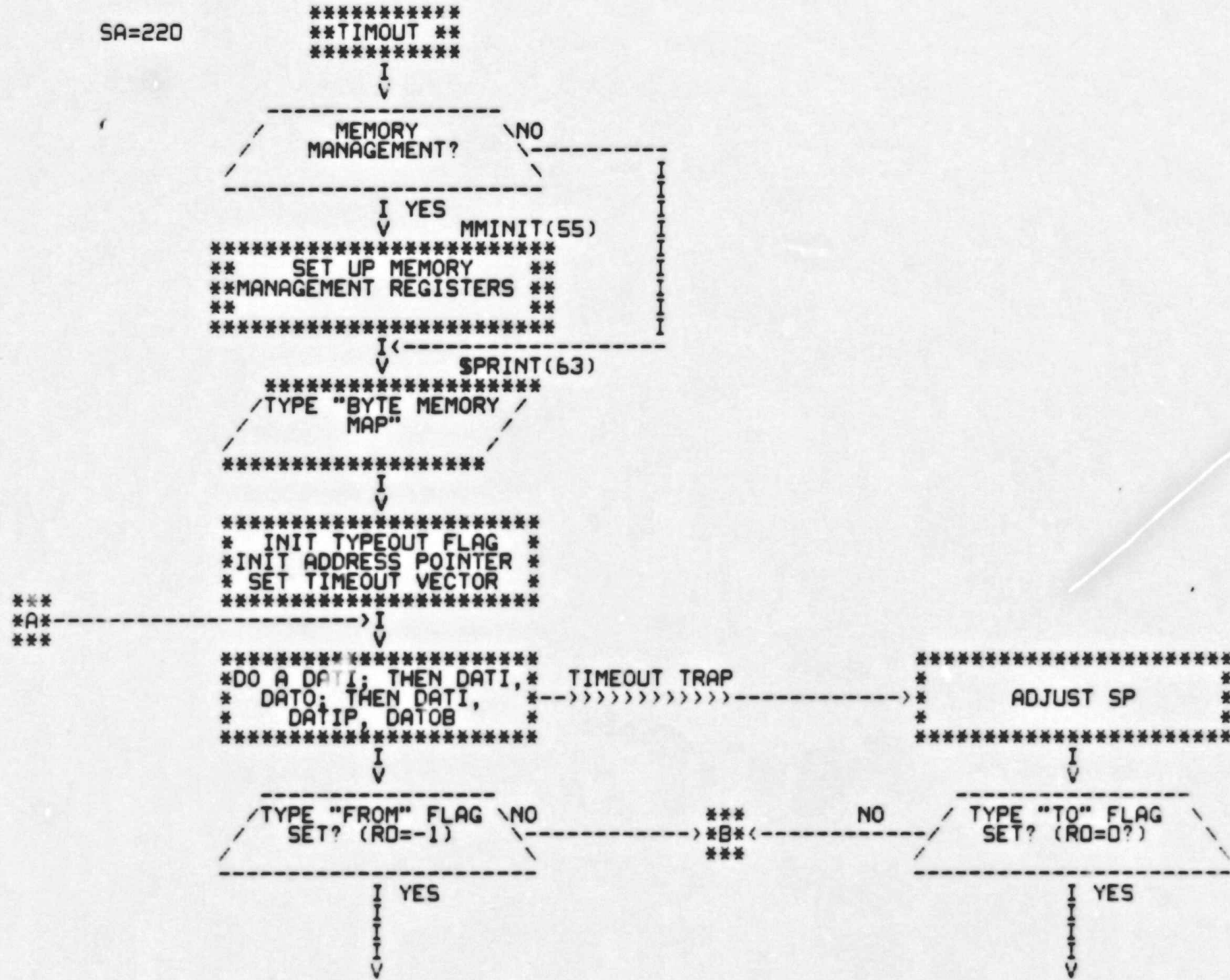
```

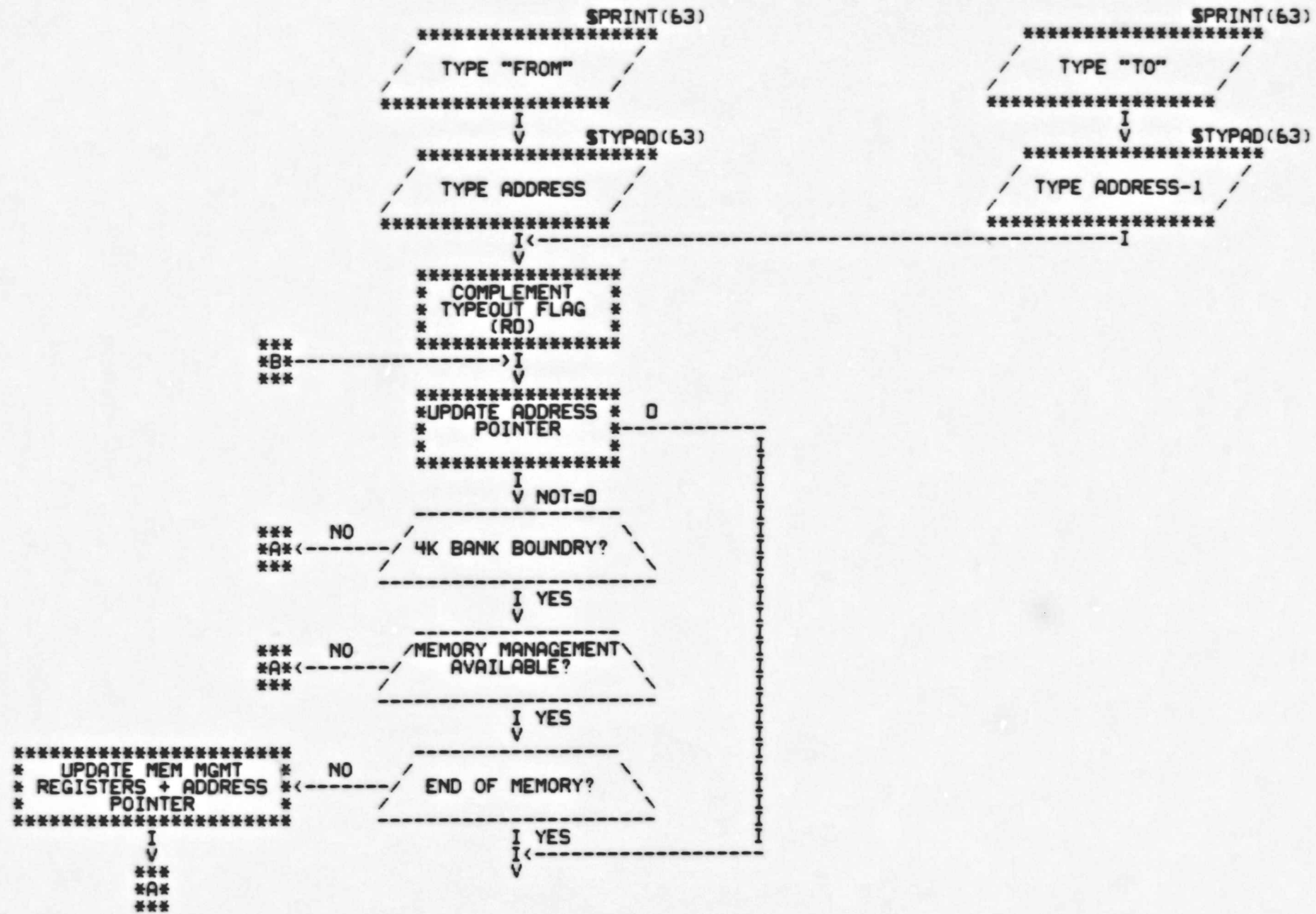


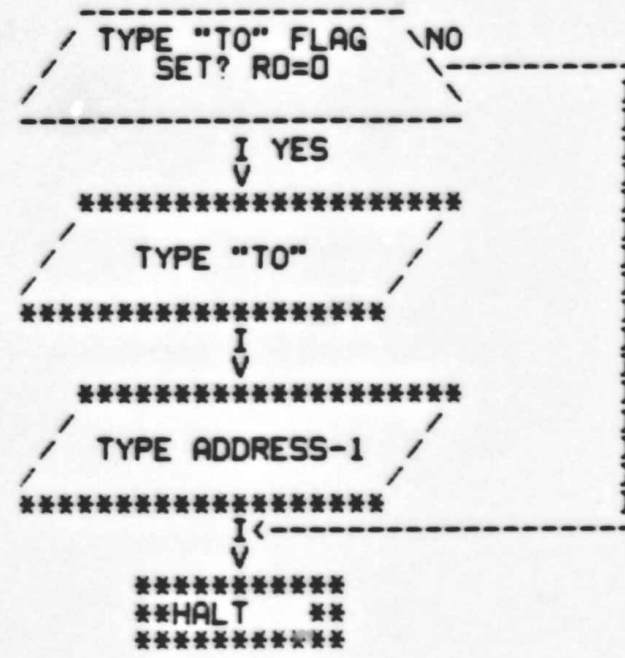




SA=220

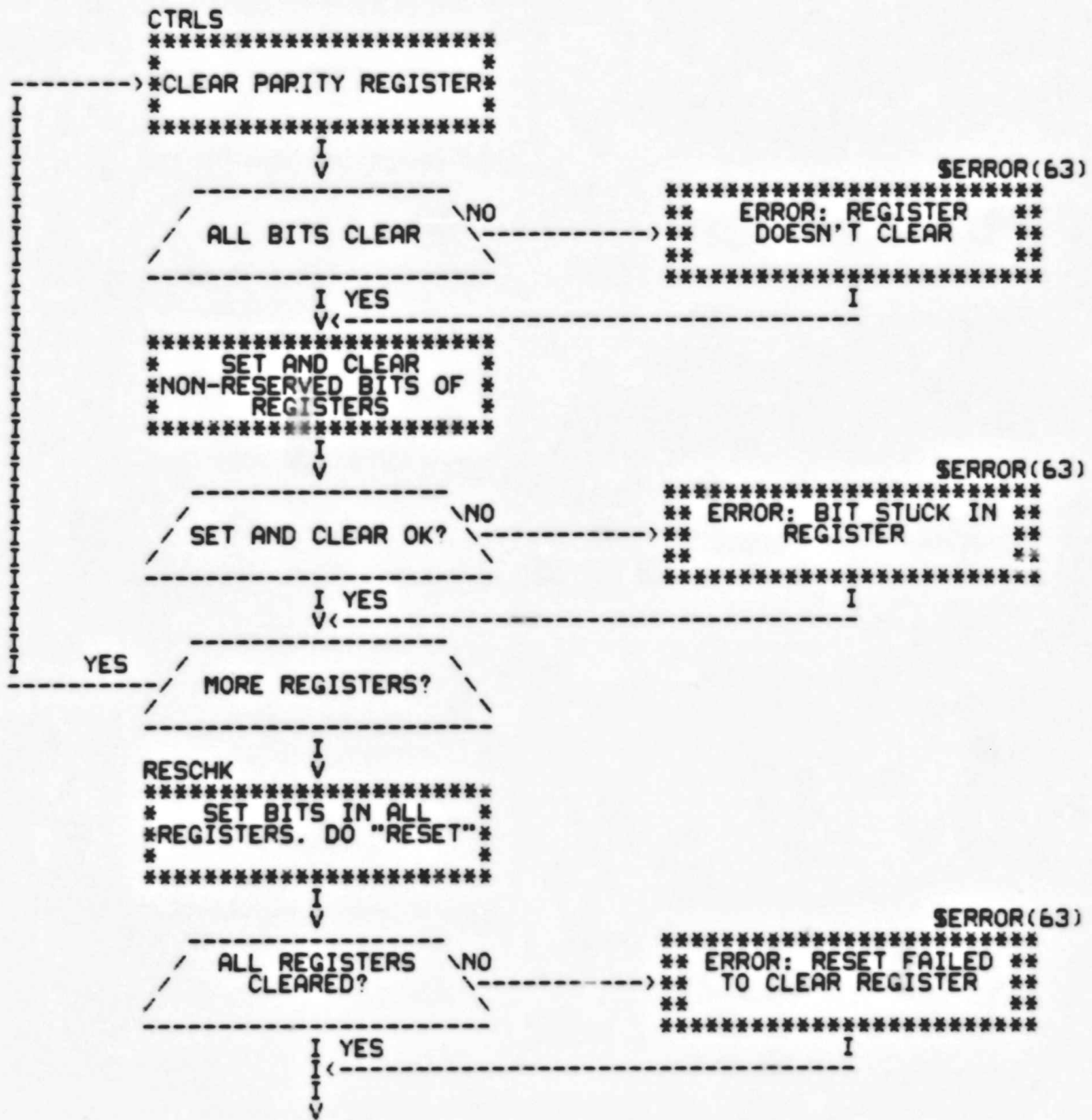




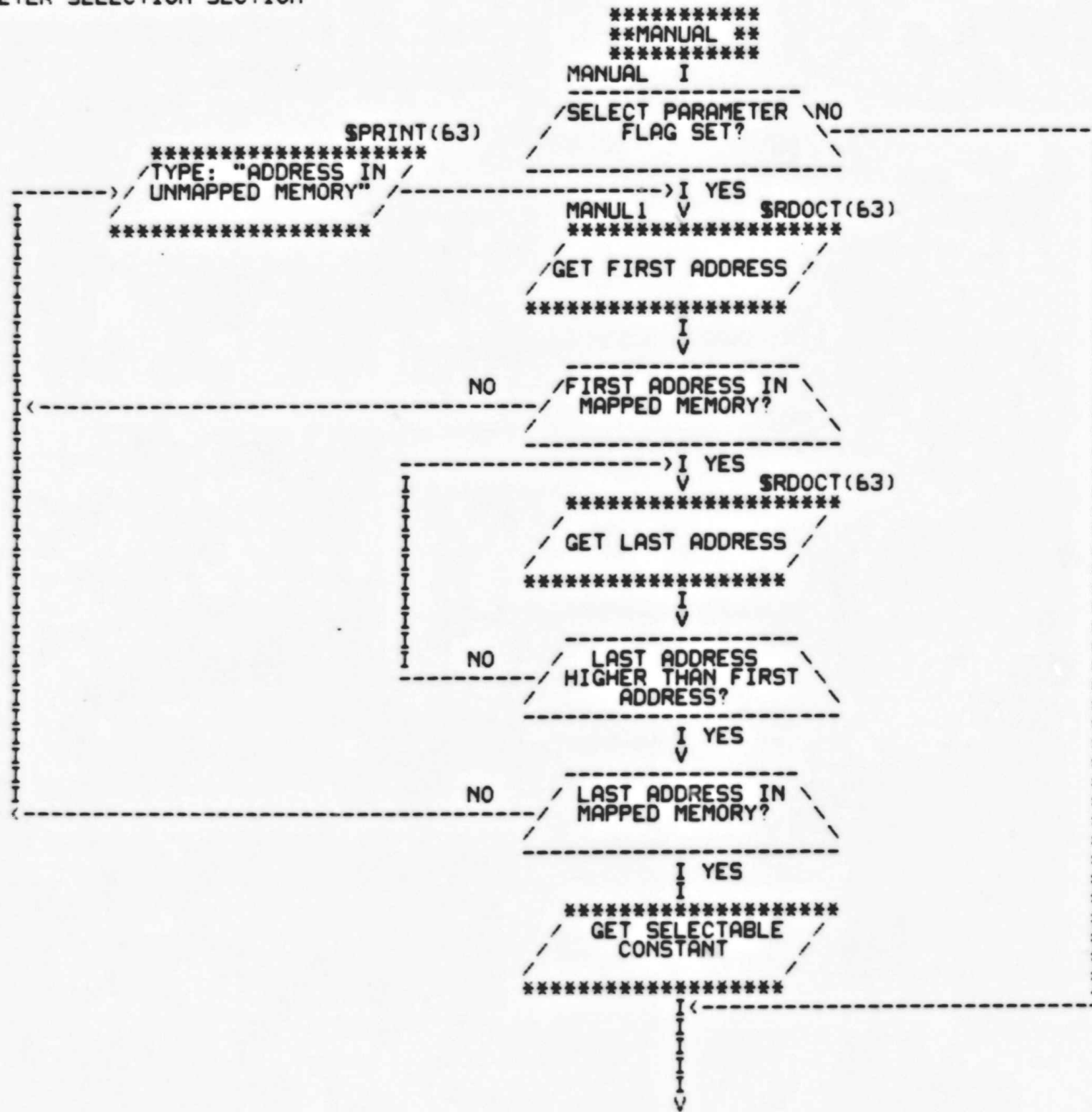






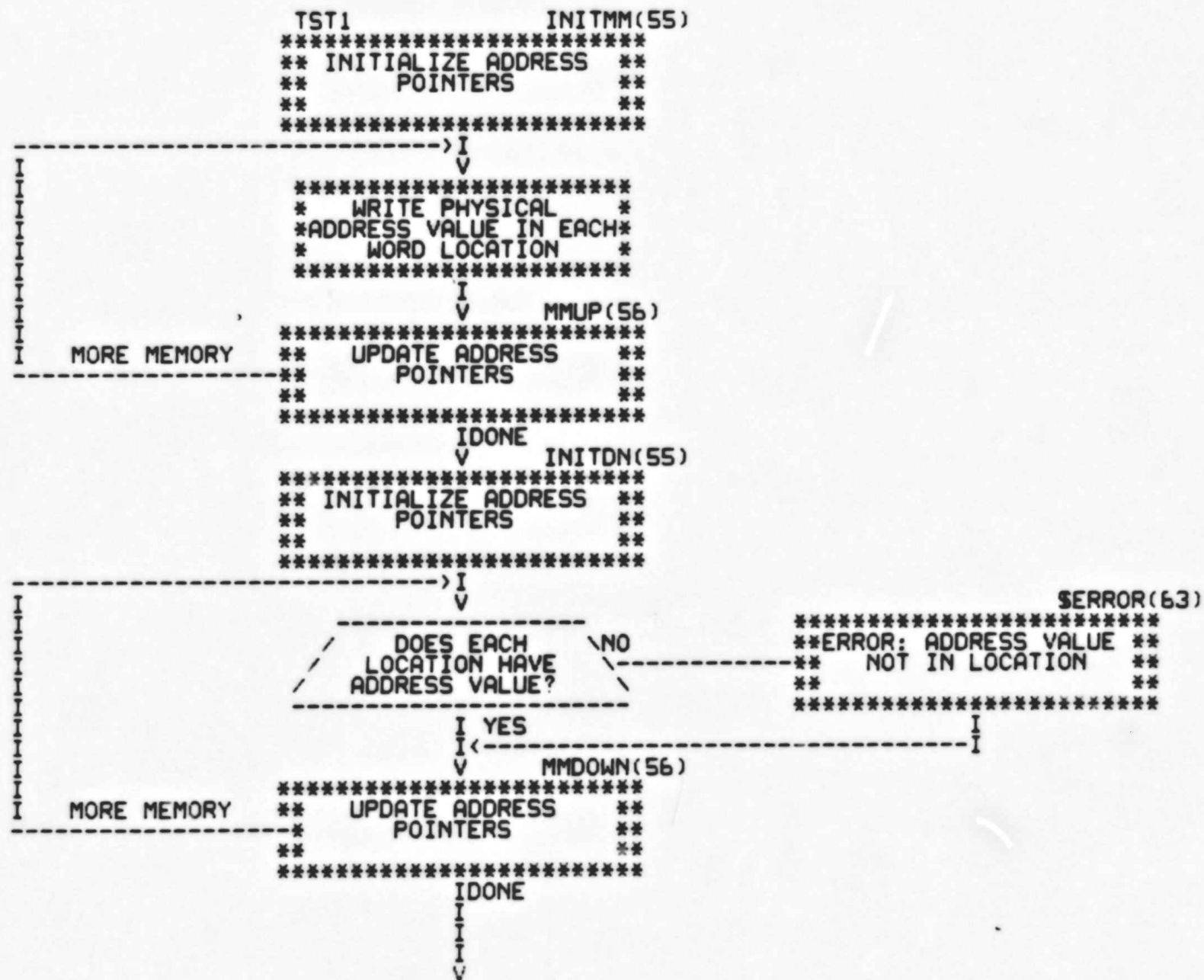


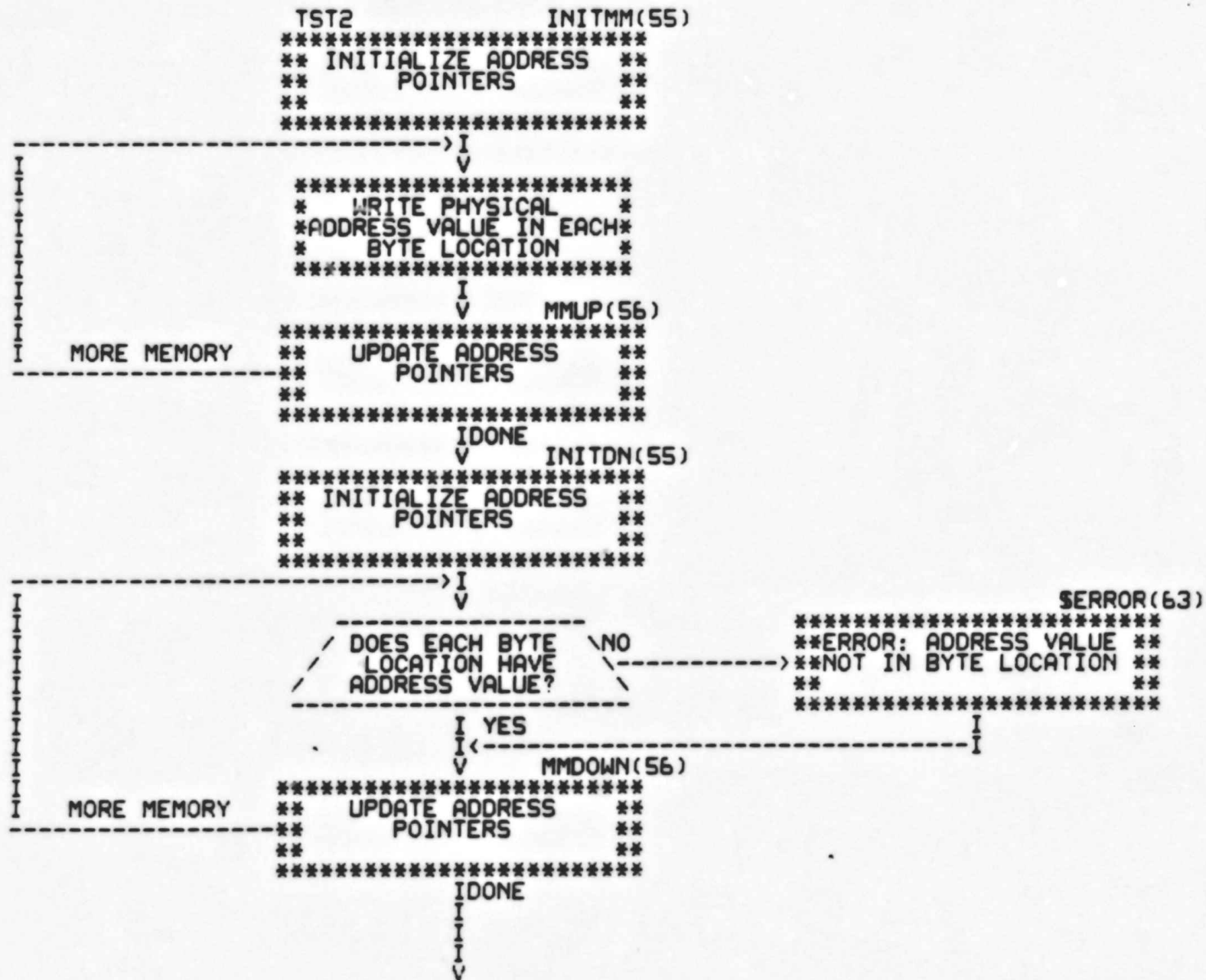


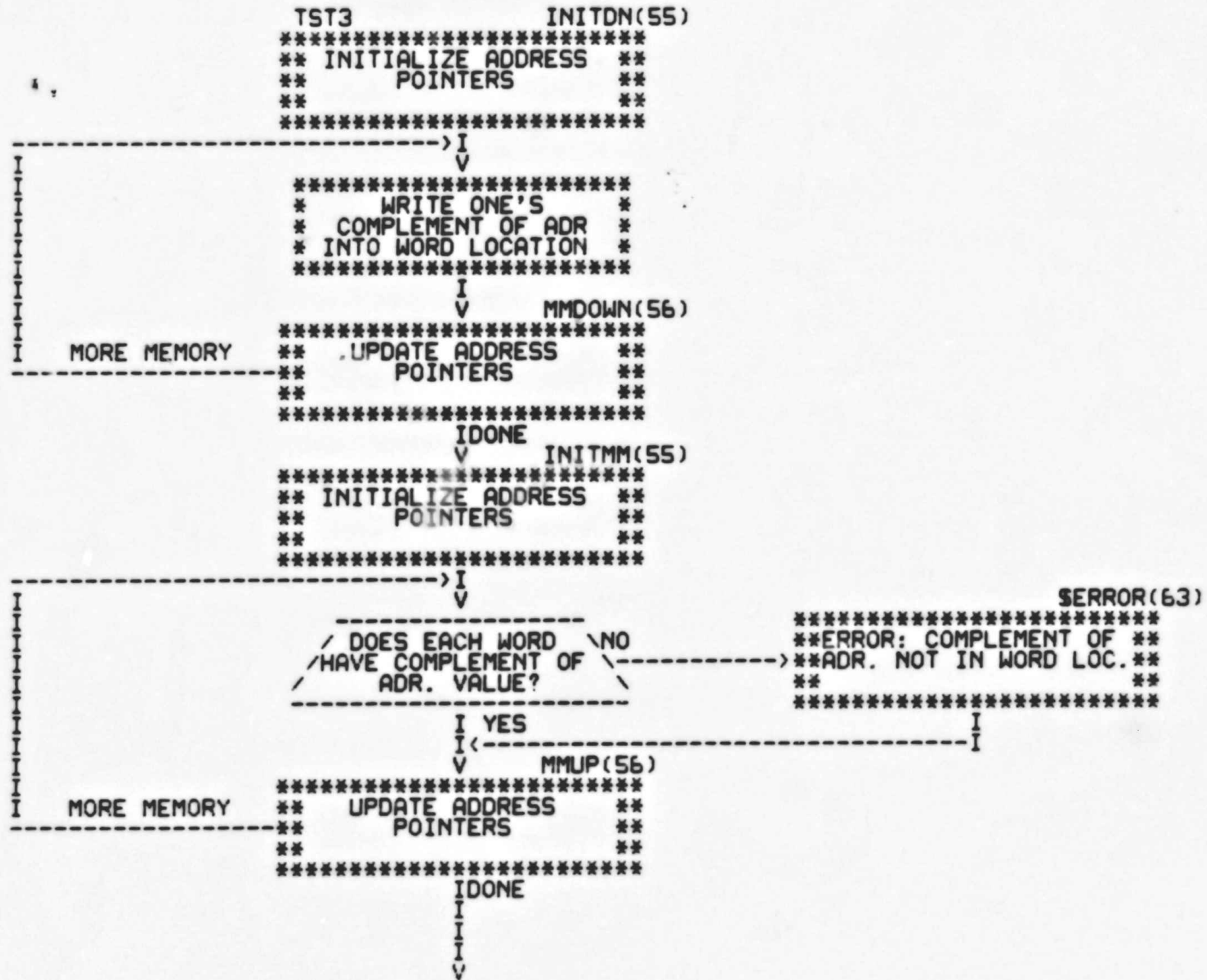


CZQMCEO 0-124K MEM EXER 16K  
START1: START OF PASS

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

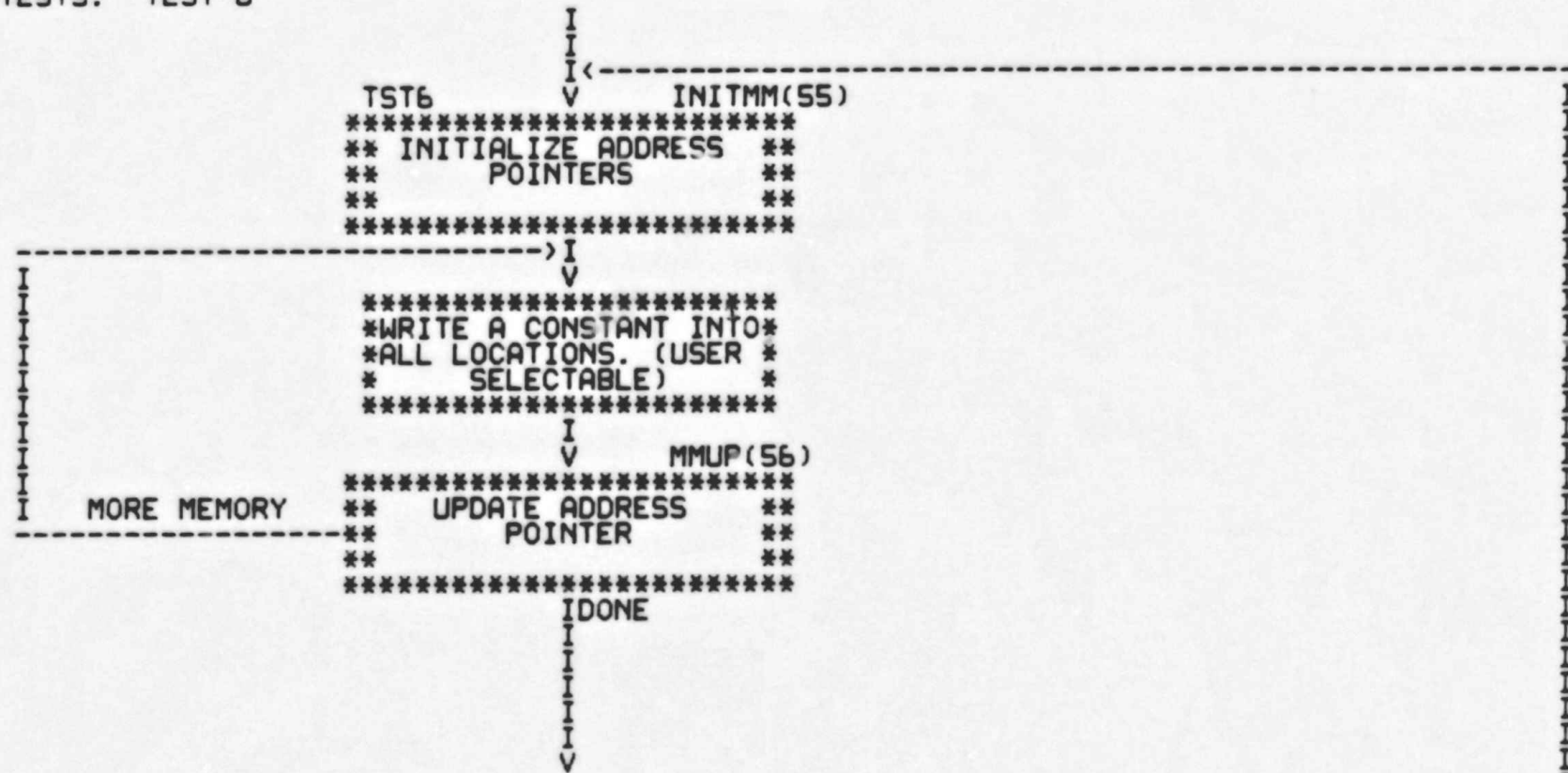




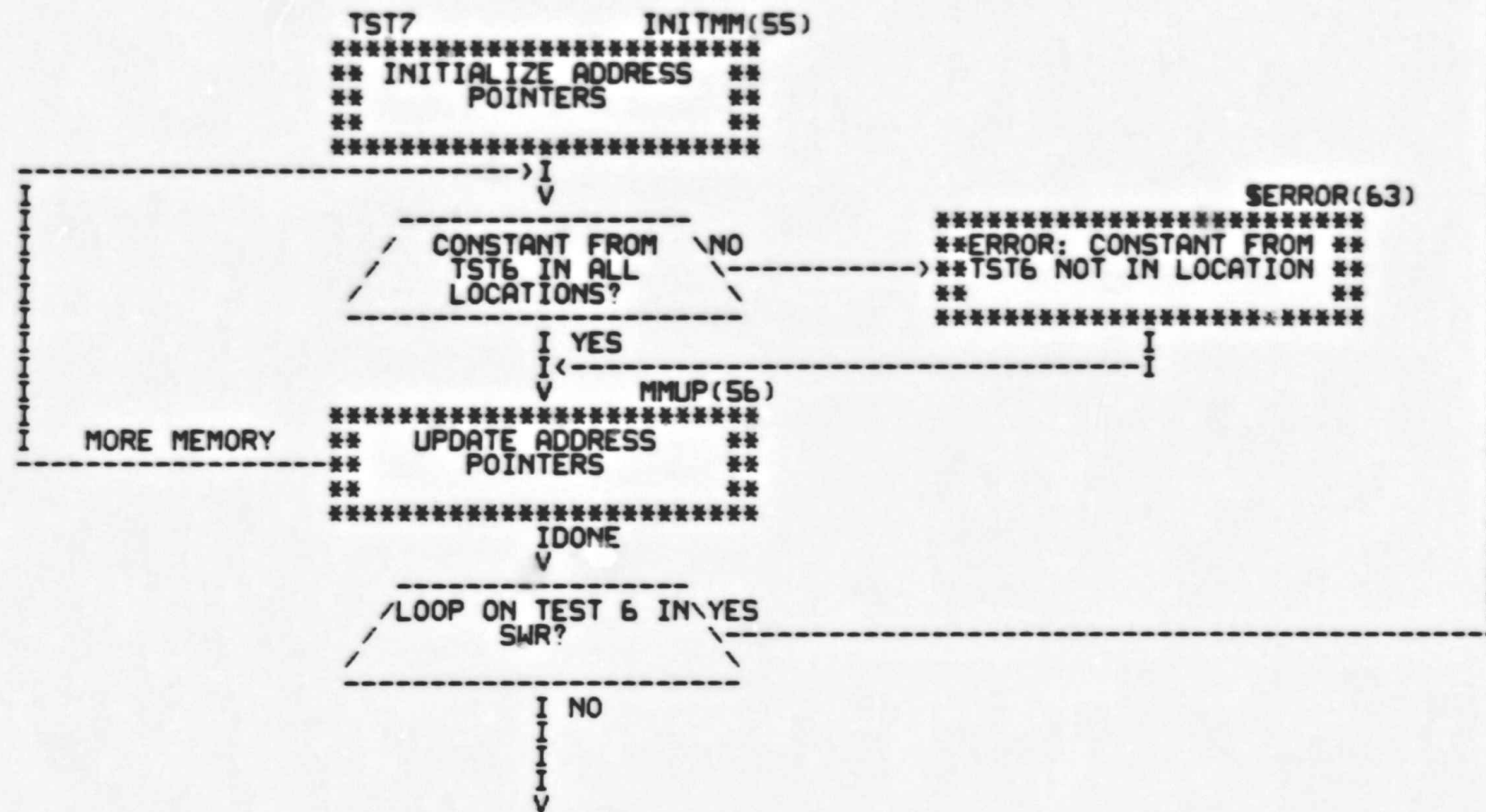


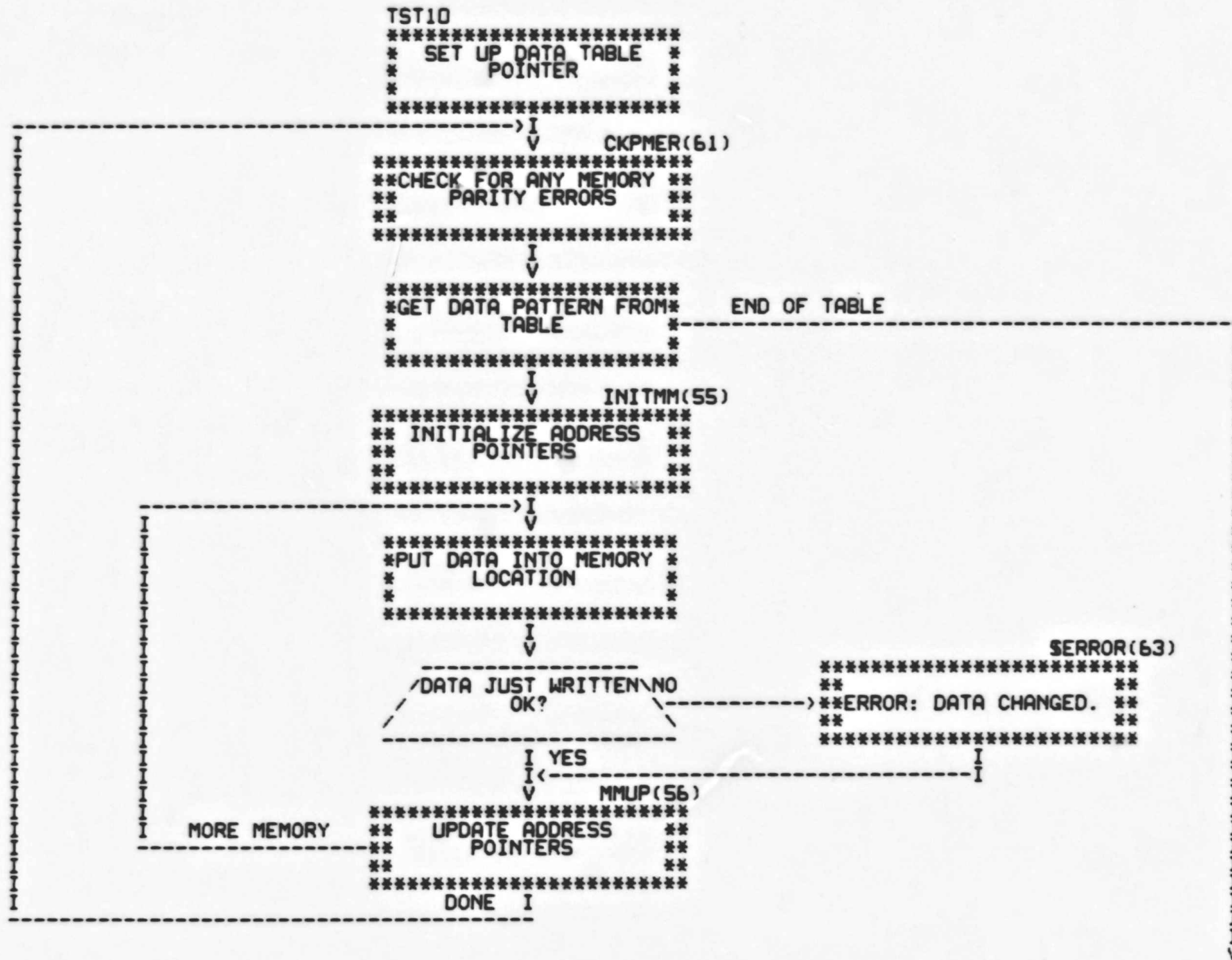


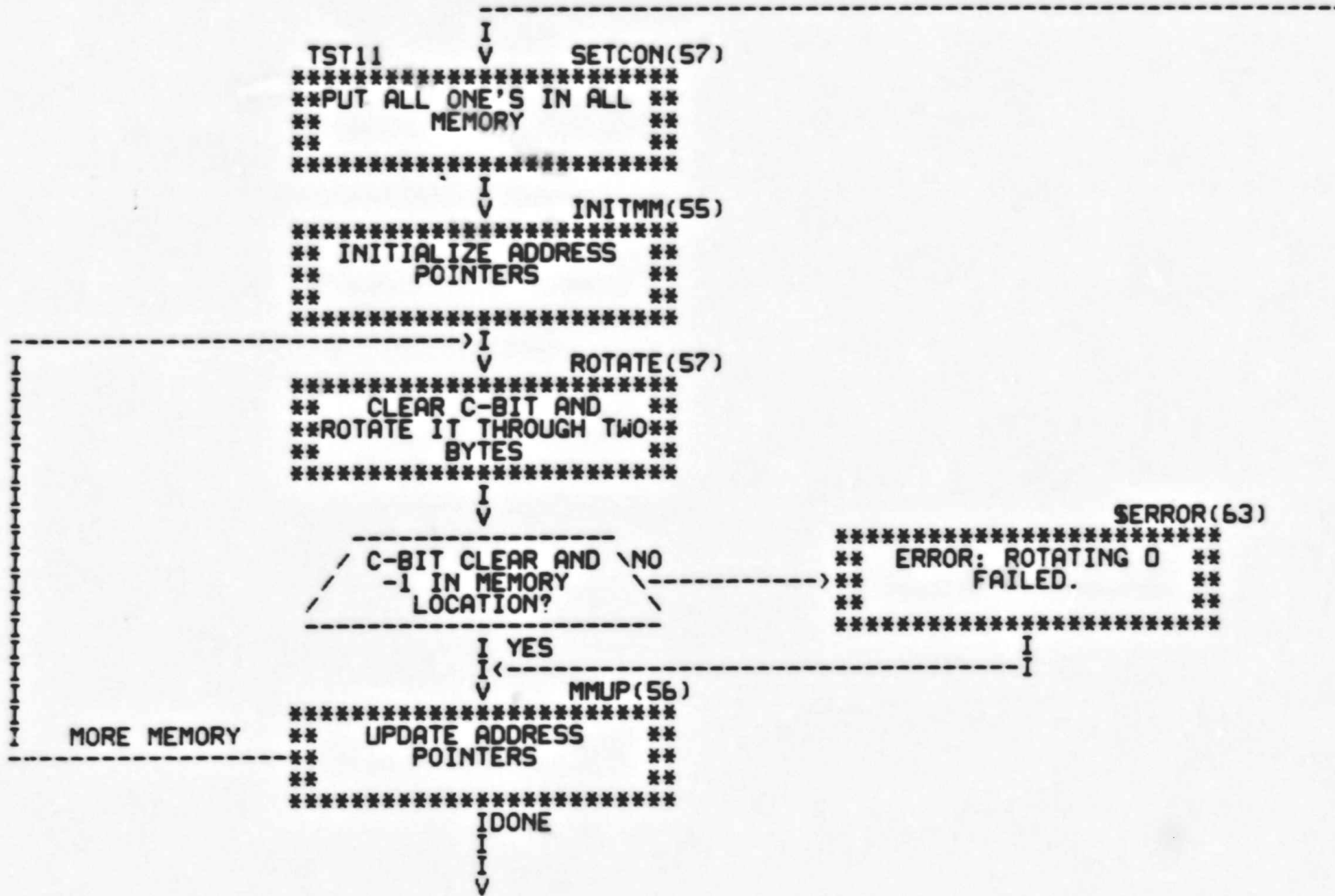


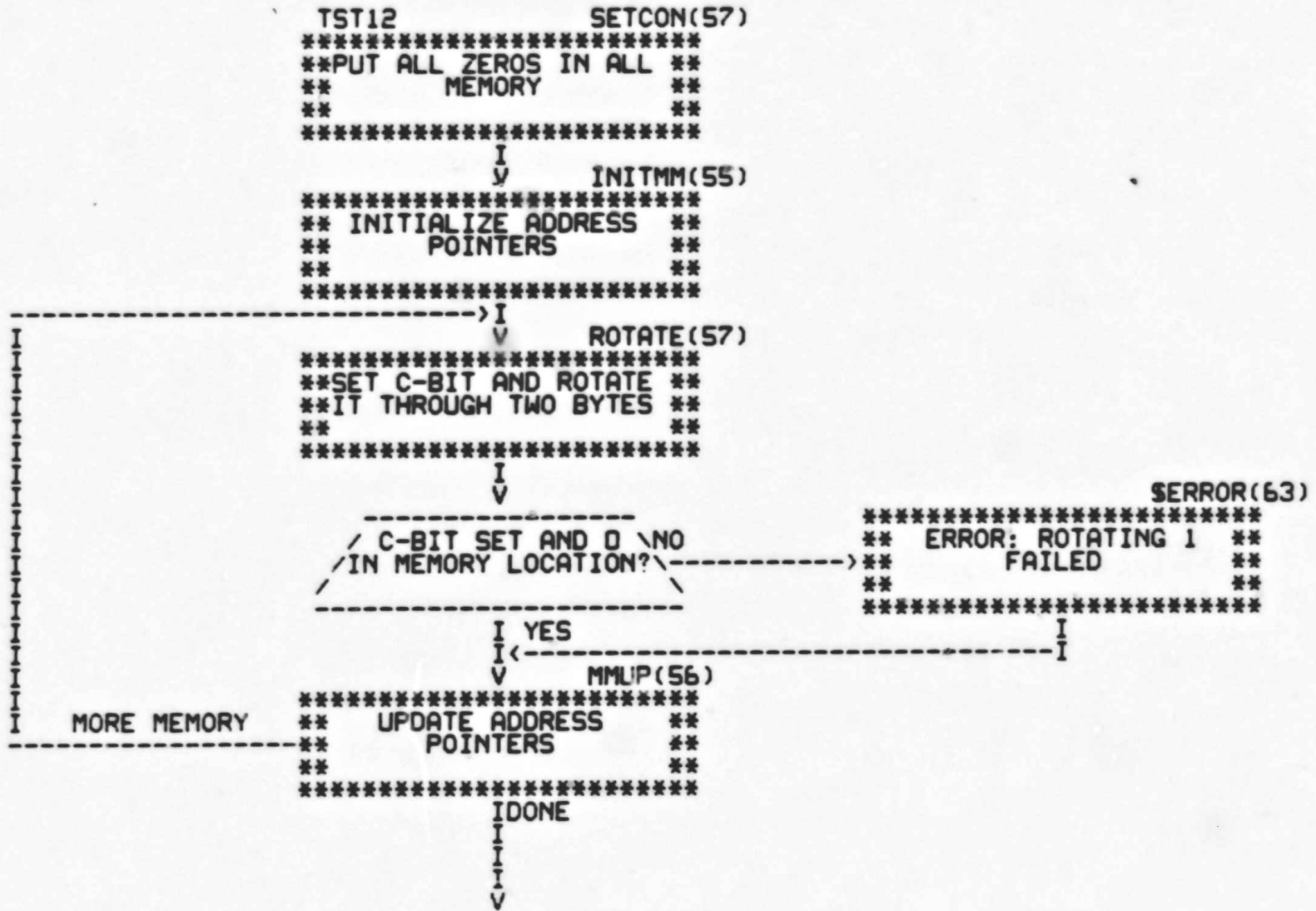






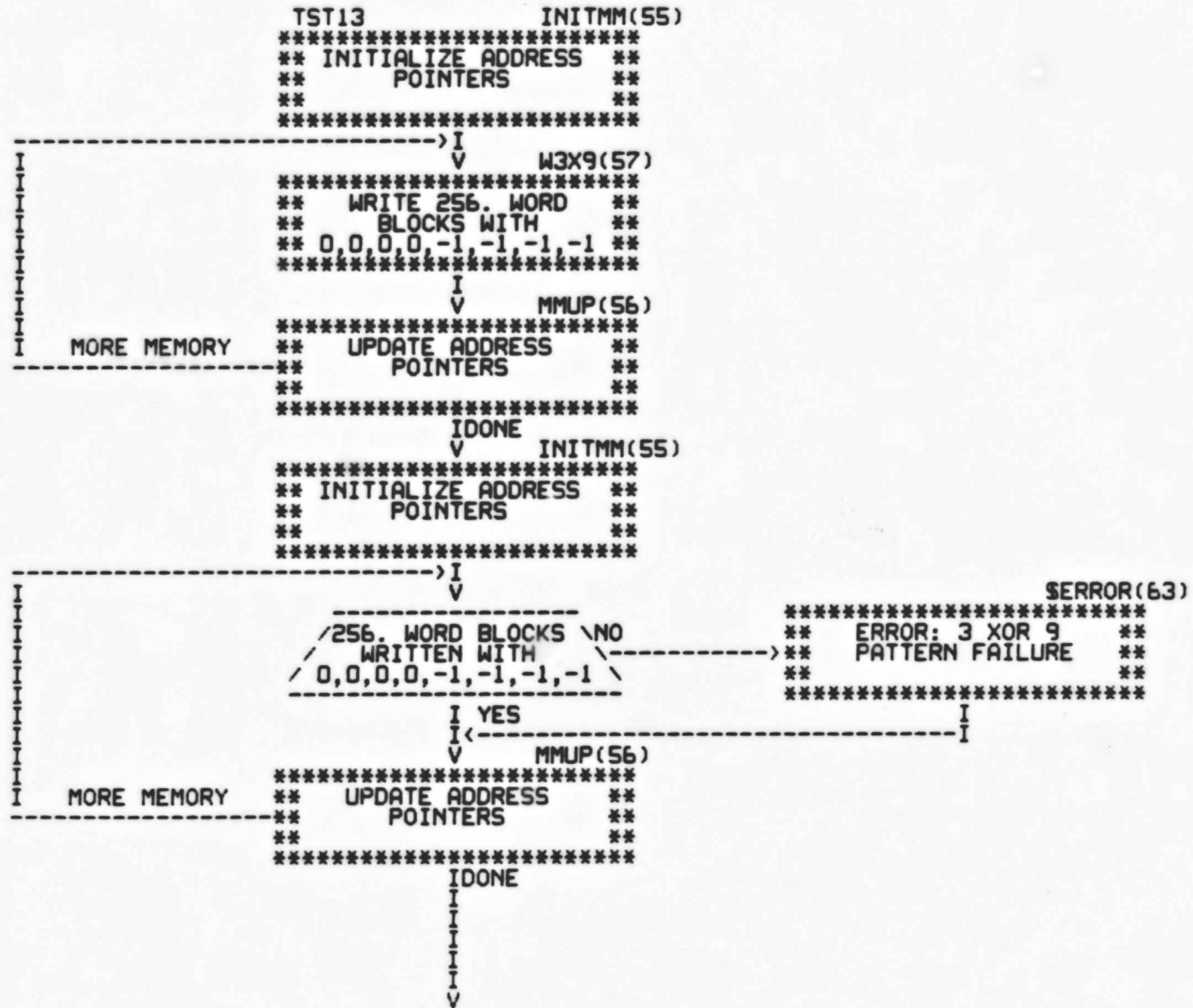


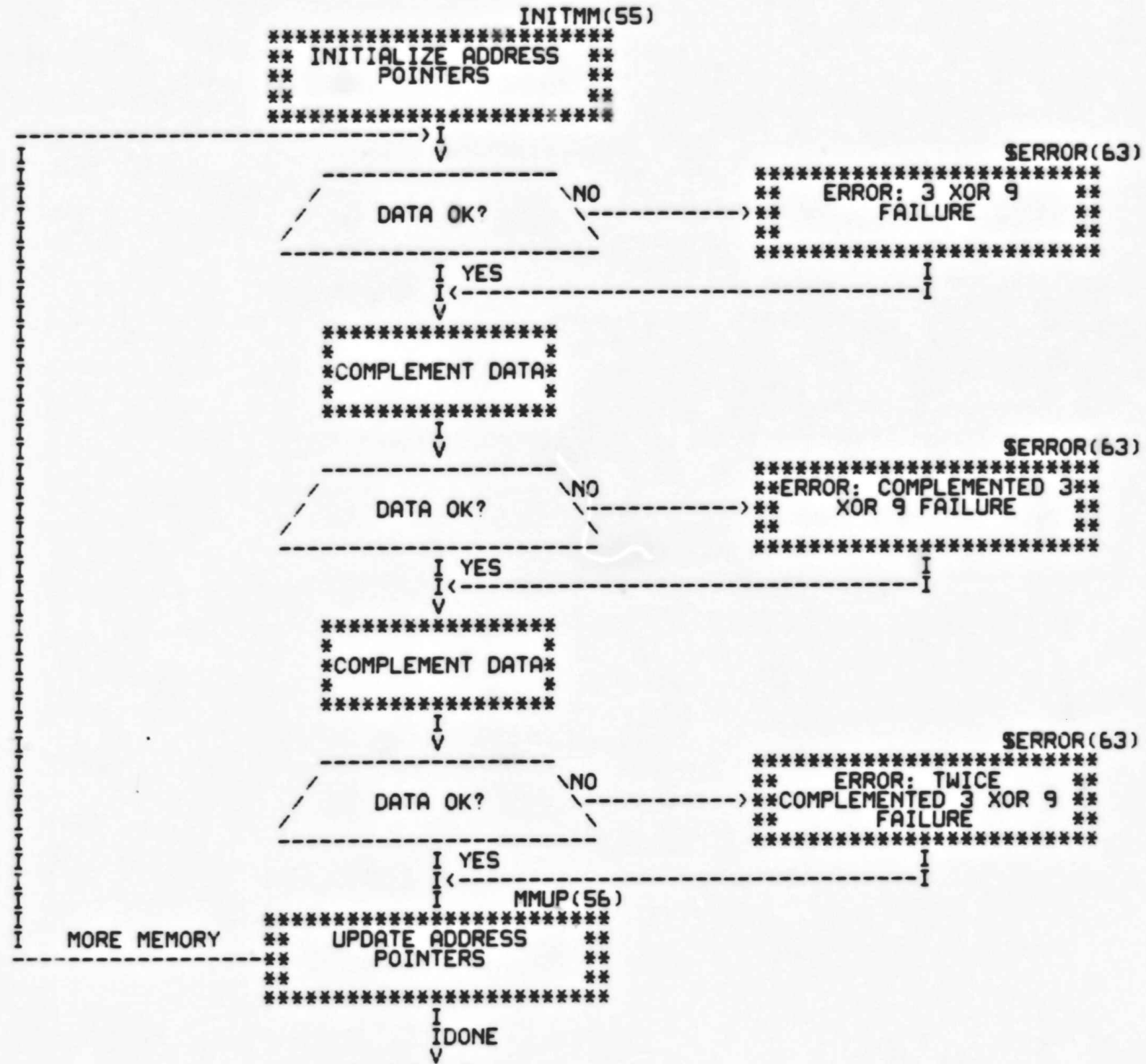




SERROR(63)

```
*****
** ERROR: ROTATING 1 **
**      FAILED        **
**                               **
*****
```



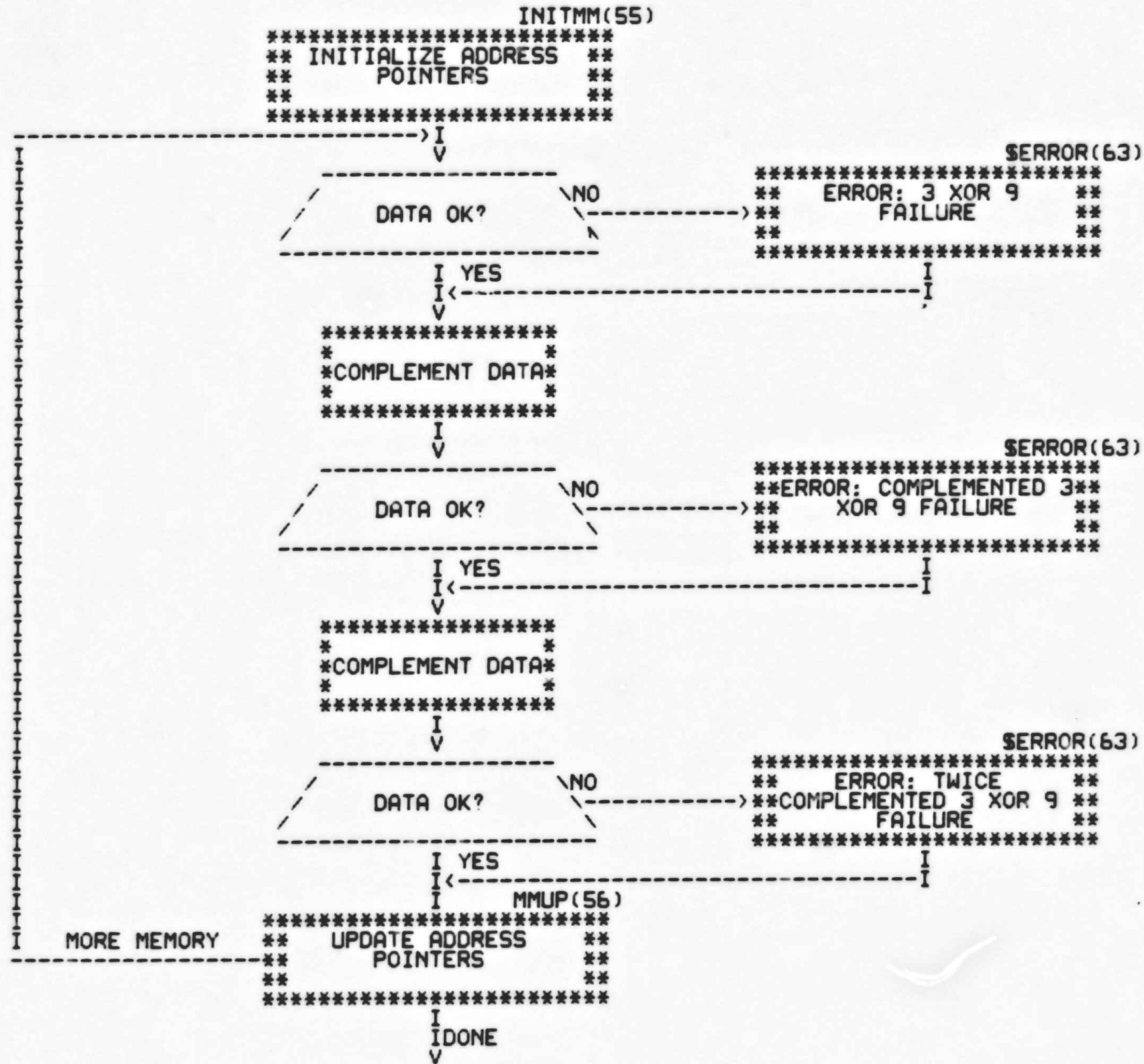


```

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

```

\*\*\*\*\* SERROR(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
          | V
  
```

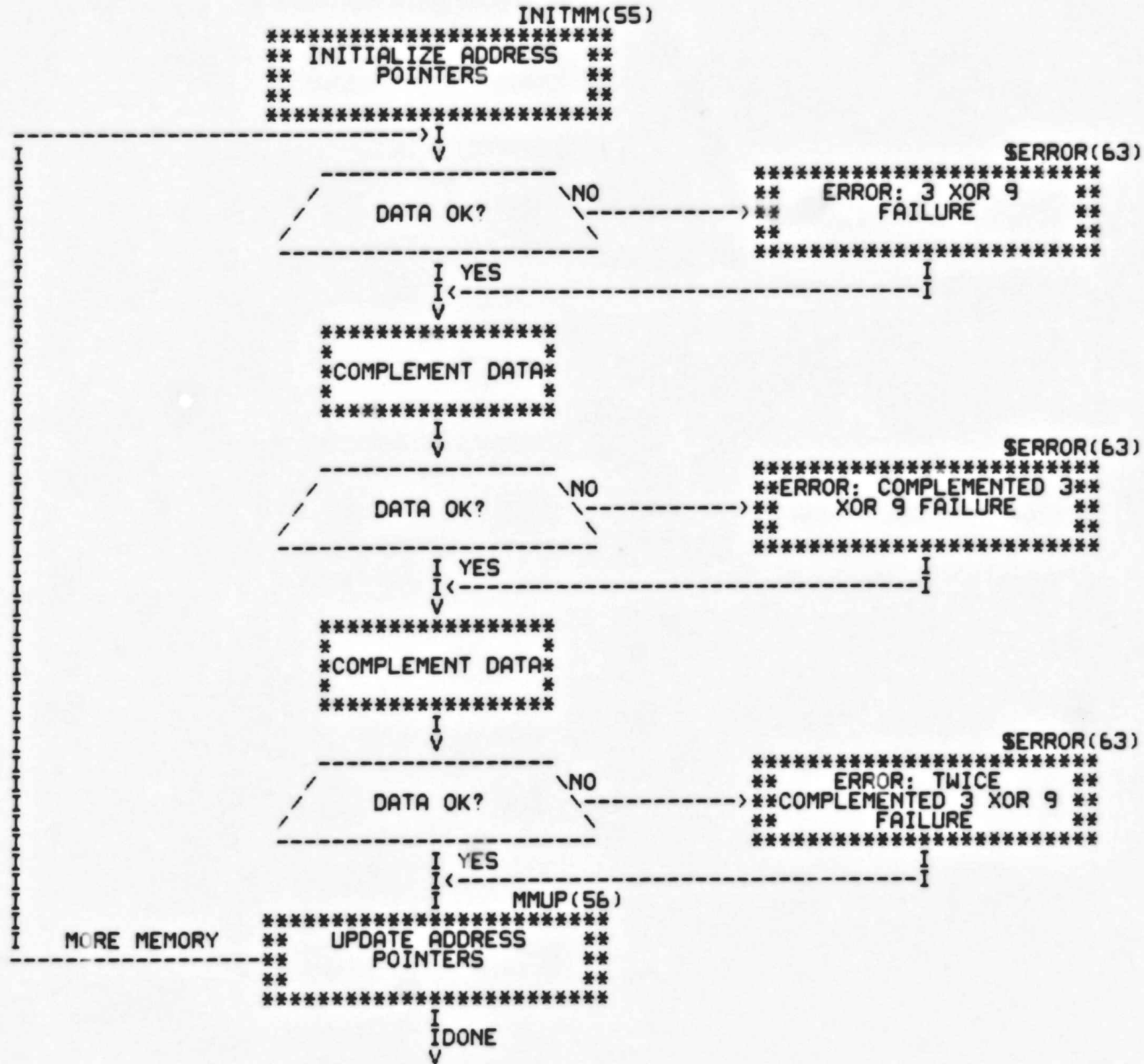
SERROR(63)

\*\*\*\*\*

\*\* ERROR: 3 XOR 9 \*\*

\*\* PATTERN FAILURE \*\*

\*\*\*\*\*



```

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

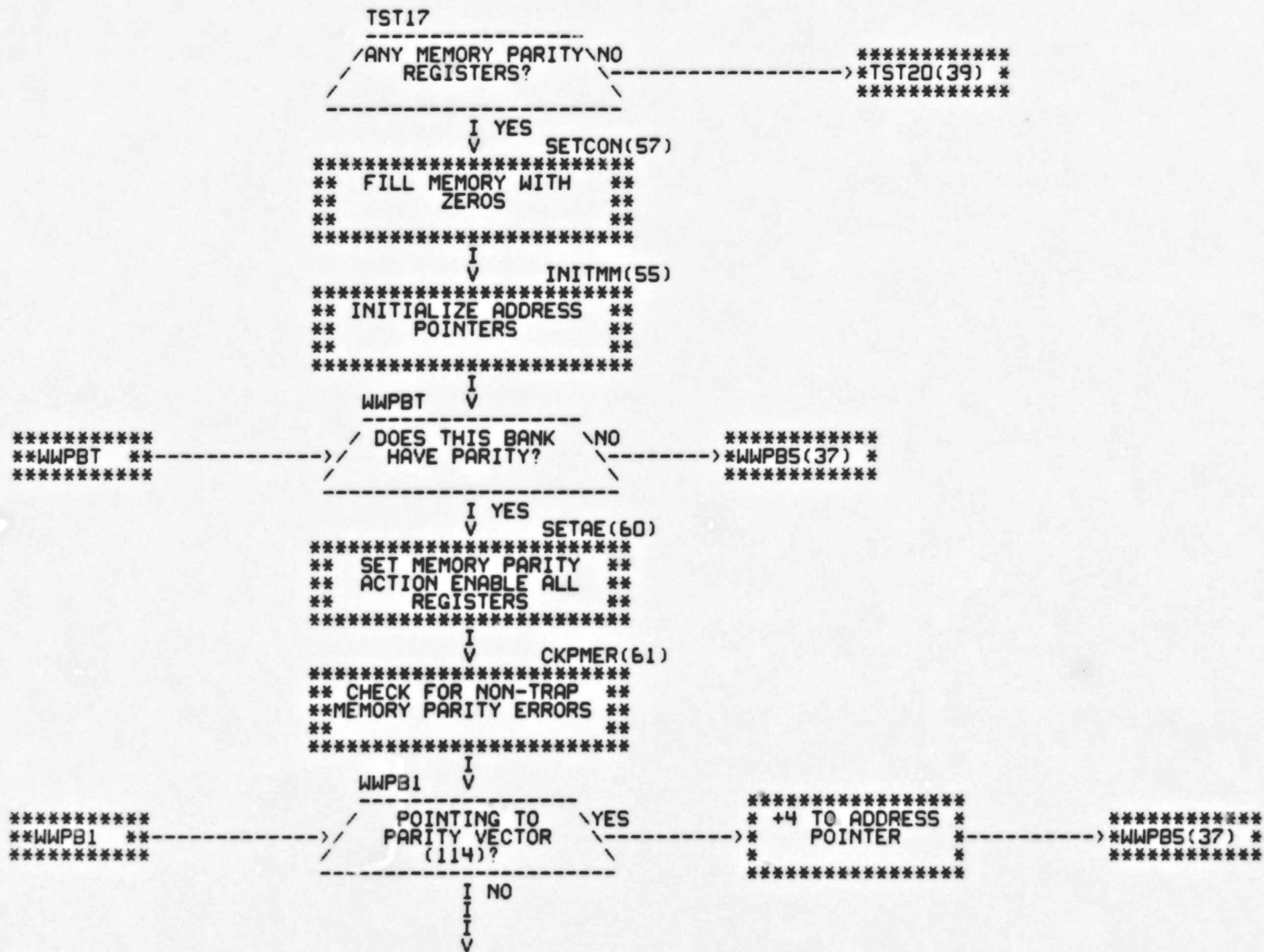
```

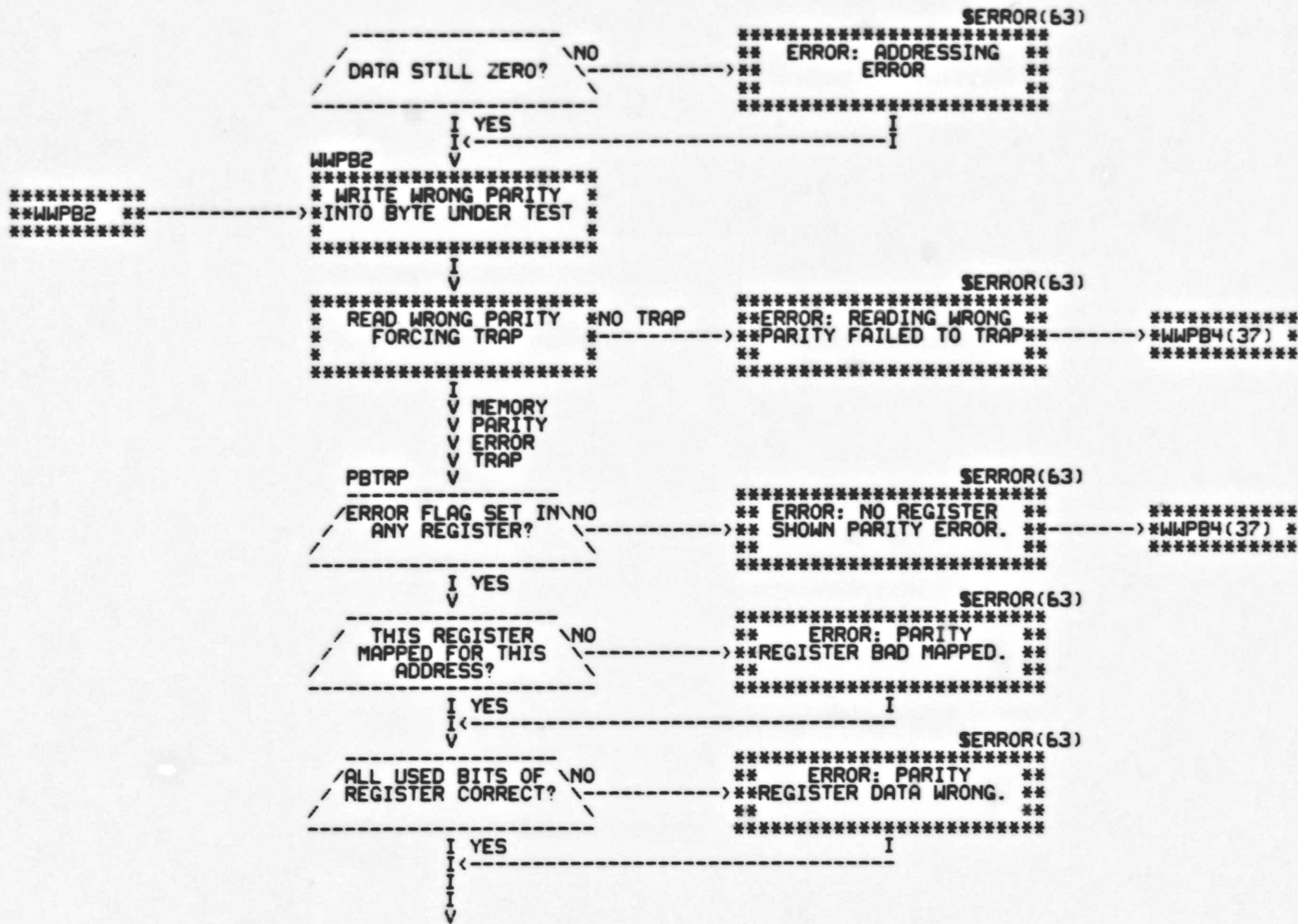
```

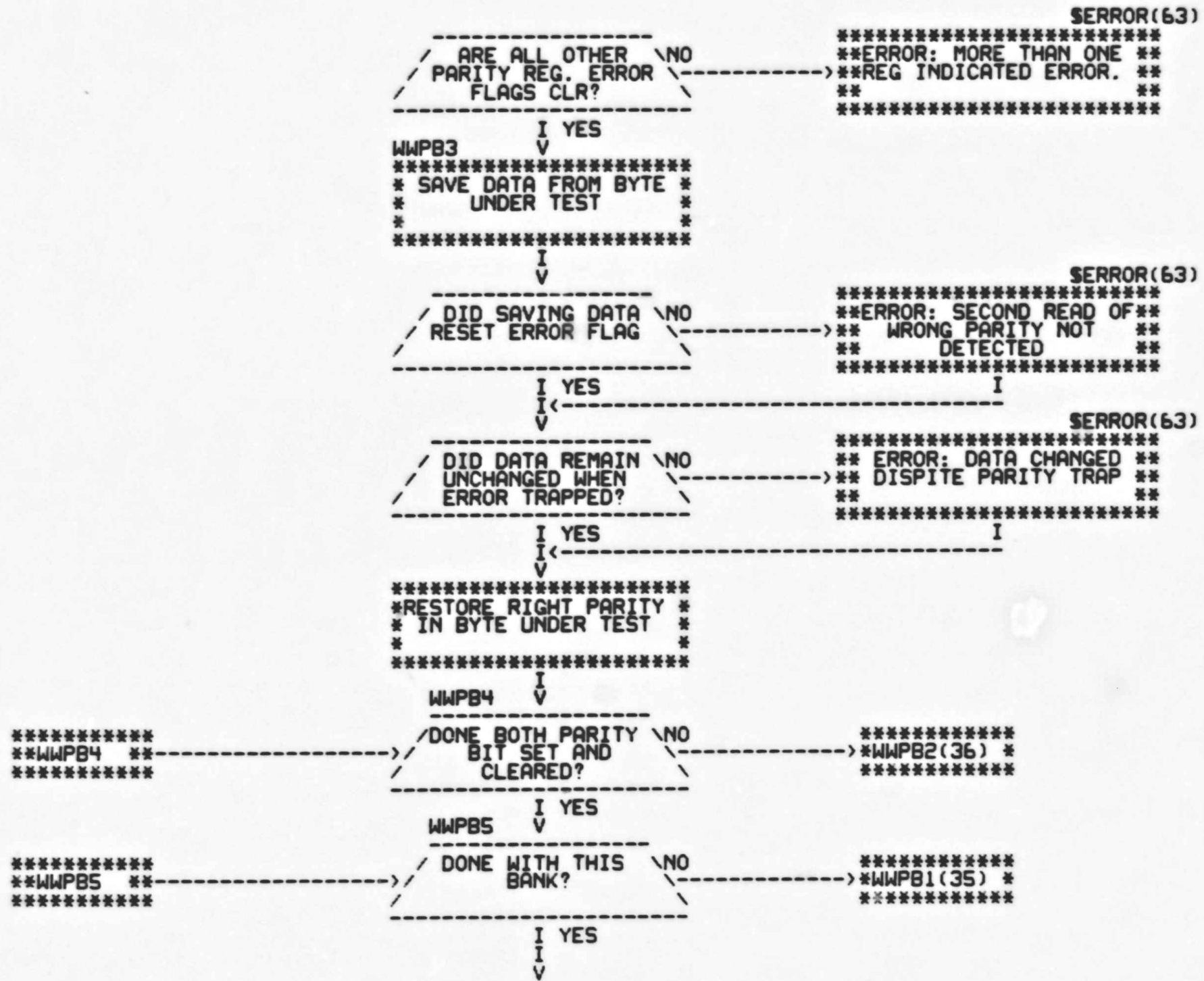
*****
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
*****

```







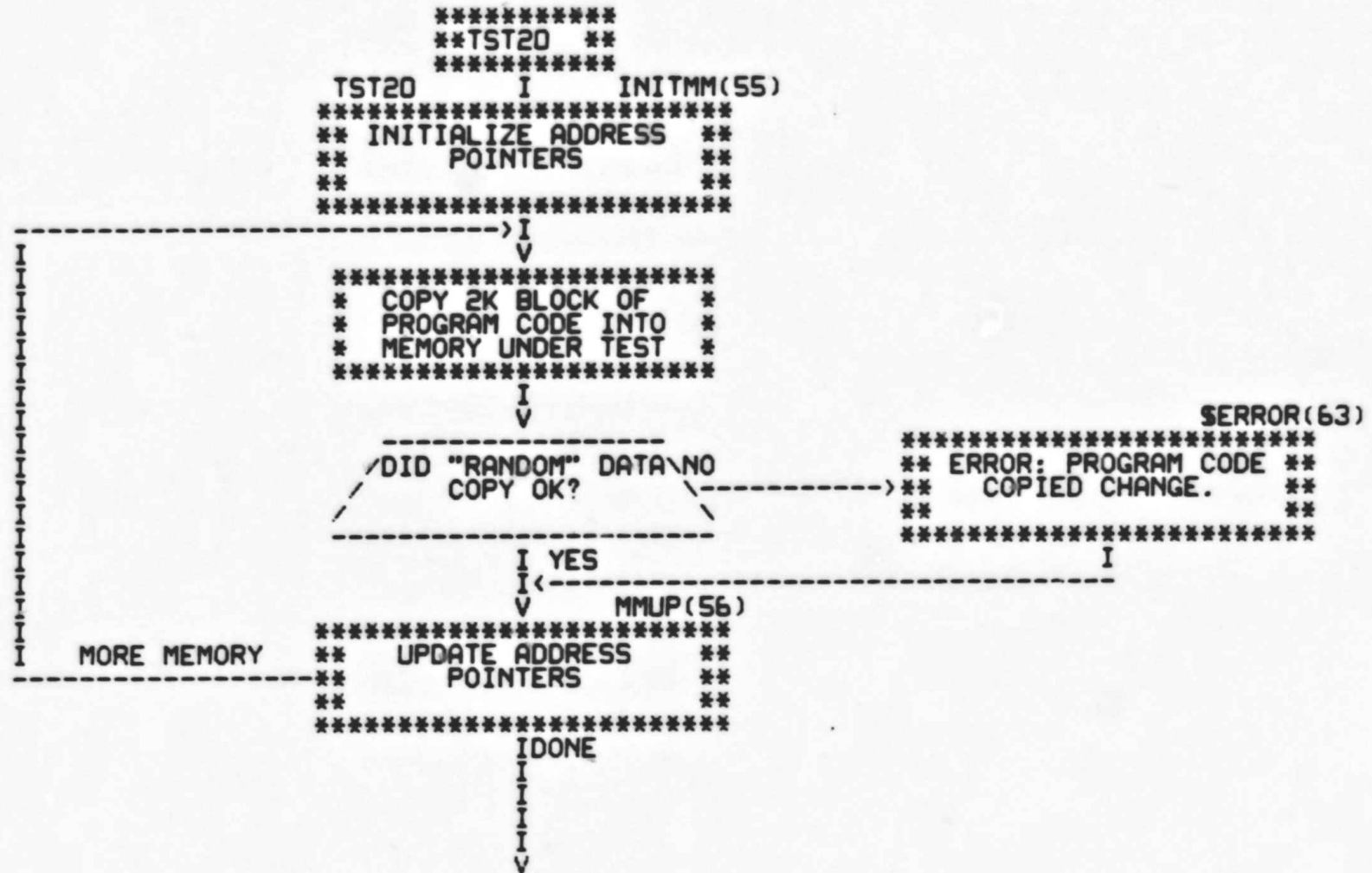


```

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

```

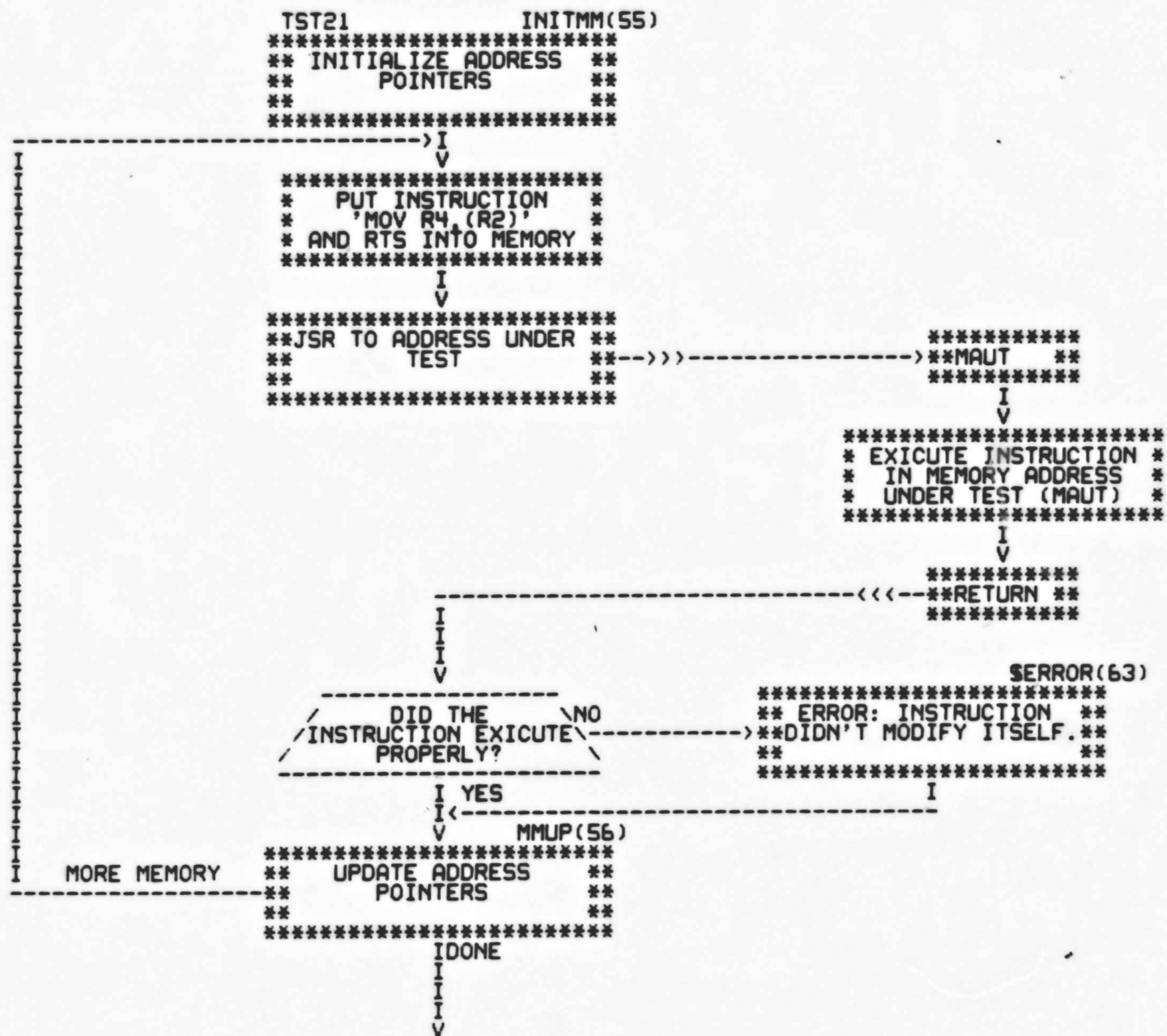




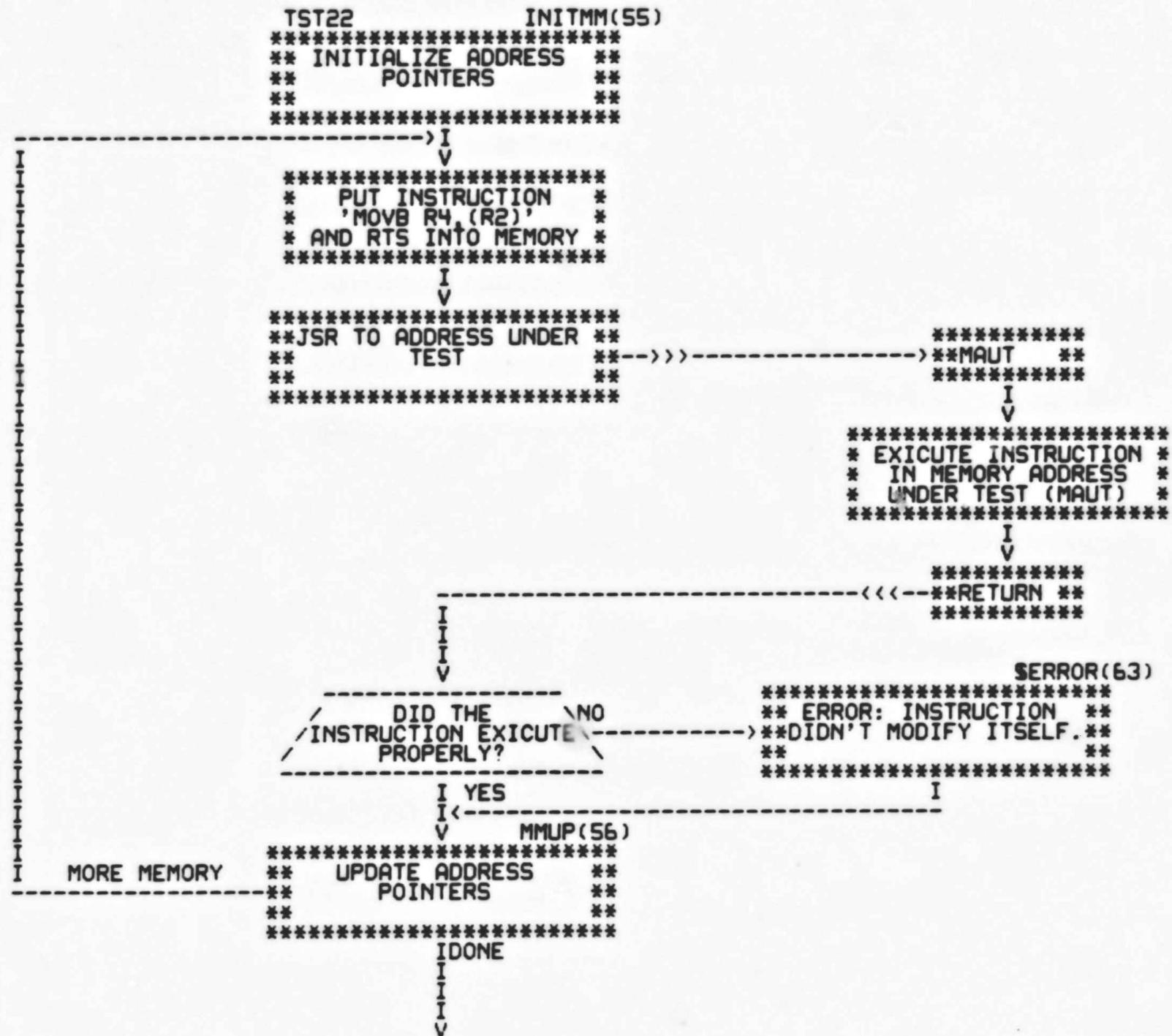
SERROR(63)

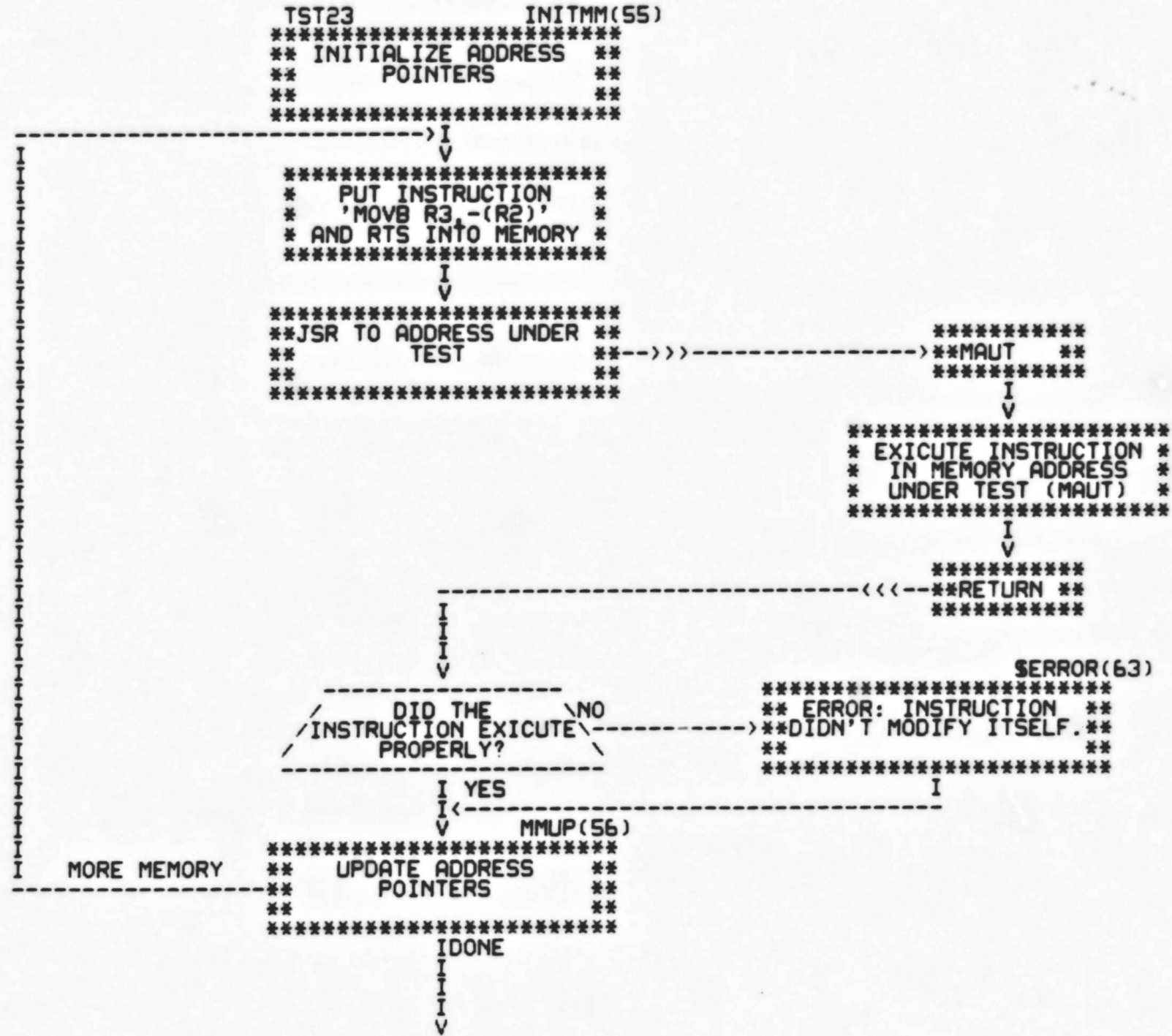
\*\*\*\*\*  
 \*\* ERROR: PROGRAM CODE \*\*  
 \*\* COPIED CHANGE. \*\*  
 \*\*\*\*\*

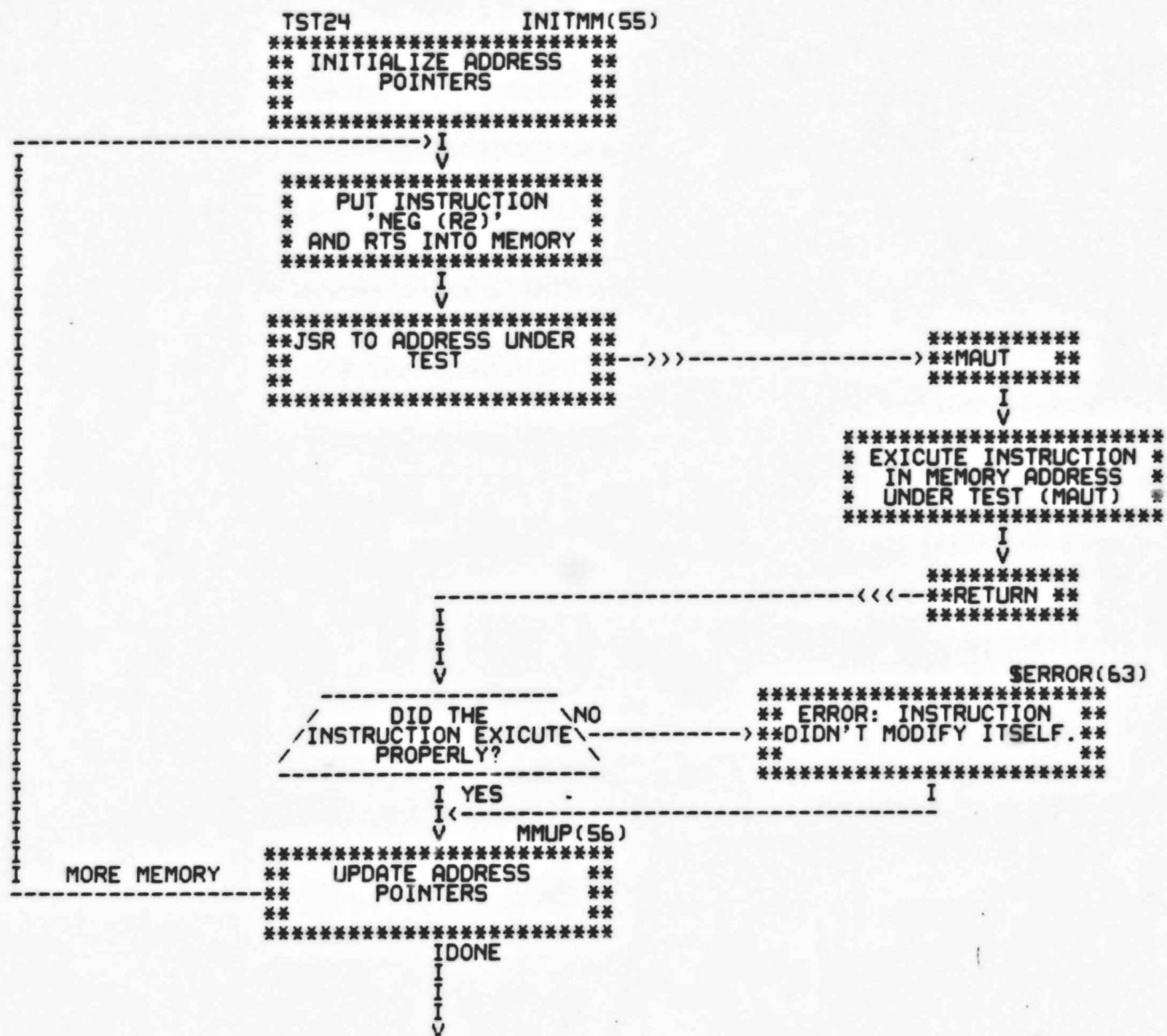
CZQMCE0 0-124K MEM EXER 16K  
TEST 21: EXICUTE DATI, DATO

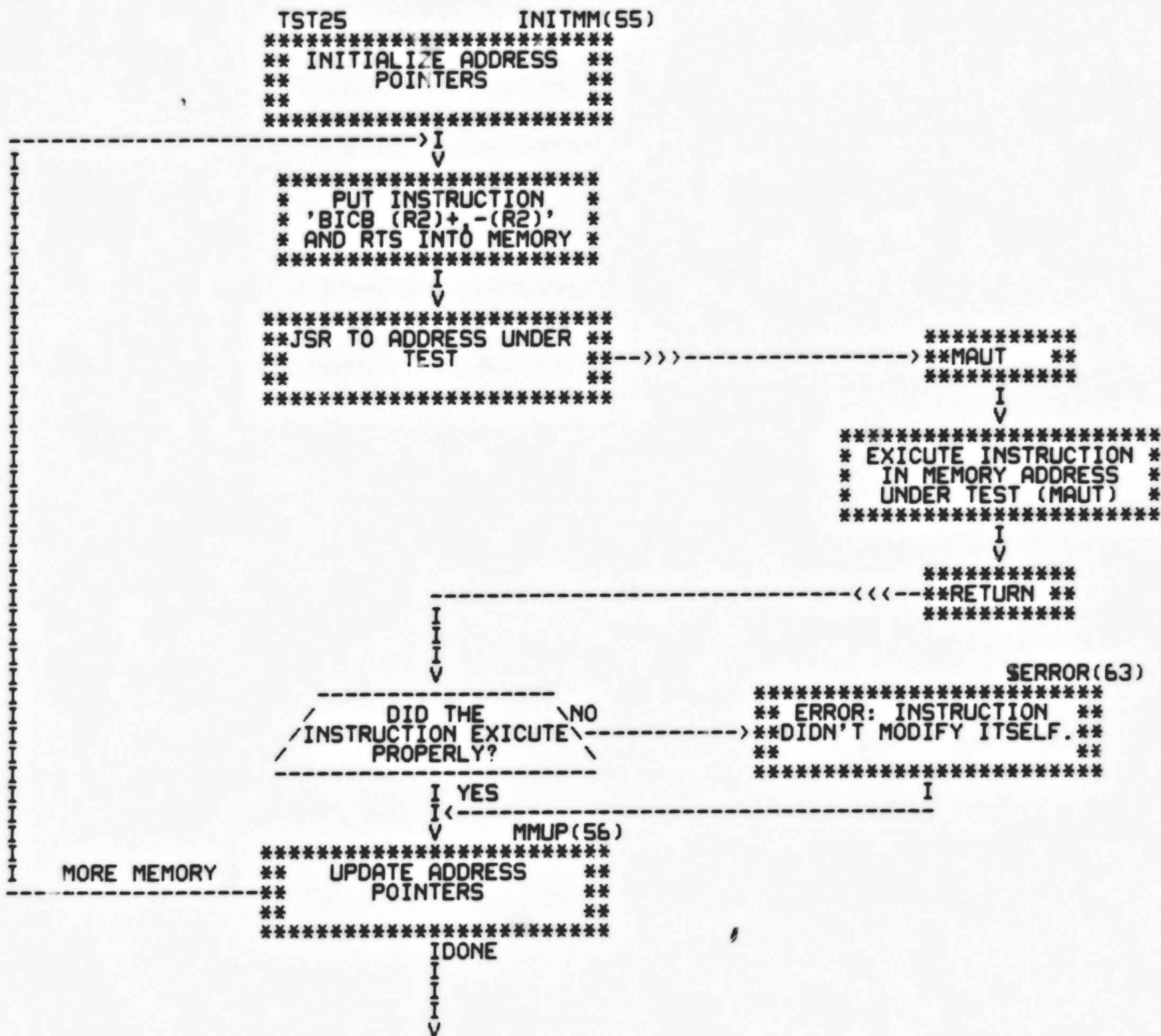


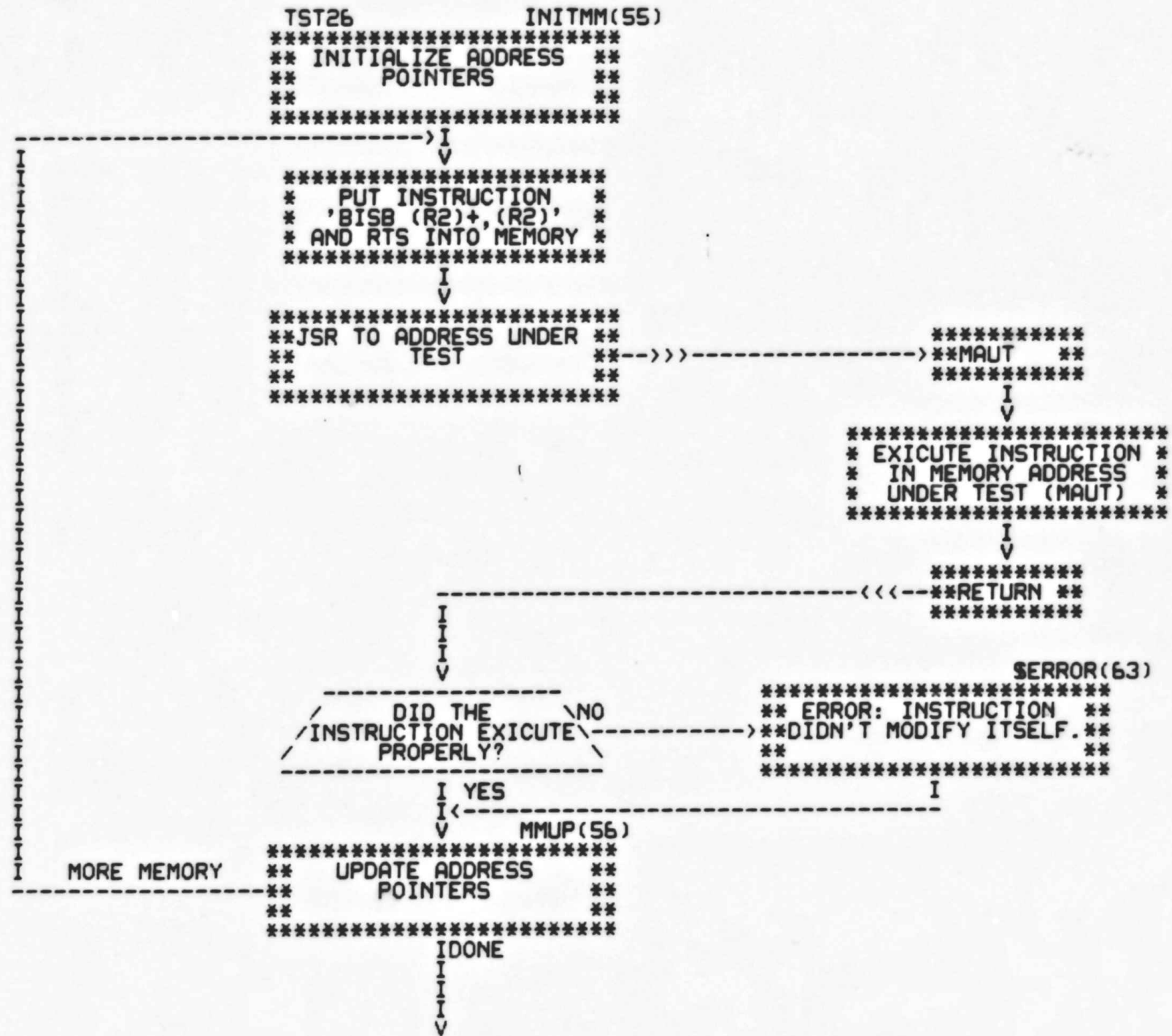
CZQMCEO 0-124K MEM EXER 16K  
TEST 22: EXICUTE DATI, DATOB (LO BYTE)

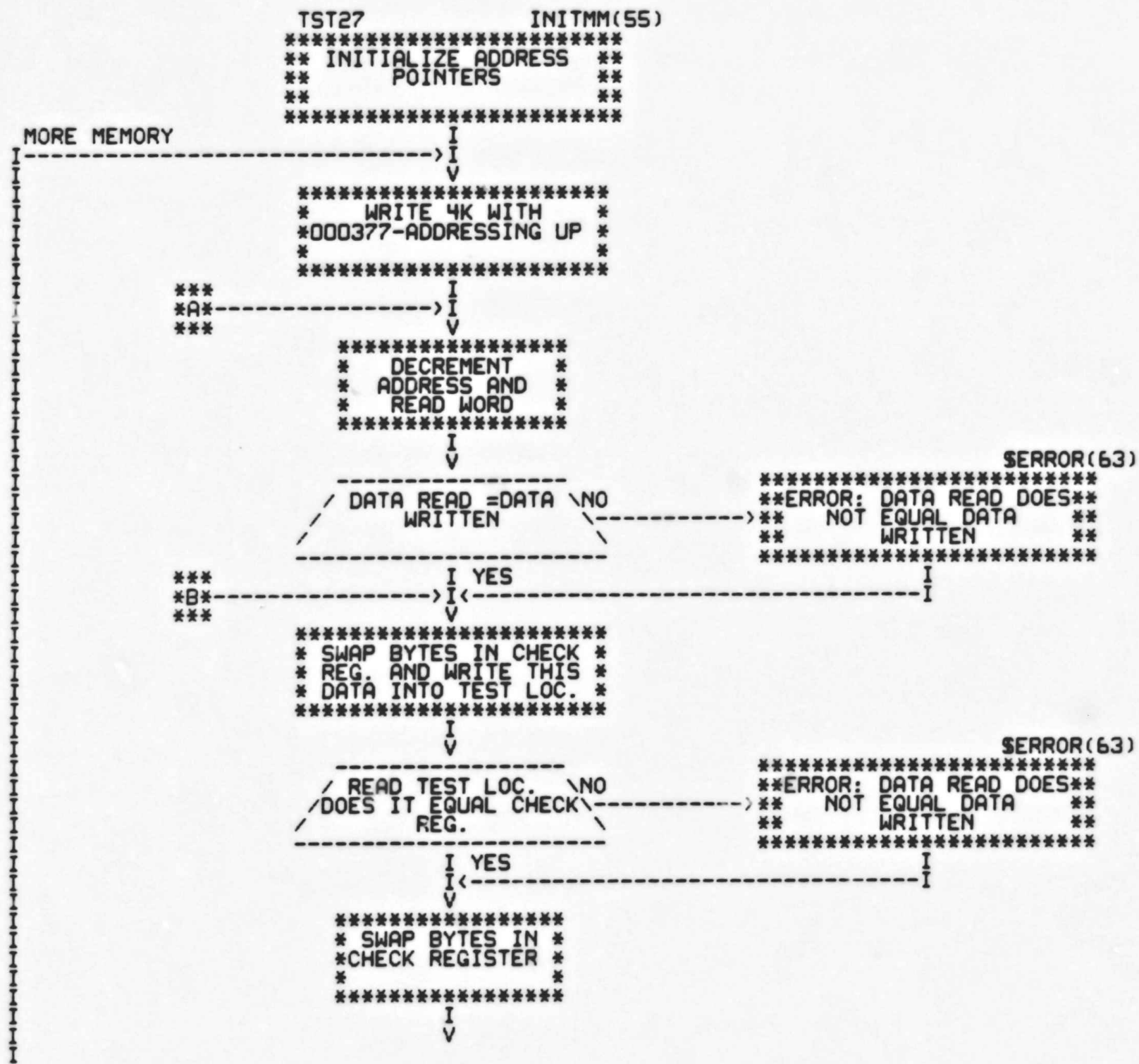




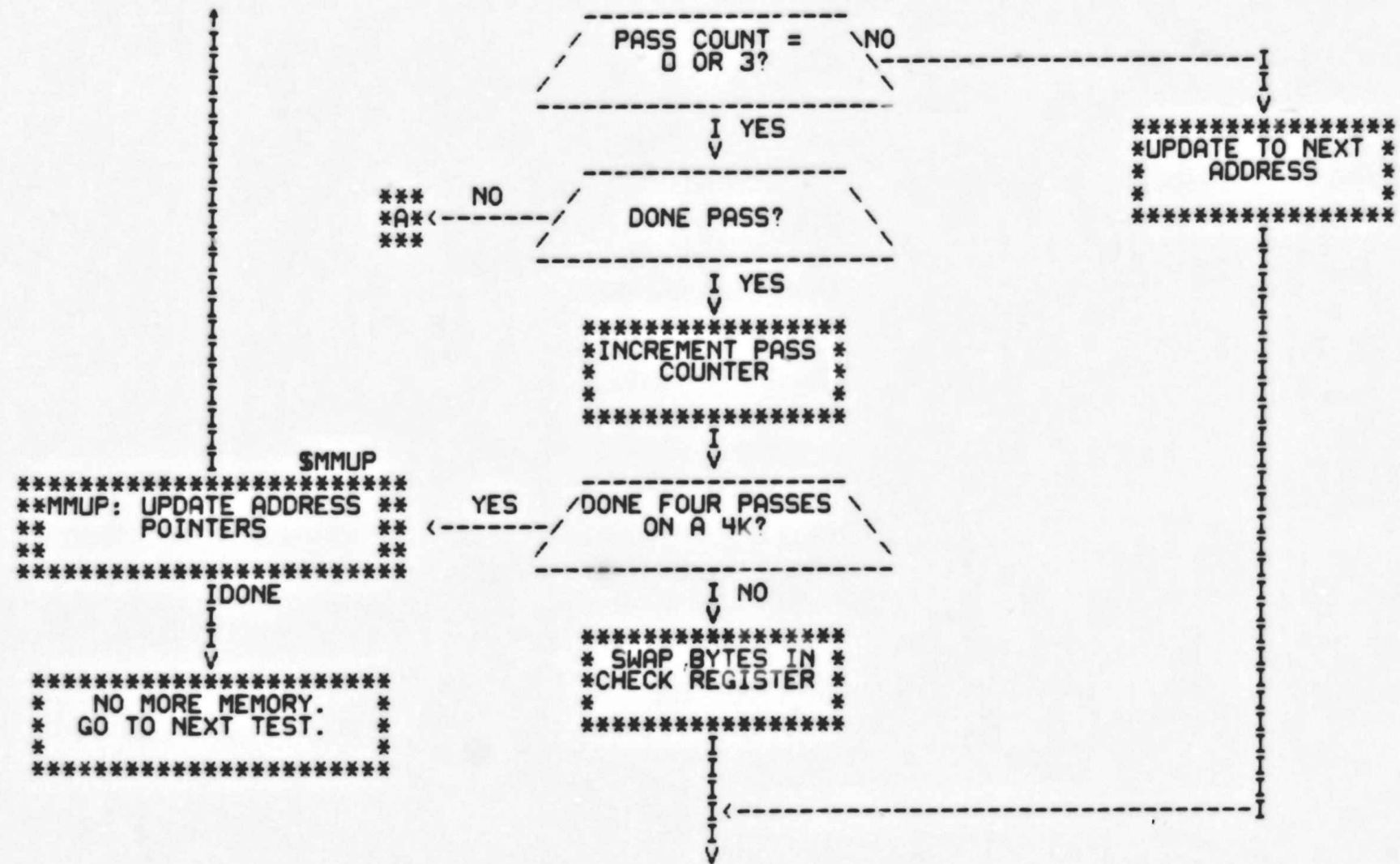


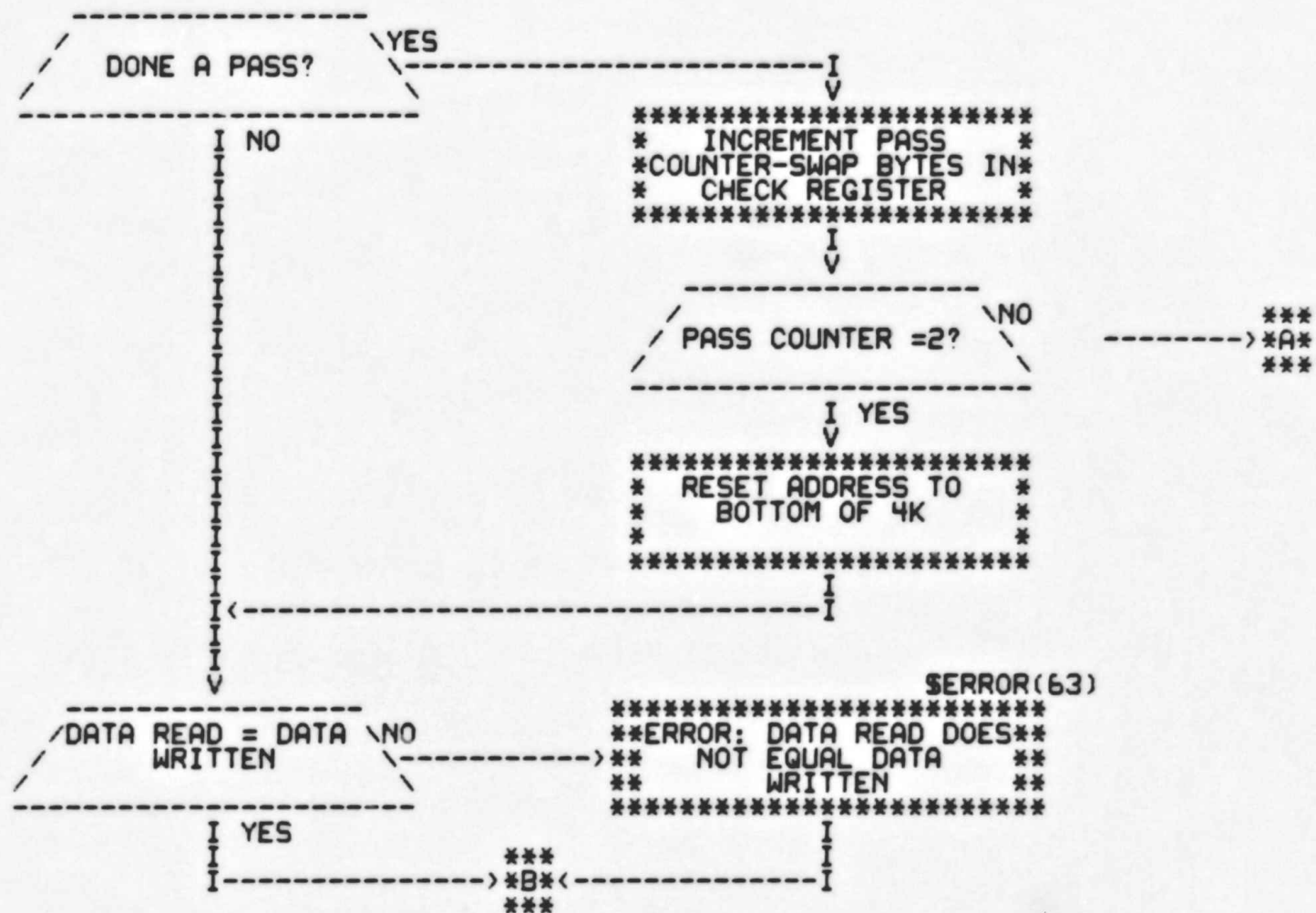


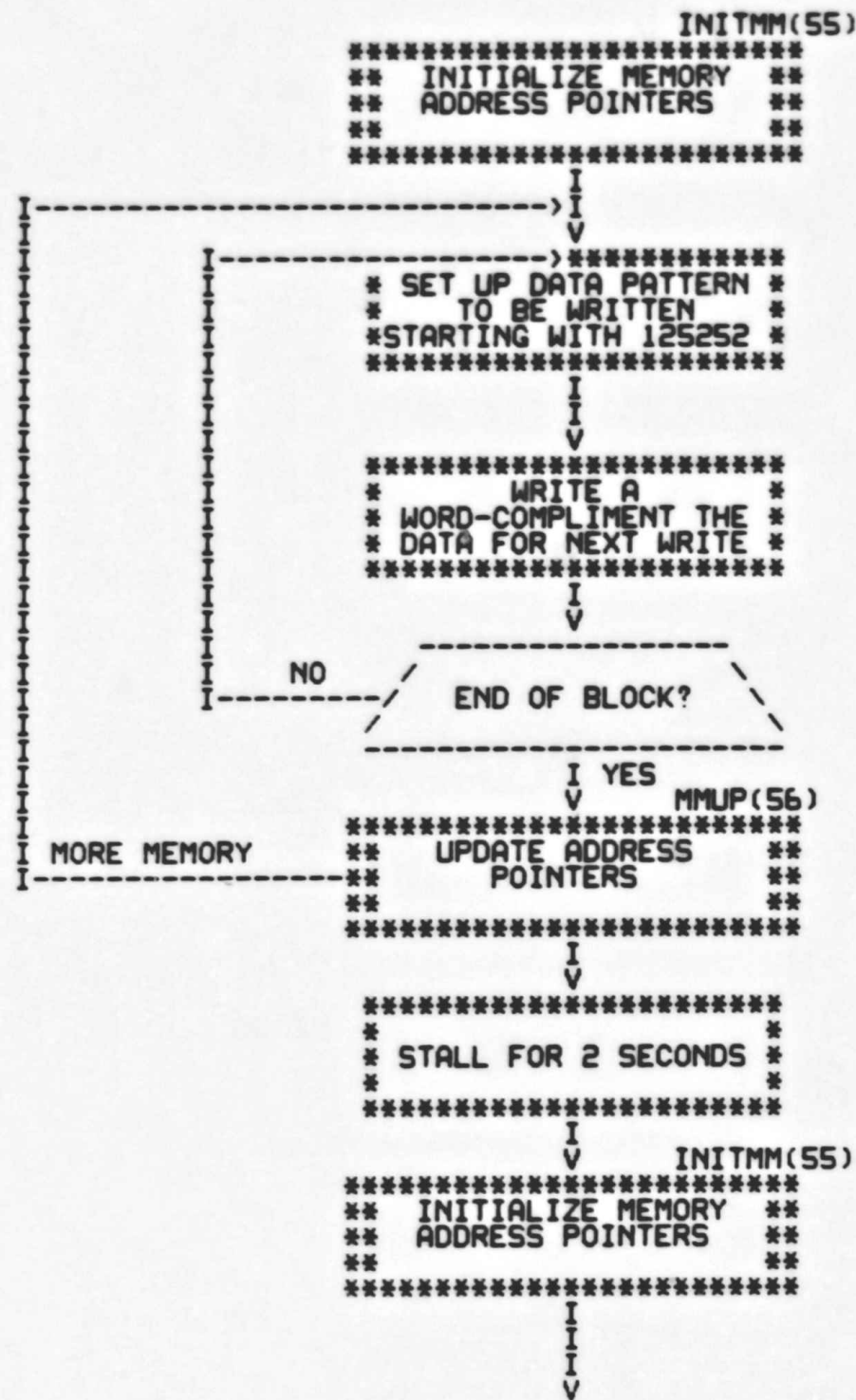


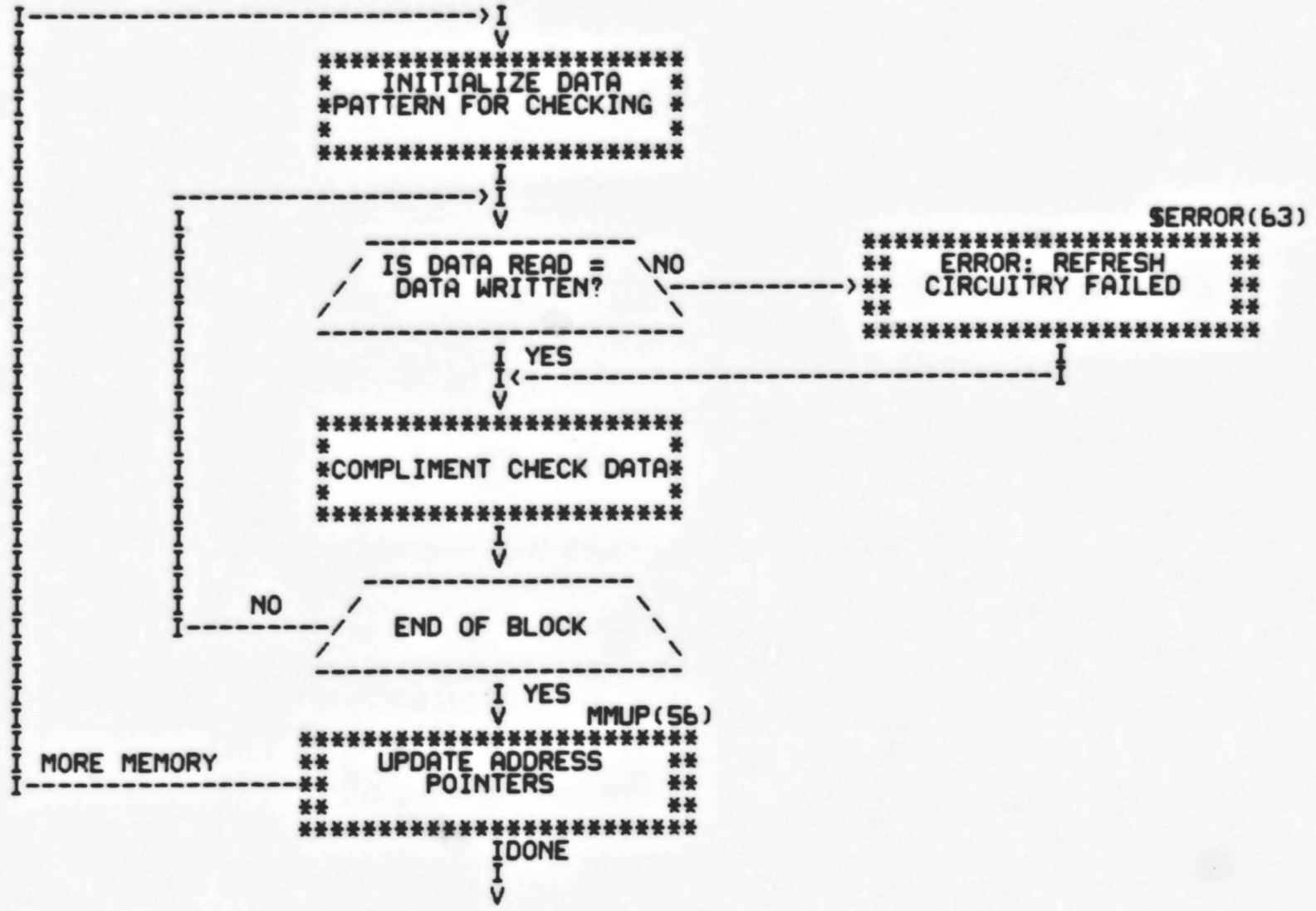


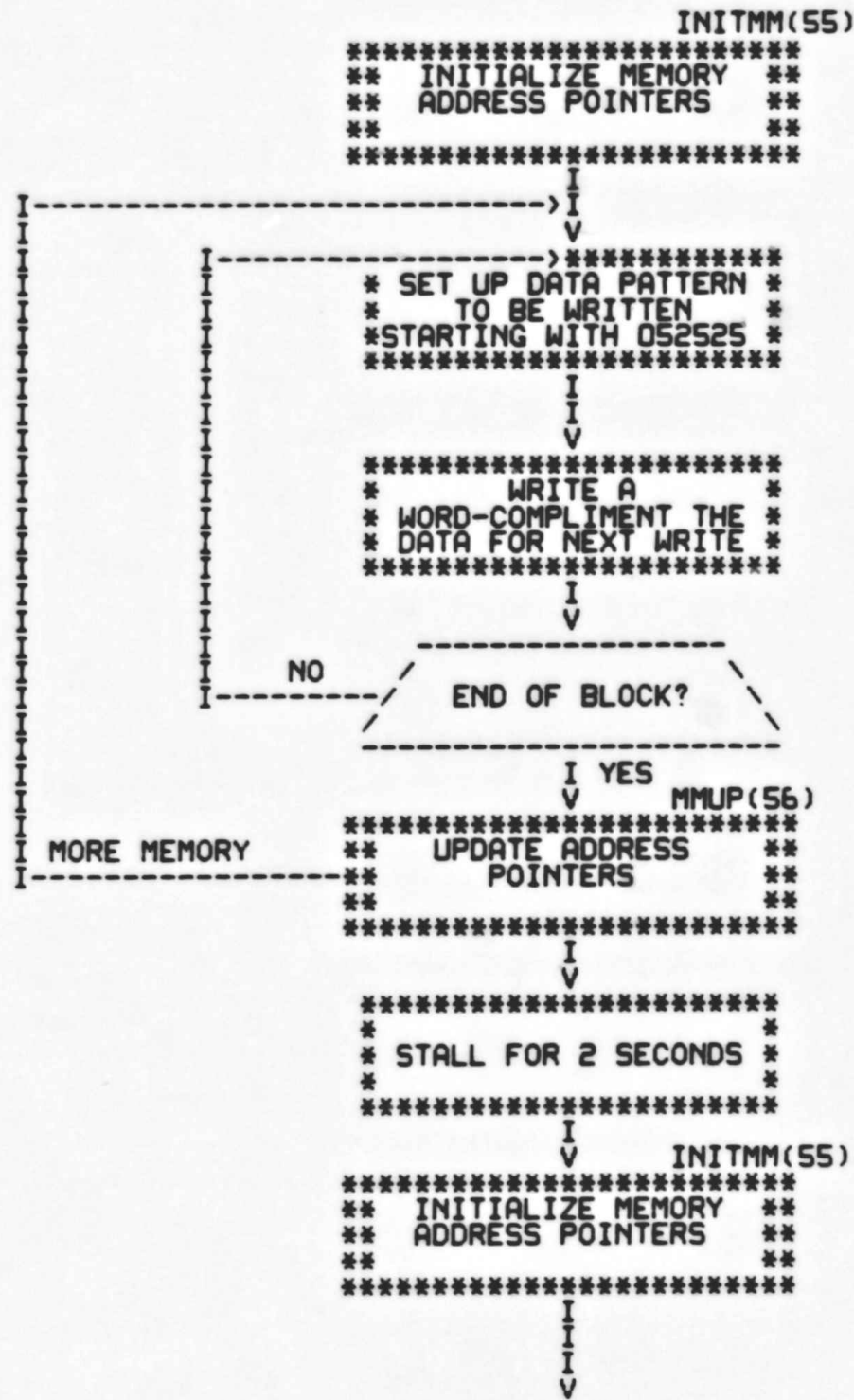




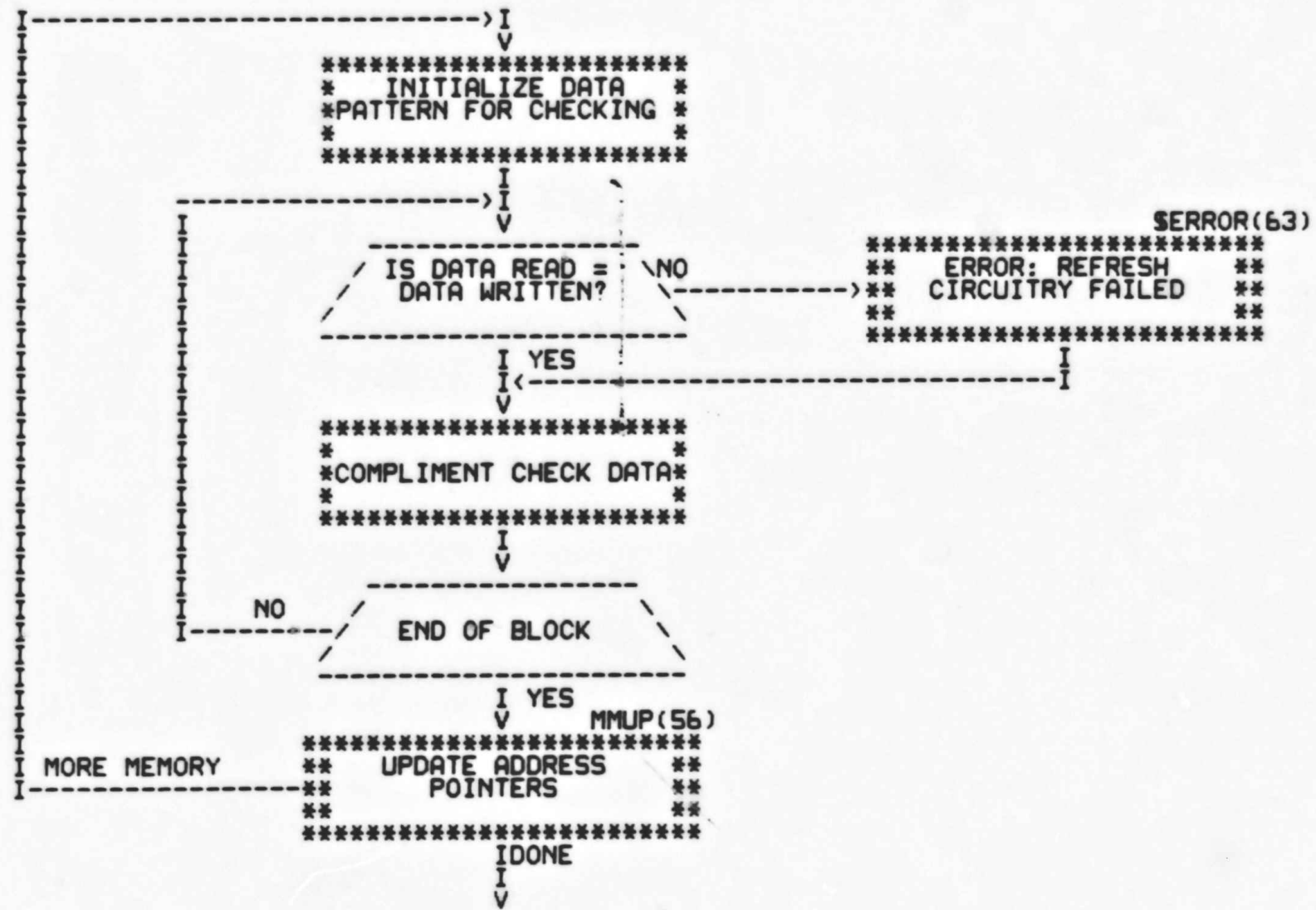


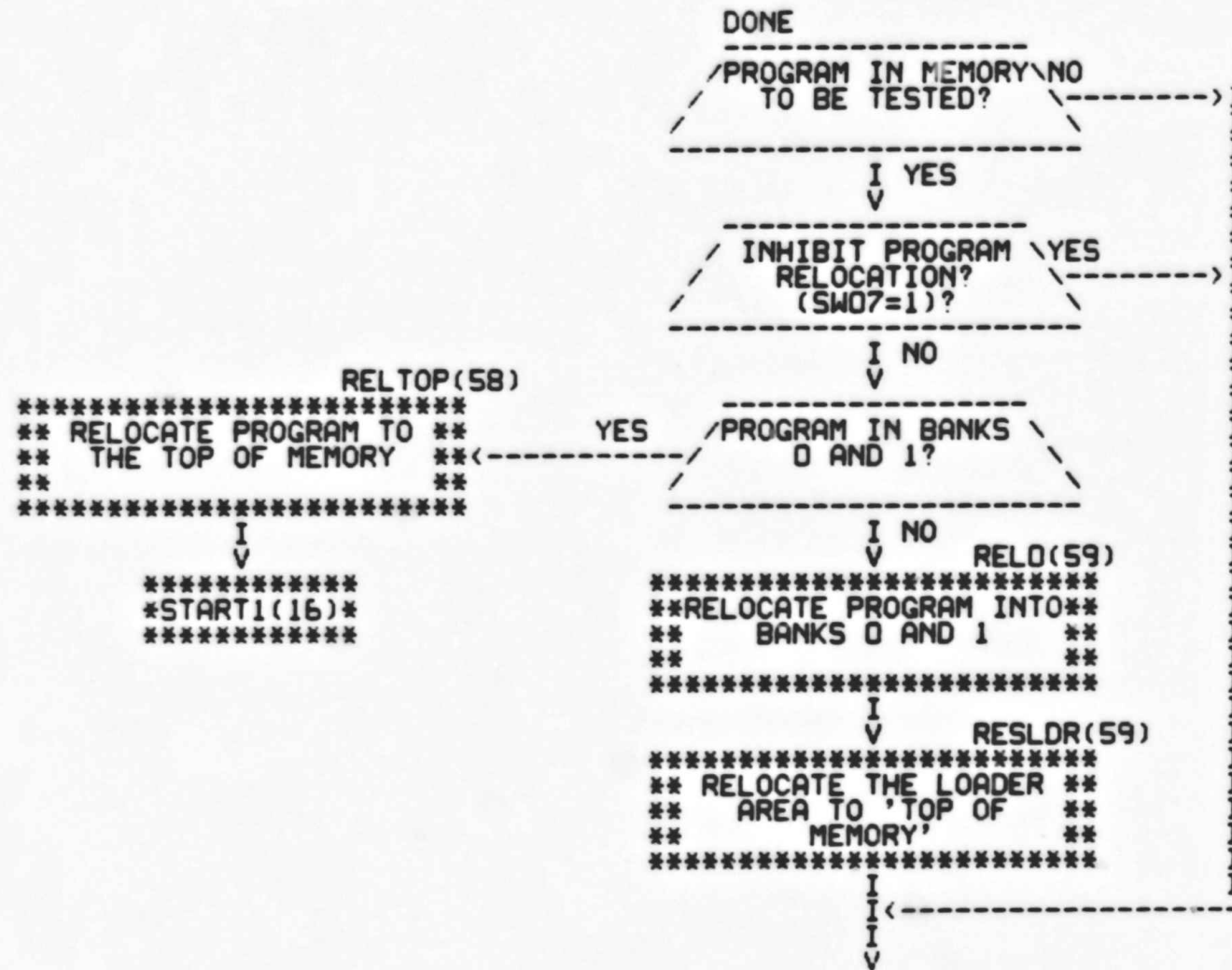


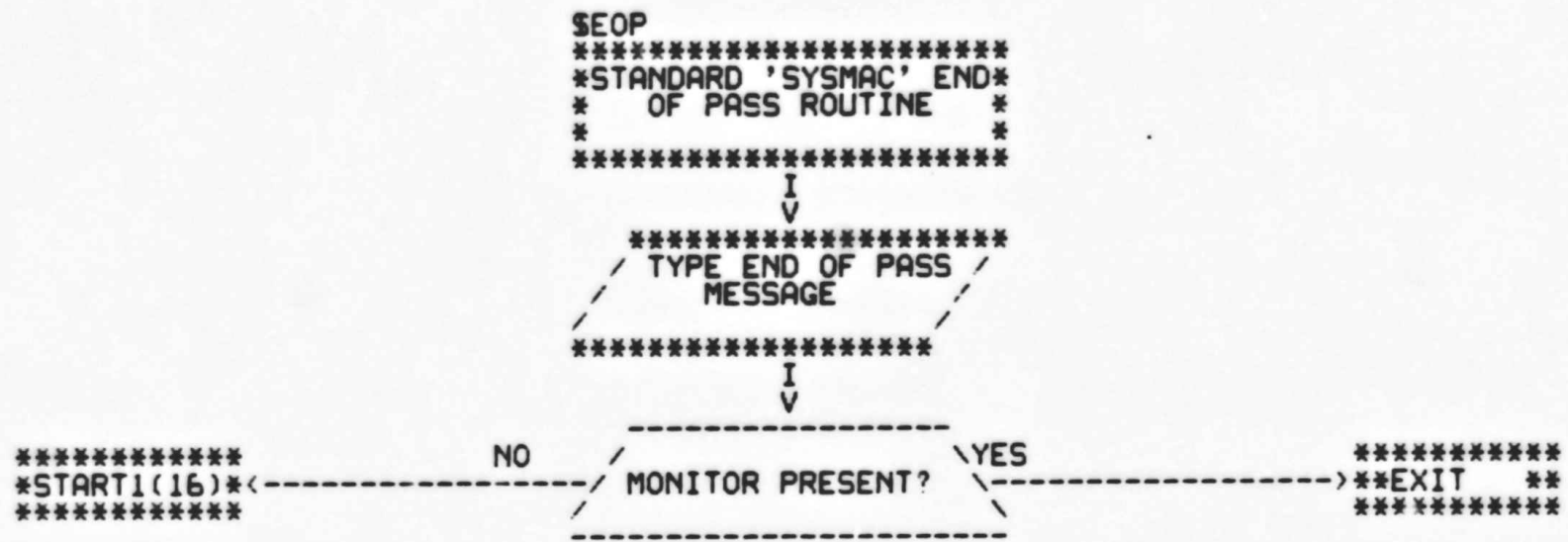




CZQMCEO 0-124K MEM EXER 16K  
TEST 31: MOS REFRESH TEST





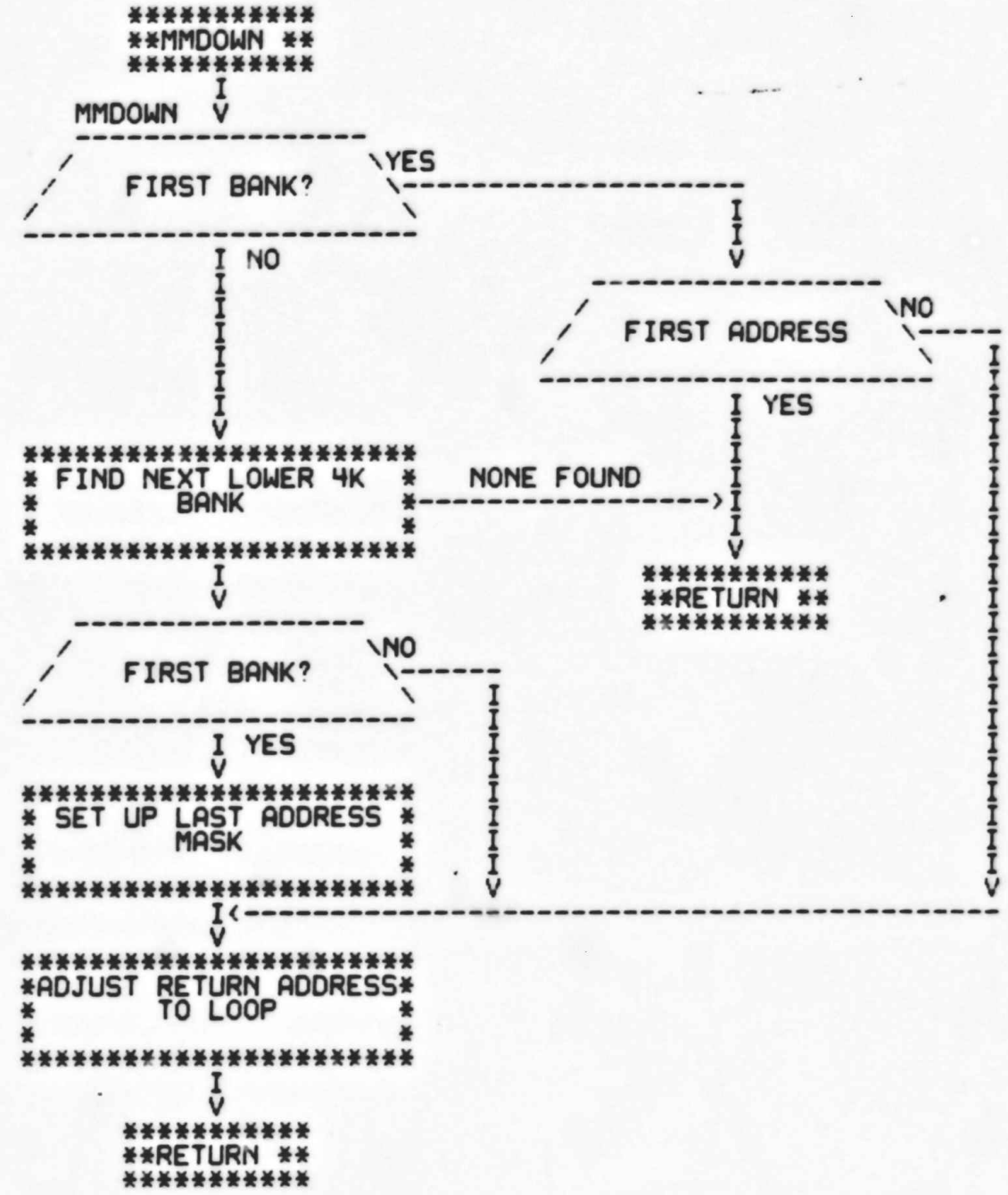
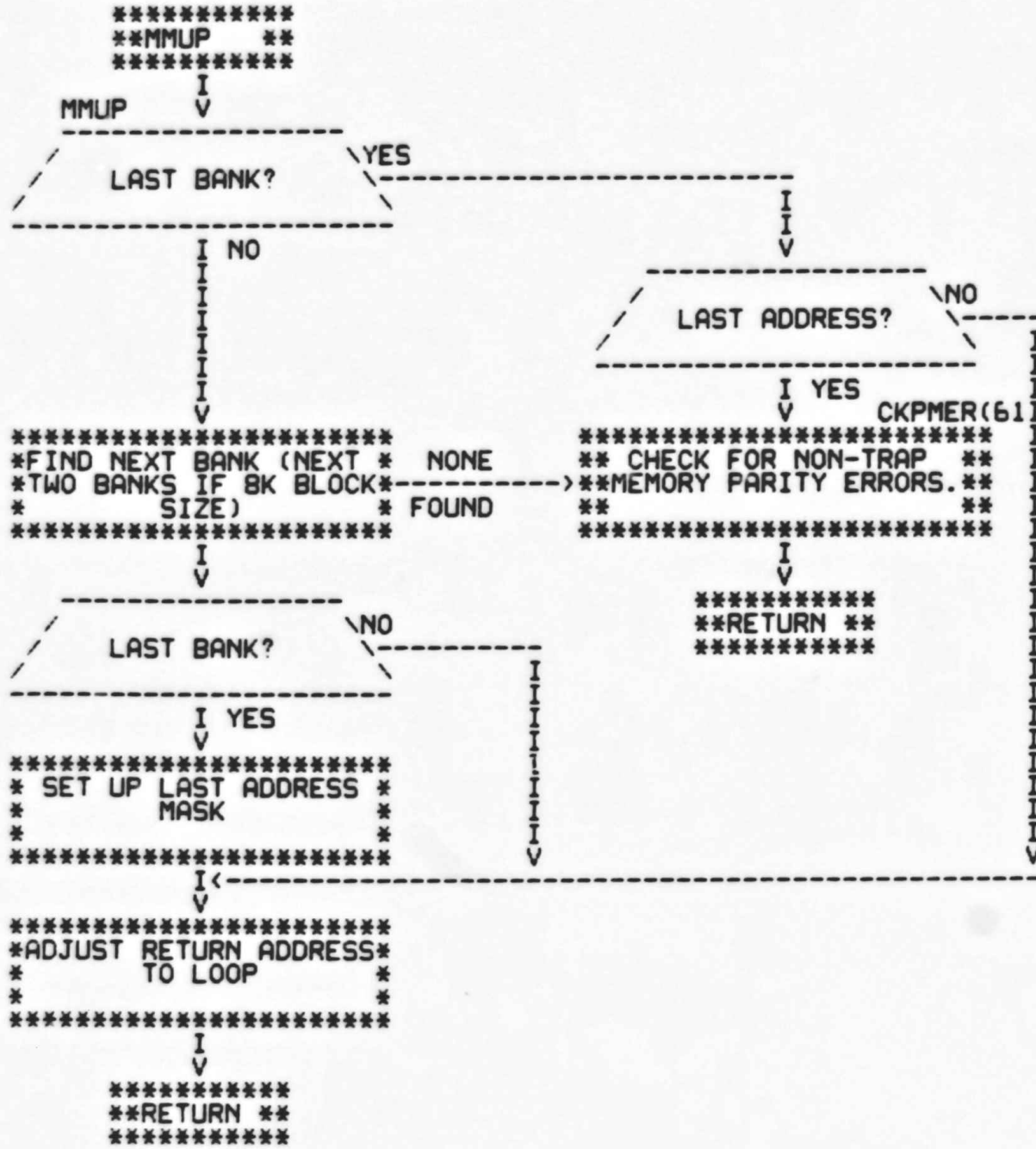




```
*****  
**MMINIT **  
*****  
MMINIT  
*****  
*SET UP ALL THE MEMORY*  
*MANAGEMENT REGISTERS.*  
*  
*****  
*****  
**RETURN **  
*****
```

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

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



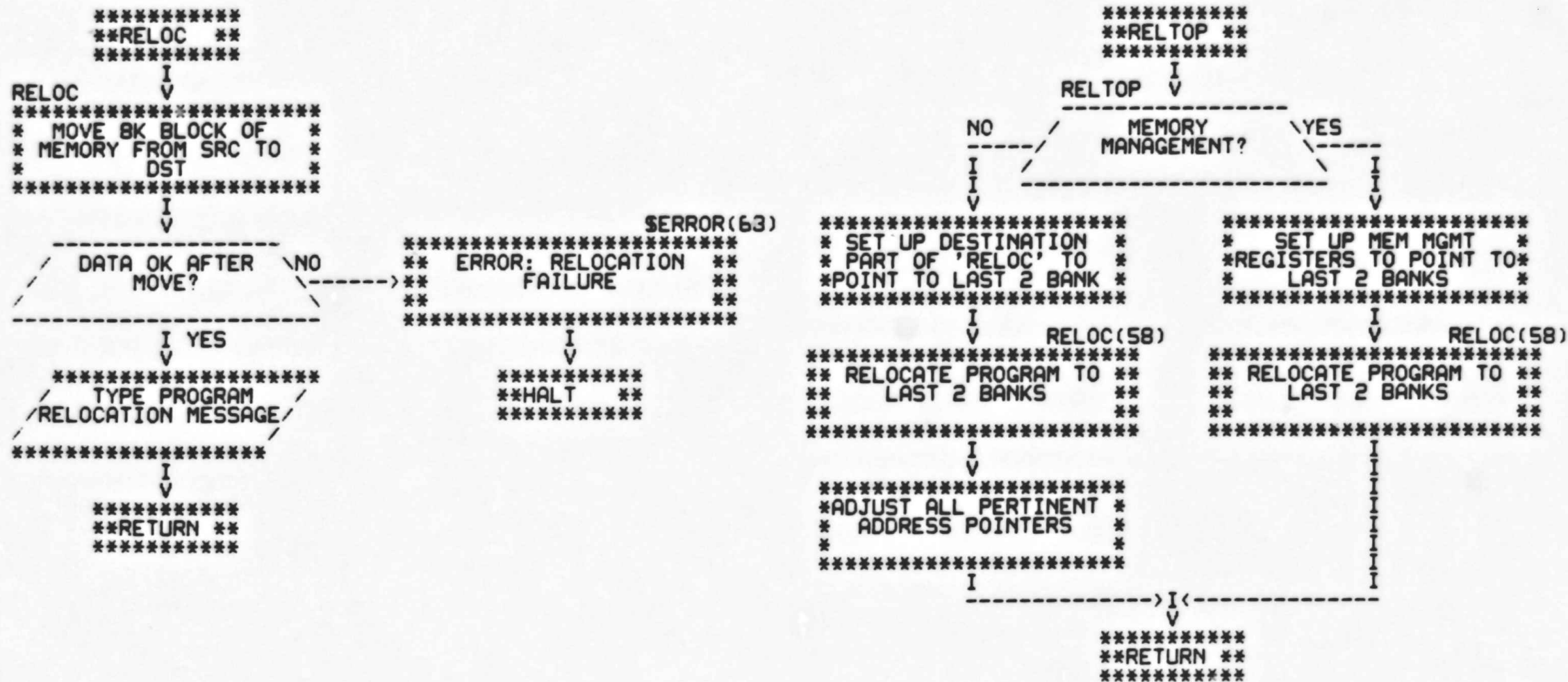
```
*****  
**PHYADR **  
*****  
PHYADR V  
*****  
* GET VIRTUAL *  
* ADDRESS (FROM *  
* R2) *  
*****  
V  
-----  
MEMORY MANAGEMENT NO  
AVAILABLE  
-----  
V YES  
*****  
* ADD INDEX FACTOR FROM *  
* KIPAR2 TO GET *  
* PHYSICAL ADR *  
*****  
V  
-----  
*****  
**RETURN **  
*****
```

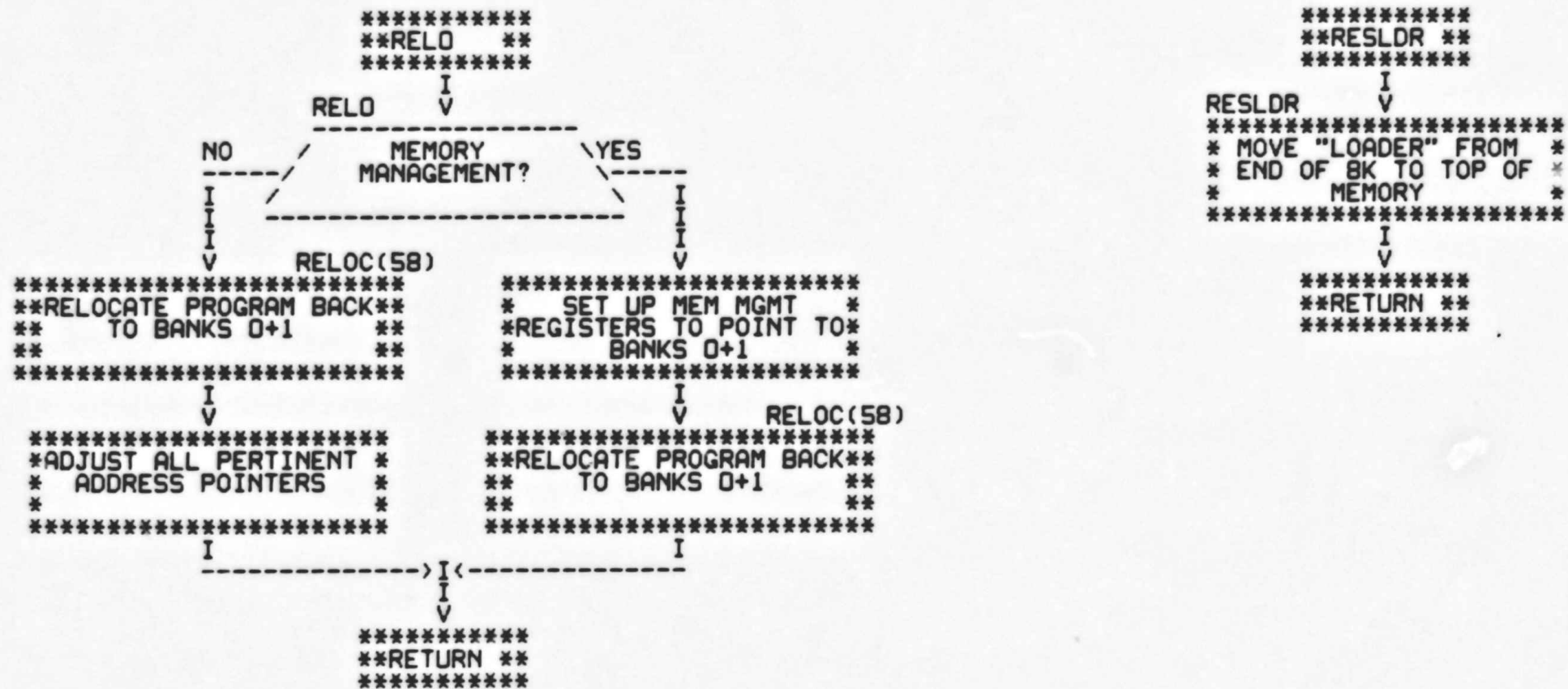
```
*****  
**BANKNO **  
*****  
BANKNO V  
*****  
* CALCULATE BANK # *  
* USING TEST MAP BANK *  
* POINTER *  
*****  
V  
*****  
**RETURN **  
*****
```

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

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

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

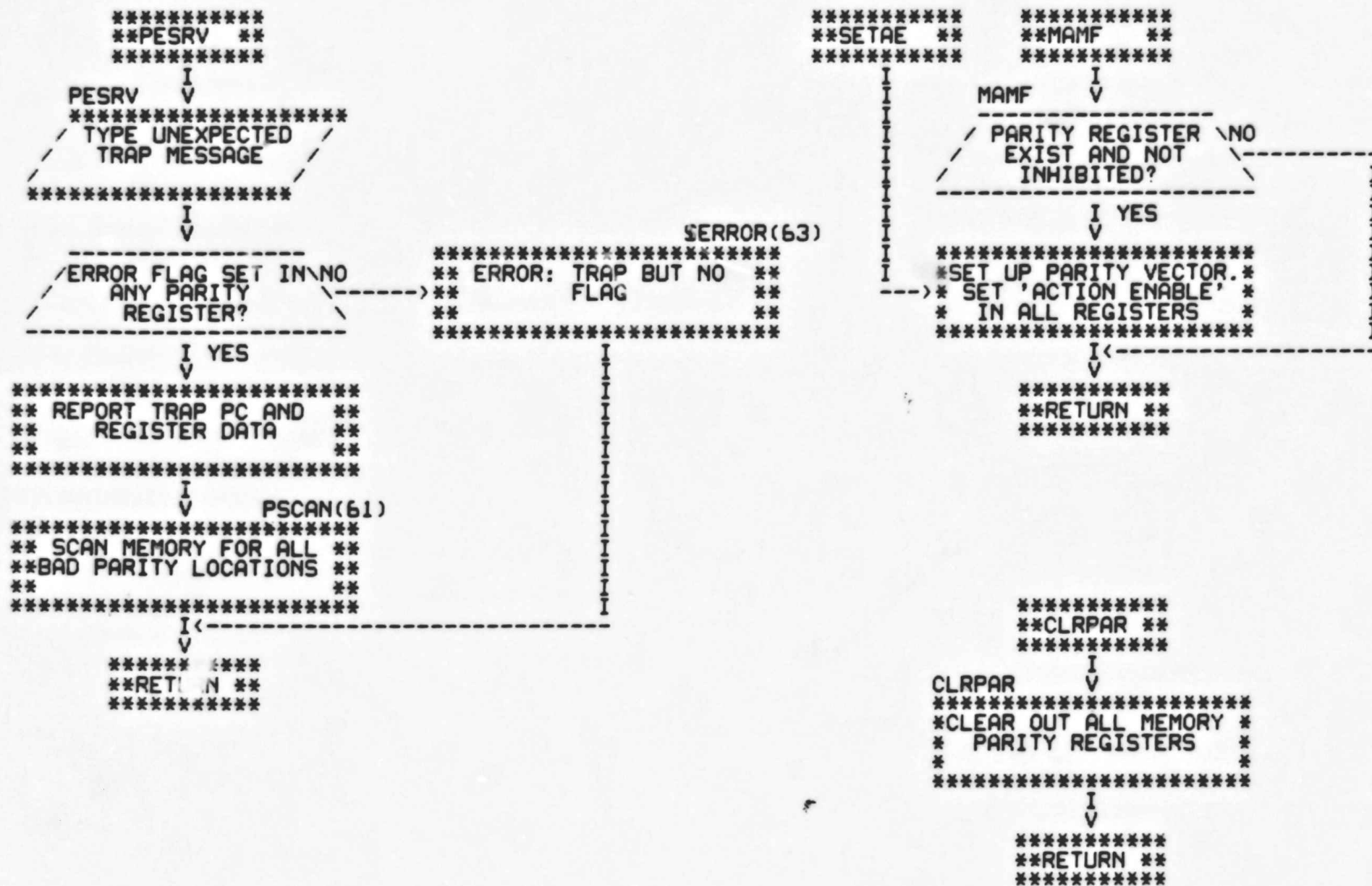


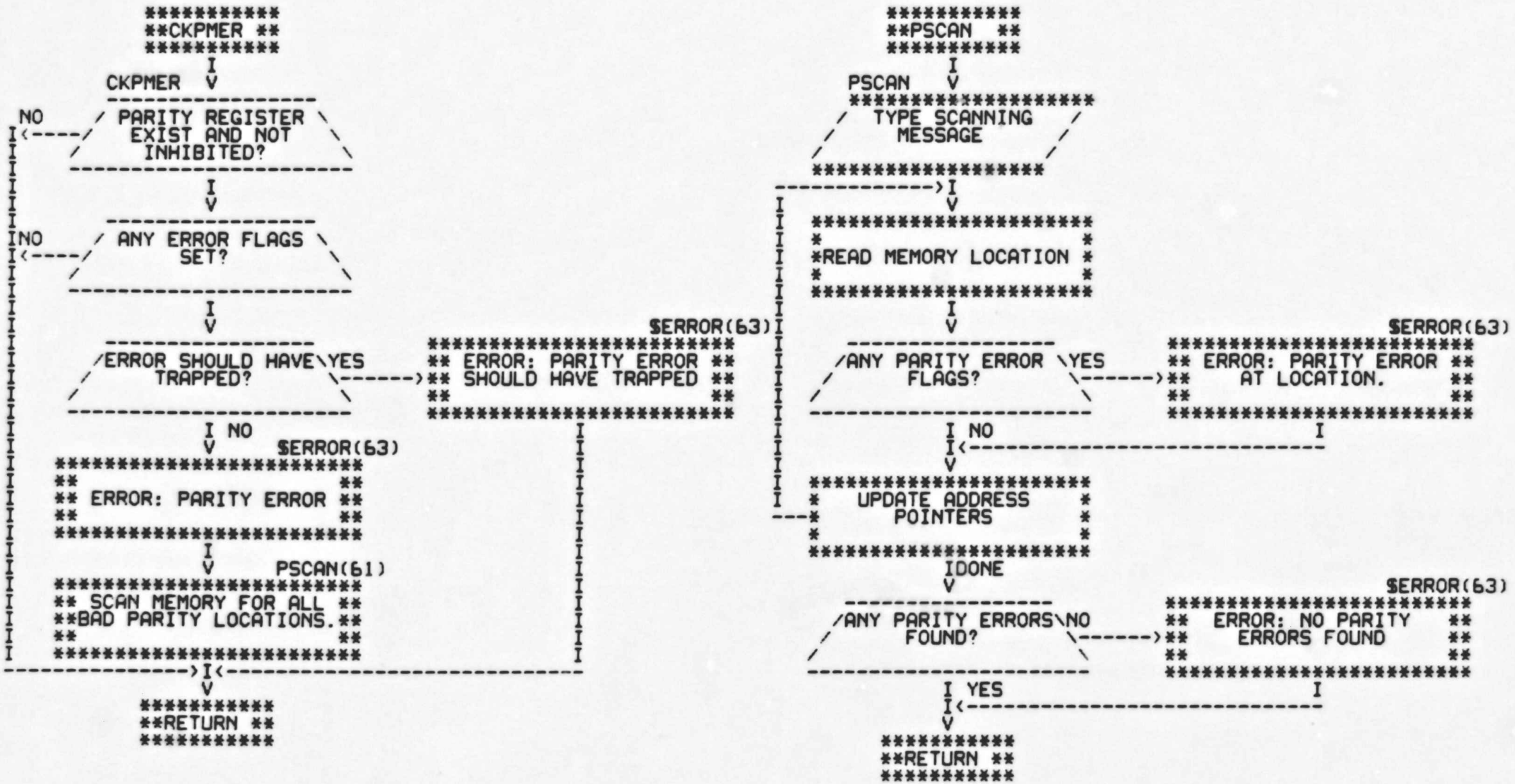


```

          *****
          **RESLDR **
          *****
              I
              V
        RESLDR
*****
*  MOVE "LOADER" FROM  *
*  END OF BK TO TOP OF *
*    MEMORY            *
*****
              I
              V
*****
**RETURN **
*****

```

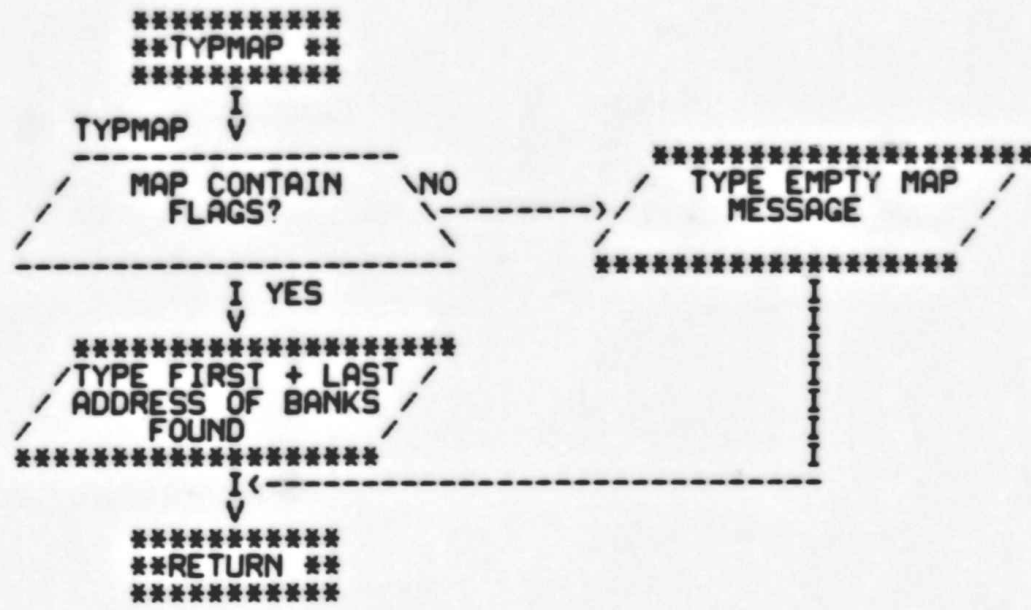




```
*****  
**SPRNT **-->  
*****  
  
*****  
**SPRNT0 **-->  
*****  
  
*****  
**SPRNT1 **-->  
*****  
  
*****  
**SPRNT2 **-->  
*****  
  
*****  
**SPRNT3 **-->  
*****
```

```
*****  
* ROUTINES TO SET UP *  
* DATA FOR ERROR *  
* TYPEOUTS. *  
*****
```

```
*****  
**RETURN **  
*****
```





```

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

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

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

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

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

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

$SPRINT
*****
**$SPRINT **--> * RELOCATES MESSAGE * **
**          * ADDRESS FOR $TYPE * **
*****

```

```

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

$TYPDS
*****
**$TYPDS **--> * TYPE A DECIMAL NUMBER * **
*****

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

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

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

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

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

```

```

*****
** .END **
*****

```





\$SCOPE	63#			
\$STYPAD	63	63#		
\$STYPDS	10	10	63	63#
\$TYPE	63	63#		
\$TYPOC	63	63#		
.END	63			

.TITLE CZQMCE0 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 TYPEOUTS
.*      12          INHIBIT KT11 (AT START TIME ONLY)
.*      11          INHIBIT ITERATIONS
.*      10          BELL ON ERROR
.*      9           LOOP ON ERROR
.*      8           LOOP ON TEST IN SWR<4:0>
.*      7           INHIBIT PROGRAM RELOCATION
.*      6           INHIBIT PARITY ERROR DETECTION
.*      5           INHIBIT EXERCISING VECTOR AREA.
    
```

.SBTTL BASIC DEFINITIONS

```

.*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100  STACK= 1100
    
```

```

.EQUIV EMT,ERROR    ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE    ;;BASIC DEFINITION OF SCOPE CALL
    
```

.\*MISCELLANEOUS DEFINITIONS

```

000011  HT= 11          ;;CODE FOR HORIZONTAL TAB
000012  LF= 12          ;;CODE FOR LINE FEED
000015  CR= 15          ;;CODE FOR CARRIAGE RETURN
000200  CRLF= 200       ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776  PS= 177776     ;;PROCESSOR STATUS WORD
177774  .EQUIV PS,PSW
177772  STKLM1= 177774 ;;STACK LIMIT REGISTER
177570  PIRQ= 177772   ;;PROGRAM INTERRUPT REQUEST REGISTER
177570  DSWR= 177570  ;;HARDWARE SWITCH REGISTER
177570  DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
    
```

.\*GENERAL PURPOSE REGISTER DEFINITIONS

```

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

```

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

```

```

113      .000001      BIT00= 1
114      .EQUIV      BIT09,BIT9
115      .EQUIV      BIT08,BIT8
116      .EQUIV      BIT07,BIT7
117      .EQUIV      BIT06,BIT6
118      .EQUIV      BIT05,BIT5
119      .EQUIV      BIT04,BIT4
120      .EQUIV      BIT03,BIT3
121      .EQUIV      BIT02,BIT2
122      .EQUIV      BIT01,BIT1
123      .EQUIV      BIT00,BIT0
124
125      ;*BASIC "CPU" TRAP VECTOR ADDRESSES
126      000004      ERRVEC= 4          ;: TIME OUT AND OTHER ERRORS
127      000010      RESVEC= 10         ;: RESERVED AND ILLEGAL INSTRUCTIONS
128      000014      TBITVEC=14        ;: "T" BIT
129      000014      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
195      000174      000174      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
196      000176      000000      SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
197
198      000200      000137      002640      .SBTTL STARTING ADDRESS(ES)
199      000204      000167      002436      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
200      ;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      003352      JMP TIMOUT ;TYPE OUT MEMORY MAP, BYTE BY BYTE.
205
206      000004      000004      .=ERRVEC
207      000006      025060      .WORD ERRTRP
208      .WORD 0
209
210      .SBTTL ACT11 HOOKS
211      ;*****
212      ;HOOKS REQUIRED BY ACT11
213      000010      $SVPC=.          ;SAVE PC
214      000046      .=46
215      000046      014174      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
216      000052      000052      .=52
217      000052      040000      .WORD BIT14      ;;2)SET LOC.52 TO BIT14
218      000010      .=$SVPC          ;; RESTORE PC

```



```

219          000300          . = 300
220          :*****
221          * THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
222          * THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
223          * THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
224          * MANAGEMENT IS DISABLED BY "CONSOLE START".
225          :*****
226 000300 005005 RESTAR: CLR R5 ;CLEAR FLAG TO INDICATE RESTART.
227 000302 000401 BR REST1 ;GO RESTORE PROGRAM BEFORE RESTARTING.
228 000304 010705 RESTOR: MOV PC R5 ;PUT DATA INTO FLAG FOR RESTORE.
229 000306 012706 001100 REST1: MOV #STACK, SP ;SET UP THE STACK POINTER.
230 000312 005767 001206 TST MEMMAP ;CHECK IF THE MEMORY HAS BEEN MAPPED.
231 000316 001002 BNE REST2 ;BR IF MEMORY MAPPED.
232 000320 000167 002330 JMP STARTA ;GO START
233 000324 005767 000256 REST2: TST MMAPA ;CHECK IF MEM MGMT AVAILABLE.
234 000330 001470 BEQ 10$ ;BR IF NO MEM MGMT.
235 000332 032737 000001 177572 BIT #BIT0, @#SRO ;CHECK IF MEM MGMT ACTIVE.
236 000340 001034 BNE 2$ ;BR IF MEM MGMT ALREADY SET UP.
237 000342 012700 172300 MOV #KIPDR0, RO ;POINT TO FIRST MEM MGMT DDATA REG.
238 000346 012701 000010 MOV #8, R1 ;SET UP COUNTER.
239 000352 012720 077406 1$: MOV #077406, (RO)+ ;MAP FIRST 28K 1-FOR-1.
240 000356 005301 DEC R1 ;COUNT REGESTERS.
241 000360 001374 BNE 1$ ;BR IF MORE REG.
242 000362 012700 172340 MOV #KIPAR0, RO ;POINT TO FIRST MEM MGMT ADDRESS REG.
243 000366 005020 CLR (RO)+ ;PAR0 MAPPED INTO BANK0.
244 000370 012720 000200 MOV #200, (RO)+ ;PAR1 MAPPED INTO BANK1.
245 000374 012720 000400 MOV #400, (RO)+ ;PAR2 MAPPED INTO BANK2.
246 000400 012720 000600 MOV #600, (RO)+ ;PAR3 MAPPED INTO BANK3.
247 000404 012720 001000 MOV #1000, (RO)+ ;PAR4 MAPPED INTO BANK4.
248 000410 012720 001200 MOV #1200, (RO)+ ;PAR5 MAPPED INTO BANK5.
249 000414 012720 001400 MOV #1400, (RO)+ ;PAR6 MAPPED INTO BANK6.
250 000420 012720 007600 MOV #7600, (RO)+ ;PAR7 MAPPED INTO BANK37.
251 000424 012737 000001 177572 MOV #BIT0, @#SRO ;ENABLE MEM MGMT.
252 000432 005000 2$: CLR RO ;INIT TEMP PAR REG.
253 000434 016701 000142 MOV PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
254 000440 016702 000140 MOV PRGMAP+2, R2 ;...HI 64K.
255 000444 006202 3$: ASR R2 ;SHIFT THE MAP POINTER...HI
256 000446 006001 ROR R1 ;...LO.
257 000450 103404 BCS 4$ ;BR WHEN FIRST BANK FOUND.
258 000452 062700 000200 ADD #200, RO ;UPDATE TMP PAR TO NEXT BANK.
259 000456 100372 BPL 3$ ;BR IF MORE.
260 000460 000000 HALT ;FATAL ERROR!!! MAP EMPTY?
261 000462 010037 172340 4$: MOV RO @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
262 000466 000137 000472 JMP @#5$ ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
263 000472 062700 000200 5$: ADD #200, RO ;KEEP UPDATING TEMP PAR REG.
264 000476 006202 ASR R2 ;SHIFT POINTER...HI
265 000500 006001 ROR R1 ;...LO
266 000502 103373 BCC 5$ ;BR IF TOP BANK NOT YET FOUND.
267 000504 010037 172342 MOV RO @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
268 000510 000410 BR 20$ ;BR TO RELOCATE SECTION.
269 000512 016700 000062 10$: MOV RELOCF, RO ;GET RELOCATION FACTOR.
270 000516 062700 001100 ADD #STACK, RO ;SET UP STACK POINTER.
271 000522 010006 MOV RO, SP ;SET STACK TO RELOCATE PROGRAM.
272 000524 062700 177432 ADD #20$-STACK, RO ;ADJUST RO TO RELOCATED "20$" ADDRESS.
273 000530 000110 JMP (RO) ;GO TO "20$" (RELOCATED).
274 000532 022767 000003 000042 20$: CMP #3, PRGMAP ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.

```

```

275 000540 001402          BEQ      21$      ;BR IF IN BANKS 0 AND 1.
276 000542 004767 016260   JSR      PC,      RELO      ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
277 000546 005705          TST      R5,      ;CHECK RESTART/RESTORE FLAG.
278 000550 001006          BNE      22$      ;BR IF RESTORE.
279 000552 005067 000412   CLR      $TIMES   ;CLEAN UP BEFORE STARTING.
280 000556 105067 000320   CLRB    $STNM
281 000562 000167 005272   JMP      START1
282 000566 004767 016442   JSR      PC,      RESLDR   ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
283 000572 000000          HALT                    ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
284 000574 000167 002054   JMP      STARTA    ;HALT AFTER RESTORING THE LOADERS.
285                                     ;CONTINUE WILL RESTART THE PROGRAM.
286                                     ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
287                                     ;* BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
288                                     ;* CIRCUMSTANCES.
288 000600 000000          RELOCF: .WORD 0      ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
289 000602 000000 000000   PRGMAP: .WORD 0,0    ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
290 000606 000000          MMAVA:  .WORD 0      ;MEMORY MANAGEMENT AVAILABLE FLAG.

```

.SBTTL POWER DOWN AND UP ROUTINES

```

291
292
293
294
295 000610 012737 000756 000024
296 000616 012737 000340 000026
297 000624 010046
298 000626 010146
299 000630 010246
300 000632 010346
301 000634 010446
302 000636 010546
303 000640 017746 000274
304 000644 010667 000112
305 000650 012737 000662 000024
306 000656 000000
307 000660 000776
308
309
310
311 000662 012737 000756 000024
312 000670 016706 000066
313 000674 005067 000062
314 000700 005267 000056
315 000704 001375
316 000706 012677 000226
317 000712 012605
318 000714 012604
319 000716 012603
320 000720 012602
321 000722 012601
322 000724 012600
323 000726 012737 000610 000024
324 000734 012737 000340 000026
325 000742 004567 022510
326 000746 025605
327 000750 012716
328 000752 000300
329 000754 000002
330 000756 000000
331 000760 000776
332 000762 000000

;*****
:POWER DOWN ROUTINE
$PWRDN: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST UP
        MOV #340,@#PWRVEC+2 ;;PRIO:7
        MOV R0,-(SP) ;;PUSH R0 ON STACK
        MOV R1,-(SP) ;;PUSH R1 ON STACK
        MOV R2,-(SP) ;;PUSH R2 ON STACK
        MOV R3,-(SP) ;;PUSH R3 ON STACK
        MOV R4,-(SP) ;;PUSH R4 ON STACK
        MOV R5,-(SP) ;;PUSH R5 ON STACK
        MOV @SWR,-(SP) ;;PUSH @SWR ON STACK
        MOV SP,$SAVR6 ;;SAVE SP
        MOV $PWRUP,@#PWRVEC ;;SET UP VECTOR
        HALT
        BR -.2 ;;HANG UP

;*****
:POWER UP ROUTINE
$PWRUP: MOV $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
        MOV $SAVR6,SP ;;GET SP
        CLR $SAVR6 ;;WAIT LOOP FOR THE TTY
        1$: INC $SAVR6 ;;WAIT FOR THE INC
        BNE 1$ ;;OF WORD
        MOV (SP)+,@SWR ;;POP STACK INTO @SWR
        MOV (SP)+,R5 ;;POP STACK INTO R5
        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
        MOV $PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
        MOV #340,@#PWRVEC+2 ;;PRIO:7
        JSR R5,$PRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
        $PWRMG: .WORD PWRMSG ;;POWER FAIL MESSAGE POINTER
        MOV (PC)+,(SP) ;;RESTART AT RESTART
        $PWRAD: .WORD RESTART ;;RESTART ADDRESS
        RTI
        $SILLUP: HALT
        BR -.2
        $SAVR6: 0

```

333  
334  
335  
336  
337  
338  
339 001100  
340 001100  
341 001100 000000  
342 001102 000  
343 001103 000  
344 001104 000000  
345 001106 000000  
346 001110 000000  
347 001112 000000  
348 001114 000  
349 001115 001  
350 001116 000000  
351 001120 000000  
352 001122 000000  
353 001124 000000  
354 001126 000000  
355 001130 000000  
356 001132 000000  
357 001134 000  
358 001135 000  
359 001136 000000  
360 001140 177570  
361 001142 177570  
362 001144 177560  
363 001146 177562  
364 001150 177564  
365 001152 177566  
366 001154 000  
367 001155 002  
368 001156 012  
369 001157 000  
370 001160 000000  
371 001162 000000  
372 001164 000000  
373 001166 000000  
374 001170 000000  
375 001172 000000  
376 001174 177607 000377  
377 001200 077  
378 001201 015  
379 001202 000012  
380  
381  
382  
383  
384  
385 001204  
386 001204 000000  
387 001206 000000  
388 001210 000000

```
.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
SCMTAG:      .=1100                ;; START OF COMMON TAGS
              .WORD                0
$TSTNM:      .BYTE                00                ;; CONTAINS THE TEST NUMBER
$ERFLG:      .BYTE                00                ;; CONTAINS ERROR FLAG
$ICNT:       .WORD                00                ;; CONTAINS SUBTEST ITERATION COUNT
$LPADR:      .WORD                00                ;; CONTAINS SCOPE LOOP ADDRESS
$LPERR:      .WORD                00                ;; CONTAINS SCOPE RETURN FOR ERRORS
$ERTTL:      .WORD                00                ;; CONTAINS TOTAL ERRORS DETECTED
$ITEMB:      .BYTE                00                ;; CONTAINS ITEM CONTROL BYTE
$ERMAX:      .BYTE                1                 ;; CONTAINS MAX. ERRORS PER TEST
$ERRPC:      .WORD                00                ;; CONTAINS PC OF LAST ERROR INSTRUCTION
$GDADR:      .WORD                00                ;; CONTAINS ADDRESS OF 'GOOD' DATA
$BDADR:      .WORD                00                ;; CONTAINS ADDRESS OF 'BAD' DATA
$GDDAT:      .WORD                00                ;; CONTAINS 'GOOD' DATA
$BDDAT:      .WORD                00                ;; CONTAINS 'BAD' DATA
              .WORD                00                ;; RESERVED--NOT TO BE USED
              .WORD                00
$AUTOB:      .BYTE                00                ;; AUTOMATIC MODE INDICATOR
$INTAG:      .BYTE                00                ;; INTERRUPT MODE INDICATOR
              .WORD                0
$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"
$TPFLG:      .BYTE                0                 ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
$TMP0:       .WORD                0                 ;; USER DEFINED
$TMP1:       .WORD                0                 ;; USER DEFINED
$TMP2:       .WORD                0                 ;; USER DEFINED
$TMP3:       .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:       .ASCII /?/                ;; QUESTION MARK
$CRLF:       .ASCII <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	SPASS:	.WORD	APASS	::	PASS COUNT
390	001214	000000	SDEVCT:	.WORD	ADEVCT	::	DEVICE COUNT
391	001216	000000	SUNIT:	.WORD	AUNIT	::	I/O UNIT NUMBER
392	001220	000000	SMSGAD:	.WORD	AMSGAD	::	MESSAGE ADDRESS
393	001222	000000	SMSGLG:	.WORD	AMSGLG	::	MESSAGE LENGTH
394	001224		SETABLE:			::	APT ENVIRONMENT TABLE
395	001224	000	SENV:	.BYTE	AENV	::	ENVIRONMENT BYTE
396	001225	000	SENVN:	.BYTE	AENVN	::	ENVIRONMENT MODE BITS
397	001226	000000	SSWREG:	.WORD	ASWREG	::	APT SWITCH REGISTER
398	001230	000000	SUSWR:	.WORD	AUSWR	::	USER SWITCHES
399	001232	000000	SCPUOP:	.WORD	ACPUOP	::	CPU TYPE, OPTIONS
400			::			::	BITS 15-11=CPU TYPE
401			::			::	11/04=01, 11/05=02, 11/20=03, 11/40=04, 11/45=05
402			::			::	11/70=06, PDQ=07, Q=10
403			::			::	BIT 10=REAL TIME CLOCK
404			::			::	BIT 9=FLOATING POINT PROCESSOR
405			::			::	BIT 8=MEMORY MANAGEMENT
406	001234	000	\$MAMS1:	.BYTE	AMAMS1	::	HIGH ADDRESS M.S. BYTE
407	001235	000	SMTYP1:	.BYTE	AMTYP1	::	MEM. TYPE, BLK#1
408			::			::	MEM. TYPE BYTE -- (HIGH BYTE)
409			::			::	900 NSEC CORE=001
410			::			::	300 NSEC BIPOLAR=002
411			::			::	500 NSEC MOS=003
412	001236	000000	\$MADR1:	.WORD	AMADR1	::	HIGH ADDRESS, BLK#1
413			::			::	MEM. LAST ADDR.=3 BYTES, THIS WORD AND LOW OF "TYPE" ABOVE
414	001240	000	\$MAMS2:	.BYTE	AMAMS2	::	HIGH ADDRESS M.S. BYTE
415	001241	000	SMTYP2:	.BYTE	AMTYP2	::	MEM. TYPE, BLK#2
416	001242	000000	\$MADR2:	.WORD	AMADR2	::	MEM. LAST ADDRESS, BLK#2
417	001244	000	\$MAMS3:	.BYTE	AMAMS3	::	HIGH ADDRESS M.S. BYTE
418	001245	000	SMTYP3:	.BYTE	AMTYP3	::	MEM. TYPE, BLK#3
419	001246	000000	\$MADR3:	.WORD	AMADR3	::	MEM. LAST ADDRESS, BLK#3
420	001250	000	\$MAMS4:	.BYTE	AMAMS4	::	HIGH ADDRESS M.S. BYTE
421	001251	000	SMTYP4:	.BYTE	AMTYP4	::	MEM. TYPE, BLK#4
422	001252	000000	\$MADR4:	.WORD	AMADR4	::	MEM. LAST ADDRESS, BLK#4
423	001254	000000	SVECT1:	.WORD	AVECT1	::	INTERRUPT VECTOR#1 BUS PRIORITY#1
424	001256	000000	SVECT2:	.WORD	AVECT2	::	INTERRUPT VECTOR#2 BUS PRIORITY#2
425	001260	000000	SBASE:	.WORD	ABASE	::	BASE ADDRESS OF EQUIPMENT UNDER TEST
426	001262	000000	SDEVN:	.WORD	ADEVN	::	DEVICE MAP
427	001264	000000	SCDW1:	.WORD	ACDW1	::	CONTROLLER DESCRIPTION WORD#1
428	001266	000000	SCDW2:	.WORD	ACDW2	::	CONTROLLER DESCRIPTION WORD#2
429	001270	000000	SDDW0:	.WORD	ADDW0	::	DEVICE DESCRIPTOR WORD#0
430	001272	000000	SDDW1:	.WORD	ADDW1	::	DEVICE DESCRIPTOR WORD#1
431	001274	000000	SDDW2:	.WORD	ADDW2	::	DEVICE DESCRIPTOR WORD#2
432	001276	000000	SDDW3:	.WORD	ADDW3	::	DEVICE DESCRIPTOR WORD#3
433	001300	000000	SDDW4:	.WORD	ADDW4	::	DEVICE DESCRIPTOR WORD#4
434	001302	000000	SDDW5:	.WORD	ADDW5	::	DEVICE DESCRIPTOR WORD#5
435	001304	000000	SDDW6:	.WORD	ADDW6	::	DEVICE DESCRIPTOR WORD#6
436	001306	000000	SDDW7:	.WORD	ADDW7	::	DEVICE DESCRIPTOR WORD#7
437	001310	000000	SDDW8:	.WORD	ADDW8	::	DEVICE DESCRIPTOR WORD#8
438	001312	000000	SDDW9:	.WORD	ADDW9	::	DEVICE DESCRIPTOR WORD#9
439	001314	000000	SDDW10:	.WORD	ADDW10	::	DEVICE DESCRIPTOR WORD#10
440	001316	000000	SDDW11:	.WORD	ADDW11	::	DEVICE DESCRIPTOR WORD#11
441	001320	000000	SDDW12:	.WORD	ADDW12	::	DEVICE DESCRIPTOR WORD#12
442	001322	000000	SDDW13:	.WORD	ADDW13	::	DEVICE DESCRIPTOR WORD#13
443	001324	000000	SDDW14:	.WORD	ADDW14	::	DEVICE DESCRIPTOR WORD#14
444	001326	000000	SDDW15:	.WORD	ADDW15	::	DEVICE DESCRIPTOR WORD#15



```

*****
*THE FOLLOWING TAGS ARE USER DEFINED
*****
501 001514 000000 $VERPC: .WORD 0 :VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
502 001516 070032 RESRVD: .WORD 070032 :CORE PARITY REG BITS RESERVED FOR FUTURE USE.
503 :NOTE: FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
504 001520 000000 LMAD: .WORD 0 :LAST CONTIGUOUS MEMORY ADDRESS (+2)
505 001522 000000 LDDISP: .WORD 0 :CONTAINS DISPLAY REGISTER IMAGE
506 001524 000000 MEMMAP: .WORD 0 :MEMORY MAP - EACH BIT CORRESPONDS TO 4K
507 001526 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
508 001528 000000 .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
509 001530 000000 TSTMAP: .WORD 0 :TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
510 001532 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
511 001534 000000 SAVTST: .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
512 .WORD 0 :SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
513 .WORD 0 :TEST EACH BANK ONCE.
514 001536 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
515 001538 000000 .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
516 001540 000000 PMEMAP: .WORD 0 :PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
517 001542 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
518 001544 000000 BITPT: .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
519 001546 000000 .WORD 0 :POINTER TO CURRENT 4K BANK OF MEMORY
520 001548 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
521 001550 000000 TMPPT: .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
522 001552 000000 .WORD 0 :TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
523 001554 000000 .WORD 0 :FIRST WORD CONTAINS LOW (0-64K) MAP
524 .WORD 0 :SECOND WORD CONTAINS HIGH (64-128K) MAP
525 .WORD 0 :LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
526 .WORD 0 :SET UP BY "INITMM" AND "INITDN" ROUTINES.
527 .WORD 0 :USED BY "MMUP" AND "MMDOWN" ROUTINES.
528 001556 000 FLAGBK: .BYTE 0 :OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
529 001557 000 OEFLG: .BYTE 0 :BK BLOCK INDICATOR. USED IN "INITMM" AND "MMUP".
530 001560 000 .EVEN :ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
531 001562 000000 FSTADR: .WORD 0 :FIRST VIRTUAL ADDRESS TO BE TESTED.
532 001564 000000 TMPFAD: .WORD 0 :FIRST ADDRESS IS USER SELECTABLE.
533 001566 000000 FADMSK: .WORD 0 :ADJUSTED FIRST ADDRESS.
534 001570 000000 000000 FADMAP: .WORD 0,0 :BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
535 001574 000000 LSTADR: .WORD 0 :TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
536 001576 000000 TMLPAD: .WORD 0 :MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
537 001600 000000 LADMSK: .WORD 0 :LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
538 001602 000000 000000 LADMAP: .WORD 0,0 :LAST ADDRESS IS USER SELECTABLE.
539 001606 000000 BLKMSK: .WORD 0 :ADJUSTED LAST ADDRESS.
540 001610 000000 .CONST: .WORD 0 :BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
541 001612 000004 WWP: .WORD 4 :TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
542 001614 000000 TEMP: .WORD 0 :MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
543 001616 000000 CASFLG: .WORD 0 :BLOCK MASK, DETERMINES THE BLOCK SIZE.
544 001620 177746 CASREG: .WORD 177746 :USER SELECTABLE CONSTANT DATA.
545 :WRITE WRONG PARITY COMMAND
546 :TEMPORARY STORAGE
547 :CACHE PRESENT FLAG
548 :CACHE CONTROL REGISTER
*****
* RELATIVE ADDRESSING TABLE.
* THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW

```

```

557
558
559 001622
560 001622 001100
561 001624 001516
562 001626 002076
563 001630 002276
564 001632 012026
565 001634 002050
566 001636 017374
567 001640 002340
568 001642 000010
569 001644 013760
570
571
572
573 001646 001116 001120 001124
574 001654 001126 000000
575 001660 001514 001116 001120
576 001666 001124 001126 000000
577 001674 001514 001116 001120
578 001702 001124 000000
579 001706 001514 001116 001160
580 001714 001120 000000
581 001720 001514 001116 001120
582 001726 001160 001124 001126
583 001734 000000
584 001736 001514 001116 001160
585 001744 001120 001124 001126
586 001752 000000
587 001754 001514 001116 001120
588 001762 001122 001124 001126
589 001770 000000
590 001772 001514 001116 001122
591 002000 000000
592 002002 001514 001116 001122
593 002010 001160 001162 000000
594 002016 001514 001116 001160
595 002024 001162 000000
596 002030 001160 001162 001120
597 002036 001126 000000
598 002042 001166 000000
599 002046 177777
600
601
602
603
604
605
606 002050 125325
607 002052 152652
608 002054 052452
609 002056 025125
610 002060 102070
611 002062 072527
612 002064 177777

```

```

;* RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUMENT TAGS.
;*****
RADTAB:
.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.
;*****
DT1: SERRPC,SGDADR,SGDDAT,SBDDAT,0
DT2: SVERPC,SERRPC,SGDADR,SGDDAT,SBDDAT,0
DT12: SVERPC,SERRPC,SGDADR,SGDDAT,0
DT14: SVERPC,SERRPC,STMPO,SGDADR,0
DT15: SVERPC,SERRPC,SGDADR,STMPO,SGDDAT,SBDDAT,0
DT21: SVERPC,SERRPC,STMPO,SGDADR,SGDDAT,SBDDAT,0
DT23: SVERPC,SERRPC,SGDADR,SBDDADR,SGDDAT,SBDDAT,0
DT24: SVERPC,SERRPC,SBDDADR,0
DT25: SVERPC,SERRPC,SBDDADR,STMPO,STMP1,0
DT26: SVERPC,SERRPC,STMPO,STMP1,0
DT30: STMPO,STMP1,SGDADR,SBDDAT,0
DT31: STMP3,0
.WORD -1 ;TABLE TERMINATOR.

```

```

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

```

```

MPPATS: 125325 ;EVEN,ODD
        152652 ;ODD,EVEN
        052452 ;EVEN,ODD
        025125 ;ODD,EVEN
        102070 ;EVEN,EVEN
        072527 ;ODD,ODD
        177777 ;EVEN,EVEN

```





669	002210	000000	0		: CONTROL MAP (LOW 64K)
670	002212	000000	0		: CONTROL MAP (HIGH 64K)
671	002214	000000	0		: MASK FOR MOS CORE, MS11-K
672	002216	172125	MPR10:	172124 +1	: PARITY STATUS REGISTER
673	002220	000000	0		: CONTROL MAP (LOW 64K)
674	002222	000000	0		: CONTROL MAP (HIGH 64K)
675	002224	000000	0		: MASK FOR MOS CORE, MS11-K
676	002226	172127	MPR11:	172126 +1	: PARITY STATUS REGISTER
677	002230	000000	0		: CONTROL MAP (LOW 64K)
678	002232	000000	0		: CONTROL MAP (HIGH 64K)
679	002234	000000	0		: MASK FOR MOS CORE, MS11-K
680	002236	172131	MPR12:	172130 +1	: PARITY STATUS REGISTER
681	002240	000000	0		: CONTROL MAP (LOW 64K)
682	002242	000000	0		: CONTROL MAP (HIGH 64K)
683	002244	000000	0		: MASK FOR MOS CORE, MS11-K
684	002246	172133	MPR13:	172132 +1	: PARITY STATUS REGISTER
685	002250	000000	0		: CONTROL MAP (LOW 64K)
686	002252	000000	0		: CONTROL MAP (HIGH 64K)
687	002254	000000	0		: MASK FOR MOS CORE, MS11-K
688	002256	172135	MPR14:	172134 +1	: PARITY STATUS REGISTER
689	002260	000000	0		: CONTROL MAP (LOW 64K)
690	002262	000000	0		: CONTROL MAP (HIGH 64K)
691	002264	000000	0		: MASK FOR MOS CORE, MS11-K
692	002266	172137	MPR15:	172136 +1	: PARITY STATUS REGISTER
693	002270	000000	0		: CONTROL MAP (LOW 64K)
694	002272	000000	0		: CONTROL MAP (HIGH 64K)
695	002274	000000	0		: MASK FOR MOS, CORE, MS11-K
696					
697	002276	000021	MPRX:	.BLKW 17.	: THIS IS THE END OF THE TABLE !
698					: TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
699					: (THE EXTRA WORD IS FOR A TERMINATOR.)

ERROR POINTER TABLE

.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:

Line	Item	DM	DH	DT	DF	Description
714	002340					
715		DM1	DH1	DT1	DF1	PARITY REGISTER DATA ERROR. PC REG, S/B, WAS
716	002340	026754				\$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 16, 16
717	002342	030333				
718	002344	001646				
719	002346	030700				
720		DM2	DH2	DT2	DF2	ADDRESS TEST ERROR(TST1-5). V/PC, P/PC, MA, S/B, WAS
721	002350	027010				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
722	002352	030352				
723	002354	001660				
724	002356	030704				
725		DM2	DH2	DT2	DF3	ADDRESS TEST ERROR(TST1-5). V/PC, P/PC, MA, S/B, WAS
726	002360	027010				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 8, 8
727	002362	030352				
728	002364	001660				
729	002366	030711				
730		DM4	DH2	DT2	DF2	CONSTANT DATA ERROR(TST6-10). V/PC, P/PC, MA, S/B, WAS
731	002370	027044				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
732	002372	030352				
733	002374	001660				
734	002376	030704				
735		DM5	DH2	DT2	DF2	ROTATING BIT ERROR(TST11-12). V/PC, P/PC, MA, S/B, WAS
736	002400	027102				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
737	002402	030352				
738	002404	001660				
739	002406	030704				
740		DM6	DH2	DT2	DF2	MOS REFRESH TEST ERROR (TST30-31). V/PC, P/PC, MA, S/B, WAS
741	002410	027140				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
742	002412	030352				
743	002414	001660				
744	002416	030704				
745		DM7	DH2	DT2	DF2	3 XOR 9 PATTERN ERROR(TST13-16). V/PC, P/PC, MA, S/B, WAS
746	002420	027204				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
747	002422	030352				
748	002424	001660				
749	002426	030704				
750		DM10	DH2	DT2	DF2	MARCHING 1'S AND 0'S ERROR(TST27). V/PC, P/PC, MA, S/B, WAS
751	002430	027245				\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT 16, 18, 18, 16, 16
752	002432	030352				
753	002434	001660				
754	002436	030704				
755						

756	002440	027311	DM11	: PARITY MEMORY ADDRESS ERROR(TST17).
757	002442	030352	DH2	: V/PC P/PC MA S/B WAS
758	002444	001660	DT2	: \$VERPC \$ERRPC \$GDADR \$GDDAT \$BDDAT
759	002446	030711	DF3	: 16, 18, 18, 8, 8
760			* ITEM 12	
761	002450	027355	DM12	: DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
762	002452	030377	DH12	: V/PC P/PC MA S/B
763	002454	001674	DT12	: \$VERPC \$ERRPC \$GDADR \$GDDAT
764	002456	030711	DF3	: 16, 18, 18, 8
765			* ITEM 13	
766	002460	027431	DM13	: WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
767	002462	030377	DH12	: V/PC P/PC MA S/B
768	002464	001674	DT12	: \$VERPC \$ERRPC \$GDADR \$GDDAT
769	002466	030711	DF3	: 16, 18, 18, 8
770			* ITEM 14	
771	002470	027521	DM14	: PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
772	002472	030420	DH14	: V/PC P/PC REG MA
773	002474	001706	DT14	: \$VERPC \$ERRPC \$TMPD \$GDADR
774	002476	030716	DF14	: 16, 18, 18, 18
775			* ITEM 15	
776	002500	026754	DM1	: PARITY REGISTER DATA ERROR.
777	002502	030441	DH15	: V/PC P/PC MAUT REG S/B WAS
778	002504	001720	DT15	: \$VERPC \$ERRPC \$GDADR \$TMPD \$GDDAT \$BDDAT
779	002506	030716	DF14	: 16, 18, 18, 18, 16, 16
780			* ITEM 16	
781	002510	027620	DM16	: MORE THAN ONE REGISTER INDICATED PARITY ERROR.
782	002512	030420	DH14	: V/PC P/PC REG MA
783	002514	001706	DT14	: \$VERPC \$ERRPC \$TMPD \$GDADR
784	002516	030716	DF14	: 16, 18, 18, 18
785			* ITEM 17	
786	002520	027677	DM17	: DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
787				: TRAPPED(TST21).
788	002522	030352	DH2	: V/PC P/PC MA S/B WAS
789	002524	001660	DT2	: \$VERPC \$ERRPC \$GDADR \$GDDAT \$BDDAT
790	002526	030711	DF3	: 16, 18, 18, 8, 8
791			* ITEM 20	
792	002530	027775	DM20	: RANDOM DATA ERROR(TST20).
793	002532	030352	DH2	: V/PC P/PC MA S/B WAS
794	002534	001660	DT2	: \$VERPC \$ERRPC \$GDADR \$GDDAT \$BDDAT
795	002536	030704	DF2	: 16, 18, 18, 16, 16
796			* ITEM 21	
797	002540	030027	DM21	: INSTRUCTION EXECUTION ERROR(TST21-26).
798	002542	030474	DH21	: V/PC P/PC IUT MA S/B WAS
799	002544	001736	DT21	: \$VERPC \$ERRPC \$TMPD \$GDADR \$GDDAT \$BDDAT
800	002546	030724	DF21	: 16, 18, 16, 18, 16, 16
801			* ITEM 22	: NOT USED
802			* ITEM 23	
803	002550	030076	DM23	: PROGRAM CODE CHANGED WHEN RELOCATED.
804	002552	030525	DH23	: V/PC P/PC SRC MA DST MA S/B WAS
805	002554	001754	DT23	: \$VERPC \$ERRPC \$GDADR \$BDADR \$GDDAT \$BDDAT
806	002556	030716	DF14	: 16, 18, 18, 18, 16, 16
807			* ITEM 24	
808	002560	030143	DM24	: TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
809	002562	030565	DH24	: V/PC P/PC TRP/PC
810	002564	001772	DT24	: \$VERPC \$ERRPC \$BDADR
811	002566	030716	DF14	: 16, 18, 18

812			;* ITEM 25	
813	002570	030217	DM25	: TRAPPED TO 114.
814	002572	030606	DH25	: V/PC P/PC TRP/PC REG, WAS
815	002574	002002	DT25	: \$VERPC, \$ERRPC, \$BDADR, \$TMPO, \$TMP1
816	002576	030716	DF14	: 16, 18, 18, 16
817			;* ITEM 26	
818	002600	030237	DM26	: FAILED TO TRAP.
819	002602	030637	DH26	: V/PC P/PC REG, WAS
820	002604	002016	DT26	: \$VERPC, \$ERRPC, \$TMPO, \$TMP1
821	002606	030704	DF2	: 16, 18, 18, 16
822			;* ITEM 27	
823	002610	030257	DM27	: (ACTION ENABLE WASN'T SET).
824	002612	030637	DH26	: V/PC P/PC REG, WAS
825	002614	002016	DT26	: \$VERPC, \$ERRPC, \$TMPO, \$BDDAT
826	002616	030704	DF2	: 16, 18, 18, 16
827			;* ITEM 30	
828	002620	000000	0	: NO MESSAGE.
829	002622	030661	DH30	: REG, WAS, MA, WAS
830	002624	002030	DT30	: \$TMPO, \$TMP1, \$GDADR, \$BDDAT
831	002626	030732	DF30	: 18, 16, 18, 8
832			;* ITEM 31	
833	002630	030313	DM31	: TRAPPED TO 4
834	002632	000000	0	: NO HEADER
835	002634	002042	DT31	: \$TMP3
836	002636	030732	DF30	: 18



```

893 003070 126727 176130 000001    CMPB   $ENV,#1          ;;ARE WE RUNNING UNDER APT?
894 003076 001411                BEQ    70$              ;;BRANCH IF YES
895 003100 026727 176034 000176    CMP    SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
896 003106 001010                BNE    71$              ;;BRANCH IF NO
897                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
898                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
899 003110 013746 177776                MOV    2#PSW,-(SP)     ;PUT THE PROCESSOR STATUS ON THE STACK
900 003114 004767 017262                JSR    PC,$GTSWR      ;GO TO THE SUBROUTINE
901 003120 000403                BR     71$
902 003122 112767 000001 176004 70$:  MOVB   #1,$AUTOB      ;;SET AUTO-MODE INDICATOR
903 003130 000405                71$:
904                                ;:69$: .ASCIZ <CRLF>,'CZQMCE0'<CRLF>
905                                68$:
906 003144                MOV    PC,RO           ;GET CURRENT PROGRAM COUNTER.
907 003144 010700                CMP    #,RO           ;CHECK IF THE PROGRAM IS RELOCATED.
908 003146 022700 003146                BEQ    10$            ;BR IF PROGRAM NOT RELOCATED.
909 003152 001402                JMP    RESTAR         ;GO TRY TO RELOCTED BEFORE CONTINUING.
910 003154 000167 175120                MOV    #3,PRGMAP      ;INITIALIZE PROGRAM MAP...LO 64K.
911 003160 012767 000003 175414 10$:  CLR    PRGMAP+2        ;...HI 64K.
912 003166 005067 175412                CLR    RELOCF        ;INIT THE RELOCATION FACTOR.
913 003172 005067 175402                JSR    PC,SAVLDR      ;GO SAVE LOADERS
914 003176 004767 014112
915
916                                ;* CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
917 003202 005067 175400                CLR    MMAVA         ;CLEAR MEM MGMT AVAILABLE FLAG
918 003206 032777 010000 175724  BIT    #SW12,2SWR     ;CHECK FOR INHIBIT KT11 SWITCH
919 003214 001014                BNE    IMPCK         ;BRANCH IF SET
920 003216 012737 003246 000004  MOV    2#IMPCK,2#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
921 003224 005037 177572                CLR    2#SRO         ;CLEAR MEM MGMT STATUS REG
922 003230 004767 011010                JSR    PC,MMINIT     ;MEM MGMT INITIALIZATION ROUTINE.
923 003234 005267 175346                INC    MMAVA         ;SET MEM MGMT AVAILABLE FLAG
924 003240 004567 020212                JSR    R5,SPRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
925 003244 025320                .WORD  MMAMES        ;ADDRESS OF MESSAGE TO BE TYPED
926                                ;"KT11 AVAILABLE"
927
928                                ;* CHECK IF CACHE PRESENT, IF SO TURN IT OFF!!!
929 003246 012706 001100  IMPCK: MOV    #STACK,SP
930 003252 005067 176340                CLR    CASFLG        ;CLEAR CACHE PRESENT FLAG
931 003256 012737 003300 000004  MOV    #MAPMEM,2#ERRVEC
932 003264 052767 000014 174454  BIS    #14,IMPCK
933 003272 012767 000001 176316  MOV    #1,CASFLG     ;SET CACHE PRESENT FLAG
934
935                                ;*****
936                                ;* ROUTINE TO MAP ALL OF MEMORY.
937                                ;* ONLY FULL 4K BANKS WILL BE RECOGNIZED.
938                                ;* R0 = MEMMAP POINTER...LO 64K.
939                                ;* R1 = MEMMAP POINTER...HI 64K.
940                                ;* R2 = ADDRESS POINTER
941                                ;* R3 = BANK POINTER...LO 64K.
942                                ;* R4 = BANK POINTER...HI 64K.
943                                ;* R5 = SCRATCH REGISTER.
944                                ;*****
945 003300 012706 001100  MAPMEM: MOV    #STACK,SP ;RESET THE STACK
946 003304 012700 001524                MOV    #MEMMAP,R0    ;SET UP MEMORY MAP POINTER...LO 64K.
947 003310 012701 001526                MOV    #MEMMAP+2,R1 ;...HI 64K.
948 003314 005010                CLR    (R0)          ;CLR MEMORY MAP...LO 64K.
    
```

# M08

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 21  
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0103

949	003316	005011				CLR	(R1)	; HI 64K.
950	003320	005002				CLR	R2	; SET ADDRESS POINTER TO 0
951	003322	012703	000001			MOV	#1, R3	; SETUP 4K BANK POINTER...LO 64K.
952	003326	005004				CLR	R4	; HI 64K.
953	003330	005067	175632			CLR	\$TMP3	; INIT TEMPORARY HIGH ADDRESS BITS.
954	003334	004567	020116			JSR	R5 SPRINT	; GO PRINT OUT THE FOLLOWING MESSAGE.
955	003340	025365				.WORD	MEMMES	; ADDRESS OF MESSAGE TO BE TYPED
956								; "MEMORY MAP:"
957	003342	012737	003456	000004		MOV	#2\$, @#ERRVEC	; SET UP TIMEOUT VECTOR
958	003350	011222			1\$:	MOV	(R2), (R2)+	; READ+WRITE ALL MEMORY
959	003352	032702	017777			BIT	#MASK4K, R2	; CHECK FOR 4K BOUNDARY
960	003356	001374				BNE	1\$	; BRANCH IF MORE IN BANK
961	003360	050310				BIS	R3, (R0)	; SET FLAG FOR BANK...LO 64K.
962	003362	050411				BIS	R4, (R1)	; HI 64K.
963	003364	010267	175574			MOV	R2, \$TMP2	; SAVE ADDRESS POINTER.
964	003370	005367	175570			DEC	\$TMP2	; ADJUST TO LAST ADR, LAST BANK.
965	003374	005767	175206			TST	MMAVA	; CHECK FOR MEM MGMT.
966	003400	001432				BEQ	3\$	; BR IF NO MEM MGMT.
967	003402	042767	160000	175554		BIC	#160000, \$TMP2	; CLEAR BANK BITS ON RELATIVE ADDRESS.
968	003410	013705	172344			MOV	@#KIPAR2, R5	; SAVE KIPAR2.
969	003414	005067	175546			CLR	\$TMP3	; MAKE SURE HI BITS ARE INIT.
970	003420	006305				ASL	R5	; SHIFT IT 6 PLACES.
971	003422	006305				ASL	R5	
972	003424	006305				ASL	R5	
973	003426	006305				ASL	R5	
974	003430	006305				ASL	R5	
975	003432	006167	175530			ROL	\$TMP3	
976	003436	006305				ASL	R5	
977	003440	006167	175522			ROL	\$TMP3	
978	003444	060567	175514			ADD	R5, \$TMP2	; MAKE LAST ADR PHYSICAL.
979	003450	005567	175512			ADC	\$TMP3	
980	003454	000404				BR	3\$	; GO TO UPDATE POINTERS.
981								
982								
983	003456	022626			2\$:	CMP	(SP)+, (SP)+	; RESTORE THE STACK POINTER
984	003460	052702	017777			BIS	#MASK4K, R2	; LAST ADDRESS OF 4K BANK
985	003464	005202				INC	R2	; FIRST ADDRESS OF NEXT BANK.
986	003466	005767	175114		3\$:	TST	MMAVA	; CHECK FOR MEM MGMT
987	003472	001411				BEQ	4\$	; BRANCH IF NO MEM MGMT
988	003474	062737	000200	172344		ADD	#200, @#KIPAR2	; UPDATE THIRD PAR
989	003502	012702	040000			MOV	#40000, R2	; POINT TO START OF THIRD PAR
990	003506	006303				ASL	R3	; UPDATE LO BANK POINTER.
991	003510	006104				ROL	R4	; UPDATE HI BANK POINTER
992	003512	100316				BPL	1\$	; BRANCH IF MORE MEMORY TO MAP.
993	003514	000402				BR	5\$	; EXIT WHEN DONE.
994								
995	003516	106303			4\$:	ASLB	R3	; UPDATE MAP POINTER
996	003520	100313				BPL	1\$	; BRANCH IF NOT YET DONE
997	003522	012737	025060	000004	5\$:	MOV	#ERRTRP, @#ERRVEC	; RESET TIMEOUT VECTOR
998	003530	004767	014622			JSR	PC, TYPMAP	; GO TYPE THE MAP.
999	003534	004567	017716			JSR	R5, SPRINT	; GO PRINT OUT THE FOLLOWING MESSAGE.
1000	003540	001201				.WORD	\$CALF	; ADDRESS OF MESSAGE TO BE TYPED
1001	003542	011067	175766			MOV	(R0), SAVTST	; SET UP TEST MAP...LO 64K.
1002	003546	011167	175764			MOV	(R1), SAVTST+2	; HI 64K.
1003	003552	011000				MOV	(R0), R0	; GET LOW MEM MAP
1004	003554	042700	177760			BIC	#177760, R0	; MASK ALL BUT BOTTOM 4 BANKS



```

1005 003560 020027 000017      CMP      R0,      #17      ;CHECK THAT BOTTOM 16K IS ALL THERE!
1006 003564 001530              BEQ      GMPR      ;BRANCH IF BOTTOM 16K EXISTS
1007 003566 004567 017664      JSR      R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
1008 003572 025470              .WORD   INSUFF      ;ADDRESS OF MESSAGE TO BE TYPED
1009                                ;"FIRST 16K OF MEMORY NOT ALL THERE!"
1010 003574 000000      6$:      HALT          ;FATAL ERROR HALT.
1011                                ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
1012                                ;*****
1013                                ;* SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
1014                                ;* DATI, DATIP, DATO, AND DATOB.
1015                                ;*****
1016 003576 012706 001100      TIMEOUT: MOV     #STACK, SP    ;SET UP THE STACK POINTER.
1017 003602 005067 175000      CLR     MMAVA      ;CLEAR MEM MGMT AVAILABLE FLAG.
1018 003606 032777 010000 175324      BIT     #SW12,    $SWR     ;CHECK IF MEM MGMT TO BE INHIBITED.
1019 003614 001011              BNE     1$,       ;BR IF NO MEM MGMT.
1020 003616 012737 003640 000004      MOV     #1$,     $#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
1021 003624 005037 177572      CLR     $#SRO     ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1022 003630 004767 010410      JSR     PC,       MMINIT  ;INIT ALL MEM MGMT REGISTERS.
1023 003634 005267 174746      INC     MMAVA     ;SET MEM MGMT AVAILABLE FLAG.
1024 003640              1$:
1025 003640 004567 017612      JSR     R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
1026 003644 025403              .WORD   BYTMS        ;ADDRESS OF MESSAGE TO BE TYPED
1027                                ;"BYTE MEMORY MAP:"
1028 003646 005000      CLR     R0        ;SET UP TYPE OUT FLAG.
1029 003650 005002      CLR     R2        ;SET ADDRESS POINTER TO ZERO.
1030 003652 012737 003716 000004      MOV     #20$,    $#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
1031 003660 105712              TSTB   (R2)       ;DO DATI ONLY.
1032 003662 032702 000001      BIT     #BIT0,   R2      ;CHECK FOR WORD ADDRESS.
1033 003666 001001              BNE     11$,     ;BR IF ODD BYTE ADDRESS.
1034 003670 011212      MOV     (R2),    (R2)    ;DO DATI, DATO...NOP FOR READ ONLY MAP.
1035 003672 151212      11$:   BISB   (R2),    (R2)    ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
1036 003674 005700              TST    R0        ;CHECK FOR PREVIOUS TYPED.
1037 003676 001023              BNE     30$,     ;BR IF ALREADY TYPED "FROM".
1038 003700 004567 017552      JSR     R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
1039 003704 025453              .WORD   FROM        ;ADDRESS OF MESSAGE TO BE TYPED
1040                                ;"FROM"
1041 003706 010246      MOV     R2,      -(SP)   ;PUT THE DATA ON THE STACK.
1042 003710 004767 021202      JSR     PC,      $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1043 003714 000413              BR     29$        ;GO TO ADDRESS POINTER UPDATE.
1044                                ;* TIME OUTS COME HERE.
1045 003716 022626      20$:   CMP     (SP)+,   (SP)+   ;POP TWO OFF STACK.
1046 003720 005700              TST    R0        ;CHECK FOR PREVIOUS TYPED.
1047 003722 001411              BEQ     30$,     ;BR IF ALREADY TYPED "TO".
1048 003724 004567 017526      JSR     R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
1049 003730 025463              .WORD   TO         ;ADDRESS OF MESSAGE TO BE TYPED
1050                                ;"TO"
1051 003732 005302      DEC     R2        ;BACK UP ONE BYTE.
1052 003734 010246      MOV     R2,      -(SP)   ;PUT THE DATA ON THE STACK.
1053 003736 004767 021154      JSR     PC,      $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1054 003742 005202      INC     R2        ;RESET ADDRESS POINTER.
1055 003744 005100      29$:   COM     R0        ;RESET PREVIOUS TYPED FLAG.
1056 003746 005202      30$:   INC     R2        ;UPDATE ADDRESS POINTER TO NEXT BYTE.
1057 003750 001423              BEQ     31$,     ;EXIT IF ZERO REACHED.
1058 003752 032702 017777      BIT     #MASK4K,R2 ;CHECK FOR 4K BANK BOUNDARY.
1059 003756 001340              BNE     10$,     ;BR IF MORE THIS 4K BANK.
1060 003760 005767 174622      TST    MMAVA     ;CHECK IF MEM MGMT IS AVAILABLE.

```

```

1061 003764 001735          BEQ      10$          ;BR IF NO MEM MGMT.
1062 003766 022737 007600 172346  CMP      #7600, @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
1063 003774 001411          BEQ      31$          ;EXIT WHEN ALL DONE.
1064 003776 012702 060000          MOV      #60000, R2    ;RESET VIRTUAL ADDRESS POINTER.
1065 004002 013737 172346 172344  MOV      @#KIPAR3, @#KIPAR2 ;SAVE MEM MGMT REG FOR TYPEOUT.
1066 004010 062737 000200 172346  ADD      #200, @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1067 004016 000720          BR       10$          ;BR BACK TO DO NEXT BANK.
1068 004020 005700          31$: TST      R0          ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1069 004022 001407          BEQ      32$          ;BR TO EXIT IF TYPING ALL DONE.
1070 004024 004567 017426          JSR      R5, SPRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1071 004030 025463          .WORD   TO          ;ADDRESS OF MESSAGE TO BE TYPED
1072                                "TO"
1073 004032 005302          DEC      R2          ;BACK ADDRESS POINTER UP ONE BYTE.
1074 004034 010246          MOV      R2, -(SP)   ;PUT THE DATA ON THE STACK.
1075 004036 004767 021054          JSR      PC, STYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1076 004042 000000          32$: HALT          ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1077                                ;* TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1078 004044 000654          BR       TIMEOUT    ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1079
1080                                .SBTTL  MAP PARITY REGISTERS
1081                                ;*****
1082                                ;* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1083                                ;* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1084                                ;*****
1085
1086 004046 012704 002276          GMPR: MOV      #MPRX, R4    ;SET UP POINTER TO PARITY REG EXIST TABLE.
1087 004052 032777 000100 175060  BIT      #SW06, @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
1088 004060 001036          BNE     GMPRD          ;BR IF INHIBIT PARITY.
1089 004062 012703 002076          MOV      #MPRO, R3     ;SET UP TABLE POINTER
1090 004066 012737 004110 000004  MOV      #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1091 004074 042713 000001          GMPRA: BIC     #1, (R3)  ;CLEAR FLAG BIT IN TABLE
1092 004100 005773 000000          TST     @ (R3)        ;DOES THIS MEMORY PARITY REGISTER EXIST.
1093                                ;* IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1094 004104 012324          MOV      (R3)+, (R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
1095 004106 000403          BR      GMPRC          ;SKIP TIMEOUT SERVICE CODE
1096                                ;* TIMEOUT COMES HERE
1097 004110 022626          GMPRB: CMP     (SP)+, (SP)+ ;RESTORE STACK POINTER
1098 004112 052723 000001          BIS     #1, (R3)+     ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1099 004116 005023          GMPRC: CLR     (R3)+   ;CLEAR THE MAP...LO 64K.
1100 004120 005023          CLR     (R3)+        ;...HI 64K.
1101 004122 005023          CLR     (R3)+        ;...AND THE MASK.
1102 004124 020327 002276          CMP     R3, #MPRX    ;HAVE WE CHECKED ALL REGISTERS?
1103 004130 103761          BLO     GMPRA        ;NO - GO BACK TO CHECK NEXT ONE
1104 004132 005014          CLR     (R4)         ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1105 004134 012737 025060 000004  MOV      #ERRTRP, @#ERRVEC ; RESTORE TRAPCATCHER
1106 004142 005767 176130          TST     MPRX          ;ANY PARITY REGISTERS PRESENT?
1107 004146 001006          BNE     MPAMEM        ;YES - GO TEST CONTROLS PRESENT
1108 004150 004567 017302          JSR      R5, SPRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1109 004154 025551          .WORD   MTR          ;ADDRESS OF MESSAGE TO BE TYPED
1110                                ;"NO MEMORY PARITY REGISTERS FOUND"
1111 004156 005014          GMPRD: CLR     (R4)   ;MAKE SURE TABLE IS CLEAR.
1112 004160 000167 001156          JMP     MANUAL        ;AND SKIP ALL CONTROLS TESTING
1113

```

```

1114 .SBTTL MAP PARITY MEMORY
1115 ;*****
1116 ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
1117 ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
1118 ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
1119 ; - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
1120 ; - PARITY GENERATE OR DETECT LOGIC FAILED
1121 ; - PARITY ERROR BIT FAILED TO SET
1122 ; - PARITY BITS IN MEMORY LOCATION FAILED
1123 ; - I.E. BIT STUCK AT GOOD PARITY VALUE
1124 ;*****
1125
1126 004164 004767 014044 MPAMEM: JSR PC, CLPAR ;INITIALIZE ALL PARITY REGISTERS
1127 004170 012767 000001 175346 MOV #1, BITPT ;INITIALIZE 4K POINTER
1128 004176 005067 175344 CLR BITPT+2 ;CLEAR HI 64K POINTER
1129 004202 012702 014000 MOV #14000, R2 ;SET ADR POINTER TO 14000.
1130 004206 005767 174374 TST MAVA ;CHECK FOR MEM MGMT
1131 004212 001404 BEQ MAPRB ;BRANCH IF NO MEM MGMT
1132 004214 012702 054000 MOV #54000, R2 ;SET ADR POINTER TO PAR2
1133 004220 004767 010020 JSR PC, MMINIT ;SET UP ALL MEMORY MGMT REGISTERS.
1134
1135 ;*****
1136 ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
1137 ;* THEN WRITE TEST LOCATION VIA DATO & READ TEST LOCATION VIA DATI
1138 ;* THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
1139 ;*****
1140
1141 004224 005067 175310 MAPRB: CLR PMEMAP ;CLEAR THE PARITY MEMORY MAP
1142 004230 005067 175306 CLR PMEMAP+2
1143 004234 012703 002076 1$: MOV #MPRO, R3 ;INITIALIZE TABLE ADDRESS
1144 004240 032713 000001 2$: BIT #1, (R3) ;IS THIS REGISTER PRESENT?
1145 004244 001052 BNE 3$ ;NO - GET THE NEXT ONE
1146 004246 013773 001612 000000 MOV @#WWP, @ (R3) ;YES - SET WRITE WRONG PARITY
1147 ;AND CLEAR REST OF REGISTER
1148 004254 011212 MOV (R2), (R2) ;WRITE WRONG PARITY
1149 004256 005712 TST (R2) ;READ WRONG PARITY
1150 004260 043773 001612 000000 BIC @#WWP, @ (R3) ;CLEAR WRITE WRONG PARITY
1151 004266 005773 000000 TST @ (R3) ;OTHERWISE, CHECK TO SEE IF THIS
1152 ;CONTROL REGISTER GOT A PARITY
1153 ;ERROR
1154 004272 100014 BPL 6$ ;BRANCH IF IT DIDN'T AND CHECK
1155 004274 032773 007740 000000 BIT #7740, @ (R3) ;IS IT A CORE PAR. REG.
1156 004302 001404 BEQ 5$ ;BRANCH IF NOT.
1157 004304 012763 070032 000006 MOV #70032, 6 (R3) ;IF IT IS SET UP MASK
1158 004312 000413 BR 7$ ;AND BRANCH TO SET BITS.
1159 004314 012763 077772 000006 5$: MOV #77772, 6 (R3) ;IF MOS SET UP MASK
1160 004322 000407 BR 7$ ;AND BRANCH TO SET BIT.
1161 004324 032773 007740 000000 6$: BIT #7740, @ (R3) ;IF ANY BITS ARE SET
1162 004332 001417 BEQ 3$ ;THEN CSR IS MS11-K.
1163 004334 012763 070000 000006 MOV #70000, 6 (R3) ;IF MS11-K SET MASK.
1164 004342 056763 175176 000002 7$: BIS BITPT, 2 (R3) ;SET FLAG IN MAP FOR THIS PARITY REGISTER
1165 004350 056763 175172 000004 BIS BITPT+2, 4 (R3)
1166 004356 056767 175162 175154 BIS BITPT, PMEMAP ;SET FLAG IN PARITY MAP
1167 004364 056767 175156 175150 BIS BITPT+2, PMEMAP+2
1168 004372 062703 000010 3$: ADD #10, R3 ;STEP UP TO NEXT REGISTER
1169 004376 020327 002276 CMP R3, #MPRX ;ARE WE DONE WITH TABLE?

```

```

1170 004402 103716 BLO 2$ :GO BACK TO CHECK FOR ANY MORE!
1171 004404 011212 MOV (R2), (R2) :CLEAR BAD PARITY
1172 004406 005767 174174 TST MMAPA :CHECK FOR MEM MGMT
1173 004412 001444 BEQ 10$ :BR IF NO MEM MGMT
1174 004414 062737 -000200 172344 4$: ADD #200, 2#KIPAR2 :UPDATE PAR TO NEXT 4K BANK.
1175 004422 006367 175116 ASL BITPT :UPDATE BANK POINTER...LO 64K.
1176 004426 006167 175114 ROL BITPT+2 :...HI 64K.
1177 004432 100441 BMI TMAP :BR IF ALL DONE.
1178 004434 023727 172344 001000 CMP 2#KIPAR2,#1000 :THIS CODE TESTS IF MS11-K IS
1179 004442 001013 BNE 12$ :PRESENT AND IF IT IS I SET
1180 004444 032737 000003 002260 BIT #3,2#MPR14+2 :THE BIT TO DISABLE ECC IN
1181 004452 001004 BNE 13$ :THE LOCATION MWP THAT IS
1182 004454 032737 000003 002270 BIT #3,2#MPR15+2 :USED AS THE COMMAND TO
1183 004462 001400 BEQ 13$ :WRITE WRONG PARITY.
1184 004464 012737 020004 001612 13$: MOV #20004,2#MWP
1185 004472 036767 175046 175024 12$: BIT BITPT, MEMMAP :CHECK IF BANK EXISTS...LO 64K.
1186 004500 001255 BNE 1$ :BR IF BANK EXISTS.
1187 004502 036767 175040 175016 BIT BITPT+2, MEMMAP+2 :...HI 64K.
1188 004510 001251 BNE 1$ :BR IF BANK EXISTS.
1189 004512 000740 BR 4$ :BR IF BANK DOESN'T EXIST.
1190 004514 036767 175024 175002 11$: BIT BITPT, MEMMAP :CHECK IF BANK EXISTS.
1191 004522 001244 BNE 1$ :BR IF BANK EXISTS.
1192 004524 062702 020000 10$: ADD #20000, R2 :UPDATE ADDRESS POINTER TO NEXT BANK.
1193 004530 106367 175010 ASLB BITPT :MOVE POINTER TO NEXT BANK.
1194 004534 100367 BPL 11$ :BR IF MORE TO LOOK FOR.

```

```

*****
* ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
* AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
*****

```

```

1200
1201 004536 004767 013472 TMAP: JSR PC, CLRPAR :INITIALIZE ALL PARITY REGISTERS PRESENT
1202 004542 004567 016710 JSR R5, $PRINT :GO PRINT OUT THE FOLLOWING MESSAGE.
1203 004546 025426 .WORD MMAP :ADDRESS OF MESSAGE TO BE TYPED
1204 : "PARITY MEMORY MAP:"
1205 004550 012703 002076 MOV #MPRO, R3 :INITIALIZE TABLE POINTER
1206 004554 032713 000001 1$: BIT #BIT0, (R3) :CHECK IF THIS REGISTER IS PRESENT.
1207 004560 001046 BNE 2$ :BR IF NOT PRESENT.
1208 004562 022763 070032 000006 CMP #70032, 6(R3)
1209 004570 001004 BNE 3$
1210 004572 004567 016660 JSR R5, $PRINT :GO PRINT OUT THE FOLLOWING MESSAGE.
1211 004576 026067 .WORD MX3 :ADDRESS OF MESSAGE TO BE TYPED
1212 : "CORE PARITY"
1213 004600 000417 BR 5$
1214 004602 022763 077772 000006 3$: CMP #77772, 6(R3)
1215 004610 001004 BNE 4$
1216 004612 004567 016640 JSR R5, $PRINT :GO PRINT OUT THE FOLLOWING MESSAGE.
1217 004616 026106 .WORD MX4 :ADDRESS OF MESSAGE TO BE TYPED
1218 : "MOS PARITY"
1219 004620 000407 BR 5$
1220 004622 022763 070000 000006 4$: CMP #70000, 6(R3)
1221 004630 001003 BNE 5$
1222 004632 004567 016620 JSR R5, $PRINT :GO PRINT OUT THE FOLLOWING MESSAGE.
1223 004636 026124 .WORD MX5 :ADDRESS OF MESSAGE TO BE TYPED
1224 : "MS11-K CSR"
1225 004640 5$:

```

```

1226 004640 004567 016612 JSR R5 SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1227 004644 026035 .WORD MX1 ;ADDRESS OF MESSAGE TO BE TYPED
1228 ;"REGISTER AT"
1229 004646 011346 MOV (R3) -(SP) ;SAVE (R3) FOR TYPEOUT
1230 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE STYPOC ROUTINE
1231 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
1232 004650 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1233 004654 004767 017774 JSR PC, STYPOC ;GO TO THE SUBROUTINE
1234 004660 004567 016572 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1235 004664 026054 .WORD MX2 ;ADDRESS OF MESSAGE TO BE TYPED
1236 ;"CONTROLS"
1237 004666 010300 MOV R3, R0 ;SET UP R0 FOR TYPMAP ROUTINE.
1238 004670 005720 TST (R0)+ ;UPDATE POINTER TO MAP.
1239 004672 004767 013460 JSR PC, TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1240 004676 062703 000010 2$: ADD #10, R3 ;UPDATE TO NEXT REGISTER IN TABLE.
1241 004702 020327 002276 CMP R3, #MPRX ;ARE WE ALL DONE WITH TABLE?
1242 004706 103722 BLO 1$ ;BRANCH IF MORE REGISTERS
1243 004710 004567 016542 JSR R5, SPRINT ;THE REASON I'M OUTPUTTING THIS CRLF
1244 004714 001201 $CRLF ;IS TO GIVE THE PRINTER ENOUGH TIME TO
1245 ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
1246 004716 022737 070000 002264 CMP #70000, @#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
1247 004724 001006 BNE 7$ ;IF NO BRANCH
1248 004726 043727 002260 001540 BIC @#MPR14+2, #PMEMAP ;IF YES THEN CLEAR THE BITS IN
1249 004734 043737 002262 001540 BIC @#MPR14+4, @#PMEMAP ;THE PARITY MEMORY MAP.
1250 004742 022737 070000 002274 7$: CMP #70000, @#MPR15+6 ;DO WE HAVE A MS11-K
1251 004750 001031 BNE 9$ ;IF NO GO TO TESTS NOW.
1252 004752 043737 002270 001540 BIC @#MPR15+2, @#PMEMAP ;IF YES I AM GOING TO
1253 004760 043737 002272 001542 BIC @#MPR15+4, @#PMEMAP+2 ;CLEAR THE PARITY INDICATORS
1254 004766 012705 002276 MOV #MPRX, R5 ;FOR THAT PORTION OF MEMORY.
1255 004772 021537 002256 6$: CMP (R5), @#MPR14 ;SEARCH FOR THIS MS11-K CSR IN
1256 004776 001004 BNE 8$ ;AND IF ITS THERE DELETE IT
1257 005000 005015 CLR (R5)
1258 005002 052737 000001 002256 8$: BIS #1, @#MPR14
1259 005010 022537 002266 CMP (R5)+, @#MPR15 ;SEARCH FOR MS11-K CSR IN
1260 005014 001366 BNE 6$ ;THE AVAILABILITY TABLE,
1261 005016 005045 CLR -(R5) ;AND CLEAR ITS ADDRESS FROM THE TABLE
1262 005020 052737 000001 002266 BIS #1, @#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
1263 005026 004567 016424 JSR R5, SPRINT ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1264 005032 026142 .WORD MX6
1265 005034 005737 002276 9$: TST @#MPRX ;ARE THERE ANY PARITY REGISTERS TO TEST?
1266 005040 001002 BNE CTRLS ;IF SO TEST THE BITS IN THE REGISTERS,
1267 005042 000167 000274 JMP MANUAL ;IF NO JUMP OVER REGISTER TESTS.
1268
1269 .SBTTL TEST PARITY REGISTERS
1270 ;*****
1271 ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1272 ;* CAN BE SET AND CLEARED.
1273 ;* THIS IS A ONCE ONLY TEST.
1274 ;*****
1275
1276 005046 012703 002076 CTRLS: MOV #MPRO, R3 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1277 005052 011302 1$: MOV (R3), R2 ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
1278 005054 062703 000010 ADD #10, R3 ;UPDATE POINTER TO NEXT PAR. REG. ADD.
1279 005060 032702 000001 BIT #1, R2 ;IS THIS REGISTER BEING USED?
1280 005064 001372 BNE 1$ ;GO TO NEXT IF NOT
1281 005066 020327 002276 CMP R3, #MPRX ;ARE WE AT END OF TABLE
    
```

```

1282 005072 003055          BGT  RESCHK          ;GO TO NEXT TEST IF YES
1283 005074 005763 177776  TST  -2(R3)          ;TEST MASK FOR PARITY REGISTER
1284 005100 001764          BEQ  1$              ;IF = 0 THEN DO NOT TEST
1285 005102 016367 177776 174406 MOV  -2(R3), RESRVD ;GET MASK FOR REGISTER WE ARE WORKING ON
1286 005110 012700 000001  MCV  #1, R0          ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
1287 005114 005012          CLR  (R2)           ;INITIALIZE THE PARITY REGISTER
1288 005116 011201          MOV  (R2), R1       ;READ THE CONTENTS OF THE PARITY REGISTER
1289 005120 046701 174372  BIC  RESRVD, R1     ;CLEAR BITS WHICH ARE RESERVED
1290 005124 001405          BEQ  2$              ;CHECK OTHER BITS - BRANCH IF OK
1291 005126 004767 013124 64$: JSR  PC, SPRT     ;SET UP VALUES FOR ERROR PRINTING.
1292 005132 004767 014446  JSR  PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1293 005136 000001          .WORD 1            ;ERROR TYPE CODE.
1294 005140 030067 174352 2$: BIT  R0, RESRVD  ;IS THIS BIT RESERVED?
1295 005144 001025          BNE  3$              ;YES - DON'T TEST IT
1296 005146 010012          MOV  R0, (R2)       ;NO - SET THIS BIT IN THE PARITY REGISTER
1297 005150 011201          MOV  (R2), R1       ;READ & SAVE CONTENTS OF THE PARITY REGISTER
1298 005152 005012          CLR  (R2)           ;CLEAR THE PARITY REGISTER
1299 005154 046701 174336  BIC  RESRVD, R1     ;CLEAR BIT LOCATIONS THAT ARE RESERVED
1300 005160 020001          CMP  R0, R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
1301 005162 001405          BEQ  66$            ;BRANCH OVER ERROR CALL IF GOOD DATA.
1302 005164 004767 013116 65$: JSR  PC, SPRT     ;SET UP VALUES FOR ERROR PRINTING.
1303 005170 004767 014410  JSR  PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1304 005174 000001          .WORD 1            ;ERROR TYPE CODE.
1305 005176          66$:
1306 005176 011201          MOV  (R2), R1       ;READ THE CONTENTS OF THE PARITY REGISTER
1307 005200 046701 174312  BIC  RESRVD, R1     ;CLEAR BITS WHICH ARE RESERVED
1308 005204 001405          BEQ  3$              ;CHECK OTHER BITS - BRANCH IF OK
1309 005206 004767 013044 67$: JSR  PC, SPRT     ;SET UP VALUES FOR ERROR PRINTING.
1310 005212 004767 014366  JSR  PC, $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
1311 005216 000001          .WORD 1            ;ERROR TYPE CODE.
1312 005220 006300          3$: ASL  R0          ;ROTATE TO GET NEXT BIT TO BE TESTED
1313 005222 103346          BCC  2$              ;BRANCH IF NOT DONE WITH ALL BITS
1314 005224 000712          BR   1$              ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.

```

```

*****
* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
* THIS IS A ONCE ONLY TEST.
*****

```

```

1320
1321 005226 012704 002076  RESCHK: MOV  #MPRO, R4 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1322 005232 010403 1$: MOV  R4, R3
1323 005234 062704 000010  ADD  #10, R4
1324 005240 032713 000001  BIT  #1, (R3) ;IS THIS REGISTER BEING USED
1325 005244 001372          BNE  1$              ;BRANCH IF NO
1326 005246 012773 177777 000000  MOV  #-1, 2(R3)     ;SET ALL BITS TO A 1
1327 005254 022704 002276  CMP  #MPRX, R4      ;ARE WE AT THE END OF THE TABLE
1328 005260 002764          BLT  1$              ;IF YES THEN WE ARE READY TO TEST
1329 005262 000005          RESET              ;RESET THE WORLD
1330 005264 012703 002076  MOV  #MPRO, R3      ;LOAD INITIAL ADDRESS FOR POINTER
1331 005270 011302 2$: MOV  (R3), R2     ;STORE PARITY REGISTER ADDRESS
1332 005272 062703 000010  ADD  #10, R3
1333 005276 032702 000001  BIT  #1, R2
1334 005302 001372          BNE  2$              ;
1335 005304 022703 002276  CMP  #MPRX, R3
1336 005310 002014          BGE  MANUAL
1337 005312 011201          MOV  (R2), R1      ;GET CONTENTS OF REGISTER

```

1338	005314	005012		CLR	(R2)			
1339	005316	042701	077772	BIC	#77772, R1			; CLEAR BITS NOT EFFECTED BY RESET
1340	005322	005701		TST	R1			; CHECK IF REST WERE CLEARED BY RESET
1341	005324	001405		BEG	65\$			; BRANCH OVER ERROR CALL IF GOOD DATA.
1342	005326	004767	012724	JSR	PC,	SPRNT		; SET UP VALUES FOR ERROR PRINTING.
1343	005332	004767	014246	JSR	PC,	\$ERROR		; *** ERROR *** (GO TYPE A MESSAGE)
1344	005336	000001		.WORD	1			; ERROR TYPE CODE.
1345	005340							
1346	005340	000753		BR	2\$			; BRANCH BACK TO CHECK NEXT REGISTER
1347								
1348								
1349	005342	012700	000014	MANUAL: MOV	#12, R0			; SET COUNTER TO CLEAR 12 WORDS.
1350	005346	012701	001562	MOV	#FSTADR, R1			; STARTING AT FSTADR.
1351	005352	005021		1\$: CLR	(R1)+			; CLEAR THE LOCATIONS.
1352	005354	005300		DEC	R0			; COUNT.
1353	005356	001375		BNE	1\$			; BR IF MORE.
1354	005360	105767	174172	TSTB	SELFLG			; CHECK FOR SELECT PARAMETERS STARTUP.
1355	005364	001005		BNE	MANUL1			; BR IF PARAMETERS TO BE SELECTED.
1356	005366	016767	173572	MOV	\$TMP2, LSTADR			; SET UP VIRTUAL LAST ADDRESS.
1357	005374	000167	000402	JMP	MANUL2			; SKIP PARAMETER SELECTION SECTION.

```

1358
1359
1360
1361
1362 005400 012700 000001
1363 005404 005001
1364 005406 005002
1365 005410 005003
1366 005412 004567 016040
1367 005416 026255
1368
1369
1370
1371 005420 013746 177776
1372 005424 004767 015654
1373 005430 042716 000001
1374 005434 005067 174074
1375 005440 005067 174072
1376 005444 062702 020000
1377 005450 005503
1378 005452 020367 015776
1379 005456 103403
1380 005460 101006
1381 005462 020216
1382 005464 101004
1383 005466 006300
1384 005470 006101
1385 005472 100364
1386 005474 000507
1387 005476 030067 174022
1388 005502 001003
1389 005504 030167 174016
1390 005510 001501
1391 005512 016704 015736
1392 005516
1393 005516 004567 015734
1394 005522 026342
1395
1396
1397
1398 005524 013746 177776
1399 005530 004767 015550
1400 005534 005716
1401 005536 001010
1402 005540 005767 015710
1403 005544 001005
1404 005546 016716 173412
1405 005552 016767 173410 015674
1406 005560 012667 174010
1407 005564 020467 015664
1408 005570 101352
1409 005572 103403
1410 005574 021667 173774
1411 005600 101346
1412 005602 032716 017777
1413 005606 001404

```

```

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

```



1414	005610	010067	173754		MOV	R0,	FADMAP	;SET UP FIRST ADDRESS MAP.
1415	005614	010167	173752		MOV	R1,	FADMAP+2	
1416	005620	050067	173710	13\$:	BIS	R0,	SAVTST	;SFT FLAG IN TEST MAP...LO 64K.
1417	005624	050167	173706		BIS	R1,	SAVTST+2	;HI 64K.
1418	005630	020367	015620	14\$:	CMP	R3,	\$HIOCT	;CHECK FOR PAST LAST ADR.
1419	005634	103404			BLO	15\$		;BR IF BELOW LAST ADR.
1420	005636	101020			BHI	16\$		;BR IF GONE PAST LAST ADR.
1421	005640	020267	173730		CMP	R2,	LSTADR	;CHECK FOR PAST LAST ADR.
1422	005644	101015			BHI	16\$		;BR IF GONE PAST LAST ADR.
1423	005646	062702	020000	15\$:	ADD	#20000,	R2	;UPDATE ADDRESS POINTER.
1424	005652	005503			ADC	R3		;...HI BITS.
1425	005654	006300			ASL	R0		;UPDATE BANK POINTER...LO 64K.
1426	005656	006101			ROL	R1		;...HI 64K.
1427	005660	100415			BMI	17\$		;BR IF OVERFLOW.
1428	005662	030067	173636		BIT	R0,	MEMMAP	;CHECK IF THIS BANK EXISTS.
1429	005666	001354			BNE	13\$		;BR IF BANK EXISTS.
1430	005670	030167	173632		BIT	R1,	MEMMAP+2	;CHECK IF THIS BANK EXISTS.
1431	005674	001351			BNE	13\$		;BR IF BANK EXISTS.
1432	005676	000754			BR	14\$		;BR IF BANK DOESN'T EXIST.
1433	005700	030067	173620	16\$:	BIT	R0,	MEMMAP	;CHECK IF THIS BANK EXISTS.
1434	005704	001010			BNE	20\$		;BR IF IT EXISTS.
1435	005706	030167	173614		BIT	R1,	MEMMAP+2	;CHECK IF THIS BANK EXISTS.
1436	005712	001005			BNE	20\$		;BR IF IT EXISTS.
1437	005714	005726		17\$:	TST	(SP)+		;ADJUST THE STACK.
1438	005716	004567	015534		JSR	R5,	SPRINT	;GO PRINT OUT THE FOLLOWING MESSAGE.
1439	005722	026365			.WORD	BADADR		;ADDRESS OF MESSAGE TO BE TYPED
1440								;"?ADDRESS IN UNMAPPED BANK?"
1441	005724	000606			BR	MANUAL		;LOOP BACK TO THE BEGINNING.
1442	005726	010067	173650	20\$:	MOV	R0,	LADMAP	;SET UP MAP FOR LAST ADDRESS.
1443	005732	010167	173646		MOV	R1,	LADMAP+2	
1444	005736	005767	172644	21\$:	TST	MMAVA		;CHECK FOR MEMORY MANAGEMENT.
1445	005742	001404			BEQ	22\$		;BR IF NO MEM MGMT.
1446	005744	042716	160000		BIC	#160000,	(SP)	;ADJUST FSTADR TO VITRUAL BANK 0.
1447	005750	062716	040000		ADD	#40000,	(SP)	;...TO VIRTUAL BANK 2.
1448	005754	012667	173602	22\$:	MOV	(SP)+,	FSTADR	;SAVE FISRT ADDRESS OFF THE STACK.
1449	005760			30\$:				
1450	005760	004567	015472		JSR	R5,	SPRINT	;GO PRINT OUT THE FOLLOWING MESSAGE.
1451	005764	026422			.WORD	CONST		;ADDRESS OF MESSAGE TO BE TYPED
1452								; "SELECT CONSTANT: "
1453					;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$RDOCT ROUTINE			
1454					;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.			
1455	005766	013746	177776		MOV	@#PSW,	-(SP)	;PUT THE PROCESSOR STATUS ON THE STACK
1456	005772	004767	015306		JSR	PC,	\$RDOCT	;GO TO THE SUBROUTINE
1457	005776	012667	173606		MOV	(SP)+,	.CONST	;SAVE THE CONSTANT
1458	006002	005767	172600	MANUL2:	TST	MMAVA		;CHECK IF MEM MGMT IS AVAILABLE.
1459	006006	001406			BEQ	31\$		;BR IF NO MEM MGMT.
1460	006010	042767	160000	173556	BIC	#160000,	LSTADR	;ADJUST LSTADR TO VIRTUAL BANK 0.
1461	006016	062767	040000	173550	ADD	#40000,	LSTADR	;...VIRTUAL BANK 2.
1462	006024	062767	000002	173542	31\$:	ADD	#2,	LSTADR
1463	006032	042767	000001	173534	BIC	#BIT0,	LSTADR	;ADJUST LAST ADDRESS UP ONE WORD.
1464	006040	032767	017777	173526	BIT	#MASK4K,	LSTADR	;MAKE SURE IT IS A WORD ADDRESS.
1465	006046	001004			BNE	START1		;CHECK IF LAST ADR IS ON BANK BOUNDRY.
1466	006050	005067	173526		CLR	LADMAP		;BR IF NOT ON BOUNDRY.
1467	006054	005067	173524		CLR	LADMAP+2		;CLEAR OUT THE LAST ADDRESS MAP.
1468								



```

1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497 006142
1498 006142 004567 012426
1499 006146 000001
1500
1501 006150 000167 005604
1502
1503
1504 006154 004467 006212
1505 006160 004767 007634
1506 006164 010012
1507 006166 012201
1508 006170 020001
1509 006172 001405
1510 006174 004767 012132
1511 006200 004767 013400
1512 006204 000002
1513 006206
1514 006206 062700 000002
1515 006212 030502
1516 006214 001363
1517 006216 004767 006726
1518
1519
1520
1521 006222 004467 006602
1522 006226 004767 007566
1523 006232 162700 000002
1524 006236 014201
1525 006240 020001
1526 006242 001405
1527 006244 004767 012036
1528 006250 004767 013330
1529 006254 000002
1530 006256
1531 006256 030502
1532 006260 001364
1533 006262 004767 007352

```

```

.SBTTL SECTION 1: MEMORY ADDRESS TESTS
*****
*TEST 1 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDRY BIT MASK.
*****
TST1:
      JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 1 ;MINIMUM BLOCK SIZE OF 1 WORDS
                        ;REQUIRED FOR THIS TEST.
      JMP    TST32 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        ;AVAILABLE FOR TEST.

;* UPWARDS WORD ADDRESSING.
      JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR    PC,    PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOV    R0,    (R2) ;WRITE VALUE OF ADDRESS INTO ADDRESS
      MOV    (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP    R0,    R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR    PC,    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.

```

```

1534
1535
1536
1537
1538
1539
1540
1541
1542
1543 006266
1544 006266 004567 012302
1545 006272 000000
1546
1547 006274 004467 006072
1548 006300 004767 007514
1549 006304 110022
1550 006306 005200
1551 006310 030502
1552 006312 001374
1553 006314 004767 006630
1554
1555
1556
1557 006320 004467 006504
1558 006324 004767 007470
1559 006330 005300
1560 006332 114201
1561 006334 120001
1562 006336 001405
1563 006340 004767 011742
1564 006344 004767 013234
1565 006350 000003
1566 006352 030502
1567 006354 001365
1568 006356 004767 007256
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580 006362
1581 006362 004567 012206
1582 006366 000000
1583
1584 006370 004467 006434
1585 006374 004767 007420
1586 006400 005100
1587 006402 062700 000002
1588 006406 010042
1589 006410 030502

```

```

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

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

65$:  BIT     R5,      R2 ;CHECK FOR END OF A BLOCK.
      BNE    4$,      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.

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

```

```

1590 006412 001373      BNE 2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1591 006414 004767 007220 JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.
1592
1593 ;* CHECK COMPLEMENT DATA WRITTEN DOWN
1594 ;* UPWARDS WORD ADDRESSING.
1595 006420 004467 005746      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1596 006424 004767 007370 3$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
1597 006430 005100      COM R0      ;COMPLEMENT IT
1598 006432
1599 006432 012201      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1600 006434 020001      CMP R0, R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
1601 006436 001405      BEQ 65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
1602 006440 004767 011665 64$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
1603 006444 004767 013134      JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1604 006450 000002      .WORD 2    ;ERROR TYPE CODE.
1605 006452
1606 006452 162700 000002 65$: SUB #2, R0   ;COUNT DOWN WITH ADDRESS
1607 006456 030502      BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1608 006460 001364      BNE 4$      ;BRANCH IF MORE IN CURRENT BLOCK.
1609 006462 004767 006462      JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.
1610
1611 ;*****
1612 ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1613 ;* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
1614 ;* R1 = DATA READ FROM MEMORY (WAS)
1615 ;* R2 = VIRTUAL ADDRESS
1616 ;* R3 = NOT USED
1617 ;* R4 = NOT USED
1618 ;* R5 = BLOCK BOUNDARY BIT MASK.
1619 ;*****
1620 006466
1621 006466 004567 012102 1ST4: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1622 006472 000000      .WORD 0    ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1623 ;* UPWARDS BYTE ADDRESSING.
1624 006474 004467 005672      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1625 006500 004767 007370 1$: JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
1626 006504 110022 2$: MOVB R0, (R2)+ ;WRITE BANK # INTO ALL ADDRESSES
1627 006506 030502      BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1628 006510 001375      BNE 2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1629 006512 004767 006432      JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
1630
1631 ;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1632 ;* UPWARDS BYTE ADDRESSING.
1633 006516 004467 005650      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1634 006522 004767 007346 3$: JSR PC, BANKNO ;GET THE BANK NUMBER INTO R0
1635 006526 112201 4$: MOVB (R2)+, R1 ;READ THE DATA OUT OF MEMORY
1636 006530 020001      CMP R0, R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
1637 006532 001405      BEQ 65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
1638 006534 004767 011554 64$: JSR PC, SPRT1 ;SET UP VALUES FOR ERROR PRINTING.
1639 006540 004767 013040      JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1640 006544 000003      .WORD 3    ;ERROR TYPE CODE.
1641 006546
1642 006546 030502      BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1643 006550 001366      BNE 4$      ;BRANCH IF MORE IN CURRENT BLOCK.
1644 006552 004767 006372      JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 3$.
1645

```

```

1646
1647
1648
1649
1650
1651
1652
1653
1654
1655 006556
1656 006556 004567 012012
1657 006562 000000
1658
1659 006564 004467 006240
1660 006570 004767 007300
1661 006574 005100
1662 006576 110042
1663 006600 030502
1664 006602 001375
1665 006604 004767 007030
1666
1667
1668
1669 006610 004467 006214
1670 006614 004767 007254
1671 006620 005100
1672 006622 114201
1673 006624 020001
1674 006626 001405
1675 006630 004767 011452
1676 006634 004767 012744
1677 006640 000003
1678 006642
1679 006642 030502
1680 006644 001366
1681 006646 004767 006766
    
```

```

*****
;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, $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * DOWNWARDS BYTE ADDRESSING.
      JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, BANKNO ;GET THE BANK 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.
      JSR R4, INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$: JSR PC, BANKNO ;GET THE BANK # INTO R0
      COM R0 ;SET 1'S COMPLEMENT OF BANK #
4$: MOVB -(R2), R1 ;READ DATA OUT OF MEMORY
      CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
      JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 3 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
      BNE 4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.
    
```

```

1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697 006652
1698 006652 004567 011716
1699 006656 000000
1700 006660 016700 172724
1701 006664 004467 005502
1702 006670 010022
1703 006672 030502
1704 006674 001375
1705 006676 004767 006246
1706
1707
1708
1709
1710
1711 006702
1712 006702 004567 011666
1713 006706 000000
1714 006710 016700 172674
1715 006714 004467 005452
1716 006720
1717 006720 012201
1718 006722 020001
1719 006724 001405
1720 006726 004767 011400
1721 006732 004767 012646
1722 006736 000004
1723 006740
1724 006740 030502
1725 006742 001366
1726 006744 004767 006200
1727
1728
1729
1730 006750 032777 000400 172162
1731 006756 001416
1732 006760 017746 172154
1733 006764 042716 177740
1734 006770 022726 000006
1735 006774 001007
1736 006776 162767 000001 172076
1737 007004 162767 000030 172074

```

```

.SBTTL SECTION 2:      WORST CASE NOISE TESTS
*****
* THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
* MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
*****
*TEST 6      WRITE A CONSTANT INTO MEMORY.
* THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
* R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
* R1 = DATA READ FROM MEMORY (WAS)
* R2 = VIRTUAL ADDRESS
* R3 = NOT USED
* R4 = NOT USED
* R5 = BLOCK BOUNDARY BIT MASK.
*****
TST6:      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
           .WORD      0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
TST6A:     MOV      CONST, R0          ;GET USER CONSTANT
           JSR      R4,      INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
IS:        MOV      R0,      (R2)+    ;WRITE CONSTANT INTO MEMORY.
           BIT      R5,      R2        ;CHECK FOR END OF A BLOCK.
           BNE     IS,      $         ;BRANCH IF MORE IN CURRENT BLOCK.
           JSR     PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO IS.
*****
*TEST 7      READ MEMORY AND COMPARE TO CONSTANT.
* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST STN.
*****
TST7:      JSR      R5,      $SCOPE    ;GO TO SCOPE ROUTINE.
           .WORD      0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
           MOV      CONST, R0          ;GET USER CONSTANT
           JSR      R4,      INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
IS:        MOV      (R2)+, R1          ;GET THE DATA FROM MEMORY UNDER TEST.
           CMP      R0,      R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
           BEQ     65$,      $         ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:      JSR      PC,      $SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
           JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
           .WORD      4              ;ERROR TYPE CODE.
65$:      BIT      R5,      R2        ;CHECK FOR END OF A BLOCK.
           BNE     IS,      $         ;BRANCH IF MORE IN CURRENT BLOCK.
           JSR     PC,      MMUP      ;FIND NEXT BLOCK AND LOOP TO IS.
* 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 SW03 ARE SET.
           BIT      #SW08, $SWR      ;CHECK THAT LOOP ON TEST BIT SET
           BEQ     TST10, $SWR      ;BRANCH IF NOT LOOP ON TEST
           MOV     $SWR, -(SP)      ;GET SWITCH REGISTER DATA.
           BIC     #177740, (SP)    ;CLEAR NON-TEST-NUMBER SWITCHES.
           CMP     #6, (SP)+        ;CHECK IF TEST 6 IN SWITCHES.
           BNE     TST10, $         ;BRANCH IF NOT TEST 6
           SUB     #1, $TSTNM      ;RESET TEST NUM
           SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR

```

```

1738 007012 000722
1739
1740
1741
1742
1743
1744 007014
1745 007014 004567 011554
1746 007020 000000
1747 007022 016704 172606
1748 007026 004767 010550
1749 007032 012400
1750 007034 001420
1751 007036 004467 005330
1752 007042 010012
1753 007044 012201
1754 007046 020001
1755 007050 001405
1756 007052 004767 011254
1757 007056 004767 012522
1758 007062 000004
1759 007064
1760 007064 030502
1761 007066 001365
1762 007070 004767 006054
1763 007074 000754

```

```

BR TST6A ;GO TO TEST 6

*****
;TEST 10 WORSE CASE NOISE (PARITY) WORD TESTING
; * CHECK MEMORY WITH A SERIES OF PATTERNS
*****
†ST10:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV .MPPAT, R4 ;INITIALIZE PATTERN TABLE POINTER
1$: JSR PC, CKPMER ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
MOV (R4)+, R0 ;GET THE DATA PATTERN.
BEQ TST11 ;BR IF END OF TABLE.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2$: MOV R0, (R2) ;PUT DATA PATTERN INTO MEMORY.
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
3$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 4 ;ERROR TYPE CODE.
65$:
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 2$.
BR 1$ ;BR BACK TO DO NEXT PATTERN

```



```

1764
1765
1766
1767 007076
1768 007076 004567 011472
1769 007102 000000
1770 007104 012700 177777
1771 007110 004767 007020
1772 007114 004467 005252
1773 007120 000241
1774 007122 004767 007026
1775 007126 016201 177776
1776 007132 103402
1777 007134 020001
1778 007136 001405
1779 007140 004767 011166
1780 007144 004767 012434
1781 007150 000005
1782 007152
1783 007152 030502
1784 007154 001361
1785 007156 004767 005766
1786
1787
1788
1789
1790 007162
1791 007162 004567 011406
1792 007166 000000
1793 007170 005000
1794 007172 004767 006736
1795 007176 004467 005170
1796 007202 000261
1797 007204 004767 006744
1798 007210 016201 177776
1799 007214 103002
1800 007216 020001
1801 007220 001405
1802 007222 004767 011104
1803 007226 004767 012352
1804 007232 000005
1805 007234
1806 007234 030502
1807 007236 001361
1808 007240 004767 005704

```

```

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

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

```

E10

```

1809
1810
1811
1812 007244
1813 007244 004567 011324
1814 007250 000777
1815
1816 007252 000167 000312
1817
1818 007256 005000
1819 007260 012703 177777
1820 007264 004467 005102
1821 007270 004767 006726
1822 007274 030502
1823 007276 001374
1824 007300 004767 005644
1825
1826
1827
1828
1829 007304 005000
1830 007306 004467 005060
1831 007312 012704 000100
1832 007316
1833 007316 012201
1834 007320 020001
1835 007322 001405
1836 007324 004767 011002
1837 007330 004767 012250
1838 007334 000007
1839 007336
1840 007336 012201
1841 007340 020001
1842 007342 001405
1843 007344 004767 010762
1844 007350 004767 012230
1845 007354 000007
1846 007356
1847 007356 012201
1848 007360 020001
1849 007362 001405
1850 007364 004767 010742
1851 007370 004767 012210
1852 007374 000007
1853 007376
1854 007376 012201
1855 007400 020001
1856 007402 001405
1857 007404 004767 010722
1858 007410 004767 012170
1859 007414 000007
1860 007416
1861 007416 005100
1862 007420 005304
1863 007422 001335
1864 007424 005100

```

```

*****
*TEST 13 3 XOR 9 TEST PATTERN.
*****
†ST13:
      JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD  777
      JMP    TST14 ;MINIMUM BLOCK SIZE OF 256. WORDS
                        REQUIRED FOR THIS TEST.
      .3X9: CLR    R0 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                        AVAILABLE FOR TEST.
                        MOV  #-1, R3 ;SET UP TEST DATA
                        JSR  R4,  R3 ;SET COM DATA REG
                        JSR  PC,  W3X9 ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:      BIT    R5,  R2 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      BNE   1$,  R2 ;CHECK FOR END OF A BLOCK.
      JSR   PC,  MMUP ;BRANCH IF MORE IN CURRENT BLOCK.
                        ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
* CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
*****
      CLR    R0 ;SET CHECK WORD
      JSR   R4,  INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$:    MOV   #64., R4 ;SET 256. WORD COUNTER
12$:
      MOV   (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP  R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ  65$, R1 ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR  PC,  SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR  PC,  $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 7 ;ERROR TYPE CODE.
65$:
      MOV   (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP  R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ  67$, R1 ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:    JSR  PC,  SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR  PC,  $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 7 ;ERROR TYPE CODE.
67$:
      MOV   (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP  R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ  69$, R1 ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$:    JSR  PC,  SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR  PC,  $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 7 ;ERROR TYPE CODE.
69$:
      MOV   (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP  R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ  71$, R1 ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$:    JSR  PC,  SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR  PC,  $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD 7 ;ERROR TYPE CODE.
71$:
      COM   R0 ;COMPLEMENT CHECK WORD
      DEC  R4 ;DECREMENT 256. WORD COUNTER
      BNE  12$, R0
      COM   R0 ;COMPLEMENT CHECK WORD

```

# F10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T13

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 40  
 3 XOR 9 TEST PATTERN.

SEQ 0122

1865	007426	030502		BIT	R5	R2	;CHECK FOR END OF A BLOCK.
1866	007430	001330		BNE	11\$		;BRANCH IF MORE IN CURRENT BLOCK.
1867	007432	004767	005512	JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO 11\$.
1868							
1869							
1870				;***** ;* CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE. ;*****			
1871							
1872	007436	005000		CLR	R0		
1873	007440	004467	004726	JSR	R4,	INITMM	;INITIALIZE THE MEMORY ADDRESS POINTERS.
1874	007444	012704	000100	21\$: MOV	#64.,	R4	;SET 256. WORD COUNTER
1875	007450	012703	000004	22\$: MOV	#4,	R3	;SET 4 WORD COUNTER
1876	007454			23\$:			
1877	007454	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1878	007456	020001		CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1879	007460	001405		BEQ	73\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
1880	007462	004767	010644	72\$: JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1881	007466	004767	012112	JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1882	007472	000007		.WORD	7		;ERROR TYPE CODE.
1883	007474			73\$:			
1884	007474	005100		COM	R0		;COMPLEMENT CHECK WORD
1885	007476	005142		COM	-(R2)		;COMPLEMENT TEST DATA
1886	007500	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1887	007502	020001		CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1888	007504	001405		BEQ	75\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
1889	007506	004767	010620	74\$: JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1890	007512	004767	012066	JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1891	007516	000007		.WORD	7		;ERROR TYPE CODE.
1892	007520			75\$:			
1893	007520	005100		COM	R0		;COMPLEMENT CHECK WORD
1894	007522	005142		COM	-(R2)		;COMPLEMENT TEST DATA
1895	007524	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
1896	007526	020001		CMP	R0,	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
1897	007530	001405		BEQ	77\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
1898	007532	004767	010574	76\$: JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
1899	007536	004767	012042	JSR	PC,	\$ERROR	;*** ERROR *** (GO TYPE A MESSAGE)
1900	007542	000007		.WORD	7		;ERROR TYPE CODE.
1901	007544			77\$:			
1902	007544	005303		DEC	R3		;DECREMENT 4 WORD COUNTER
1903	007546	001342		BNE	23\$		;BR IF NOT DONE.
1904	007550	005100		COM	R0		;COMPLEMENT CHECK WORD
1905	007552	005304		DEC	R4		;DECREMENT 256. WORD COUNTER
1906	007554	001335		BNE	22\$		;BR IF NOT DONE.
1907	007556	005100		COM	R0		;COMPLEMENT CHECK WORD
1908	007560	030502		BIT	R5	R2	;CHECK FOR END OF A BLOCK.
1909	007562	001330		BNE	21\$		;BRANCH IF MORE IN CURRENT BLOCK.
1910	007564	004767	005360	JSR	PC,	MMUP	;FIND NEXT BLOCK AND LOOP TO 21\$.

# G10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
CZQMCE.P11 10-JAN-78 12:56 T14

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 41  
COMPLEMENT 3 XOR 9 TEST PATTERN

SEQ 0123

```

1911
1912
1913
1914 007570
1915 007570 004567 011000
1916 007574 000777
1917
1918 007576 000167 000316
1919
1920 007602 012700 177777
1921 007606 005003
1922 007610 004467 004556
1923 007614 004767 006402
1924 007620 030502
1925 007622 001374
1926 007624 004767 005320
1927
1928
1929
1930
1931
1932 007630 012700 177777
1933 007634 004467 004532
1934 007640 012704 000100
1935 007644
1936 007644 012201
1937 007646 020001
1938 007650 001405
1939 007652 004767 010454
1940 007656 004767 011722
1941 007662 000007
1942 007664
1943 007664 012201
1944 007666 020001
1945 007670 001405
1946 007672 004767 010434
1947 007676 004767 011702
1948 007702 000007
1949 007704
1950 007704 012201
1951 007706 020001
1952 007710 001405
1953 007712 004767 010414
1954 007716 004767 011662
1955 007722 000007
1956 007724
1957 007724 012201
1958 007726 020001
1959 007730 001405
1960 007732 004767 010374
1961 007736 004767 011642
1962 007742 000007
1963 007744
1964 007744 005100
1965 007746 005304
1966 007750 001335

```

```

*****
*TEST 14      COMPLEMENT 3 XOR 9 TEST PATTERN
*****
†ST14:
      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.
*****
11$:  MOV      #-1,     R0   ;SET CHECK WORD
      JSR      R4,     INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
12$:  MOV      #64.,    R4   ;SET 256. WORD COUNTER
      MOV      (R2)+,  R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,     R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,     SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7         ;ERROR TYPE CODE.
65$:  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,     $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7         ;ERROR TYPE CODE.
67$:  MOV      (R2)+,  R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,     R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     69$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$:  JSR      PC,     SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7         ;ERROR TYPE CODE.
69$:  MOV      (R2)+,  R1   ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP      R0,     R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     71$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$:  JSR      PC,     SPRT2  ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,     $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    7         ;ERROR TYPE CODE.
71$:  COM      R0       ;COMPLEMENT CHECK WORD
      DEC     R4       ;DECREMENT 256. WORD COUNTER
      BNE     12$,    ;

```

H10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
CZQMCE.P11 10-JAN-78 12:56 T14

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 42  
COMPLEMENT 3 XOR 9 TEST PATTERN

SEQ 0124

1967 007752 005100  
 1968 007754 030502  
 1969 007756 001330  
 1970 007760 004767 005164  
 1971  
 1972  
 1973  
 1974  
 1975 007764 012700 177777  
 1976 007770 004467 004376  
 1977 007774 012704 000100  
 1978 010000 012703 000004  
 1979 010004  
 1980 010004 012201  
 1981 010006 020001  
 1982 010010 001405  
 1983 010012 004767 010314  
 1984 010016 004767 011562  
 1985 010022 000007  
 1986 010024  
 1987 010024 005100  
 1988 010026 005142  
 1989 010030 012201  
 1990 010032 020001  
 1991 010034 001405  
 1992 010036 004767 010270  
 1993 010042 004767 011536  
 1994 010046 000007  
 1995 010050  
 1996 010050 005100  
 1997 010052 005142  
 1998 010054 012201  
 1999 010056 020001  
 2000 010060 001405  
 2001 010062 004767 010244  
 2002 010066 004767 011512  
 2003 010072 000007  
 2004 010074  
 2005 010074 005303  
 2006 010076 001342  
 2007 010100 005100  
 2008 010102 005304  
 2009 010104 001335  
 2010 010106 005100  
 2011 010110 030502  
 2012 010112 001330  
 2013 010114 004767 005030

```

COM R0 ;COMPLEMENT CHECK WORD
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 11$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.

*****
* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
*****

MOV #-1, R0 ;SET UP CHECK WORD.
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
21$: MOV #64., R4 ;SET 256. WORD COUNTER
22$: MOV #4, R3 ;SET 4 WORD COUNTER
23$:

MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 73$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
72$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
73$:

COM R0 ;COMPLEMENT CHECK WORD
COM -(R2) ;COMPLEMENT TEST DATA
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 75$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
74$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
75$:

COM R0 ;COMPLEMENT CHECK WORD
COM -(R2) ;COMPLEMENT TEST DATA
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 77$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
76$: JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
77$:

DEC R3 ;DECREMENT 4 WORD COUNTER
BNE 23$ ;BR IF NOT DONE.
COM R0 ;COMPLEMENT CHECK WORD
DEC R4 ;DECREMENT 256. WORD COUNTER
BNE 22$ ;BR IF NOT DONE.
COM R0 ;COMPLEMENT CHECK WORD
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 21$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.

```

```

2014
2015
2016
2017 010120
2018 010120 004567 010450
2019 010124 000777
2020
2021 010126 000167 000610
2022
2023 010132 012700 000401
2024 010136 012703 177777
2025 010142 004467 004224
2026 010146 004767 006050
2027 010152 030502
2028 010154 001374
2029 010156 004767 004766
2030
2031
2032
2033
2034 010162 012700 000401
2035 010166 012703 177777
2036 010172 004467 004174
2037 010176 012704 000100
2038 010202
2039 010202 012201
2040 010204 020001
2041 010206 001405
2042 010210 004767 010116
2043 010214 004767 011364
2044 010220 000007
2045 010222
2046 010222 012201
2047 010224 020001
2048 010226 001405
2049 010230 004767 010076
2050 010234 004767 011344
2051 010240 000007
2052 010242
2053 010242 012201
2054 010244 020001
2055 010246 001405
2056 010250 004767 010056
2057 010254 004767 011324
2058 010260 000007
2059 010262
2060 010262 012201
2061 010264 020001
2062 010266 001405
2063 010270 004767 010036
2064 010274 004767 011304
2065 010300 000007
2066 010302
2067 010302 010046
2068 010304 010300
2069 010306 012603

```

```

*****
*TEST 15      MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
*****
†T15:
      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
      .WORD   777
      JMP      TST16
      MOV      #401,    R0          ;SET UP PARITY "ALL ZEROS" PATTERN
      MOV      #-1,    R3          ;SET COM DATA REG
      JSR      R4,      INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      W3X9       ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      BIT      R5,      R2          ;CHECK FOR END OF A BLOCK.
      BNE     1$,      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR      PC,      MMUP       ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
* CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
*****
      MOV      #401,    R0          ;RESET PARITY "ALL ZEROS" PATTERN.
      MOV      #-1,    R3          ;RESET PARITY ALL ONES PATTERN.
      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,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7
65$:
      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,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7
67$:
      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,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7
69$:
      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,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   7
71$:
      MOV      R0,      -(SP)      ;SAVE R0
      MOV      R3,      R0          ;PUT R3 INTO R0
      MOV      (SP)+,   R3          ;PUT SAVED R0 INTO R3

```

# J10

CZQMCE0 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T15

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 44  
 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

SEQ 0126

2070	010310	005304		DEC	R4	; COUNT 256. WORDS
2071	010312	001333		BNE	12\$	; BRANCH IF MORE
2072	010314	010046		MOV	RO, -(SP)	; SAVE RO
2073	010316	010300		MOV	R3, RO	; PUT R3 INTO RO
2074	010320	012603		MOV	(SP)+, R3	; PUT SAVED RO INTO R3
2075	010322	030502		BIT	R5, R2	; CHECK FOR END OF A BLOCK.
2076	010324	001324		BNE	11\$	; BRANCH IF MORE IN CURRENT BLOCK.
2077	010326	004767	004616	JSR	PC, MMUP	; FIND NEXT BLOCK AND LOOP TO 11\$.
2078						
2079						
2080						
2081						
2082	010332	012700	000401			
2083	010336	012703	177777			
2084	010342	004467	004024			
2085	010346	012704	000100			
2086	010352					
2087	010352	012201				
2088	010354	020001				
2089	010356	001405				
2090	010360	004767	007746			
2091	010364	004767	011214			
2092	010370	000007				
2093	010372					
2094	010372	005100				
2095	010374	005142				
2096	010376	012201				
2097	010400	020001				
2098	010402	001405				
2099	010404	004767	007722			
2100	010410	004767	011170			
2101	010414	000007				
2102	010416					
2103	010416	005100				
2104	010420	005142				
2105	010422	012201				
2106	010424	020001				
2107	010426	001405				
2108	010430	004767	007676			
2109	010434	004767	011144			
2110	010440	000007				
2111	010442					
2112	010442	012201				
2113	010444	020001				
2114	010446	001405				
2115	010450	004767	007656			
2116	010454	004767	011124			
2117	010460	000007				
2118	010462					
2119	010462	005100				
2120	010464	005142				
2121	010466	012201				
2122	010470	020001				
2123	010472	001405				
2124	010474	004767	007632			
2125	010500	004767	011100			

```

;*****
; * CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
;*****
MOV #401, RO ; SET UP PARITY "ALL ZEROS" PATTERN.
MOV #-1, R3 ; SET UP ALL ONES PATTERN.
JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
MOV #64., R4 ; SET 256. WORD COUNTER

21$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
22$: CMP RO, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 73$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
72$: JSR PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ; ERROR TYPE CODE.

73$: COM RO ; COMPLEMENT CHECK WORD
COM -(R2) ; COMPLEMENT TEST DATA
MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 75$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
74$: JSR PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ; ERROR TYPE CODE.

75$: COM RO ; COMPLEMENT CHECK WORD
COM -(R2) ; RESTORE DATA
MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 77$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
76$: JSR PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ; ERROR TYPE CODE.

77$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 79$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
78$: JSR PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ; ERROR TYPE CODE.

79$: COM RO ; COMPLEMENT CHECK WORD
COM -(R2) ; COMPLEMENT TEST DATA
MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
CMP RO, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 81$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
80$: JSR PC, SPRT2 ; SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)

```

2126	010504	000007			.WORD	7		; ERROR TYPE CODE.
2127	010506		81\$:					
2128	010506	005100			COM	RO		; COMPLEMENT CHECK WORD
2129	010510	005142			COM	-(R2)		; RESTORE DATA
2130	010512	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2131	010514	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2132	010516	001405			BEQ	83\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2133	010520	004767	007606	82\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2134	010524	004767	011054		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2135	010530	000007			.WORD	7		; ERROR TYPE CODE.
2136	010532		83\$:					
2137	010532	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2138	010534	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2139	010536	001405			BEQ	85\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2140	010540	004767	007566	84\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2141	010544	004767	011034		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2142	010550	000007			.WORD	7		; ERROR TYPE CODE.
2143	010552		85\$:					
2144	010552	005100			COM	RO		; COMPLEMENT CHECK WORD
2145	010554	005142			COM	-(R2)		; COMPLEMENT TEST DATA
2146	010556	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2147	010560	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2148	010562	001405			BEQ	87\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2149	010564	004767	007542	86\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2150	010570	004767	011010		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2151	010574	000007			.WORD	7		; ERROR TYPE CODE.
2152	010576		87\$:					
2153	010576	005100			COM	RO		; COMPLEMENT CHECK WORD
2154	010600	005142			COM	-(R2)		; RESTORE DATA
2155	010602	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2156	010604	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2157	010606	001405			BEQ	89\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2158	010610	004767	007516	88\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2159	010614	004767	010764		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2160	010620	000007			.WORD	7		; ERROR TYPE CODE.
2161	010622		89\$:					
2162	010622	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2163	010624	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2164	010626	001405			BEQ	91\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2165	010630	004767	007476	90\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2166	010634	004767	010744		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2167	010640	000007			.WORD	7		; ERROR TYPE CODE.
2168	010642		91\$:					
2169	010642	005100			COM	RO		; COMPLEMENT CHECK WORD
2170	010644	005142			COM	-(R2)		; COMPLEMENT TEST DATA
2171	010646	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2172	010650	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2173	010652	001405			BEQ	93\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2174	010654	004767	007452	92\$:	JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2175	010660	004767	010720		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2176	010664	000007			.WORD	7		; ERROR TYPE CODE.
2177	010666		93\$:					
2178	010666	005100			COM	RO		; COMPLEMENT CHECK WORD
2179	010670	005142			COM	-(R2)		; RESTORE DATA
2180	010672	012201			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2181	010674	020001			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.



# L10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T15

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 46  
 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

SEQ 0128

```

2182 010676 001405          BEQ      95$      ; BRANCH OVER ERROR CALL IF GOOD DATA.
2183 010700 004767 007426 94$: JSR      PC,      SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
2184 010704 004767 010674 JSR      PC,      $ERROR   ; *** ERROR *** (GO TYPE A MESSAGE)
2185 010710 000007          .WORD    7              ; ERROR TYPE CODE.
2186 010712          95$:
2187 010712 010046          MOV      R0,      -(SP)   ; SAVE R0
2188 010714 010300          MOV      R3,      R0     ; PUT R3 INTO R0
2189 010716 012603          MOV      (SP)+,   R3     ; PUT SAVED R0 INTO R3
2190 010720 005304          DEC      R4           ; DECREMENT 256. WORD COUNTER
2191 010722 001213          BNE     22$         ; BRANCH IF MORE.
2192 010724 010046          MOV      R0,      -(SP)   ; SAVE R0
2193 010726 010300          MOV      R3,      R0     ; PUT R3 INTO R0
2194 010730 012603          MOV      (SP)+,   R3     ; PUT SAVED R0 INTO R3
2195 010732 030502          BIT      R5,      R2     ; CHECK FOR END OF A BLOCK.
2196 010734 001204          BNE     21$         ; BRANCH IF MORE IN CURRENT BLOCK.
2197 010736 004767 004206 JSR      PC,      MMUP    ; FIND NEXT BLOCK AND LOOP TO 21$.
2198
2199
2200 ; *****
2201 ; *TEST 16      COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2202 ; *****
2203 010742          TST16:
2204 010742 004567 007626 JSR      R5,      $SCOPE  ; GO TO SCOPE ROUTINE.
2205 010746 000777          .WORD    777          ; MINIMUM BLOCK SIZE OF 256. WORDS
2206 010750 000167 000610 JMP      TST17         ; REQUIRED FOR THIS TEST.
2207 ; SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2208 010754 012700 177777          MOV      #-1,     R0     ; AVAILABLE FOR TEST.
2209 010760 012703 000401          MOV      #401,   R3     ; SET UP ALL ONES PATTERN
2210 010764 004467 003402          JSR      R4,     INITMM  ; SET UP PARITY "ALL ZEROS" PATTERN
2211 010770 004767 005226 1$: JSR      PC,      W3X9   ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2212 010774 030502          BIT      R5,      R2     ; WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2213 010776 001374          BNE     1$,      MMUP    ; CHECK FOR END OF A BLOCK.
2214 011000 004767 004144 JSR      PC,      MMUP    ; BRANCH IF MORE IN CURRENT BLOCK.
2215 ; FIND NEXT BLOCK AND LOOP TO 1$.
2216
2217 ; *****
2218 ; * CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2219 ; *****
2220 011004 012700 177777          MOV      #-1,     R0     ; SET UP ALL ONES PATTERN
2221 011010 012703 000401          MOV      #401,   R3     ; SET UP PARITY "ALL ZEROS" PATTERN
2222 011014 004467 003352          JSR      R4,     INITMM  ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2223 011020 012704 000100 11$: MOV      #64.,   R4     ; SET 256. WORD COUNTER
2224 011024          12$:
2225 011026 012201          MOV      (R2)+,  R1     ; GET THE DATA FROM MEMORY UNDER TEST.
2226 011030 020001          CMP      R0,      R1     ; COMPARE THE CHECK WORD WITH THE DATA READ.
2227 011032 004767 007274 64$: BEQ      65$         ; BRANCH OVER ERROR CALL IF GOOD DATA.
2228 011036 004767 010542 JSR      PC,      SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
2229 011042 000007          .WORD    7              ; *** ERROR *** (GO TYPE A MESSAGE)
2230 011044          65$:
2231 011044 012201          MOV      (R2)+,  R1     ; GET THE DATA FROM MEMORY UNDER TEST.
2232 011046 020001          CMP      R0,      R1     ; COMPARE THE CHECK WORD WITH THE DATA READ.
2233 011050 001405          BEQ      67$         ; BRANCH OVER ERROR CALL IF GOOD DATA.
2234 011052 004767 007254 66$: JSR      PC,      SPRINT2 ; SET UP VALUES FOR ERROR PRINTING.
2235 011056 004767 010522 JSR      PC,      $ERROR   ; *** ERROR *** (GO TYPE A MESSAGE)
2236 011062 000007          .WORD    7              ; ERROR TYPE CODE.
2237 011064          67$:

```

M10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T16

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 47  
 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0129

2238	011064	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2239	011066	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2240	011070	001405		BEQ	69\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
2241	011072	004767	007234	JSR	PC,	SPRNT2		;SET UP VALUES FOR ERROR PRINTING.
2242	011076	004767	010502	JSR	PC,	SERROR		;*** ERROR *** (GO TYPE A MESSAGE)
2243	011102	000007		.WORD	7			;ERROR TYPE CODE.
2244	011104						69\$:	
2245	011104	012201		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2246	011106	020001		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2247	011110	001405		BEQ	71\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
2248	011112	004767	007214	JSR	PC,	SPRNT2		;SET UP VALUES FOR ERROR PRINTING.
2249	011116	004767	010462	JSR	PC,	SERROR		;*** ERROR *** (GO TYPE A MESSAGE)
2250	011122	000007		.WORD	7			;ERROR TYPE CODE.
2251	011124						71\$:	
2252	011124	010046		MOV	RO,	-(SP)		;SAVE RO
2253	011126	010300		MOV	R3,	RO		;PUT R3 INTO RO
2254	011130	012603		MOV	(SP)+,	R3		;PUT SAVED RO INTO R3
2255	011132	005304		DEC	R4			;COUNT 256. WORDS
2256	011134	001333		BNE	12\$			;BRANCH IF MORE
2257	011136	010046		MOV	RO,	-(SP)		;SAVE RO
2258	011140	010300		MOV	R3,	RO		;PUT R3 INTO RO
2259	011142	012603		MOV	(SP)+,	R3		;PUT SAVED RO INTO R3
2260	011144	030502		BIT	R5,	R2		;CHECK FOR END OF A BLOCK.
2261	011146	001324		BNE	11\$			;BRANCH IF MORE IN CURRENT BLOCK.
2262	011150	004767	003774	JSR	PC,	MMUP		;FIND NEXT BLOCK AND LOOP TO 11\$.
2263								
2264								
2265								
2266								
2267	011154	012700	177777					
2268	011160	012703	000401					
2269	011164	004467	003202					
2270	011170	012704	000100					
2271	011174						21\$:	
2272	011174	012201					22\$:	
2273	011176	020001		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2274	011200	001405		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2275	011202	004767	007124	BEQ	73\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
2276	011206	004767	010372	JSR	PC,	SPRNT2		;SET UP VALUES FOR ERROR PRINTING.
2277	011212	000007		JSR	PC,	SERROR		;*** ERROR *** (GO TYPE A MESSAGE)
2278	011214			.WORD	7			;ERROR TYPE CODE.
2279	011214	005100					73\$:	
2280	011216	005142		COM	RO			;COMPLEMENT CHECK WORD
2281	011220	012201		COM	-(R2)			;COMPLEMENT TEST DATA
2282	011222	020001		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2283	011224	001405		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2284	011226	004767	007100	BEQ	75\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
2285	011232	004767	010346	JSR	PC,	SPRNT2		;SET UP VALUES FOR ERROR PRINTING.
2286	011236	000007		JSR	PC,	SERROR		;*** ERROR *** (GO TYPE A MESSAGE)
2287	011240			.WORD	7			;ERROR TYPE CODE.
2288	011240	005100					75\$:	
2289	011242	005142		COM	RO			;COMPLEMENT CHECK WORD
2290	011244	012201		COM	-(R2)			;RESTORE DATA
2291	011246	020001		MOV	(R2)+,	R1		;GET THE DATA FROM MEMORY UNDER TEST.
2292	011250	001405		CMP	RO,	R1		;COMPARE THE CHECK WORD WITH THE DATA READ.
2293	011252	004767	007054	BEQ	77\$			;BRANCH OVER ERROR CALL IF GOOD DATA.
				JSR	PC,	SPRNT2		;SET UP VALUES FOR ERROR PRINTING.

```

*****
* CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
*****

```

# N10

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
CZQMCE.P11 10-JAN-78 12:56 T16

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 48  
COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0130

2294	011256	004767	010322	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2295	011262	000007		.WORD	7		;ERROR TYPE CODE.
2296	011264		77\$:				
2297	011264	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2298	011266	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2299	011270	001405		BEQ	79\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2300	011272	004767	007034	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2301	011276	004767	010302	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2302	011302	000007		.WORD	7		;ERROR TYPE CODE.
2303	011304		79\$:				
2304	011304	005100		COM	RO		;COMPLEMENT CHECK WORD
2305	011306	005142		COM	-(R2)		;COMPLEMENT TEST DATA
2306	011310	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2307	011312	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2308	011314	001405		BEQ	81\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2309	011316	004767	007010	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2310	011322	004767	010256	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2311	011326	000007		.WORD	7		;ERROR TYPE CODE.
2312	011330		81\$:				
2313	011330	005100		COM	RO		;COMPLEMENT CHECK WORD
2314	011332	005142		COM	-(R2)		;RESTORE DATA
2315	011334	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2316	011336	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2317	011340	001405		BEQ	83\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2318	011342	004767	006764	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2319	011346	004767	010232	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2320	011352	000007		.WORD	7		;ERROR TYPE CODE.
2321	011354		83\$:				
2322	011354	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2323	011356	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2324	011360	001405		BEQ	85\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2325	011362	004767	006744	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2326	011366	004767	010212	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2327	011372	000007		.WORD	7		;ERROR TYPE CODE.
2328	011374		85\$:				
2329	011374	005100		COM	RO		;COMPLEMENT CHECK WORD
2330	011376	005142		COM	-(R2)		;COMPLEMENT TEST DATA
2331	011400	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2332	011402	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2333	011404	001405		BEQ	87\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2334	011406	004767	006720	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2335	011412	004767	010166	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2336	011416	000007		.WORD	7		;ERROR TYPE CODE.
2337	011420		87\$:				
2338	011420	005100		COM	RO		;COMPLEMENT CHECK WORD
2339	011422	005142		COM	-(R2)		;RESTORE DATA
2340	011424	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2341	011426	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2342	011430	001405		BEQ	89\$		;BRANCH OVER ERROR CALL IF GOOD DATA.
2343	011432	004767	006674	JSR	PC,	SPRNT2	;SET UP VALUES FOR ERROR PRINTING.
2344	011436	004767	010142	JSR	PC,	\$ERROR	*** ERROR *** (GO TYPE A MESSAGE)
2345	011442	000007		.WORD	7		;ERROR TYPE CODE.
2346	011444		89\$:				
2347	011444	012201		MOV	(R2)+,	R1	;GET THE DATA FROM MEMORY UNDER TEST.
2348	011446	020001		CMP	RO	R1	;COMPARE THE CHECK WORD WITH THE DATA READ.
2349	011450	001405		BEQ	91\$		;BRANCH OVER ERROR CALL IF GOOD DATA.

# B11

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T16

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 49  
 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEQ 0131

2350	011452	004767	006654	90\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2351	011456	004767	010122		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2352	011462	000007			.WORD	7		:ERROR TYPE CODE.
2353	011464			91\$:				
2354	011464	005100			COM	RO		:COMPLEMENT CHECK WORD
2355	011466	005142			COM	-(R2)		:COMPLEMENT TEST DATA
2356	011470	012201			MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2357	011472	020001			CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2358	011474	001405			BEQ	93\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2359	011476	004767	006630	92\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2360	011502	004767	010076		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2361	011506	000007			.WORD	7		:ERROR TYPE CODE.
2362	011510			93\$:				
2363	011510	005100			COM	RO		:COMPLEMENT CHECK WORD
2364	011512	005142			COM	-(R2)		:RESTORE DATA
2365	011514	012201			MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2366	011516	020001			CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2367	011520	001405			BEQ	95\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2368	011522	004767	006604	94\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2369	011526	004767	010052		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2370	011532	000007			.WORD	7		:ERROR TYPE CODE.
2371	011534			95\$:				
2372	011534	010046			MOV	RO,	-(SP)	:SAVE RO
2373	011536	010300			MOV	R3,	RO	:PUT R3 INTO RO
2374	011540	012603			MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
2375	011542	005304			DEC	R4		:DECREMENT 256. WORD COUNTER
2376	011544	001213			BNE	22\$		:BRANCH IF MORE.
2377	011546	010046			MOV	RO,	-(SP)	:SAVE RO
2378	011550	010300			MOV	R3,	RO	:PUT R3 INTO RO
2379	011552	012603			MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
2380	011554	030502			BIT	R5,	R2	:CHECK FOR END OF A BLOCK.
2381	011556	001204			BNE	21\$		:BRANCH IF MORE IN CURRENT BLOCK.
2382	011560	004767	003364		JSR	PC,	MMUP	:FIND NEXT BLOCK AND LOOP TO 21\$.

```

2383 .....*****
2384 *TEST 17 WORSE CASE NOISE PARITY BYTE TESTING
2385 * CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
2386 * 1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
2387 * 2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
2388 * 3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
2389 * 4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
2390 .....*****
2391 TST17:
2392 011564 004567 007004 JSR R5, $SCOPE ; GO TO SCOPE ROUTINE.
2393 011570 000000 .WORD 0 ; NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2394 011572 005767 170500 WWPB0: TST MPRX ; CHECK FOR ANY PARITY MEMORY.
2395 011576 001404 BEQ 1$ ; BR IF NO PARITY MEMORY.
2396 011600 032777 000100 167332 BIT #SW06, @SWR ; CHECK FORINHIBIT PARITY SWITCH.
2397 011606 001402 BEQ 2$ ; BR IF NOT SET.
2398 011610 000167 000622 1$: JMP TST20 ; SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2399 011614 005000 CLR R0 ; ZERO TO BE PUT IN ALL MEMORY.
2400 011616 004767 004312 JSR PC, SETCON ; ROUTINE TO LOAD ALL MEMORY.
2401 011622 004467 002544 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2402 011626 036767 167712 167704 WWPBYT: BIT BITPT, PMEMAP ; CHECK IF CURRENT BANK HAS PARITY MEMORY.
2403 011634 001010 BNE 2$ ; BR IF PARITY MEM.
2404 011636 036767 167704 167676 BIT BITPT+2, PMEMAP+2 ; ...HI 64K.
2405 011644 001004 BNE 2$ ; BR IF PARITY MEM.
2406 011646 050502 BIS R5, R2 ; POINT TO END OF BLOCK.
2407 011650 005202 INC R2 ; FIRST ADR OF NEXT BLOCK.
2408 011652 000167 000540 JMP WWPB5 ; BR TO FIND NEXT BLOCK.
2409 011656 004767 005664 2$: JSR PC, SETAE ; SET ACTION ENABLE (EVEN IF BANK0.)
2410 011662 004767 005714 JSR PC, CKPMER ; CHECK FOR ANY NON TRAP PARITY ERRORS.
2411 011666 020227 000114 WWPB1: CMP R2, #114 ; CKECK IF POINTING TO PARITY ERROR VECTOR.
2412 011672 001004 BNE 3$ ; BR IF NOT AT VECTOR.
2413 011674 062702 000004 ADD #4, R2 ; SKIP PARITY VECTOR.
2414 011700 000167 000512 JMP WWPB5 ; CHECK FOR BLOCK END.
2415 011704 111201 3$: MOVB (R2), R1 ; CHECK IF BYTE STILL CLEARED.
2416 011706 001405 BEQ 65$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
2417 011710 004767 006342 64$: JSR PC, SPRT ; SET UP VALUES FOR ERROR PRINTING.
2418 011714 004767 007664 JSR PC, SERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2419 011720 000011 .WORD 11 ; ERROR TYPE CODE.
2420 011722 65$:
2421 011722 105067 167632 CLRB OEFLG ; CLEAR ODD/EVEN FLAG.
2422 011726 112700 000252 MOVB #252, R0 ; SET UP DATA...EVEN, SETS PARITY BIT.
2423 011732 110012 WWPB2: RO, (R2) ; MOV DATA INTO TEST LOCATION.
2424 011734 016703 167670 MOV .MPRX, R3 ; GET PARITY REGISTER TABLE POINTER.
2425 011740 056773 167646 000000 10$: BIS WWP, @ (R3) ; SET WRITE WRONG PARITY.
2426 011746 052733 000001 BIS #AE, @ (R3)+
2427 011752 005713 TST (R3) ; CHECK FOR TABLE TERMINATOR.
2428 011754 001371 BNE 10$ ; BR IF MORE REGS IN TABLE.
2429 ;* SET WRONG PARITY IN LOCATION UNDER TEST.
2430 011756 110012 MOVB RO, (R2) ; WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
2431 011760 016703 167644 MOV .MPRX, R3 ; GET PARITY REG TABLE POINTER.
2432 011764 046733 167622 11$: BIC WWP, @ (R3)+ ; CLEAR WRITE WRONG PARITY.
2433 011770 005713 TST (R3) ; CHECK FOR TABLE TERMINATOR.
2434 011772 001374 BNE 11$ ; BR IF MORE PARITY REGISTERS.
2435 011774 016737 167632 000114 MOV .PBTRP, @#PARVEC ; SET UP VECTOR FOR EXPECTED TRAP.
2436 ;* DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
2437 012002 105412 NEGB (R2) ; DATIP (DATOB AND COM PARITY BIT.)
2438 ;* SHOULD HAVE TRAPPED TO PBTRP.

```

```

2439 012004 016737 167626 000114 MOV .PESRV, 3#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
2440 012012 004767 006270 64$: JSR PC, SPRTD ;SET UP VALUES FOR ERROR PRINTING.
2441 012016 004767 007562 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2442 012022 000012 .WORD 12 ;ERROR TYPE CODE.
2443 012024 000562 BR WWPB4 ;SKIP TRAP SERVICE.
2444
2445
2446 012026 016737 167604 000114 ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
2447 012034 022626 PBTRP: MOV .PESRV, 3#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2448 012036 016703 CMP (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
2449 012042 032713 MOV .MPRO, R3 ;GET PARITY REG AND MAP TABLE POINTER.
2450 012046 001003 21$: BIT #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
2451 012050 017301 BNE 22$ ;BR IF IT DOESN'T EXIST.
2452 012054 100413 MOV 2(R3), R1 ;GET THE CONTENTS.
2453 012056 062703 22$: BMI 23$ ;BR IF ERROR FLAG SET.
2454 012062 020367 ADD #10, R3 ;MOVE POINTER TO NEXT REG.
2455 012066 103765 BLO 21$ .MPRX ;CHECK FOR END OF TABLE.
2456 012070 004767 64$: JSR PC, SPRTD ;SET UP VALUES FOR ERROR PRINTING.
2457 012074 004767 007504 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2458 012100 000013 .WORD 13 ;ERROR TYPE CODE.
2459 012102 000533 BR WWPB4 ;EXIT AFTER ERROR.
2460 012104 036763 167434 000002 23$: BIT BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
2461 012112 001011 BNE 24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2462 012114 036763 167426 000004 BIT BITPT+2, 4(R3) ;CHECK THE HI 64K.
2463 012122 001005 BNE 24$ ;BR IF THIS REGISTER CONTROLS THIS BANK.
2464 012124 004767 006152 65$: JSR PC, SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2465 012130 004767 007450 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2466 012134 000014 .WORD 14 ;ERROR TYPE CODE.
2467 012136 24$:
2468 012136 010046 MOV R0, -(SP) ;: PUSH R0 ON STACK
2469 012140 010200 MOV R2, R0 ;: GET THE ADDRESS POINTER.
2470 012142 042700 003777 BIC #3777, R0 ;: CLEAR LOW ADDRESS BITS.
2471 012146 000300 SWAB R0 ;: SHIFT 6 PLACES RIGHT.
2472 012150 006300 ASL R0
2473 012152 006300 ASL R0
2474 012154 005767 166426 TST MMAVA ;: CHECK FOR MEM MGMT.
2475 012160 001404 BEQ 25$ ;: BR IF NO MEM MGMT.
2476 012162 042700 177600 BIC #177600, R0 ;: CLEAR BANK BITS
2477 012166 063700 172344 ADD 3#KIPAR2, R0 ;: ADD MEM MGMT OFFSET.
2478 012172 052700 100001 25$: BIS #BIT15+BIT0, R0 ;: SET ERROR AND AE BIT IN CHECK WORD.
2479 012176 016367 000006 167312 MOV 6(R3), RESRVD ;: GET APPROPRIATE MASK.
2480 012204 046700 167306 BIC RESRVD, R0 ;: CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2481 012210 046701 167302 BIC RESRVD, R1 ;: CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2482 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2483 012214 020001 CMP R0, R1 ;: COMPARE THE CHECK WORD WITH THE DATA READ.
2484 012216 001405 BEQ 67$ ;: BRANCH OVER ERROR CALL IF GOOD DATA.
2485 012220 004767 006056 66$: JSR PC, SPRNTP ;: SET UP VALUES FOR ERROR PRINTING.
2486 012224 004767 007354 JSR PC, $ERROR ;: *** ERROR *** (GO TYPE A MESSAGE)
2487 012230 000015 .WORD 15 ;: ERROR TYPE CODE.
2488 012232 67$:
2489 012232 005073 000000 CLR 2(R3) ;: CLEAR REG INCLUDING ACTION ENABLE.
2490 012236 010346 MOV R3, -(SP) ;: PUSH R3 ON STACK
2491 012240 062703 000010 26$: ADD #10, R3 ;: UPDATE POINTER TO NEXT PARITY REG + MAP.
2492 012244 020367 167360 CMP R3, .MPRX ;: CHECK FOR END OF TABLE.
2493 012250 101014 BHI WWPB3 ;: BR IF END OF TABLE REACHED.
2494 012252 032713 000001 BIT #BIT0, (R3) ;: CHECK IF NEXT REG EXISTS.

```

```

2495 012256 001370      BNE      26$      ;BR IF THIS PARITY REG DOESN'T EXIST.
2496 012260 017301 000000  MOV      @R3), R1 ;SAVE AND CHECK FOR ERROR FLAG.
2497 012264 100365      BPL      26$      ;BR IF NO ERROR FLAG.
2498 012266 004767 006010 68$: JSR      PC,      SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2499 012272 004767 007306  JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2500 012276 000016      .WORD   16       ;ERROR TYPE CODE.
2501 012300 000757      BR       26$      ;BR AFTER ERROR.
2502 012302 111204  WWPB3: MOVB   (R2), R4 ;GET THE DATA FOR CHECKING.
2503 ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
2504 ;* ACTION ENABLE IS NOT SET IN CONTROLLING REG, SO NO TRAP SHOULD OCCURE.
2505 012304 111212      MOVB   (R2), (R2) ;RESTORE RIGHT PARITY
2506 ;NOTE: THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
2507 ; WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
2508 012306 012603      MOV      (SP)+, R3 ;POP STACK INTO R3
2509 012310 017301 000000  MOV      @R3), R1 ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
2510 012314 046701 167176  BIC      RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2511 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2512 012320 042700 000001  BIC      #AE, R0 ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
2513 012324 020001      CMP      R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2514 012326 001405      BEQ     65$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2515 012330 004767 005746 64$: JSR      PC,      SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2516 012334 004767 007244  JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2517 012340 000015      .WORD   15       ;ERROR TYPE CODE.
2518 012342
2519 012342 012773 000001 000000 65$: MOV      #1, @R3) ;CLEAR ALL BUT ACTION ENABLE.
2520 012350 010401      MOV      R4, R1 ;GET DATA READ FROM MEMORY FOR TESTING.
2521 012352 012600      MOV      (SP)+, R0 ;POP STACK INTO R0
2522 012354 120001      CMPB   R0, R1 ;CHECK THE DATA.
2523 012356 001405      BEQ     67$      ;BRANCH OVER ERROR CALL IF GOOD DATA.
2524 012360 004767 005722 66$: JSR      PC,      SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2525 012364 004767 007214  JSR      PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2526 012370 000017      .WORD   17       ;ERROR TYPE CODE.
2527 012372
2528 012372 110012  WWPB4: MOVB   R0, (R2) ;RESTORE DATA.
2529 012374 105712      TSTB   (R2) ;DO A DATI TO BE SURE RIGHT PARITY.
2530 012376 012700 000253  MOV      #253, R0 ;SET ODD PARITY DATA.
2531 012402 105167 167152  COMB   OEFLG ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
2532 012406 100002      BPL      27$      ;BR IF DONE BOTH EVEN AND ODD.
2533 012410 000167 177316  JMP      WWPB2 ;LOOP BACK AND DO ODD(PARITY BIT CLR).
2534 012414 005202 27$: INC      R2 ;MOVE POINTER TO NEXT MEMORY BYTE.
2535 012416 030502  WWPB5: BIT      R5, R2 ;CHECK FOR END OF BLOCK.
2536 012420 001402      BEQ     30$      ;BR IF END OF BLOCK FOUND.
2537 012422 000167 177240  JMP      WWPB1 ;LOOP BACK TO TEST NEXT BYTE.
2538 012426 004767 002516 30$: JSR      PC,      MMUP ;FIND NEXT BLOCK AND LOOP TO WWPBYT
2539 012432 004767 005044  JSR      PC,      MAMF ;GO RESET PARITY REGISTERS.

```

```

2540      ;:*****
2541      ;:TEST 20      RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
2542      ;:*****
2543      †ST20:
2544      012436      004567      006132      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
2545      012436      000000      .WORD      0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2546      012442      010703      RANTST:  MOV      PC,      R3      ;GET CURRENT PROGRAM COUNTER.
2547      012444      042703      007777      BIC      #7777,   R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
2548      012446      004467      001714      JSR      R4,      INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2549      012452      010246      1$:      MOV      R2,      -(SP)     ;SAVE MEMORY POINTER.
2550      012456      010346      MOV      R3,      -(SP)     ;SAVE "DATA" POINTER.
2551      012462      012322      2$:      MOV      (R3)+,   (R2)+   ;MOV CODE INTO TEST MEMORY.
2552      012464      032703      007777      BIT      #7777,   R3      ;CHECK FOR END OF "DATA TABLE"
2553      012470      001002      BNE      3$      ;BRANCH IF MORE
2554      012472      162703      010000      SUB      #10000,  R3      ;RESET POINTER TO START OF "RANDOM DATA"
2555      012476      030502      3$:      BIT      R5,      R2      ;CHECK FOR END OF BLOCK
2556      012500      001370      BNE      2$      ;BRANCH IF MORE
2557      012502      012603      MOV      (SP)+,   R3      ;RESET "DATA" POINTER.
2558      012504      012602      MOV      (SP)+,   R2      ;RESET MEMORY POINTER.
2559      012506      012300      4$:      MOV      (R3)+,   R0      ;GET S/B DATA.
2560      012510      012201      MOV      (R2)+,   R1      ;GET THE DATA FROM MEMORY UNDER TEST.
2561      012512      020001      CMP      R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
2562      012514      001405      BEQ      65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
2563      012516      004767      005610      64$:      JSR      PC,      SPRINT2   ;SET UP VALUES FOR ERROR PRINTING.
2564      012522      004767      007056      JSR      PC,      $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
2565      012526      000020      .WORD      20          ;ERROR TYPE CODE.
2566      012530      032703      007777      65$:      BIT      #7777,   R3      ;CHECK FOR END OF "DATA TABLE"
2567      012530      001002      BNE      5$      ;BR IF MORE.
2568      012534      001002      SUB      #10000,  R3      ;RESET POINTER TO TOP OF "DATA TABLE".
2569      012536      162703      010000      5$:      BIT      R5,      R2      ;CHECK FOR END OF A BLOCK.
2570      012542      030502      BNE      4$      ;BRANCH IF MORE IN CURRENT BLOCK.
2571      012542      001360      JSR      PC,      MMUP     ;FIND NEXT BLOCK AND LOOP TO 1$.
2572      012544      004767      002376
2573      012546

```



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  
2614  
2615  
2616  
2617  
2618  
2619  
2620  
2621  
2622  
2623

012552  
012552 004567 006016  
012556 000003  
012560 000167 000056  
012564 012703 010412  
012570 012704 000205  
012574 010400  
012576 004467 001570  
012602 010322  
012604 010412  
012606 004542  
012610 012201  
012612 020001  
012614 001405  
012616 004767 005504  
012622 004767 006756  
012626 000021  
012630  
012630 010322  
012632 030502  
012634 001363  
012636 004767 002306

```
.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 LOCATION      INSTRUCTION      CONTENTS OF MEMORY 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.
*****
†ST21: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST22 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DIDO: MOV #010412,R3 ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
        MOV #205,R4 ;GET 'RTS R5'
        MOV R4,R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3,(R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4,(R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5,-(R2) ;GO EXECUTE THE IUT.
        MOV (R2)+,R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0,R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC,SPRNT3 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC,$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 21 ;ERROR TYPE CODE.
65$: MOV R3,(R2)+ ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT R5,R2 ;CHECK FOR END OF A BLOCK.
        BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR PC,MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
```

# H11

```

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 012642
2650 012642 004567 005726
2651 012646 000003
2652
2653 012650 000167 000060
2654
2655 012654 012703 110412
2656 012660 012704 000205
2657 012664 012700 110605
2658 012670 004467 001476
2659 012674 010322
2660 012676 010412
2661 012700 004542
2662 012702 012201
2663 012704 020001
2664 012706 001405
2665 012710 004767 005412
2666 012714 004767 006664
2667 012720 000021
2668 012722
2669 012722 010322
2670 012724 030502
2671 012726 001363
2672 012730 004767 002214
  
```

```

*****
*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 MEMOY 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.
*****
TST22: 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$.
  
```

```

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 012734
2699 012734 004567 005634
2700 012740 000003
2701
2702 012742 000167 000064
2703
2704 012746 012703 110342
2705 012752 012704 000205
2706 012756 012700 161342
2707 012762 004467 001404
2708 012766 010322
2709 012770 010412
2710 012772 004562 177776
2711 012776 005302
2712 013000 012201
2713 013002 020001
2714 013004 001405
2715 013006 004767 005314
2716 013012 004767 006566
2717 013016 000021
2718 013020
2719 013020 010322
2720 013022 030502
2721 013024 001361
2722 013026 004767 002116

```

```

*****
*TEST 23 EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOVB R3, -(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY 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.
*****
TST23: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST24 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DIDBH: MOV #110342, R3 ;GET 'MOVB R3, -(R2)' INSTRUCTION (IUT).
        MOV #205, R4 ;GET 'RTS R5'
        MOV #161342, R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: 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$.

```

# J11

CZQMCEO 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCE.P11 10-JAN-78 12:56 T24

MACY11 30A(1052) 10-JAN-78 13:12 PAGE 57  
 EXECUTE DATI, DATIP, DATO THRU MEMORY.

SEQ 0139

```

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 013032
2749 013032 004567 005536
2750 013036 000003
2751
2752 013040 000167 000060
2753
2754 013044 012703 005412
2755 013050 012704 000205
2756 013054 012700 172366
2757 013060 004467 001306
2758 013064 010322
2759 013066 010412
2760 013070 004542
2761 013072 012201
2762 013074 020001
2763 013076 001405
2764 013100 004767 005222
2765 013104 004767 006474
2766 013110 000021
2767 013112
2768 013112 010322
2769 013114 030502
2770 013116 001363
2771 013120 004767 002024
  
```

```

*****
*TEST 24 EXECUTE DATI, DATIP, DATO THRU MEMORY.
* EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
*          MEMORY          INSTRUCTION          CONTENTS OF MEMORY LOCATION
*          LOCATION        PLACED THERE        AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000        005412             172366
* THRU TEST / 40002        000205             000205
*
* 2ND PASS / 40002        005412             172366
* THRU TEST / 40004        000205             000205
*
*          ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
†ST24:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   3                ;MINIMUM BLOCK SIZE OF 2 WORDS
                                   ;REQUIRED FOR THIS TEST.
      JMP      TST25          ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                   ;AVAILABLE FOR TEST.
DIPD0: MOV      #005412,R3    ;GET 'NEG (R2)' INSTRUCTION (IUT).
      MOV      #205,R4       ;GET 'RTS R5'
      MOV      #172366,R0    ;SET UP S/B DATA AFTER EXECUTION.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV      R3,      (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:   MOV      R4,      (R2)  ;PUT 'RTS R5' FOLLOWING IUT.
      JSR      R5,      -(R2) ;GO EXECUTE THE IUT.
      MOV      (R2)+,    R1    ;GET THE DATA FROM THE MEM ADR UNDER TEST.
      CMP      R0,      R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      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$.
  
```

2772  
2773  
2774  
2775  
2776  
2777  
2778  
2779  
2780  
2781  
2782  
2783  
2784  
2785  
2786  
2787  
2788  
2789  
2790  
2791  
2792  
2793  
2794  
2795  
2796  
2797  
2798  
2799  
2800  
2801  
2802  
2803  
2804  
2805  
2806  
2807  
2808  
2809  
2810  
2811  
2812  
2813  
2814  
2815  
2816  
2817  
2818  
2819  
2820

013124  
013124 004567 005444  
013130 000003  
013132 000167 000060  
013136 012703 142242  
013142 012704 000205  
013146 012700 142000  
013152 004467 001214  
013156 010322  
013160 010412  
013162 004542  
013164 012201  
013166 020001  
013170 001405  
013172 004767 005130  
013176 004767 006402  
013202 000021  
013204  
013204 010322  
013206 030502  
013210 001363  
013212 004767 001732

\*\*\*\*\*  
\*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  
\* 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.  
\*\*\*\*\*

TST25: 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, \$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\$.

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  
2861  
2862  
2863  
2864  
2865  
2866  
2867  
2868  
2869  
2870

013216  
013216 004567 005352  
013222 000003  
013224 000167 000062  
013230 012703 152212  
013234 012704 000205  
013240 012700 157212  
013244 004467 001122  
013250 010322  
013252 010412  
013254 004542  
013256 005302  
013260 012201  
013262 020001  
013264 001405  
013266 004767 005034  
013272 004767 006306  
013276 000021  
013300  
013300 010322  
013302 030502  
013304 001362  
013306 004767 001636

```
*****
*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 BOUNDRY BIT MASK.
*****
TST26: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST27 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DPDBH: MOV #152212, R3 ;GET 'BISB (R2)+, (R2)' INSTRUCTION (IUT).
        MOV #205, R4 ;GET 'RTS R5'
        MOV #157212, R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
        MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
        MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) ;GO EXECUTE THE IUT.
        DEC R2 ;RESET R2 TO POINT TO IUT.
        MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
        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$.
1$:
2$:
```

2871  
2872  
2873  
2874  
2875  
2876  
2877  
2878  
2879  
2880  
2881  
2882  
2883  
2884  
2885  
2886  
2887  
2888  
2889  
2890  
2891  
2892  
2893  
2894  
2895  
2896 013312  
2897 013312 004567 005256  
2898 013316 000000  
2899 013320 004467 001046  
2900 013324 010267 166264  
2901 013330 005003  
2902 013332 012700 000377  
2903 013336 010022  
2904 013340 030502  
2905 013342 001375  
2906 013344 014201  
2907 013346 020001  
2908 013350 001405  
2909 013352 004767 004754  
2910 013356 004767 006222  
2911 013362 000010  
2912 013364  
2913 013364 000300  
2914 013366 010012  
2915 013370 011201  
2916 013372 020001  
2917 013374 001405  
2918 013376 004767 004730  
2919 013402 004767 006176  
2920 013406 000010  
2921 013410  
2922 013410 000300  
2923 013412 005703  
2924 013414 001403  
2925 013416 020327 000003  
2926 013422 001010

```
.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, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
        MOV R2, TEMP ;SAVE BANK STARTING ADDRESS
        CLR R3 ;CLEAR PASS COUNTER
1$:     MOV #000377, R0 ;SETUP TO WRITE PATTERN
        MOV R0, (R2)+ ;WRITE PATTERN
2$:     BIT R5, R2 ;END OF 4K?
        BNE 2$ ;CONTINUE WRITING IF NO.
3$:     MOV -(R2), R1 ;GET DATA WRITEN
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 10 ;ERROR TYPE CODE.
65$:    SWAB R0 ;SWAP BYTES OF DATA
4$:     MOV R0, (R2) ;WRITE SWAPPED WORD
        MOV (R2), R1 ;GET DATA WRITEN
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$:    JSR PC, SPRT2 ;SET UP VALUES FOR ERROR PRINTING.
        JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD 10 ;ERROR TYPE CODE.
67$:    SWAB R0 ;PUT DATA BACK TO ORINGINAL
        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
```

```

2927 013424 030502          5$: BIT      R5,R2      ;DONE A PASS?
2928 013426 001346          BNE      3$         ;IF NO CONTINUE
2929 013430 005203          INC      R3         ;IF YES INCREMENT PASS COUNTER
2930 013432 022703 000004     CMP      #4,R3     ;ARE WE DONE ALL PASSES FOR THIS 4K?
2931 013436 001427          BEQ      9$         ;IF YES BRANCH
2932 013440 000300          SWAB    R0         ;ELSE SET UP NEW READ WORD
2933 013442 000404          BR      7$         ;GO TO START OF ADDRESS UP
2934 013444 062702 000002     6$: ADD      #2,R2     ;UPDATE TO NEXT ADDRESS
2935 013450 030502          BIT      R5,R2     ;DONE A PASS
2936 013452 001411          BEQ      8$         ;IF YES BRANCH
2937 013454 011201          7$: MOV      (R2),R1 ;GET DATA WRITTEN
2938 013456 020001          CMP      R0,R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2939 013460 001405          BEQ      69$        ;BRANCH OVER ERROR CALL IF GOOD DATA.
2940 013462 004767 004644     68$: JSR      PC,SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2941 013466 004767 006112     JSR      PC,$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2942 013472 000010          .WORD   10        ;ERROR TYPE CODE.
2943 013474
2944 013474 000733          69$: BR      4$
2945 013476 005203          8$: INC      R3         ;INCREMENT PASS COUNTER
2946 013500 000300          SWAB    R0         ;SET UP NEW READ WORD
2947 013502 020327 000002     CMP      R3,#2     ;ADDRESSING UP?
2948 013506 001316          BNE      3$         ;IF NO GO TO DOWN SEQUENCE
2949 013510 016702 166100     MOV      TEMP,R2   ;IF YES RESET ADDRESS TO START
2950 013514 000757          BR      7$         ;GO TO UP SEQUENCE
2951 013516 004467 000650     9$: JSR      R4,INITMM ;INITIALIZE MEMORY ADDRESS POINTERS
2952 013522 004767 001422     JSR      PC,MMUP   ;UPDATE TO NEW BANK IF EXISTS
2953
2954
2955 ;*****
2956 ;*TEST 30 WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2957 ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY STALL
2958 ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
2959 ;* DETERIORATE BETWEEN REFRESH CYCLES.
2960 ;*
2961 ;* RO=DATA WRITTEN INTO MEMORY(SHOULD BE)
2962 ;* R1=DATA READ FROM MEMORY(WAS)
2963 ;* R2=VIRTUAL ADDRESS
2964 ;* R3=SMALL LOOP COUNTER FOR STALL
2965 ;* R4=NUMBER OF TIMES SMALL LOOP DONE
2966 ;* R5=BLOCK BOUNDARY BIT MASK.
2967 ;*****
2967 013526          TST30:
2968 013526 004567 005042     JSR      R5,$SCOPE ;GO TO SCOPE ROUTINE.
2969 013532 000000          .WORD   0         ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2970 013534 004467 000632     JSR      R4,INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2971 013540 012700 125252     MOV      #125252,R0 ;SETUP DATA PATTERN
2972 013544 010022          1$: MOV      R0,(R2)+ ;WRITE A WORD
2973 013546 005100          COM      R0         ;COMPLEMENT DATA
2974 013550 030502          BIT      R5,R2     ;CHECK FOR END OF A BLOCK.
2975 013552 001374          BNE      1$         ;BRANCH IF MORE IN CURRENT BLOCK.
2976 013554 004767 001370     JSR      PC,MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
2977 013560 005003          CLR      R3         ;SET UP COUNTER FOR STALL
2978 013562 012704 000046     2$: MOV      #46,R4   ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
2979 013566 005303          DEC      R3
2980 013570 001376          BNE      2$
2981 013572 005304          DEC      R4
2982 013574 001374          BNE      2$

```



```

2983 013576 004467 000570 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
2984 013602 012700 125252 MOV #125252, R0 ; INIT DATA FOR CHECKING
2985 013606 3$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
2986 013606 012201 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
2987 013610 020001 BEQ 65$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
2988 013612 001405 64$: JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
2989 013614 004767 004512 JSR PC, SERROR ; *** ERROR *** (GO TYPE A MESSAGE)
2990 013620 004767 005760 .WORD 6 ; ERROR TYPE CODE.
2991 013624 000006 65$: COM R0 ; CHECK FOR END OF A BLOCK.
2992 013626 BIT R5, R2 ; BRANCH IF MORE IN CURRENT BLOCK.
2993 013626 005100 BNE 3$ ; FIND NEXT BLOCK AND LOOP TO 1$.
2994 013630 030502 JSR PC, MMUP
2995 013632 001365
2996 013634 004767 001310
2997 *****
2998 ; *TEST 31 WRITE CHECKERBOARD STARTING WITH 052525 DATA
2999 *****
3000 013640
3001 013640 004567 004730 JSR R5, $SCOPE ; GO TO SCOPE ROUTINE.
3002 013644 000000 .WORD 0 ; NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3003 013646 004467 000520 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
3004 013652 012700 052525 MOV #052525, R0 ; SETUP DATA PATTERN
3005 013656 010022 1$: MOV R0, (R2)+ ; WRITE A WORD
3006 013660 005100 COM R0
3007 013662 030502 BIT R5, R2 ; CHECK FOR END OF A BLOCK.
3008 013664 001374 BNE 1$ ; BRANCH IF MORE IN CURRENT BLOCK.
3009 013666 004767 001256 JSR PC, MMUP ; FIND NEXT BLOCK AND LOOP TO 1$.
3010 013672 005003 CLR R3 ; SET COUNTER FOR LOOP
3011 013674 012704 000046 MOV #46, R4 ; DO LOOP 46 TIMES OR 2 SEC. TOTAL
3012 013700 005303 2$: DEC R3
3013 013702 001376 BNE 2$
3014 013704 005304 DEC R4
3015 013706 001374 BNE 2$
3016 013710 004467 000456 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
3017 013714 012700 052525 MOV #052525, R0 ; INIT PATTERN FOR CHECKING
3018 013720 3$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
3019 013720 012201 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
3020 013722 020001 BEQ 65$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
3021 013724 001405 64$: JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
3022 013726 004767 004400 JSR PC, SERROR ; *** ERROR *** (GO TYPE A MESSAGE)
3023 013732 004767 005646 .WORD 6 ; ERROR TYPE CODE.
3024 013736 000006 65$: COM R0 ; CHECK FOR END OF A BLOCK.
3025 013740 BIT R5, R2 ; BRANCH IF MORE IN CURRENT BLOCK.
3026 013740 005100 BNE 3$ ; FIND NEXT BLOCK AND LOOP TO 1$.
3027 013742 030502 JSR PC, MMUP
3028 013744 001365
3029 013746 004767 001176

```

3030					.SBTTL	DONE:	RELOCATE PROGRAM AND REPEAT ALL TESTS.
3031	013752				DONE:		
3032	013752	004567	004616			JSR	RS, \$SCOPE ;GO TO SCOPE ROUTINE.
3033	013756	000000				.WORD	0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3034	013760	005067	165204		TST32:	CLR	\$TIMES ;RESET ITERATION COUNTER FOR RESTARTING TEST.
3035	013764	105067	165112			CLRB	\$TSTNM ;RESET TEST NUMBER.
3036	013770	036767	164606	165536	1\$:	BIT	PRGMAP, SAVTST ;CHECK IF PROGRAM IS IN TEST AREA.
3037	013776	001004				BNE	2\$ ;BR IF IT PROG IN MEM TO BE TESTED.
3038	014000	036767	164600	165530		BIT	PRGMAP+2, SAVTST+2 ;CHECK HI 64K
3039	014006	001434				BEQ	3\$ ;BR IF PROG NOT IN MEM TO BE TESTED.
3040	014010	032777	000200	165122	2\$:	BIT	#SW07, @SWR ;CHECK FOR INHIBIT RELOCATION SWITCH.
3041	014016	001030				BNE	4\$ ;SKIP RELOCATION IF SWITCH SET.
3042	014020	022767	000003	164554		CMP	#3, PRGMAP ;CHECK IF PROGRAM IN FIRST BK.
3043	014026	001012				BNE	4\$ ;BR IF NOT IN FIRST BK.
3044	014030	005737	000042			TST	@#42 ;CHECK FOR A ACT11.
3045	014034	001014				BNE	5\$ ;BR IF A ACT11.
3046	014036	105737	001224			TSTB	@\$SENV ;CHECK FOR APT
3047	014042	001011				BNE	5\$ ;IF APT DO NOT RELOCATE
3048	014044	004767	002354			JSR	PC, RELTOP ;RELOCATE PROGRAM TO TOP OF MEMORY.
3049	014050	000167	172004		3\$:	JMP	START1 ;LOOP BACK AND RUN ALL TESTS AGAIN.
3050							
3051	014054	004767	002746		4\$:	JSR	PC, RELO ;RELOCATE PROGRAM BACK TO FIRST BK.
3052	014060	005737	000042			TST	@#42 ;TEST FOR XXDP
3053	014064	001402				BEQ	6\$ ;IF NOT RUNNING UNDER MON. DONT
3054	014066	004767	003142		5\$:	JSR	PC, RESLDR ;RESTORE LOADERS.
3055	014072				6\$:		
3056	014072	004567	007360			JSR	RS, \$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3057	014076	001201				.WORD	\$CRLF ;ADDRESS OF MESSAGE TO BE TYPED

```

3058 ;*****
3059
3060 .SBTTL END OF PASS ROUTINE
3061
3062 ;*INCREMENT THE PASS NUMBER ($PASS)
3063 ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
3064 ;*IF THERES A MONITOR GO TO IT
3065 ;*IF THERE ISN'T JUMP TO START1
3066
3067 SEOP:
3068 014100 NOP
3069 014100 000240 CLR $TIMES ;:ZERO THE NUMBER OF ITERATIONS
3070 014102 005067 165062 INC $PASS ;:INCREMENT THE PASS NUMBER
3071 014106 005267 165100 BIC #100000,$PASS ;:DON'T ALLOW A NEG. NUMBER
3072 014112 042767 100000 165072 DEC (PC)+ ;:LOOP?
3073 014122 000001 SEOPCT: .WORD 1
3074 014124 003035 BGT $DOAGN ;:YES
3075 014126 012737 MOV (PC)+,a(PC)+ ;:RESTORE COUNTER
3076 014130 000001 SENDCT: .WORD 1
3077 014132 014122 SEOPCT
3078 014134 004567 007316 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3079 014140 014224 .WORD $ENDMG ;ADDRESS OF MESSAGE TO BE TYPED
3080 014142 016746 165044 MOV $PASS,-(SP) ;SAVE $PASS FOR TYPEOUT
3081 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
3082 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
3083 014146 013746 177776 MOV a#PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3084 014152 004767 010220 JSR PC,$STYPDS ;GO TO THE SUBROUTINE
3085 014156 004567 007274 JSR R5,$PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3086 014162 014241 .WORD $ENULL ;ADDRESS OF MESSAGE TO BE TYPED
3087 014164
3088 $GET42:
3089 014164 016700 163652 MOV 42,RO ;:GET MONITOR ADDRESS
3090 014170 001413 BEQ $DOAGN ;:BRANCH IF NO MONITOR
3091 014172 000005 RESET ;:CLEAR THE WORLD
3092 014174 004710 JSR PC,(RO) ;:GO TO MONITOR
3093 014176 000240 NOP ;:SAVE ROOM
3094 014200 000240 NOP ;:FOR
3095 014202 000240 NOP ;:ACT11
3096 014204 023737 000042 000046 CMP a#42,a#46 ;:ARE WE UNDER ACT11 OR XXDP
3097 014212 001402 BEQ $DOAGN ;:IF ACT11 THEN RESTART
3098 014214 004767 003074 JSR PC,$AVLDR ;:IF XXDP FIRST SAVE MONITOR
3099 014220 $DOAGN:
3100 014220 000167 171634 JMP START1 ;:RETURN*****
3101 014224 005015 047105 020104 SENDMG: .ASCIZ <15><12>/END PASS #/
3102 014232 040520 051523 021440
3103 014240 000
3104 014241 377 377 000 $ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
3105
3106 .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
3107 .SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
3108 ;*****
3109 ;* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
3110 ;* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
3111 ;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
3112 ;* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
3113 ;* PARS 4, 5, AND 6 ARE UNUSED.
;*****

```

```

3114 014244 MMINIT:
3115 014244 012737 077406 172300 MOV #200-1*400+UP+RW, @#KIPDR0 ;SET KIPDR0 = RW UP 200 BLOCKS
3116 014252 012737 077406 172302 MOV #200-1*400+UP+RW, @#KIPDR1 ;SET KIPDR1 = RW UP 200 BLOCKS
3117 014260 012737 077406 172304 MOV #200-1*400+UP+RW, @#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
3118 014266 012737 077406 172306 MOV #200-1*400+UP+RW, @#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
3119 014274 005037 172310 CLR @#KIPDR4
3120 014300 005037 172312 CLR @#KIPDR5
3121 014304 005037 172314 CLR @#KIPDR6
3122 014310 012737 077406 172316 MOV #200-1*400+UP+RW, @#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
3123 014316 005037 172340 @#KIPAR0 ;MAP PAR0 INTO BANK0
3124 014322 012737 000200 172342 MOV #200, @#KIPAR1 ;MAP PAR1 INTO BANK1
3125 014330 005037 172344 CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
3126 014334 005037 172346 CLR @#KIPAR3
3127 014340 005037 172350 CLR @#KIPAR4
3128 014344 005037 172352 CLR @#KIPAR5
3129 014350 005037 172354 CLR @#KIPAR6
3130 014354 012737 007600 172356 MOV #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
3131 014362 012737 000001 177572 MOV #1, @#SRO ;ENABLE MEMORY MANAGEMENT
3132 014370 000207 RTS PC ;RETURN
3133
3134
3135
3136
3137

```

```

*****
* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
*****
3138 014372 012767 000001 165144 INITMM: MOV #BIT0, BITPT ;SET POINTER TO BANK0
3139 014400 005067 165142 CLR BITPT+2 ;CLEAR HI 64K BANK POINTERS
3140 014404 005002 R2 ;SET ADDRESS POINTER TO 0
3141 014406 016705 165174 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3142 014412 005767 164170 TST MMAPVA ;CHECK FOR MEM MGMT AVAILABLE
3143 014416 001514 BEQ 10$ ;BRANCH IF NO MEM MGMT
3144 014420 005037 172344 CLR @#KIPAR2 ;SET UP 3RD PAR TO BANK0
3145 014424 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3146 014430 036767 165110 1$: BIT BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED
3147 014436 001015 BNE 2$ ;BRANCH IF MATCH
3148 014440 036767 165102 165064 BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
3149 014446 001011 BNE 2$ ;BRANCH IF MATCH
3150 014450 062737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3151 014456 006367 165062 BITPT ;UPDATE LO POINTER TO NEXT BANK.
3152 014462 006167 165060 ROL BITPT+2 ;...HI POINTER.
3153 014466 100360 BPL 1$ ;BR IF MORE.
3154 014470 000000 HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
3155 014472 036767 165046 165102 2$: BIT BITPT, LADMAP ;CHECK IF LAST BANK.
3156 014500 001004 BNE 3$ ;BR IF LAST BANK.
3157 014502 036767 165040 165074 BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
3158 014510 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3159 014512 016705 165062 3$: MOV LADMSK, R5 ;SET MASK TO FIND LAST ADR.
3160 014516 042767 020000 165052 BIC #20000, TMAPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
3161 014524 013737 172344 172346 4$: MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3162 014532 016767 165006 165010 MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3163 014540 016767 165002 165004 MOV BITPT+2, TMPPT+2 ;...HI 64K.
3164 014546 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3165 014552 001505 BEQ 21$ ;BRANCH IF NOT 8K.
3166 014554 062737 000200 172346 5$: ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3167 014562 006367 164762 ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3168 014566 006167 164760 ROL TMPPT+2 ;...HI POINTER.
3169 014572 100473 BMI 20$ ;BR IF NO MORE.

```

3170	014574	036767	164750	164726		BIT	TMPPT, TSTMAP	;CHECK IF BANK TO BE TESTED.
3171	014602	001004				BNE	6\$	;BRANCH IF A MATCH.
3172	014604	036767	164742	164720		BIT	TMPPT+2, TSTMAP+2	;CHECK FOR HI 64K BANKS.
3173	014612	001760				BEQ	5\$	;BRANCH IF NO MEMORY
3174	014614	036767	164730	164760	6\$:	BIT	TMPPT, LADMAP	;CHECK IF LAST BANK.
3175	014622	001004				BNE	7\$	;BRANCH IF A MATCH
3176	014624	036767	164722	164752		BIT	TMPPT+2, LADMAP+2	;CHECK HI 64K
3177	014632	001455				BEQ	21\$	;BR IF NOT LAST BANK.
3178	014634	016705	164740		7\$:	MOV	LADMSK, R5	;RESET MASK TO FIND LAST ADR.
3179	014640	052767	020000	164730		BIS	#20000, TEMPLAD	;MAKE SURE LAST ADDRESS IS IN BANK 3.
3180	014646	000447				BR	21\$	;BR TO FINISH UP.
3181								
3182	014650	036767	164670	164652	10\$:	BIT	BITPT, TSTMAP	;CHECK IF THIS BANK TO BE TESTED.
3183	014656	001006				BNE	11\$	;BR IF MATCH.
3184	014660	062702	020000			ADD	#20000, R2	;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3185	014664	106367	164654			ASLB	BITPT	;UPDATE BANK POINTER TO NEXT BANK.
3186	014670	100367				BPL	10\$	;BR IF MORE BANKS.
3187	014672	000000				HALT		;FATAL ERROR!!! NO 4K BANK FOUND?
3188	014674	016767	164644	164646	11\$:	MOV	BITPT, TMPPT	;COPY BANK POINTER.
3189	014702	036767	164636	164672		BIT	BITPT, LADMAP	;CHECK IF LAST BANK.
3190	014710	001021				BNE	12\$	;BR IF LAST BANK.
3191	014712	032705	020000			BIT	#BIT13, R5	;CHECK FOR 8K BLOCK SIZE.
3192	014716	001423				BEQ	21\$	;BRANCH IF SMALLER BLOCK SIZE.
3193	014720	106367	164624			ASLB	TMPPT	;POINT TO NEXT BANK.
3194	014724	100416				BMI	20\$	;BRANCH IF OVERFLOW.
3195	014726	036767	164616	164574		BIT	TMPPT, TSTMAP	;CHECK IF BANK TO BE TESTED.
3196	014734	001412				BEQ	20\$	;BRANCH IF NOT TO BE TESTED.
3197	014736	112767	000011	164613		MOVB	#11, FLAG8K	;SET 8K BLOCK SIZE FLAG.
3198	014744	036767	164600	164630		BIT	TMPPT, LADMAP	;CHECK FOR LAST BANK.
3199	014752	001403				BEQ	20\$	;BR IF NOT LAST BANK.
3200	014754	016705	164620		12\$:	MOV	LADMSK, R5	;RESET MASK TO FIND LAST ADR.
3201	014760	000402				BR	21\$	;SKIP MASK RESET.
3202	014762	012705	017777		20\$:	MOV	#MASK4K, R5	;RESET MASK TO 4K BLOCK SIZE.
3203	014766	056767	164552	164554	21\$:	BIS	BITPT, TMPPT	;SET TMPPT FOR FLAGING LAST BANK.
3204	014774	056767	164546	164550		BIS	BITPT+2, TMPPT+2	
3205	015002	036767	164536	164560		BIT	BITPT, FADMAP	;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3206	015010	001004				BNE	22\$	;BR IF FIRST BANK.
3207	015012	036767	164530	164552		BIT	BITPT+2, FADMAP+2	;CHECK HI 64K.
3208	015020	001450				BEQ	INITEX	;BR IF NOT FIRST BANK.
3209	015022	016702	164536		22\$:	MOV	TMPPAD, R2	;RESET ADDRESS POINTER TO FIRST ADR.
3210	015026	000445				BR	INITEX	
3211								
3212	015030	016705	164552		INITDN:	MOV	BLKMSK, R5	;RESET R5 TO CURRENT BLOCK MASK.
3213	015034	005002				CLR	R2	;INIT ADDRESS POINTER.
3214	015036	005767	163544			TST	MMAVA	;CHECK FOR MEM MGMT
3215	015042	001411				BEQ	31\$	;BRANCH IF NO MEM MGMT
3216	015044	012767	100000	164474		MOV	#BIT15, BITPT+2	;SET POINTER TO TOP BIT
3217	015052	005067	164466			CLR	BITPT	
3218	015056	012737	007600	172344		MOV	#7600, #KIPAR2	;SET PAR TO TOP OF MEM
3219	015064	000403				BR	32\$	;BRANCH TO COMMON AREA
3220								
3221	015066	012767	000400	164450	31\$:	MOV	#BIT8, BITPT	;SET UP BANK POINTER
3222	015074	012767	015116	164452	32\$:	MOV	#33\$, MMORE	;SET "MMDOWN" EXIT ADDRESS.
3223	015102	066767	163472	164444		ADD	RELOC, MMORE	;ADD OFFSET
3224	015110	004767	000524			JSR	PC, MMDOWN	;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3225	015114	000000				HALT		;FATAL ERROR!!! NO MEM INDICATED IN MEM MAP ABOVE 8K!

```

3226 015116 036767 164422 164456 33$: BIT BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADDR.
3227 015124 001004 BNE 34$ ;BR IF LAST BANK FLAG FOUND.
3228 015126 036767 164414 164450 BIT BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADDR.
3229 015134 001402 BEQ INITEX ;BR IF NO LAD FLG FOUND.
3230 015136 016702 164432 34$: MOV LSTADR, R2 ;SET UP R2.
3231 015142 010467 164406 INITEX: MOV R4, MMORE ;PUT RETURN PC INTO "MMORE"
3232 015146 000204 RTS R4 ;RETURN
3233
3234 ;*****
3235 ;* COMMON UPWARDS ADDRESSING ROUTINE
3236 ;* FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
3237 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS
3238 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3239 ;*****
3240 015150 036767 164374 164424 MMUP: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3241 015156 001122 BNE 10$ ;BR IF LAST BANK.
3242 015160 036767 164366 164416 BIT TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3243 015166 001116 BNE 10$ ;BR IF LAST BANK.
3244 015170 016705 164412 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3245 015174 005767 163406 TST MAVA ;CHECK FOR MEM MGMT AVAILABLE
3246 015200 001515 BEQ 20$ ;BRANCH IF NO MEM MGMT
3247 015202 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3248 015206 062737 000200 172344 1$: ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3249 015214 006367 164324 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3250 015220 006167 164322 ROL BITPT+2 ;...HI POINTER.
3251 015224 100577 BMI 32$ ;BR IF ALL DONE.
3252 015226 036767 164312 164274 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS
3253 015234 001004 BNE 2$ ;BRANCH IF MATCH
3254 015236 036767 164304 164266 BIT BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3255 015244 001760 BEQ 1$ ;BRANCH IF NO MATCH
3256 015246 036767 164272 164326 2$: BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3257 015254 001004 BNE 3$ ;BRANCH IF LAST BANK FLAG.
3258 015256 036767 164264 164320 BIT BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3259 015264 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3260 015266 016705 164306 3$: MOV LADMSK, R5 ;RESET MASK.
3261 015272 042767 020000 164276 BIC #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3262 015300 016767 164240 164242 4$: MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3263 015306 016767 164234 164236 MOV BITPT+2,TMPPT+2 ;...HI 64K.
3264 015314 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
3265 015320 001530 BEQ 31$ ;BRANCH IF NOT.
3266 015322 013737 172344 172346 MOV @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3267 015330 062737 000200 172346 5$: ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3268 015336 006367 164206 TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3269 015342 006167 164204 ROL TMPPT+2 ;...HI POINTER.
3270 015346 100513 BMI 30$ ;BR IF NO MORE.
3271 015350 036767 164174 164152 6$: BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3272 015356 001004 BNE 7$ ;BRANCH IF A MATCH.
3273 015360 036767 164166 164144 BIT TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3274 015366 001760 BEQ 5$ ;BRANCH IF NO MEMORY
3275 015370 036767 164154 164204 7$: BIT TMPPT,LADMAP ;CHECK FOR LAST BANK FLAG.
3276 015376 001004 BNE 8$ ;BRANCH IF A MATCH
3277 015400 036767 164146 164176 BIT TMPPT+2,LADMAP+2 ;CHECK HI 64K
3278 015406 001475 BEQ 31$ ;BR IF NO LAST BANK FLAG.
3279 015410 016705 164164 8$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.
3280 015414 052767 020000 164154 BIS #20000, TEMPLAD ;SET VIRTUAL ADR TO BANK 3.
3281 015422 000467 BR 31$

```

```

3282
3283 015424 026702 164146      10$:  CMP      TEMPLAD, R2      ;CHECK IF LAST ADR REACHED.
3284 015430 001064              BNE      31$              ;BR IF MORE.
3285 015432 000474              BR       32$              ;BR IF ALL DONE.
3286
3287 015434 106267 164117      20$:  ASRB     FLAG8K          ;SHIFT 8K FLAG
3288 015440 001407              BEQ     22$              ;BR IF NOT 8K BLOCK.
3289 015442 103455              BCS     30$              ;BR IF ANOTHER 4K.
3290 015444 105067 164107      CLRB     FLAG8K          ;CLEAR OUT ALL FLAGS.
3291 015450 162702 040000      SUB     #40000, R2       ;BACK UP 8K.
3292 015454 062702 020000      21$:  ADD     #20000, R2       ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3293 015460 106367 164060      22$:  ASLB     BITPT          ;UPDATE POINTER.
3294 015464 100457              BMI     32$              ;BRANCH WHEN END IS REACHED.
3295 015466 036767 164052 164034  BIT     BITPT, TSTMAP    ;CHECK IF THIS BANK EXISTS.
3296 015474 001767              BEQ     21$              ;BRANCH IF NO MATCH.
3297 015476 036767 164042 164076  BIT     BITPT, LADMAP    ;CHECK FOR LAST BANK FLAG.
3298 015504 001402              BEQ     23$              ;BR IF NO MATCH.
3299 015506 016705 164066      MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADR.
3300 015512 016767 164026 164030  23$:  MOV     BITPT, TMPPT    ;SET UP TMP POINTER.
3301 015520 032705 020000      BIT     #BIT13, R5       ;CHECK FOR 8K BLOCK SIZE.
3302 015524 001426              BEQ     31$              ;BRANCH IF SMALLER BLOCK SIZE.
3303 015526 106367 164016      ASLB     TMPPT          ;POINT TO NEXT BANK.
3304 015532 100421              BMI     30$              ;BRANCH IF OVERFLOW.
3305 015534 036767 164010 163766  BIT     TMPPT, TSTMAP    ;CHECK IF BANK TO BE TESTED.
3306 015542 001415              BEQ     30$              ;BRANCH IF NOT TO BE TESTED.
3307 015544 036767 163774 164030  BIT     BITPT, LADMAP    ;CHECK FOR LAST BANK FLAG.
3308 015552 112767 000011 163777  MOVB    #11, FLAG8K      ;SET 8K BLOCK FLAG.
3309 015560 036767 163760 164014  BIT     BITPT, LADMAP    ;CHECK FOR LAST BANK FLAG.
3310 015566 001403              BEQ     30$              ;BR IF NO FLAG.
3311 015570 016705 164004      MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADR.
3312 015574 000402              BR      31$
3313 015576 012705 017777      30$:  MOV     #MASK4K, R5      ;SET MASK TO 4K.
3314 015602 056767 163736 163740  31$:  BITPT  TMPPT          ;SET TMPPT FOR FINDING LAST ADR.
3315 015610 056767 163732 163734  BIS     BITPT+2, TMPPT+2
3316 015616 016716 163732      MOV     MMORE, (SP)      ;FUDGE RETURN ADDRESS TO LOOP.
3317 015622 000207              RTS     PC               ;RETURN
3318
3319 015624 005767 164446      32$:  TST     MPRX           ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3320 015630 001402              BEQ     33$              ;BR IF NONE.
3321 015632 004767 001744      JSR     PC, CKPMER       ;CHECK FOR PARITY MEMORY ERRORS.
3322 015636 000207      33$:  RTS     PC               ;STRAIGHT RETURN.
3323
3324 ;*****
3325 ;* MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
3326 ;* FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
3327 ;* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
3328 ;* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
3329 ;*****
3330 015640 036767 163700 163722  MMDOWN: BIT     BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
3331 015646 001004              BNE     1$              ;BR IF FIRST ADR IN THIS BANK.
3332 015650 036767 163672 163714  BIT     BITPT+2, FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
3333 015656 001404              BEQ     2$              ;BR IF NO FLAG
3334 015660 026702 163700      1$:  CMP     TMPFAD, R2       ;CHECK IF FIRST ADDRESS REACHED.
3335 015664 001052              BNE     9$              ;BR IF MORE.
3336 015666 000453              BR      10$             ;BR IF ALL DONE.
3337 015670 005767 162712      2$:  TST     MMAVA          ;CHECK IF MEM MGMT IS AVAILABLE

```

3338	015674	001425				BEQ	6\$		; BRANCH IF NOT
3339	015676	162737	000200	172344	3\$:	SUB	#200, 2#KIPAR2		; LOWER MEM MGMT PAR BY 4K
3340	015704	006067	163636			ROR	BITPT+2		; MOV POINTER TO NEXT LOWER BANK...HI MAP.
3341	015710	006067	163630			ROR	BITPT		; ...LO MAP.
3342	015714	103440				BCS	10\$		; BR IF NO MORE.
3343	015716	036767	163622	163604		BIT	BITPT, TSTMAP		; CHECK FOR BANK EXISTING
3344	015724	001004				BNE	4\$		; BR IF BANK TO BE TESTED.
3345	015726	036767	163614	163576		BIT	BITPT+2, TSTMAP+2		; CHECK FOR BANK IN HI MAP.
3346	015734	001760				BEQ	3\$		; BR IF NOT THERE.
3347	015736	012702	060000		4\$:	MOV	#60000, R2		; SET ADR POINTER TO TOP OF BANK
3348	015742	000411				BR	7\$		; GO TO COMMON EXIT
3349	015744	162702	020000		5\$:	SUB	#20000, R2		; BACK POINTER DOWN ONE BANK
3350	015750	006267	163570		6\$:	ASR	BITPT		; MOVE POINTER TO NEXT LOWER BANK
3351	015754	103420				BCS	10\$		; BRANCH TO EXIT IF NO MORE MEM
3352	015756	036767	163562	163544		BIT	BITPT, TSTMAP		; CHECK IF BANK EXISTS
3353	015764	001767				BEQ	5\$		; BRANCH IF BANK DOESN'T EXIST
3354	015766	036767	163552	163574	7\$:	BIT	BITPT, FADMAP		; CHECK IF FIRST BANK FLAG.
3355	015774	001004				BNE	8\$		; BR IF FIRST BANK.
3356	015776	036767	163544	163566		BIT	BITPT+2, FADMAP+2		; CHECK IF FIRST BANK FLAG.
3357	016004	001402				BEQ	9\$		; BR IF NO FLAG FOUND.
3358	016006	016705	163554		8\$:	MOV	FADMSK, R5		; SET UP R5 TO FIND FIRST ADDRESS.
3359	016012	016716	163536		9\$:	MOV	MMORE, (SP)		; RESET RETURN ADDRESS
3360	016016	000207			10\$:	RTS	PC		; RETURN



3361  
3362  
3363  
3364  
3365  
3366 016020 010200  
3367 016022 005067 163132  
3368 016026 005767 162554  
3369 016032 001417  
3370 016034 010146  
3371 016036 013701 172344  
3372 016042 006301  
3373 016044 006301  
3374 016046 006301  
3375 016050 006301  
3376 016052 006301  
3377 016054 006167 163100  
3378 016060 006301  
3379 016062 006167 163072  
3380 016066 060100  
3381 016070 012601  
3382 016072 000207  
3383  
3384  
3385  
3386  
3387 016074 005000  
3388 016076 010146  
3389 016100 010246  
3390 016102 016701 163436  
3391 016106 016702 163434  
3392 016112 006202  
3393 016114 006001  
3394 016116 103403  
3395 016120 105200  
3396 016122 100373  
3397 016124 000000  
3398 016126  
3399 016126 012602  
3400 016130 012601  
3401 016132 000207  
3402  
3403  
3404  
3405  
3406 016134  
3407 016134 004467 176232  
3408 016140 010022  
3409 016142 030502  
3410 016144 001375  
3411 016146 004767 176776  
3412 016152 000207

```

.SBTTL SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
*****
;* SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN RO (BOTTOM 16 BITS).
;* BITS 16 AND 17 ARE IN $TMP0.
*****
PHYADR: MOV R2, RO ;VIRTUAL INTO RO
        CLR $TMP0 ;CLEAR TEMP SAVE OF HIGH BITS
        TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
        BEQ 1$ ;BRANCH IF NO MEM MGMT
        MOV R1, -(SP) ;PUSH R1 ON STACK
        MOV @#KIPAR2, R1 ;GET PAR TO BE ADDED TO VIRTUAL
        ASL R1 ;SHIFT IT 6 TIMES
        ASL R1
        ASL R1
        ASL R1
        ASL R1
        ROL $TMP0 ;SAVE EXTRA BITS
        ASL R1
        ROL $TMP0
        ADD R1, RO ;ADD SHIFTED PAR TO VIRTUAL
        MOV (SP)+, R1 ;POP STACK INTO R1
1$: RTS PC ;RETURN

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

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

```

3413
3414
3415
3416 016154 106112
3417 016156 106112
3418 016160 106112
3419 016162 106112
3420 016164 106112
3421 016166 106112
3422 016170 106112
3423 016172 106112
3424 016174 106122
3425 016176 106112
3426 016200 106112
3427 016202 106112
3428 016204 106112
3429 016206 106112
3430 016210 106112
3431 016212 106112
3432 016214 106112
3433 016216 106122
3434 016220 000207
3435
3436
3437
3438
3439 016222 012704 000020
3440
3441 016226 010022
3442 016230 010022
3443 016232 010022
3444 016234 010022
3445
3446 016236 010322
3447 016240 010322
3448 016242 010322
3449 016244 010322
3450
3451 016246 010022
3452 016250 010022
3453 016252 010022
3454 016254 010022
3455
3456 016256 010322
3457 016260 010322
3458 016262 010322
3459 016264 010322
3460
3461 016266 005304
3462 016270 001356
3463 016272 010046
3464 016274 010300
3465 016276 012603
3466 016300 000207

```

```

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

```

```

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

```

```

3467 .SBTTL RELOCATION SUBROUTINES.
3468 ;*****
3469 ;* ROUTINE TO RELOCATE PROGRAM CODE
3470 ;*****
3471 RELOC:
3472   MOV     R2,-(SP)      ;; PUSH R2 ON STACK
3473   MOV     R3,-(SP)      ;; PUSH R3 ON STACK
3474   MOV     R4,-(SP)      ;; PUSH R4 ON STACK
3475   4$:    MOV     (R5)+, R2  ;; GET FIRST LOCATION.
3476   MOV     (R5)+, R3  ;; GET FIRST LOCATION OF DESTINATION.
3477   MOV     #20000, R4  ;; SET UP BK COUNTER.
3478   1$:    MOV     (R2)+, (R3)+ ;; MOV THE DATA.
3479   DEC     R4           ;; COUNT THE WORDS.
3480   BNE    1$           ;; BR IF MORE.
3481   MOV     #20000, R4  ;; RESET THE COUNTER.
3482   2$:    CMP     -(R2), -(R3) ;; CHECK THE DATA JUST MOVED.
3483   BEQ    3$           ;; BR IF DATA OK.
3484   MOV     (R2), $GDDAT ;; GET SOURCE DATA.
3485   MOV     (R3), $BDDAT ;; GET DESTINATION DATA.
3486   MOV     R2, $GDADR  ;; GET SOURCE ADDRESS.
3487   MOV     R3, $BDADR  ;; GET DESTINATION ADDRESS.
3488   JSR    PC, $ERRROR ;; *** ERROR *** (GO TYPE A MESSAGE)
3489   .WORD  23           ;; ERROR TYPE CODE.
3490   HALT                    ;; FATAL ERROR!!! RELOCATION FAILED.
3491   SUB     #4, R5        ;; ADJUST RETURN POINTER.
3492   BR     4$           ;; GO BACK AND TRY AGAIN.
3493   3$:    DEC     R4           ;; COUNT WORDS.
3494   BNE    2$           ;; BR IF MORE.
3495   JSR    R5, $PRINT   ;; GO PRINT OUT THE FOLLOWING MESSAGE.
3496   .WORD  PRELOC      ;; ADDRESS OF MESSAGE TO BE TYPED
3497   "PROGRAM RELOCATED TO "
3498   MOV     R3, -(SP)    ;; PUT THE DATA ON THE STACK.
3499   JSR    PC, $STYPAD  ;; DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
3500   MOV     (SP)+, R4    ;; POP STACK INTO R4
3501   MOV     (SP)+, R3    ;; POP STACK INTO R3
3502   MOV     (SP)+, R2    ;; POP STACK INTO R2
3503   RTS     R5          ;; RETURN
3504 ;*****
3505 ;* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
3506 ;*****
3507   RELTOP: CMP     #3, PRGMAP ;; CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
3508   BEQ    1$           ;; BR IF OK
3509   HALT                    ;; FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
3510   1$:    MOV     R0,-(SP) ;; PUSH R0 ON STACK
3511   MOV     R1,-(SP) ;; PUSH R1 ON STACK
3512   TST    MMVA
3513   BEQ    10$
3514   MOV     #7600, @#KIPAR3 ;; SET PAR TO TOP OF MEM
3515   CLR    R0           ;; INIT BANK POINTER...LO 64K
3516   MOV     #BIT15, R1  ;; ...HI 64K.
3517   SUB    #200, @#KIPAR3 ;; BACK DOWN ONE BANK.
3518   ROR    R1           ;; MOVE POINTER...HI 64K.
3519   ROR    R0           ;; ...LO 64K.
3520   BCS    90$
3521   BIT    R1, MEMMAP+2 ;; CHECK FOR BANK EXISTS.
3522
016302 010246
016304 010346
016306 010446
016310 012502
016312 012503
016314 012704 020000
016320 012223
016322 005304
016324 001375
016326 012704 020000
016332 024243
016334 001417
016336 011267 162562
016342 011367 162560
016346 010267 162546
016352 010367 162544
016356 004767 003222
016362 000023
016364 000000
016366 162705 000004
016372 000746
016374 005304
016376 001355
016400 004567 005052
016404 026506
016406 010346
016410 004767 006502
016414 012604
016416 012603
016420 012602
016422 000205
016424 022767 000003 162150
016432 001401
016434 000000
016436
016436 010046
016440 010146
016442 005767 162140
016446 001465
016450 012737 007600 172346
016456 005000
016460 012701 100000
016464 162737 000200 172346
016472 006001
016474 006000
016476 103500
016500 030167 163022

```

```

3523 016504 001003          BNE      3$          ;BR IF AVAILABLE
3524 016506 030067 163012   BIT      RO,        MEMMAP ;CHECK FOR BANK EXISTS.
3525 016512 001764          BEQ      2$          ;BR IF NO BANK FOUND.
3526 016514 013737 172346 172344 3$:  MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3527 016522 010046          MOV     RO,-(SP)     ;PUSH RO ON STACK
3528 016524 010146          MOV     R1,-(SP)    ;PUSH R1 ON STACK
3529 016526 162737 000200 172344 4$:  SUB     #200, @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3530 016534 006001          ROR     R1          ;SHIFT POINTER.
3531 016536 006000          ROR     RO          ;...LO 64K.
3532 016540 103457          BCS     90$        ;BR IF OVERFLOW.
3533 016542 030167 162760          5$:  BIT     R1,        MEMMAP+2 ;CHECK IF BANK EXISTS...HI 64K.
3534 016546 001003          BNE     6$          ;BR IF BANK EXISTS.
3535 016550 030067 162750   BIT     RO,        MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
3536 016554 001764          BEQ     4$          ;BR IF BANK DOESN'T EXIST.
3537 016556 052601          6$:  BIS     (SP)+,    R1      ;GET SECOND BANK POINTER.
3538 016560 052600          BIS     (SP)+,    RO      ;...LO 64K.
3539 016562 030067 162014   BIT     RO,        PRGMAP ;CHECK FOR CONFLICT.
3540 016566 001044          BNE     90$        ;ABORT IF DESTINATION OVERLAYS SOURCE.
3541 016570 004567 177506   JSR     R5,        RELOC   ;GO RELOCATE PROGRAM.
3542 016574 000000          .WORD  0           ;SOURCE FIRST ADDRESS.
3543 016576 040000          .WORD  40000       ;DESTINATION FIRST ADDRESS.
3544 016600 013737 172344 172340   MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3545 016606 013737 172346 172342   MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
3546                                ;* PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3547 016614 010167 161764   MOV     R1,        PRGMAP+2 ;RESET PROGRAM MAP.
3548 016620 000473          BR      30$        ;BR TO COMMON EXIT.
3549
3550 016622 012700 000400          10$:  MOV     #BIT8,    RO      ;SET BANK POINTER TO TOP OF MEM.
3551 016626 005001          CLR     R1          ;SET ADDRESS POINTER TO TOP.
3552 016630 162701 020000          11$:  SUB     #20000,   R1      ;BACK DOWN ONE BANK.
3553 016634 006200          ASR     RO          ;MOVE POINTER DOWN ONE BANK.
3554 016636 103420          BCS     90$        ;BR IF OVERFLOW.
3555 016640 030067 162660          BIT     RO,        MEMMAP ;CHECK IF THIS BANK EXISTS.
3556 016644 001771          BEQ     11$        ;BR IF NON-EXISTANT BANK.
3557 016646 162701 020000          SUB     #20000,   R1      ;BACK DOWN TO NEXT BANK.
3558 016652 006200          ASR     RO          ;MOV POINTER DOWN ONE BANK.
3559 016654 103411          BCS     90$        ;BR IF OVERFLOW.
3560 016656 030067 162642          BIT     RO,        MEMMAP ;CHECK IF THIS BANK EXISTS.
3561 016662 001762          BEQ     11$        ;BR TO START OVER IF NO LOWER BANK.
3562 016664 010046          MOV     RO,        -(SP)  ;SAVE THE POINTER.
3563 016666 006300          ASL     RO          ;RESET POINTER TO HI BANK.
3564 016670 052600          BIS     (SP)+,    RO      ;SET BIT FOR LO BANK.
3565 016672 030067 161704          BIT     RO,        PRGMAP ;CHECK FOR A PROGRAM CONFLICT.
3566 016676 001401          BEQ     12$        ;BR IF NO CONFLICT.
3567 016700          90$:  HALT                    ;FATAL ERROR!!! NOT ENOUGH MEMORY??
3568 016700 000000          13$:  MOV     R1,        13$   ;SET DATA FOR RELOCATION SUBROUTINE.
3569 016702 010167 000006          JSR     R5,        RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3570 016706 004567 177370          .WORD  0           ;SOURCE STARTING ADDRESS.
3571 016712 000000          .WORD  0           ;DESTINATION STARTING ADDRESS.
3572 016714 000000          13$:  MOV     R1,        RELOC  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3573 016716 010167 161656          ADD     R1,        PC     ;JUMP TO RELOCATED PROGRAM
3574 016722 060107          ;* PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3575                                ADD     R1,        SP     ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3576 016724 060106          MOV     R1,        RELOC  ;SET THE RELOCATION FACTOR.
3577 016726 010167 161646          ADD     R1,        @#ERRVEC ;ADJUST ERROR VECTOR.
3578 016732 060137 000004

```

```

3579 016736 060137 000024      ADD    R1,    @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3580 016742 060137 000114      ADD    R1,    @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3581 016746 026727 162166 177570  CMP    SWR,   #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
3582 016754 001404          BEQ    14$    ;BR IF HARDWARE SWITCH REGISTER.
3583 016756 060167 162156      ADD    R1,    SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3584 016762 060167 162154      ADD    R1,    DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3585 016766 062701 001622 14$:  ADD    #RDTAB,R1 ;POINT TO THE RELATIVE RELOCATION TABLE.
3586 016772 066721 161602 15$:  ADD    RELOCF,(R1)+ ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3587 016776 005721 161602 16$:  TST    (R1)+ ;CHECK FOR INTERUM TERMINATOR.
3588 017000 001776          BEQ    16$    ;BR SO AS TO NOT MODIFY ZERO.
3589 017002 024127 177777      CMP    -(R1), #-1    ;CHECK FOR END OF TABLE.
3590 017006 001371          BNE    15$    ;BR IF MORE IN TABLE.
3591 017010 010067 161566 30$:  MOV    R0,    PRGMAP ;SET NEW PROGRAM MAP...LO 64K.
3592 017014 012601          MOV    (SP)+,R1 ;POP STACK INTO R1
3593 017016 012600          MOV    (SP)+,R0 ;POP STACK INTO R0
3594 017020 066716 161554      ADD    RELOCF,(SP) ;ADJUST RETURN ADDRESS.
3595 017024 000207          RTS    PC      ;RETURN
3596
3597 ;*****
3598 ;* SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3599 ;*****
3600 017026 032767 000003 161546 RELO:  BIT    #3,    PRGMAP ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3601 017034 001401          BEQ    1$     ;BR IF NO CONFLICT.
3602 017036 000000          HALT          ;FATAL ERROR!!! PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3603 017040 005767 161542 1$:  TST    MAVA   ;CHECK FOR MEM MGMT.
3604 017044 001417          BEQ    10$    ;BR IF NO MEMMGMT.
3605 017046 005037 172344      CLR    @#KIPAR2 ;SET PAR 2 TO BANK 0.
3606 017052 012737 000200 172346      MOV    #200, @#KIPAR3 ;SET PAR 3 TO BANK 1.
3607 017060 004567 177216      JSR    R5,    RELOC  ;GO MOVE BK INTO BANKS 0 AND 1.
3608 017064 000000          .WORD    0      ;SOURCE STARTING ADDRESS.
3609 017066 040000          .WORD    40000   ;DESTINATION STARTING ADDRESS.
3610 017070 005037 172340      CLR    @#KIPAR0 ;RESTORE PAR 0 TO BANK0.
3611 017074 012737 000200 172342      MOV    #200, @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
3612 ;* PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3613 017102 000444          BR     30$    ;BR TO COMMON EXIT.
3614
3615 017104 016746 161470 10$:  MOV    RELOCF,-(SP) ;PUT RELOCATION FACTOR ONTO THE STACK.
3616 017110 011667 000004      MOV    (SP), 20$    ;SET DATA FOR RELOC SUBROUTINE.
3617 017114 004567 177162      JSR    R5,    RELOC ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3618 017120 000000          .WORD    0      ;SOURCE STARTING ADDRESS.
3619 017122 000000          .WORD    0      ;DESTINATION STARTING ADDRESS.
3620 017124 161607          SUB    (SP),  PC   ;JUMP TO RELOCATED PROGRAM.
3621 ;* THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3622 017126 161606          SUB    (SP),  SP   ;RESET THE STACK POINTER.
3623 017130 010046          MOV    R0,-(SP)  ;PUSH R0 ON STACK
3624 017132 012700 001622      MOV    #RDTAB,R0 ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3625 017136 166620 000002 21$:  SUB    2(SP), (R0)+ ;RESET ADDRESSES TO UNRELOCATED VALUES.
3626 017142 005720 22$:  TST    (R0)+ ;CHECK FOR TERMINATORS.
3627 017144 001776          BEQ    22$    ;BR OVER TERMINATORS.
3628 017146 024027 177777      CMP    -(R0), #-1  ;CHECK FOR END OF TABLE INDICATOR.
3629 017152 001371          BNE    21$    ;BR IF MORE ADDRESSES IN TABLE.
3630 017154 012600          MOV    (SP)+,R0 ;POP STACK INTO R0
3631 017156 161637 000004      SUB    (SP), @#ERRVEC ;ADJUST ERROR VECTOR.
3632 017162 161637 000024      SUB    (SP), @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3633 017166 161637 000114      SUB    (SP), @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3634 017172 026727 161742 177570  CMP    SWR,   #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.

```

```

3635 017200 001404 BEQ 23$ ;BR IF HARDWARE SWITCH REGISTER.
3636 017202 161667 161732 SUB (SP), SWR ;ADJUST SOFTWARE SWITCH REGISTER.
3637 017206 161667 161730 SUB (SP), DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3638 017212 162616 23$: SUB (SP), (SP) ;ADJUST RETURN ADDRESS.
3639 017214 005067 161360 30$: CLR RELOC ;RESET RELOCATION FACTOR.
3640 017220 012767 000003 161354 MOV #3, PRGMAP ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3641 017226 005067 161352 CLR PRGMAP+2 ;HI 64K.
3642 017232 000207 RTS PC ;RETURN.

```

```

*****
; 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.
*****

```

```

3648 017234 016700 162260 RESLDR: MOV LMAD, R0 ;CHECK IF THE LOADERS WERE SAVED.
3649 017240 001001 BNE IS ;BR IF LOADER AREA WAS SAVED.
3650 017242 000000 HALT ;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3651 017244 005767 161336 1$: TST MMAVA ;CHECK FOR MEM MGMT.
3652 017250 001402 BEQ 2$ ;SKIP IF NO MEM MGMT.
3653 017252 005037 177572 CLR #SRO ;DISABLE MEM MGMT.
3654 017256 012701 040000 2$: MOV #40000, R1 ;GET END OF BK, ASSUME PROG NOT RELOCATED.
3655 017262 012702 002734 MOV #1500., R2 ;GET COUNTER.
3656 017266 014140 3$: MOV -(R1), -(R0) ;MOVE THE LOADER AREA.
3657 017270 005302 DEC R2 ;COUNT HOW LONG THE AREA IS.
3658 017272 001375 BNE 3$ ;BR IF NOT MORE TO MOVE.
3659 017274 005067 162220 CLR LMAD ;CLEAR MONITOR SAVED FLAG
3660 017300 005767 161302 TST MMAVA ;CHECK FOR MEM MGMT.
3661 017304 001402 BEQ 4$ ;BR IF NO MEM MGMT.
3662 017306 005237 177572 INC #SRO ;ENABLE MEM MGMT.
3663 017312 000207 4$: RTS PC ;RETURN.

```

```

; * ROUTINE TO SAVE THE LOADERS AT THE END OF BK.

```

```

3666 017314 005767 162200 SAVLDR: TST LMAD ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3667 017320 001024 BNE 4$ ;BRANCH IF ALREADY SAVED
3668 017322 012700 040000 MOV #40000, R0 ;GET END OF BK
3669 017326 010001 MOV R0, R1 ;GET END OF BK
3670 017330 012737 017342 000004 MOV #2$, #ERRVEC ;SET UP TIMEOUT VECTOR
3671 017336 011020 1$: MOV (R0), (R0)+ ;SEARCH FOR END OF MEMORY
3672 017340 000776 BR 1$ ;KEEP SEARCHING
3673 017342 022626 2$: CMP (SP)+, (SP)+ ;RESTORE STACK POINTER
3674 017344 012737 025060 000004 MOV #ERRTRP, #ERRVEC ;RESET TIMEOUT VECTOR.
3675 017352 010046 MOV R0, -(SP) ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3676 017354 012702 002734 MOV #1500., R2 ;SET UP WORD COUNTER
3677 017360 014041 3$: MOV -(R0), -(R1) ;SAVE THE LOADERS
3678 017362 005302 DEC R2 ;COUNT THE WORDS
3679 017364 001375 BNE 3$ ;BRANCH IF MORE WORDS
3680 017366 012667 162126 MOV (SP)+, LMAD ;SAVE LAST MEMORY ADDRESS
3681 017372 000207 4$: RTS PC ;RETURN

```

```

3683 .SBTTL PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
3684 .....
3685 * PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
3686 * FIND OUT WHICH REGISTER DETECTED THE ERROR.
3687 * THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
3688 .....
3689 PESRV: MOV (SP), SBDADR ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
3690 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3691 .WORD UNEXPT ;ADDRESS OF MESSAGE TO BE TYPED
3692 ;"UNEXPECTED MEMORY PARITY TRAP."
3693 MOV R1,-(SP) ;PUSH R1 ON STACK
3694 MOV R3,-(SP) ;PUSH R3 ON STACK
3695 .MPRX, R3 ;GET POINTER TO PARITY REGISTERS.
3696 1$: TST @R3+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3697 BMI 3$ ;BR IF THIS REGISTER SHOWS THE ERROR.
3698 TST (R3) ;CHECK FOR TABLE TERMINATOR.
3699 BNE 1$ ;BR IF MORE REGISTERS.
3700 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3701 ;***ERROR*** NO REGISTER INDICATED ERROR
3702 .WORD 24 ;ERROR TYPE CODE.
3703 BR 4$ ;EXIT
3704 2$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
3705 BEQ 4$ ;BR IF NO MORE PARITY REGISTERS.
3706 TST @R3+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3707 BPL 2$ ;BR IF NO ERROR FLAG.
3708 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3709 .WORD MTOE ;ADDRESS OF MESSAGE TO BE TYPED
3710 ;"MORE THAN ONE ERROR FOUND."
3711 3$:
3712 64$: JSR PC, SPRNTQ ;SET UP VALUES FOR ERROR PRINTING.
3713 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3714 .WORD 25 ;ERROR TYPE CODE.
3715 JSR PC, PSCAN ;GO SCAN MEMORY FOR BAD PARITY.
3716 BR 2$ ;GO LOOK FOR MORE ERRORS.
3717 4$:
3718 MOV (SP)+,R3 ;POP STACK INTO R3
3719 MOV (SP)+,R1 ;POP STACK INTO R1
3720 RTI ;RETURN.
3721 .....
3722 ;*****
3723 ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
3724 ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
3725 ;*****
3726 MAMF: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
3727 BEQ MAMF2 ;EXIT IF NO PARITY REGISTERS.
3728 BIT #SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
3729 BNE MAMF2 ;EXIT IF NO PARITY ERROR DETECTION.
3730 TST RELOCF ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
3731 BEQ SETAE ;BR IF PROG IN BANK 0.
3732 BIT #SW5, @SWR ;CHECK IF VECTORS PROTECTED.
3733 BNE SETAE ;BR IF VECTOR AREA PROTECTED.
3734 CMP FSTADR, #1000 ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
3735 BLO MAMF2 ;EXIT IF VECTORS EXPOSED TO TESTING.

```

```

3736 017546 016737 162064 000114 SETAE: MOV .PESRV, 2#PARVEC ;SET PARITY ERROR TRAP VECTOR
3737 017554 005037 000116 CLR 2#PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
3738 017560 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
3739 017562 016703 162042 MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
3740 017566 052733 000001 MAMF1: BIS #AE, 2(R3)+ ;SET ACTION ENABLE BIT IN PARITY REG
3741 017572 005713 TST (R3) ;CHECK FOR END OF TABLE.
3742 017574 001374 BNE MAMF1 ;BR IF MORE PARITY REGISTERS.
3743 017576 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3744 017600 000207 MAMF2: RTS PC ;RETURN.
3745
3746 ;*****
3747 ;* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3748 ;*****
3749 017602 005767 162470 CKPMER: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
3750 017606 001437 BEQ 4$ ;BR IF NO PARITY REGISTERS.
3751 017610 032777 000100 161322 BIT #SW6, 2SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3752 017616 001033 BNE 4$ ;BR IF PARITY ERROR CHECKING INHIBITED.
3753 017620 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
3754 017622 016703 162002 MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
3755 017626 005733 1$: TST 2(R3)+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3756 017630 100023 BPL 3$ ;BR IF NO ERROR
3757 017632 032773 000001 177776 BIT #BIT0, 2-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3758 017640 001010 BNE 2$ ;BR IF NO ACTION ENABLE.
3759 017642 004767 000422 64$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
3760 017646 004767 001732 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3761 017652 000026 .WORD 26 ;ERROR TYPE CODE.
3762 017654 000411 BR 3$
3763 017656 004767 000026 JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3764 017662 2$:
3765 017662 004767 000402 65$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
3766 017666 004767 001712 JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3767 017672 000027 .WORD 27 ;ERROR TYPE CODE.
3768 017674 004767 000010 JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3769 017700 005713 3$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
3770 017702 001351 BNE 1$ ;BR IF MORE.
3771 017704 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3772 017706 000207 4$: RTS PC ;RETURN.
3773
3774 ;*****
3775 ;* THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY
3776 ;* TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
3777 ;* LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
3778 ;*****

```



```

3779 017710          PSCAN:
3780 017710 010046   MOV      R0,-(SP)      ;; PUSH R0 ON STACK
3781 017712 010146   MOV      R1,-(SP)      ;; PUSH R1 ON STACK
3782 017714 010246   MOV      R2,-(SP)      ;; PUSH R2 ON STACK
3783 017716 010346   MOV      R3,-(SP)      ;; PUSH R3 ON STACK
3784 017720 010446   MOV      R4,-(SP)      ;; PUSH R4 ON STACK
3785 017722 013746 000114  MOV      @#114,-(SP)    ;; PUSH @#114 ON STACK
3786 017726 013746 000116  MOV      @#116,-(SP)    ;; PUSH @#116 ON STACK
3787 017732 004567 003520  JSR      R5, $SPRINT    GO PRINT OUT THE FOLLOWING MESSAGE.
3788 017736 026602          .WORD    SCANM          ADDRESS OF MESSAGE TO BE TYPED
3789                                "SCANNING MEMORY FOR BAD PARITY."
3790 017740 012700 000001   MOV      #BIT0, R0      SET BIT POINTER TO FIRST BANK.
3791 017744 005001   CLR      R1            CLR HI 64K POINTER.
3792 017746 005002   CLR      R2            INIT ADDRESS POINTER.
3793 017750 005004   CLR      R4            INIT ERROR DETECTED FLAG.
3794 017752 004767 000256   JSR      PC, CLRPAR    CLEAR THE PARITY REGISTERS.
3795 017756 012737 000116 000114  MOV      @#116, @#114  HALT IF ANOTHER PARITY TRAP.
3796 017764 005037 000116   CLR      @#116
3797 017770 005767 160612   TST     MMAVA          CHECK FOR MEMORY MANAGEMENT.
3798 017774 001406   BEQ     1$            BR IF NO MEM MGMT.
3799 017776 013746 172344   MOV      @#KIPAR2,-(SP) ;; PUSH @#KIPAR2 ON STACK
3800 020002 005037 172344   CLR      @#KIPAR2      INIT MEM MGMT TO POINT TO BANK 0.
3801 020006 012702 040000   MOV      #40000, R2    SET ADR POINTER TO PAR2.
3802 020012 030067 161506   1$: BIT     R0, MEMMAP    CHECK IF THIS BANK OF MEM EXISTS.
3803 020016 001003   BNE     2$            BR IF THIS BANK EXISTS.
3804 020020 030167 161502   BIT     R1, MEMMAP+2  CHECK HI 64K MAP.
3805 020024 001442   BEQ     10$           BR IF THIS BANK DOESN'T EXIST.
3806 020026          2$:
3807 020026 010146   MOV      R1,-(SP)      ;; PUSH R1 ON STACK
3808 020030 111201   MOV     (R2), R1        READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
3809 020032 016703 161572   MOV     .MPRX, R3      SET UP POINTER TO PARITY REGISTERS.
3810 020036 005733   4$: TST     @ (R3)+      CHECK FOR THE ERROR FLAG.
3811 020040 100024   BPL     6$            BR IF NO ERROR FLAG.
3812 020042 005704   TST     R4            CHECK IF FIRST ERROR, THIS SCAN.
3813 020044 001003   BNE     5$            BR IF MORE THAN ONE ERROR FOUND.
3814 020046 005367 161040   DEC     $ERTTL        ADJUST ERROR COUNT.
3815 020052 005204   INC     R4            SET FLAG TO INDICATE ERROR FOUND.
3816 020054          5$:
3817 020054 004767 000210 64$: JSR      PC, $SPRINTQ   SET UP VALUES FOR ERROR PRINTING.
3818 020060 004767 001520   JSR     PC, $ERROR     *** ERROR *** (GO TYPE A MESSAGE)
3819 020064 000030   .WORD   30            ERROR TYPE CODE.
3820 020066 111212   MOV     (R2), (R2)     REWRITE THE LOCATION TO CLEAR BAD PARITY.
3821 020070 005053   CLR     @-(R3)         CLEAR THE ERROR FLAG.
3822 020072 105712   TST    (R2)           CHECK IF THE PARITY ERROR WAS CLEARED.
3823 020074 005733   TST    @ (R3)+       CHECK FOR THE ERROR FLAG.
3824 020076 100005   BPL     6$            BR IF IT IS OK.
3825 020100 004567 003352   JSR     R5, $SPRINT    GO PRINT OUT THE FOLLOWING MESSAGE.
3826 020104 026644   .WORD   PEWNC         ADDRESS OF MESSAGE TO BE TYPED
3827                                "PARITY ERROR WILL NOT CLEAR."
3828 020106 005073 177776   CLR     @-2(R3)       CLEAR OUT THE PARITY ERROR FLAG.
3829 020112 005713   TST    (R3)          CHECK FOR THE END OF REG ADR TABLE.
3830 020114 001350   BNE     4$            BR IF MORE PARITY REGISTERS.
3831 020116 005202   INC     R2            GO TO NEXT MEMORY ADDRESS.
3832 020120 032702 017777   BIT     #MASK4K, R2   CHECK FOR END OF 4K BANK.
3833 020124 001341   BNE     3$            BR IF MORE MEMORY THIS BANK.
3834 020126 012601   MOV     (SP)+, R1     ;; POP STACK INTO R1

```

```

3835 020130 000402 BR 11$ ;BR TO CHECK FOR NEXT BANK.
3836 020132 062702 020000 10$: ADD #20000, R2 ;SKIP BANKS THAT AREN'T THERE.
3837 020136 005767 160444 11$: TST MMAVA ;CHECK FOR MEM MGMT.
3838 020142 001413 BEQ 12$ ;BR IF NO MEM MGMT.
3839 020144 062737 000200 172344 ADD #200, @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
3840 020152 012702 040000 MOV #40000, R2 ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
3841 020156 006300 ASL R0 ;UPDATE BANK POINTER.
3842 020160 006101 ROL R1 ;HI 64K.
3843 020162 100313 BPL 1$ ;BR IF MORE BANKS.
3844 020164 012637 172344 MOV (SP)+, @#KIPAR2 ;POP STACK INTO @#KIPAR2
3845 020170 000402 BR 20$ ;GO CHECK IF ANY ERRORS FOUND.
3846 020172 106300 12$: ASLB R0 ;UPDATE POINTER TO NEXT BANK.
3847 020174 100306 BPL 1$ ;BR IF MORE BANKS.
3848 020176 005704 20$: TST R4 ;CHECK IF ANY PARITY ERRORS DETECTED.
3849 020200 001003 BNE 21$ ;BR IF ERRORS DETECTED.
3850 020202 004567 003250 JSR R5, SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3851 020206 025652 .WORD NOPE$ ;ADDRESS OF MESSAGE TO BE TYPED
3852 020210 21$: MOV (SP)+, @#116 ;POP STACK INTO @#116
3853 020210 012637 000116 MOV (SP)+, @#114 ;POP STACK INTO @#114
3854 020214 012637 000114 MOV (SP)+, R4 ;POP STACK INTO R4
3855 020220 012604 MOV (SP)+, R3 ;POP STACK INTO R3
3856 020222 012603 MOV (SP)+, R2 ;POP STACK INTO R2
3857 020224 012602 MOV (SP)+, R1 ;POP STACK INTO R1
3858 020226 012601 MOV (SP)+, R0 ;POP STACK INTO R0
3859 020230 012600 RTS PC ;RETURN.
3860 020232 000207
3861
3862 ;*****
3863 ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
3864 ;*****
3865 020234 CLRPAR:
3866 020234 010346 161366 MOV R3, -(SP) ;PUSH R3 ON STACK
3867 020236 016703 1$: MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
3868 020242 005713 TST (R3) ;CHECK FOR THE TABLE TERMINATOR.
3869 020244 001402 2$: BEQ 2$ ;BR IF DONE ALL PARITY REGISTERS.
3870 020246 005033 CLR @R3+ ;CLEAR THE PARITY REGISTER.
3871 020250 000774 BR 1$ ;BR FOR MORE
3872 020252 2$: MOV (SP)+, R3 ;POP STACK INTO R3
3873 020252 012603 RTS PC ;RETURN.
3874 020254 000207
3875
3876 .SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3877 ;*****
3878 ;* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.SCMTAG)
3879 ;* FOR ERROR PRINTOUT BY .SERROR & .SERRTYP ROUTINES FROM **SYSMAC**.
3880 ;*****
3881 020256 010267 160636 SPRINT: MOV R2, $GDADR ;SAVE THE ADDRESS UNDER TEST.
3882 020262 005067 160636 CLR $GDADR ;SHOULD BE DATA IS "0".
3883 020266 000430 BR SPRINTB
3884
3885 020270 014367 160664 SPRINTQ: MOV -(R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
3886 020274 013367 160662 MOV @R3+, $TMP1 ;GET THE CONTENTS OF THE PARITY REG.
3887 020300 000402 BR SPRINTO
3888
3889 020302 011367 160652 SPRINTP: MOV (R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
3890 020306 010267 160606 SPRINTO: MOV R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
    
```

```

3891 020312 000414 BR SPRNTA ;BR TO COMMON SECTION.
3892
3893 020314 010267 160600 SPRNT1: MOV R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
3894 020320 005367 160574 DEC $GDADR ;ADJUST IT FOR PRINTOUT.
3895 020324 000407 BR SPRNTA ;BR TO COMMON SECTION.
3896
3897 020326 010367 160626 SPRNT3: MOV R3, $TMPD ;GET THE DATA IN R3.
3898 020332 010267 160562 SPRNT2: MOV R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED
3899 020336 162767 000002 160554 SUB #2, $GDADR ;ADJUST IT FOR PRINTOUT.
3900 020344 010067 160554 SPRNTA: MOV R0, $GDDAT ;GET WHAT THE DATA SHOULD BE
3901 020350 010167 160552 SPRNTB: MOV R1, $BDDAT ;GET WHAT THE DATA WAS
3902 020354 000207 RTS PC ;RETURN TO ENTER ERROR ROUTINES
3903
3904 ;*****
3905 ;* SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
3906 ;* R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
3907 ;*****
3908 020356 005710 TYPMAP: TST (R0) ;CHECK IF ANY MEMORY IN MAP...LO 64K.
3909 020360 001007 BNE 1$ ;BR IF MEMORY IN MAP.
3910 020362 005760 000002 TST 2(R0) ;...HI 64K.
3911 020366 001004 BNE 1$ ;BR IF MEMORY IN MAP.
3912 020370 004567 003062 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3913 020374 026232 .WORD NOMEM ;ADDRESS OF MESSAGE TO BE TYPED
3914 ;"NO MEMORY FOUND."
3915 020376 000475 BR 6$ ;EXIT
3916 020400 1$:
3917 020400 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
3918 020402 010246 MOV R2,-(SP) ;PUSH R2 ON STACK
3919 020404 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
3920 020406 010446 MOV R4,-(SP) ;PUSH R4 ON STACK
3921 020410 012701 000001 MOV #BIT0, R1 ;SET UP BANK POINTER...LO 64K.
3922 020414 005002 CLR R2 ;HI 64K.
3923 020416 012703 177777 MOV #-1, R3 ;SET UP ADDRESS POINTER TO -1.
3924 020422 010304 MOV R3, R4 ;HI BITS OF ADDRESS AS WILL.
3925 020424 030110 2$: BIT R1, (R0) ;CHECK THE MAP FOR THIS BANK.
3926 020426 001014 BNE 3$ ;BR IF THIS BANK PRESENT.
3927 020430 030260 000002 BIT R2, 2(R0) ;CHECK HI 64K MAP.
3928 020434 001011 BNE 3$ ;BR IF THIS BANK PRESENT.
3929 020436 105703 TSTB R3 ;CHECK FOR PREVIOUS PRINTOUT.
3930 020440 001042 BNE 5$ ;BR IF ALREADY TYPED "TO"
3931 020442 162703 000001 SUB #1, R3 ;BACK UP TO LAST ADR OF PREVIOUS BANK.
3932 020446 005604 SBC R4 ;...HI ADDRESS BITS.
3933 020450 004567 003002 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3934 020454 025463 .WORD TO ;ADDRESS OF MESSAGE TO BE TYPED
3935 020456 000410 BR 4$ ;GO TO TYPE THE ADDRESS.
3936 020460 105703 3$: TSTB R3 ;CHECK FOR PREVIOUS TYPEOUT.
3937 020462 001431 BEQ 5$ ;BR IF ALREADY TYPE "FROM".
3938 020464 062703 000001 ADD #1, R3 ;POINT TO FIRST ADDRESS OF THIS BANK.
3939 020470 005504 ADC R4 ;...HI BITS OF ADDRESS.
3940 020472 004567 002760 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3941 020476 025453 .WORD FROM ;ADDRESS OF MESSAGE TO BE TYPED
3942 020500 4$:
3943 020500 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
3944 020502 010446 MOV R4,-(SP) ;PUSH R4 ON STACK
3945 020504 006303 ASL R3 ;BIT 15 INTO C-BIT
3946 020506 006104 ROL R4 ;BIT 15 INTO R4.
    
```

```

3947 020510 006003          ROR    R3          ;RESTORE BITS 14-0.
3948 020512 010446          MOV    R4,-(SP)    ;SAVE R4 FOR TYPEOUT
3949                                ;TYPE ADDRESS BITS 21-15
3950                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3951                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3952 020514 013746 177776    MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3953 020520 004767 004104    JSR    PC, $TYPOS ;GO TO THE SUBROUTINE
3954 020524 003              .BYTE 3           ;TYPE 3 DIGIT(S)
3955 020525 000              .BYTE 0           ;SUPPRESS LEADING ZEROS
3956 020526 010346          MOV    R3,-(SP)    ;SAVE R3 FOR TYPEOUT
3957                                ;TYPE ADDRESS BITS 14-0
3958                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3959                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3960 020530 013746 177776    MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3961 020534 004767 004070    JSR    PC, $TYPOS ;GO TO THE SUBROUTINE
3962 020540 005              .BYTE 5           ;TYPE 5 DIGIT(S)
3963 020541 001              .BYTE 1           ;TYPE LEADING ZEROS
3964 020542 012604          MOV    (SP)+,R4    ;POP STACK INTO R4
3965 020544 012603          MOV    (SP)+,R3    ;POP STACK INTO R3
3966 020546 062703 020000    5$:   ADD    #20000, R3 ;UPDATE TO NEXT BANK.
3967 020552 005504          ADC    R4          ;HI ADDRESS BITS.
3968 020554 006301          ASL    R1          ;SHIFT POINTER...LO 64K.
3969 020556 006102          ROL    R2          ;HI 64K.
3970 020560 103321          BCC    2$         ;BR IF MORE BANKS.
3971 020562 012604          MOV    (SP)+,R4    ;POP STACK INTO R4
3972 020564 012603          MOV    (SP)+,R3    ;POP STACK INTO R3
3973 020566 012602          MOV    (SP)+,R2    ;POP STACK INTO R2
3974 020570 012601          MOV    (SP)+,R1    ;POP STACK INTO R1
3975 020572 000207          6$:   RTS    PC     ;RETURN.
3976                                .SBTTL SCOPE HANDLER ROUTINE
3977                                ;*****
3978                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3979                                ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3980                                ;*AND LOAD THE ERROR FLAG($ERFLG) INTO DISPLAY<15:08>
3981                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3982                                ;*SW14=1 LOOP ON TEST
3983                                ;*SW11=1 INHIBIT ITERATIONS
3984                                ;*SW09=1 LOOP ON ERROR
3985                                ;*SW08=1 LOOP ON TEST IN SWR<4:0>
3986                                ;*CALL
3987                                ;* SCOPE ;;SCOPE=IOT
3988
3989
3990
3991 020574          $SCOPE:
3992                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
3993                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
3994 020574 013746 177776    MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3995 020600 004767 001524    JSR    PC, $CKSWR ;GO TO THE SUBROUTINE
3996 020604 012504          MOV    (R5)+, R4   ;SAVE MINIMUM BLOCK MASK NEXT TEST.
3997 020606 010516          MOV    R5, (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
3998 020610 032777 040000 160322 1$:   BIT    #BIT14, @SWR ;LOOP ON PRESENT TEST?
3999 020616 001117          BNE    $OVER       ;YES IF SW14=1
4000                                ;*****START OF CODE FOR THE XOR TESTER*****
4001 020620 000416          $XTSTR: BR    6$   ;IF RUNNING ON THE "XOR" TESTER CHANGE
4002                                ;THIS INSTRUCTION TO A "NOP" (NOP=240)
    
```

4003	020622	013746	000004		MOV	@#ERRVEC, -(SP)	;; SAVE THE CONTENTS OF THE ERROR VECTOR
4004	020626	012737	020646	000004	MOV	#5\$, @#ERRVEC	;; SET FOR TIMEOUT
4005	020634	005737	177060		TST	@#177060	;; TIME OUT ON XOR?
4006	020640	012637	000004		MOV	(SP)+, @#ERRVEC	;; RESTORE THE ERROR VECTOR
4007	020644	000466			BR	\$SVLAD	;; GO TO THE NEXT TEST
4008	020646	022626			5\$: CMP	(SP)+, (SP)+	;; CLEAR THE STACK AFTER A TIME OUT
4009	020650	012637	000004		MOV	(SP)+, @#ERRVEC	;; RESTORE THE ERROR VECTOR
4010	020654	000426			BR	7\$	;; LOOP ON THE PRESENT TEST
4011	020656				6\$: ; #####	END OF CODE FOR THE XOR TESTER#####	
4012	020656	032777	000400	160254	BIT	#BIT08, @SWR	;; LOOP ON SPEC. TEST?
4013	020664	001407			BEQ	2\$	;; BR IF NO
4014	020666	017746	160246		MOV	@SWR, -(SP)	;; SET DESIRED TEST NUM. FROM SWR
4015	020672	042716	000340		BIC	##\$WARMK, (SP)	;; STRIP AWAY UNDESIRED BITS
4016	020676	122667	160200		CMPB	(SP)+, \$TSTNM	;; ON THE RIGHT TEST?
4017	020702	001465			BEQ	\$OVER	;; BR IF YES
4018	020704	105767	160173		2\$: TSTB	\$ERFLG	;; HAS AN ERROR OCCURRED?
4019	020710	0C1421			BEQ	3\$	;; BR IF NO
4020	020712	126767	160177	160163	CMPB	\$ERMAX, \$ERFLG	;; MAX. ERRORS FOR THIS TEST OCCURRED?
4021	020720	101015			BHI	3\$	;; BR IF NO
4022	020722	032777	001000	160210	BIT	#BIT09, @SWR	;; LOOP ON ERROR?
4023	020730	001404			BEQ	4\$	;; BR IF NO
4024	020732	016767	160152	160146	7\$: MOV	\$LPERR, \$LPADR	;; SET LOOP ADDRESS TO LAST SCOPE
4025	020740	000446			BR	\$OVER	
4026	020742	105067	160135		4\$: CLRB	\$ERFLG	;; ZERO THE ERROR FLAG
4027	020746	005067	160216		CLR	\$TIMES	;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
4028	020752	000415			BR	1\$	;; ESCAPE TO THE NEXT TEST
4029	020754	032777	004000	160156	3\$: BIT	#BIT11, @SWR	;; INHIBIT ITERATIONS?
4030	020762	001011			BNE	1\$	;; BR IF YES
4031	020764	005767	160222		TST	\$PASS	;; IF FIRST PASS OF PROGRAM
4032	020770	001406			BEQ	1\$	;; INHIBIT ITERATIONS
4033	020772	005267	160106		INC	\$ICNT	;; INCREMENT ITERATION COUNT
4034	020776	026767	160166	160100	CMP	\$TIMES, \$ICNT	;; CHECK THE NUMBER OF ITERATIONS MADE
4035	021004	002024			BGE	\$OVER	;; BR IF MORE ITERATION REQUIRED
4036	021006	012767	000001	160070	1\$: MOV	#1, \$ICNT	;; REINITIALIZE THE ITERATION COUNTER
4037	021014	016767	000552	160146	MOV	\$MXCNT, \$TIMES	;; SET NUMBER OF ITERATIONS TO DO
4038	021022	105267	160054		##\$SVLAD: INCB	\$TSTNM	;; COUNT TEST NUMBERS
4039	021026	116767	160050	160154	MOVB	\$TSTNM, \$TESTN	;; SET TEST NUMBER IN APT MAILBOX
4040	021034	011667	160046		MOV	(SP), \$LPADR	;; SAVE SCOPE LOOP ADDRESS
4041	021040	011667	160044		MOV	(SP), \$LPERR	;; SAVE ERROR LOOP ADDRESS
4042	021044	005067	160122		CLR	\$ESCAPE	;; CLEAR THE ESCAPE FROM ERROR ADDRESS
4043	021050	112767	000001	160037	MOVB	#1, \$ERMAX	;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4044	021056	016777	160020	160056	##\$OVER: MOV	\$TSTNM, @DISPLAY	;; DISPLAY TEST NUMBER
4045	021064	016716	160016		MOV	\$LPADR, (SP)	;; FUDGE RETURN ADDRESS
4046	021070	020516			INSERT: CMP	R5, (SP)	;; CHECK FOR LOOP ON TEST.
4047	021072	001402			BEQ	1\$	;; BR IF START NEXT TEST.
4048	021074	000167	000470		JMP	ENDINS	;; JMP IF LOOP ON LAST TEST.
4049	021100	012767	037777	160500	1\$: MOV	#37777, \$BLKMSK	;; SET BK BOUNDARY MASK.
4050	021106	005767	160100		TST	\$PASS	;; CHECK FOR PASS 0.
4051	021112	001404			BEQ	2\$	;; BR IF PASS 0
4052	021114	126727	157762	000021	CMPB	\$TSTNM, #21	;; CHECK IF IN SECTION 3.
4053	021122	103002			BHIS	3\$	;; BR IF IN SECTION 3.
4054	021124	006267	160456		2\$: ASR	\$BLKMSK	;; RESET BOUNDARY TO 4K.
4055	021130	016767	160426	160426	3\$: MOV	\$FSTADR, \$TMPFAD	;; GET FIRST ADDRESS.
4056	021136	005767	157436		TST	\$RELOC	;; CHECK IF PRG RELOCATED.
4057	021142	001430			BEQ	4\$	;; BR IF NOT RELOCATED.
4058	021144	032777	000040	157766	BIT	##\$W05, @SWR	;; CHECK IF LOC 0-776 TO BE PROTECTED.

4059	021152	001424			BEQ	4\$			:BR IF SW NOT SET.
4060	021154	026727	160404	001000	CMP	TMPFAD, #1000			:CHECK IF NOT BEING TESTED.
4061	021162	103020			BHIS	4\$			:BR IF ALREADY PROTECTED.
4062	021164	012767	001000	160372	MOV	#1000, TMPFAD			:RESET FIRST ADDRESS.
4063	021172	052767	000001	160370	BIS	#BITO, FADMAP			:SET FLAG IN FIRST BANK.
4064	021200	026727	160370	001000	CMP	LSTADR, #1000			:CHECK IF GONE PAST LAST ADR.
4065	021206	101006			BHI	4\$			:BR IF ENOUGH MEMORY.
4066	021210	004567	002242		JSR	R5, SPRINT			:GO PRINT OUT THE FOLLOWING MESSAGE.
4067	021214	026703			.WORD	NOMTST			:ADDRESS OF MESSAGE TO BE TYPED
4068									: "NO MEMORY TESTED"
4069	021216	016716	160422		MOV	.TST32, (SP)			:ADJUST RETURN ADR FOR ABORT.
4070	021222	000207			RTS	PC			:ABORT.
4071	021224	016767	160344	160344	4\$:	MOV	LSTADR, TEMPLAD		:GET LAST ADDRESS.
4072	021232	016767	160276	160270	MOV	SAVTST, TSTMAP			:GET TEST MAP, LO 64K.
4073	021240	016767	160272	160264	MOV	SAVTST+2, TSTMAP+2			:HI 64K.
4074	021246	046767	157330	160254	BIC	PRGMAP, TSTMAP			:DON'T TEST OVER THE PROGRAM.
4075	021254	046767	157324	160250	BIC	PRGMAP+2, TSTMAP+2			
4076	021262	005767	157724		TST	\$PASS			:CHECK FOR FIRST PASS
4077	021266	001011			BNE	10\$			:BR IF NOT FIRST PASS.
4078	021270	032767	000003	160232	BIT	#3, TSTMAP			:CHECK IF FIRST TWO BANKS AVAILABLE.
4079	021276	001405			BEQ	10\$			:NOT TESTING FIRST 2 BANKS.
4080	021300	042767	177774	160222	BIC	#177774, TSTMAP			:CLR ALL BUT FIRST 2 BANKS.
4081	021306	005067	160220		CLR	TSTMAP+2			
4082	021312	005704			10\$:	TST	R4		:CHECK FOR A MINIMUM BLOCK SIZE.
4083	021314	001503			BEQ	20\$			:BR IF NO MIN BLOCK SIZE.
4084	021316	030467	160242		BIT	R4, TMPFAD			:CHECK IF FIRST ADR ON BLOCK BOUNDARY.
4085	021322	001416			BEQ	11\$			:BR IF FIRST ADR ON BLOCK BOUNDARY.
4086	021324	050467	160234		BIS	R4, TMPFAD			:ADJUST FIRST ADR TO END OF BLOCK.
4087	021330	005267	160230		INC	TMPFAD			:FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4088	021334	032767	017777	160222	BIT	#MASK4K, TMPFAD			:CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4089	021342	001006			BNE	11\$			:BR IF NOT ON 4K BOUNDARY.
4090	021344	046767	160220	160156	BIC	FADMAP, TSTMAP			:DON'T TEST FIRST BANK.
4091	021352	046767	160214	160152	BIC	FADMAP+2, TSTMAP+2			
4092	021360	030467	160212		11\$:	BIT	R4, TEMPLAD		:CHECK IF LAST ADR ON BLOCK BOUNDARY.
4093	021364	001414			BEQ	12\$			:BR IF ON BLOCK BOUNDARY.
4094	021366	040467	160204		BIC	R4, TEMPLAD			:ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4095	021372	032767	017777	160176	BIT	#MASK4K, TEMPLAD			:CHECK IF ADJUSTED TO 4K BOUNDARY.
4096	021400	001006			BNE	12\$			:BR IF NOT ON 4K BOUNDARY.
4097	021402	046767	160174	160120	BIC	LADMAP, TSTMAP			:SKIP TESTING LAST BANK.
4098	021410	046767	160170	160114	BIC	LADMAP+2, TSTMAP+2			
4099	021416	036767	160146	160156	12\$:	BIT	FADMAP, LADMAP		:CHECK IF FIRST AND LAST IN SAME BANK.
4100	021424	001004			BNE	13\$			:BR IF IN SAME BANK.
4101	021426	036767	160140	160150	BIT	FADMAP+2, LADMAP+2			:...UPPER 64K.
4102	021434	001404			BEQ	14\$			:BR IF FIRST AND LAST NOT SAME BANK.
4103	021436	026767	160134	160120	13\$:	CMP	TEMPLAD, TMPFAD		:CHECK IF ANY MEMORY LEFT.
4104	021444	101406			BLOS	15\$			:BR IF NO MEMORY TO TEST.
4105	021446	005767	160056		14\$:	TST	TSTMAP		:CHECK IF ANY BANKS LEFT TO TEST!!
4106	021452	001017			BNE	16\$			:BR IF TEST MAP NOT EMPTY.
4107	021454	005767	160052		TST	TSTMAP+2			:CHECK FOR ANY BANKS.
4108	021460	001014			BNE	16\$			:BR IF TEST MAP NOT EMPTY.
4109	021462				15\$:				
4110	021462	004567	001770		JSR	R5, SPRINT			:GO PRINT OUT THE FOLLOWING MESSAGE.
4111	021466	026727			.WORD	SKPMES			:ADDRESS OF MESSAGE TO BE TYPED
4112									: "SKIPPING TEST #"
4113	021470	005046			CLR	-(SP)			:CLEAR THE WORD ON THE STACK.
4114	021472	116716	157404		MOVB	\$TSTNM, (SP)			:PUT THE DATA ON THE STACK.

```

4115      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4116      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4117      021476 013746 177776      MOV      @#PSW, -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
4118      021502 004767 003122      JSR      PC,      $TYPOS  ;GO TO THE SUBROUTINE
4119      021506      003      .BYTE      3      ;TYPE 3 DIGITS.
4120      021507      001      .BYTE      1      ;TYPE LEADING ZEROS.
4121      021510 000427      BR      ENDINS      ;RETURN TO SKIP TEST.
4122      021512 062716 000004      ADD      #4,      (SP)      ;SKIP THE SKIP ON RETURN.
4123      021516 062767 000004 157362      ADD      #4,      $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
4124      021524 012767 017777 160034 20$:      MOV      #MASK4K, FADMSK ;GET 4K MASK.
4125      021532 016705 160026      MOV      TMPFAD, R5      ;GET FIRST ADR.
4126      021536 040567 160024      21$:      BIC      R5,      FADMSK ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4127      021542 006305      ASL      R5      ;MOVE LOWEST BIT UP ONE.
4128      021544 001374      BNE      21$      ;LOOP UNTIL OVERFLOW.
4129      021546 012767 017777 160024      MOV      #MASK4K, LADMSK ;SET MASK BITS
4130      021554 016705 160016      MOV      TEMPLAD, R5     ;GET LAST ADR.
4131      021560 040567 160014      22$:      BIC      R5,      LADMSK ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4132      021564 006305      ASL      R5      ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4133      021566 001374      BNE      22$      ;LOOP UNTIL OVERFLOW.
4134      021570 000207      ENDINS: RTS      PC      ;EXIT SCOPE ROUTINE BACK TO TEST.
4135      021572 000004      $MXCNT: 4      ;MAX. NUMBER OF ITERATIONS
4136      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4137      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4138      021574 013746 177776      MOV      @#PSW, -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
4139      021600 004767 000524      JSR      PC,      $CKSWR  ;GO TO THE SUBROUTINE
4140      .SBTTL  ERROR HANDLER ROUTINE
4141
4142      ;*****
4143      ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4144      ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4145      ;AND GO TO $ERRTYP ON ERROR
4146      ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4147      ;$SW15=1      HALT ON ERROR
4148      ;$SW13=1      INHIBIT ERROR TYPEOUTS
4149      ;$SW10=1      BELL ON ERROR
4150      ;$SW09=1      LOOP ON ERROR
4151      ;CALL
4152      ;*      ERROR      N      ;;ERROR=EMT AND N=ERROR ITEM NUMBER
4153
4154      021604      $ERROR:
4155      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4156      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4157      021604 013746 177776      MOV      @#PSW, -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
4158      021610 004767 000514      JSR      PC,      $CKSWR  ;GO TO THE SUBROUTINE
4159      021614 062716 000002      ADD      #2,      (SP)      ;ADJUST POINTER PAST CODE WORD.
4160      021620 105267 157257      7$:      INCB      $ERFLG      ;SET THE ERROR FLAG
4161      021624 001775      BEQ      7$      ;DON'T LET THE FLAG GO TO ZERO
4162      021626 016777 157250 157306      MOV      $STNM, @DISPLAY ;DISPLAY TEST NUMBER AND ERROR FLAG
4163      021634 032777 002000 157276      BIT      #BIT10, @SWR     ;BELL ON ERROR?
4164      021642 001403      BEQ      1$      ;NO - SKIP
4165      021644 004567 001606      JSR      R5,      $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
4166      021650 001174      .WORD      $BELL      ;ADDRESS OF MESSAGE TO BE TYPED
4167      021652 005267 157234      1$:      INC      $ERTTL      ;COUNT THE NUMBER OF ERRORS
4168      021656 011667 157234      MOV      (SP), $ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
4169      021662 162767 000002 157226      SUB      #2, $ERRPC
4170      021670 117767 157222 157216      MOV      @ $ERRPC, $ITEMB ;;STRIP AND SAVE THE ERROR ITEM CODE

```

```

4171 021676 032777 020000 157234      BIT      #BIT13,JSWR      ;;SKIP TYPEOUT IF SET
4172 021704 001005                    BNE      20$           ;;SKIP TYPEOUTS
4173 021706 004767 000116                    JSR      PC,$ERRTYP    ;;GO TO USER ERROR ROUTINE
4174 021712 004567 001540                    JSR      RS,$SPRINT    ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4175 021716 001201                    .WORD    $CRLF         ;;ADDRESS OF MESSAGE TO BE TYPED
4176 021720                    20$:  CMPB     #APTENV,$ENV    ;;RUNNING IN APT MODE
4177 021720 122767 000001 157276                    BNE      2$           ;;NO SKIP APT ERROR REPORT
4178 021726 001007                    MOV      $ITEMB,21$   ;;SET ITEM NUMBER AS ERROR NUMBER
4179 021730 116767 157160 000004                    JSR      PC,$SATY4    ;;REPORT FATAL ERROR TO APT
4180 021736 004767 002044                    .BYTE    0
4181 021742 000                    21$:  .BYTE    0
4182 021743 000                    .BYTE    0
4183 021744 000777                    22$:  BR      22$
4184 021746 005777 157166                    2$:   TST      JSWR
4185 021752 100005                    BPL      3$           ;;APT ERROR LOOP
4186 021754 000000                    HALT     3$           ;;HALT ON ERROR
4187                    ;;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4188                    ;;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4189 021756 013746 177776                    MOV      2#PSW,-(SP)  ;;PUT THE PROCESSOR STATUS ON THE STACK
4190 021762 004767 000342                    JSR      PC,$CKSWR    ;;GO TO THE SUBROUTINE
4191 021766 032777 001000 157144                    3$:   BIT      #BIT09,JSWR  ;;LOOP ON ERROR SWITCH SET?
4192 021774 001402                    BEQ      4$           ;;BR IF NO
4193 021776 016716 157106                    MOV      $LPERR,(SP)  ;;FUDGE RETURN FOR LOOPING
4194 022002 005767 157164                    4$:   TST      $ESCAPE    ;;CHECK FOR AN ESCAPE ADDRESS
4195 022006 001402                    BEQ      5$           ;;BR IF NONE
4196 022010 016716 157156                    MOV      $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4197 022014                    5$:   CMP      #SENDAD,2#42  ;;ACT-11 AUTO-ACCEPT?
4198 022014 022737 014174 000042                    BNE      6$           ;;BRANCH IF NO
4199 022022 001001                    HALT     6$           ;;YES
4200 022024 000000
4201 022026
4202 022026 000207                    6$:   RTS      PC
4203                    ;;*****
4204
4205                    .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
4206
4207                    ;;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4208                    ;;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4209                    ;;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4210
4211                    $ERRTYP:
4212 022030 004567 001422                    JSR      RS,$SPRINT  ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4213 022034 001201                    .WORD    $CRLF         ;;ADDRESS OF MESSAGE TO BE TYPED
4214 022036 010046                    MOV      RO,-(SP)     ;;SAVE RO
4215 022040 005000                    CLR      RO           ;;PICKUP THE ITEM INDEX
4216 022042 156700 157046                    BISB    $ITEMB,RO
4217 022046 001007                    BNE      1$           ;;IF ITEM NUMBER IS ZERO, JUST
4218                    ;;TYPE THE PC OF THE ERROR
4219 022050 016746 157042                    MOV      $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
4220                    ;;ERROR ADDRESS
4221                    ;;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4222                    ;;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4223 022054 013746 177776                    MOV      2#PSW,-(SP)  ;;PUT THE PROCESSOR STATUS ON THE STACK
4224 022060 004767 002570                    JSR      PC,$TYPOC    ;;GO TO THE SUBROUTINE
4225 022064 000513                    BR      10$          ;;GET OUT
4226 022066 016767 157024 157420 1$:  MOV      $ERRPC,$VERPC ;;SET UP VIRTUAL PC FOR TYPEOUT.

```



```

4227 022074 166767 156500 157412 SUB RELOCF, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
4228 022102 005300 DEC RO ;ADJUST THE INDEX SO THAT IT WILL
4229 022104 006300 ASL RO ; WORK FOR THE ERROR TABLE
4230 022106 006300 ASL RO
4231 022110 006300 ASL RO
4232 022112 066700 157522 ADD .ERRTB, RO ;FORM TABLE POINTER
4233 022116 012067 000006 MOV (RO)+, 2$ ;PICKUP "ERROR MESSAGE" POINTER
4234 022122 001406 BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
4235 022124 004567 001326 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4236 022130 000000 2$: .WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
4237 022132 004567 001320 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4238 022136 001201 .WORD SCALF ;ADDRESS OF MESSAGE TO BE TYPED
4239 022140 012067 000006 3$: MOV (RO)+, 4$ ;PICKUP "DATA HEADER" POINTER
4240 022144 001406 BEQ 5$ ;SKIP TYPEOUT IF 0
4241 022146 004567 001304 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4242 022152 000000 4$: .WORD 0 ;"DATA HEADER" POINTER GOES HERE
4243 022154 004567 001276 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4244 022160 001201 .WORD SCALF ;ADDRESS OF MESSAGE TO BE TYPED
4245 022162 010146 5$: MOV R1, -(SP) ;SAVE R1
4246 022164 012001 MOV (RO)+, R1 ;PICKUP "DATA TABLE" POINTER
4247 022166 001451 BEQ 9$ ;BR IF NO DATA TO BE TYPED
4248 022170 066701 156404 ADD RELOCF, R1 ;ADJUST POINTER
4249 022174 012000 MOV (RO)+, RO ;PICKUP "DATA FORMAT" POINTER
4250 022176 066700 156376 ADD RELOCF, RO ;ADJUST POINTER.
4251 022202 105720 6$: TSTB (RO)+ ;CHECK THE FORMAT
4252 022204 001006 BNE 7$ ;BR IF NOT 16-BIT OCTAL
4253 022206 013146 MOV @ (R1)+, -(SP) ;SAVE @ (R1)+ FOR TYPEOUT
4254 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4255 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4256 022210 013746 177776 MOV @ #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4257 022214 004767 002434 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4258 022220 000426 BR 8$
4259 022222 100406 7$: BMI 17$ ;BRANCH IF NOT DECIMAL
4260 022224 013146 MOV @ (R1)+, -(SP) ;SAVE @ (R1)+ FOR TYPEOUT
4261 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
4262 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4263 022226 013746 177776 MOV @ #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4264 022232 004767 002140 JSR PC, $TYPDS ;GO TO THE SUBROUTINE
4265 022236 000417 BR 8$ ;SKIP
4266 022240 122760 177777 177777 17$: CMPB #-1, -1(RO) ;CHECK FOR 18-BIT ADDRESS FORMAT.
4267 022246 001004 BNE 18$ ;BR IF NOT 18-BIT ADDRESS FORMAT.
4268 022250 013146 MOV @ (R1)+, -(SP) ;PUT THE DATA ON THE STACK.
4269 022252 004767 002640 JSR PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4270 022256 000407 BR 8$ ;SKIP
4271 022260 18$: CLR -(SP) ;CLEAR THE WORD ON THE STACK.
4272 022260 005046 MOV @ (R1)+, (SP) ;PUT THE DATA ON THE STACK.
4273 022262 113116 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4274 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4275 022264 013746 177776 MOV @ #PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4276 022270 004767 002334 JSR PC, $TYPOS ;GO TO THE SUBROUTINE
4277 022274 003 .BYTE 3 ;TYPE 3 DIGITS.
4278 022274 001 .BYTE 1 ;TYPE LEADING ZEROS.
4279 022275 001 8$: TST (R1) ;IS THERE ANOTHER NUMBER?
4280 022276 005711 BEQ 9$ ;BR IF NO
4281 022300 001404 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4282 022302 004567 001150

```

```

4283 022306 022326 .WORD 11$ ;ADDRESS OF MESSAGE TO BE TYPED
4284 022310 000734 BR 6$ ;LOOP
4285
4286 022312 012601 9$: MOV (SP)+,R1 ;RESTORE R1
4287 022314 012600 10$: MOV (SP)+,R0 ;RESTORE R0
4288 022316 004567 001134 JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4289 022322 001201 .WORD $CALF ;ADDRESS OF MESSAGE TO BE TYPED
4290 022324 000207 RTS PC ;RETURN
4291 022326 000011 11$: .ASCIZ / / ;TAB CHARACTER.
4292 .EVEN
4293 .SBTTL TTY INPUT ROUTINE
4294
4295 ;*****
4296 .ENABL LSB
4297
4298 ;*****
4299 ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
4300 ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
4301 ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
4302 ;*WHEN OPERATING IN TTY FLAG MODE.
4303 022330 022767 000176 156602 $CKSWR: CMP #SWREG,SWR ;: IS THE SOFT-SWR SELECTED?
4304 022336 001104 BNE 15$ ;: BRANCH IF NO
4305 022340 105777 156600 TSTB @STKS ;: CHAR THERE?
4306 022344 100101 BPL 15$ ;: IF NO, DON'T WAIT AROUND
4307 022346 117746 156574 MOVB @STKB,-(SP) ;: SAVE THE CHAR
4308 022352 042716 177600 BIC #C177,(SP) ;: STRIP-OFF THE ASCII
4309 022356 022726 000007 CMP #7,(SP)+ ;: IS IT A CONTROL G?
4310 022362 001072 BNE 15$ ;: NO, RETURN TO USER
4311 022364 126727 156544 000001 CMPB $AUTOB,#1 ;: ARE WE RUNNING IN AUTO-MODE?
4312 022372 001466 BEQ 15$ ;: BRANCH IF YES
4313
4314 022374 004567 001056 JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4315 022400 023255 .WORD $CNTLG ;ADDRESS OF MESSAGE TO BE TYPED
4316 022402
4317 022402 004567 001050 $GTSWR: JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4318 022406 023262 .WORD $MSWR ;ADDRESS OF MESSAGE TO BE TYPED
4319 022410 016746 155562 MOV SWREG,-(SP) ;: SAVE SWREG FOR TYPEOUT
4320 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4321 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4322 022414 013746 177776 MOV @PSW,-(SP) ;: PUT THE PROCESSOR STATUS ON THE STACK
4323 022420 004767 002230 JSR PC,$TYPOC ;: GO TO THE SUBROUTINE
4324 022424 004567 001026 JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4325 022430 023273 .WORD $MNEW ;ADDRESS OF MESSAGE TO BE TYPED
4326 022432 005046 19$: CLR -(SP) ;: CLEAR COUNTER
4327 022434 005046 7$: CLR -(SP) ;: THE NEW SWR
4328 022436 105777 156502 TSTB @STKS ;: CHAR THERE?
4329 022442 100375 BPL 7$ ;: IF NOT TRY AGAIN
4330
4331 022444 117746 156476 MOVB @STKB,-(SP) ;: PICK UP CHAR
4332 022450 042716 177600 BIC #C177,(SP) ;: MAKE IT 7-BIT ASCII
4333
4334
4335
4336 022454 021627 000025 9$: CMP (SP),#25 ;: IS IT A CONTROL-U?
4337 022460 001006 BNE 10$ ;: BRANCH IF NOT
4338 022462 004567 000770 JSR R5 $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.

```

```

4339 022466 023250          WORD  SCNTLU      ; ADDRESS OF MESSAGE TO BE TYPED
4340 022470 062706 000006 20$:  ADD      86,SP      ; IGNORE PREVIOUS INPUT
4341 022474 000756          BR       19$      ; LET'S TRY IT AGAIN
4342
4343
4344 022476 021627 000015 10$:  CMP      (SP),#15  ; IS IT A <CR>?
4345 022502 001023          BNE          ; BRANCH IF NO
4346 022504 005766 000004          TST      4(SP)    ; YES, IS IT THE FIRST CHAR?
4347 022510 001403          BEQ          ; BRANCH IF YES
4348 022512 016677 000002 156420 11$:  MOV      2(SP),@SMR ; SAVE NEW SMR
4349 022520 062706 000006 14$:  ADD      86,SP      ; CLEAR UP STACK
4350
4351 022524 004567 000726          JSR      R5, SPRINT ; GO PRINT OUT THE FOLLOWING MESSAGE.
4352 022530 001201          .WORD  SCALF      ; ADDRESS OF MESSAGE TO BE TYPED
4353 022532 126727 156377 000001  CMPB    $INTAG,#1 ; RE-ENABLE TTY KBD INTERRUPTS?
4354 022540 001003          BNE          ; BRANCH IF NOT
4355 022542 012777 000100 156374 15$:  MOV      #100,@STKS ; RE-ENABLE TTY KBD INTERRUPTS
4356 022550 000002          RTI          ; RETURN
4357 022552 004767 001142 16$:  JSR      PC,STYPEC ; ECHO CHAR
4358 022556 021627 000060          CMP      (SP),#60 ; CHAR < 0?
4359 022562 002420          BLT          ; BRANCH IF YES
4360 022564 021627 000067          CMP      (SP),#67 ; CHAR > 7?
4361 022570 003015          BGT          ; BRANCH IF YES
4362 022572 042726 000060          BIC      #60,(SP)+ ; STRIP-OFF ASCII
4363 022576 005766 000002          TST      2(SP)    ; IS THIS THE FIRST CHAR
4364 022602 001403          BEQ          ; BRANCH IF YES
4365 022604 006316          ASL      (SP)    ; NO, SHIFT PRESENT
4366 022606 006316          ASL      (SP)    ; CHAR OVER TO MAKE
4367 022610 006316          ASL      (SP)    ; ROOM FOR NEW ONE.
4368 022612 005266 000002 17$:  INC      2(SP)    ; KEEP COUNT OF CHAR
4369 022616 056616 177776          BIS      -2(SP),(SP) ; SET IN NEW CHAR
4370 022622 000705          BR       7$      ; GET THE NEXT ONE
4371 022624
4372 022624 004567 000626          JSR      R5, SPRINT ; GO PRINT OUT THE FOLLOWING MESSAGE.
4373 022630 001200          .WORD  $QUES     ; ADDRESS OF MESSAGE TO BE TYPED
4374 022632 000716          BR       20$     ; SIMULATE CONTROL-U
4375 .DSABL  LSB
4376
4377
4378 ; *****
4379 ; THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
4380 ; CALL:
4381 ; RDCHR          ; INPUT A SINGLE CHARACTER FROM THE TTY
4382 ; RETURN HERE   ; CHARACTER IS ON THE STACK
4383 ;              ; WITH PARITY BIT STRIPPED OFF
4384 ;
4385 ;
4386 022634 011646          $RDCHR: MOV      (SP),-(SP) ; PUSH DOWN THE PC
4387 022636 016666 000004 000002  MOV      4(SP),2(SP) ; SAVE THE PS
4388 022644 105777 156274 1$:  TSTB   @STKS      ; WAIT FOR
4389 022650 100375          BPL      1$      ; A CHARACTER
4390 022652 117766 156270 000004  MOVB   @STKB,4(SP) ; READ THE TTY
4391 022660 042766 177600 000004  BIC      #177,4(SP) ; GET RID OF JUNK IF ANY
4392 022666 026627 000004 000023  CMP      4(SP),#23 ; IS IT A CONTROL-5?
4393 022674 001013          BNE          ; BRANCH IF NO
4394 022676 105777 156242 2$:  TSTB   @STKS      ; WAIT FOR A CHARACTER

```

```

4395 022702 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
4396 022704 117746 156236  MOVB     2$TKB, -(SP) ;; GET CHARACTER
4397 022710 042716 177600  BIC     #177, (SP)   ;; MAKE IT 7-BIT ASCII
4398 022714 022627 000021  CMP     (SP)+, #21   ;; IS IT A CONTROL-Q?
4399 022720 001366          BNE     2$          ;; IF NOT DISCARD IT
4400 022722 000750          BR      1$          ;; YES, RESUME
4401 022724 026627 000004 000140 3$:    CMP     4(SF), #140  ;; IS IT UPPER CASE?
4402 022732 002407          BLT     4$          ;; BRANCH IF YES
4403 022734 026627 000004 000175  CMP     4(SF), #175  ;; IS IT A SPECIAL CHAR?
4404 022742 003003          BGT     4$          ;; BRANCH IF YES
4405 022744 042766 000040 000004  BIC     #40, 4(SP)   ;; MAKE IT UPPER CASE
4406 022752 000002 4$:    RTI          ;; GO BACK TO USER
4407                                     ;; *****
4408                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4409                                     ;; *CALL:
4410                                     ;; *   RDLIN          ;; INPUT A STRING FROM THE TTY
4411                                     ;; *   RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4412                                     ;; *                                     ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
4413
4414 022754 010346          $RDLIN: MOV     R3, -(SP)   ;; SAVE R3
4415 022756 005046          CLR     -(SP)       ;; CLEAR THE RUBOUT KEY
4416 022760 012703 023240  1$:    MOV     #STTYIN, R3 ;; GET ADDRESS
4417 022764 022703 023250  2$:    CMP     #STTYIN+8., R3 ;; BUFFER FULL?
4418 022770 101467          BLOS   4$          ;; BR IF YES
4419                                     ;; * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4420                                     ;; * WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4421 022772 013746 177776  MOV     #PSW, -(SP) ;; PUT THE PROCESSOR STATUS ON THE STACK
4422 022776 004767 177632  JSR     PC, $RDCHR  ;; GO TO THE SUBROUTINE
4423 023002 112613          MOVB   (SP)+, (R3)  ;; GET CHARACTER
4424 023004 122713 000177  10$:   CMPB   #177, (R3)   ;; IS IT A RUBOUT
4425 023010 001024          BNE    5$          ;; BR IF NO
4426 023012 005716          TST   (SP)         ;; IS THIS THE FIRST RUBOUT?
4427 023014 001010          BNE    6$          ;; BR IF NO
4428 023016 112767 000134 000212  MOVB   #' \, 9$    ;; TYPE A BACK SLASH
4429 023024 004567 000426          JSR   R5, $PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4430 023030 023236          .WORD 9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4431 023032 012716 177777          MOV   #-1, (SP)   ;; SET THE RUBOUT KEY
4432 023036 005303          DEC   R3          ;; BACKUP BY ONE
4433 023040 020327 023240          CMP   R3, #STTYIN ;; STACK EMPTY?
4434 023044 103441          BLO   4$          ;; BR IF YES
4435 023046 111367 000164          MOVB (R3), 9$     ;; SETUP TO TYPEOUT THE DELETED CHAR.
4436 023052 004567 000400          JSR   R5, $PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4437 023056 023236          .WORD 9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4438 023060 000741          BR    2$          ;; GO READ ANOTHER CHAR.
4439 023062 005716          TST  (SP)         ;; RUBOUT KEY SET?
4440 023064 001407          BEQ   7$          ;; BR IF NO
4441 023066 112767 000134 000142  MOVB   #' \, 9$    ;; TYPE A BACK SLASH
4442 023074 004567 000356          JSR   R5, $PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4443 023100 023236          .WORD 9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4444 023102 005016          CLR  (SP)         ;; CLEAR THE RUBOUT KEY
4445 023104 122713 000025          CMPB #25, (R3)    ;; IS CHARACTER A CTRL U?
4446 023110 001004          BNE   8$          ;; BR IF NO
4447 023112 004567 000340          JSR   R5, $PRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4448 023116 023250          .WORD $CNTLU     ;; ADDRESS OF MESSAGE TO BE TYPED
4449 023120 000717          BR    1$          ;; GO START OVER
4450 023122 122713 000022  8$:    CMPB   #22, (R3)  ;; IS CHARACTER A "↑"?

```

```

4451 023126 001014      BNE      3$          ;; BRANCH IF NO
4452 023130 105013      CLR      (R3)       ;; CLEAR THE CHARACTER
4453 023132 004567 000320 JSR      R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4454 023136 001201      .WORD   $CRLF       ;; ADDRESS OF MESSAGE TO BE TYPED
4455 023140 004567 000312 JSR      R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4456 023144 023240      .WORD   $TTYIN      ;; ADDRESS OF MESSAGE TO BE TYPED
4457 023146 000706      BR       2$          ;; GO PICKUP ANOTHER CHARACTER
4458 023150
4459 023150 004567 000302 JSR      R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4460 023154 001200      .WORD   $QUES       ;; ADDRESS OF MESSAGE TO BE TYPED
4461 023156 000700      BR       1$          ;; CLEAR THE BUFFER AND LOOP
4462 023160 111367 000052 JSR      (R3), 9$     ;; ECHO THE CHARACTER
4463 023164 004567 000266 JSR      R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4464 023170 023236      .WORD   9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4465 023172 122723 000015 CMP      #15, (R3)+   ;; CHECK FOR RETURN
4466 023176 001272      BNE      2$          ;; LOOP IF NOT RETURN
4467 023200 105063 177777 CLR      -1(R3)      ;; CLEAR RETURN (THE 15)
4468 023204 004567 000246 JSR      R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4469 023210 001202      .WORD   $LF         ;; ADDRESS OF MESSAGE TO BE TYPED
4470 023212 005726      TST     (SP)+        ;; CLEAN RUBOUT KEY FROM THE STACK
4471 023214 012603      MOV     (SP)+, R3    ;; RESTORE R3
4472 023216 011646      MOV     (SP), -(SP)  ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4473 023220 016666 000004 000002 MOV     4(SP), 2(SP)  ;; FIRST ASCII CHARACTER ON IT
4474 023226 012766 023240 000004 MOV     #TTYIN, 4(SP)
4475 023234 000002      RTI              ;; RETURN
4476 023236 000      9$: .BYTE 0          ;; STORAGE FOR ASCII CHAR. TO TYPE
4477 023237 000      .BYTE 0          ;; TERMINATOR
4478 023240 000010      $TTYIN: .BLKB 8     ;; RESERVE 8 BYTES FOR TTY INPUT
4479 023250 052536 005015 000      $CNTLU: .ASCIZ /↑U/<15><12> ;; CONTROL "U"
4480 023255 136 006507 000012 $CNTLG: .ASCIZ /↑G/<15><12> ;; CONTROL "G"
4481 023262 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
4482 023270 020075 000
4483 023273 040 047040 053505 $MNEW: .ASCIZ / NEW = /
4484 023300 036440 000040
4485 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
4486
4487 *****
4488 *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4489 *CHANGE IT TO BINARY.
4490 *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
4491 *OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
4492 *FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
4493 *THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
4494 *CALL:
4495 * RDOCT          ;; READ AN OCTAL NUMBER
4496 * RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
4497 *              ;; HIGH ORDER BITS ARE IN $HI OCT
4498
4499 023304 011646 000004 000002 $RDOCT: MOV     (SP), -(SP)  ;; PROVIDE SPACE FOR THE
4500 023306 016666      MOV     4(SP), 2(SP)  ;; INPUT NUMBER
4501 023314 010046      MOV     R0, -(SP)    ;; PUSH R0 ON STACK
4502 023316 010146      MOV     R1, -(SP)    ;; PUSH R1 ON STACK
4503 023320 010246      MOV     R2, -(SP)    ;; PUSH R2 ON STACK
4504 023322
4505 1$: * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
4506 * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.

```

```

4507 023322 013746 177776      MOV      2#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4508 023326 004767 177422      JSR      PC, $RDLIN ;GO TO THE SUBROUTINE
4509 023332 012600                MOV      (SP)+,R0 ;GET ADDRESS OF 1ST CHARACTER
4510 023334 010067 000102      MOV      R0,$5 ;AND SAVE IT
4511 023340 005001                CLR      R1 ;CLEAR DATA WORD
4512 023342 005002                CLR      R2
4513 023344 112046 2$:      MOVB     (R0)+,-(SP) ;PICKUP THIS CHARACTER
4514 023346 001420                BEQ      3$ ;IF ZERO GET OUT
4515 023350 122716 000060      CMPB     #'0,(SP) ;MAKE SURE THIS CHARACTER
4516 023354 003026                BGT      4$ ;IS AN OCTAL DIGIT
4517 023356 122716 000067      CMPB     #'7,(SP)
4518 023362 002423                BLT      4$
4519 023364 006301                ASL      R1 ;;*2
4520 023366 006102                ROL      R2
4521 023370 006301                ASL      R1 ;;*4
4522 023372 006102                ROL      R2
4523 023374 006301                ASL      R1 ;;*8
4524 023376 006102                ROL      R2
4525 023400 042716 177770      BIC      #1C7,(SP) ;STRIP THE ASCII JUNK
4526 023404 062601                ADD      (SP)+,R1 ;ADD IN THIS DIGIT
4527 023406 000756                BR       2$ ;LOOP
4528 023410 005726 3$:      TST      (SP)+ ;CLEAN TERMINATOR FROM STACK
4529 023412 010166 000012      MOV      R1,12(SP) ;SAVE THE RESULT
4530 023416 010267 000032      MOV      R2,$SHIOCT
4531 023422 012602                MOV      (SP)+,R2 ;POP STACK INTO R2
4532 023424 012601                MOV      (SP)+,R1 ;POP STACK INTO R1
4533 023426 012600                MOV      (SP)+,R0 ;POP STACK INTO R0
4534 023430 000002                RTI ;RETURN
4535 023432 005726 4$:      TST      (SP)+ ;CLEAN PARTIAL FROM STACK
4536 023434 105010                CLRB     (R0) ;SET A TERMINATOR
4537 023436 004567 000014      JSR      R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4538 023442 000000 5$:      .WORD   0
4539 023444 004567 000006      JSR      R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4540 023450 001200                .WORD   $QUES ;ADDRESS OF MESSAGE TO BE TYPED
4541 023452 000723                BR       1$ ;TRY AGAIN
4542 023454 000000  $SHIOCT: .WORD 0 ;HIGH ORDER BITS GO HERE

```

```

4543
4544
4545
4546
4547
4548
4549 023456 012567 000016
4550 023462 066767 155112 000010

```

```

*****
;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
;* CALL:      JSR      R5, $SPRINT
;*            <MESSAGE VIRTUAL ADDRESS>
*****
$SPRINT: MOV      (R5)+, 1$ ;GET THE MESSAGE VIRTUAL ADDRESS.
         ADD      RELOC, 1$ ;MAKE IT PHYSICAL.
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
         MOV      2#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
         JSR      PC, $TYPE ;GO TO THE SUBROUTINE
1$:      .WORD   0 ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
         RTS      R5 ;RETURN.

```

```

4551
4552
4553 023470 013746 177776
4554 023474 004767 000004
4555 023500 000000
4556 023502 000205
4557
4558
4559
4560
4561
4562

```

```

.SBTTL TYPE ROUTINE
*****
;*ROUTINE TO TYPE ASCII MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

```

```

4563      ;*NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4564      ;*NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4565      ;*NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4566      ;*
4567      ;*CALL:
4568      ;*1) USING A TRAP INSTRUCTION
4569      ;*          TYPE          ,MESADR          ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4570      ;*OR
4571      ;*          TYPE
4572      ;*          MESADR
4573      ;*
4574
4575 023504 105767 155447 $TYPE: TSTB $TPFLG          ;; IS THERE A TERMINAL?
4576 023510 100002          BPL 1$          ;; BR IF YES
4577 023512 000000          HALT          ;; HALT HERE IF NO TERMINAL
4578 023514 000430          BR 3$          ;; LEAVE
4579 023516 010046          MOV RO,-(SP)          ;; SAVE RO
4580 023520 017600 000002          MOV @2(SP),RO          ;; GET ADDRESS OF ASCIZ STRING
4581 023524 122767 000001 155472          CMPB #APTENV,$ENV          ;; RUNNING IN APT MODE
4582 023532 001011          BNE 62$          ;; NO GO CHECK FOR APT CONSOLE
4583 023534 132767 000100 155463          BITB #APTPOOL,$ENVM          ;; SPOOL MESSAGE TO APT
4584 023542 001405          BEQ 62$          ;; NO GO CHECK FOR CONSOLE
4585 023544 010067 000004          MOV RO,61$          ;; SETUP MESSAGE ADDRESS FOR APT
4586 023550 004767 000222          JSR PC,$ATY3          ;; SPOOL MESSAGE TO APT
4587 023554 000000          .WORD 0          ;; MESSAGE ADDRESS
4588 023556 132767 000040 155441          BITB #APTCSUP,$ENVM          ;; APT CONSOLE SUPPRESSED
4589 023564 001003          BNE 60$          ;; YES, SKIP TYPE OUT
4590 023566 112046          MOVB (RO)+,-(SP)          ;; PUSH CHARACTER TO BE TYPED ONTO STACK
4591 023570 001005          BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
4592 023572 005726          TST (SP)+          ;; IF TERMINATOR POP IT OFF THE STACK
4593 023574 012600          MOV (SP)+,RO          ;; RESTORE RO
4594 023576 062716 000002          ADD #2,(SP)          ;; ADJUST RETURN PC
4595 023602 000002          RTI          ;; RETURN
4596 023604 122716 000011          CMPB #HT,(SP)          ;; BRANCH IF <HT>
4597 023610 001431          BEQ 8$          ;;
4598 023612 122716 000200          CMPB #CRLF,(SP)          ;; BRANCH IF NOT <CRLF>
4599 023616 001007          BNE 5$          ;;
4600 023620 005726          TST (SP)+          ;; POP <CR><LF> EQUIV
4601 023622 004567 177630          JSR R5, SPRINT          ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4602 023626 001201          $CRLF
4603 023630 105067 000130          CLRB $CHARCNT          ;; CLEAR CHARACTER COUNT
4604 023634 000754          BR 2$          ;; GET NEXT CHARACTER
4605 023636 004767 000056          JSR PC,$TYPEC          ;; GO TYPE THIS CHARACTER
4606 023642 126726 155310          CMPB $FILLC,(SP)+          ;; IS IT TIME FOR FILLER CHARS.?
4607 023646 001347          BNE 2$          ;; IF NO GO GET NEXT CHAR.
4608 023650 016746 155300          MOV $NULL,-(SP)          ;; GET # OF FILLER CHARS. NEEDED
4609          AND THE NULL CHAR.
4610 023654 105366 000001          DECB 1(SP)          ;; DOES A NULL NEED TO BE TYPED?
4611 023660 002770          BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
4612 023662 004767 000032          JSR PC,$TYPEC          ;; GO TYPE A NULL
4613 023666 105367 000072          DECB $CHARCNT          ;; DO NOT COUNT AS A COUNT
4614 023672 000770          BR 7$          ;; LOOP
4615
4616          ;HORIZONTAL TAB PROCESSOR
4617
4618 023674 112716 000040          8$: MOVB #' ,(SP)          ;; REPLACE TAB WITH SPACE
    
```

```

4619 023700 004767 000014 9$: JSR PC,$TYPEC ;;TYPE A SPACE
4620 023704 132767 000007 000052 BITB #7,$SCHARCNT ;;BRANCH IF NOT AT
4621 023712 001372 BNE 9$ ;;TAB STOP
4622 023714 005726 TST (SP)+ ;;POP SPACE OFF STACK
4623 023716 000723 BR 2$ ;;GET NEXT CHARACTER
4624 023720 105777 155224 $TYPEC: TSTB @STPS ;;WAIT UNTIL PRINTER IS READY
4625 023724 100375 BPL $TYPEC
4626 023726 116677 000002 155216 MOVB 2(SP),@STPB ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4627 023734 122766 000015 000002 CMPB #CR,2(SP) ;;IS CHARACTER A CARRIAGE RETURN?
4628 023742 001003 BNE 1$ ;;BRANCH IF NO
4629 023744 105067 000014 CLRB $SCHARCNT ;;YES--CLEAR CHARACTER COUNT
4630 023750 000406 BR $TYPEX ;;EXIT
4631 023752 122766 000012 000002 1$: CMPB #LF,2(SP) ;;IS CHARACTER A LINE FEED?
4632 023760 001402 BEQ $TYPEX ;;BRANCH IF YES
4633 023762 105227 INCB (PC)+ ;;COUNT THE CHARACTER
4634 023764 000000 $SCHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
4635 023766 000207 $TYPEX: RTS PC
4636
4637 .SBTTL APT COMMUNICATIONS ROUTINE
4638
4639 *****
4640 023770 112767 000001 000376 $ATY1: MOVB #1,$FFLG ;;TO REPORT FATAL ERROR
4641 023776 112767 000001 000366 $ATY3: MOVB #1,$MFLG ;;TO TYPE A MESSAGE
4642 024004 000403 BR $ATYC
4643 024006 112767 000001 000360 $ATY4: MOVB #1,$FFLG ;;TO ONLY REPORT FATAL ERROR
4644 024014 $ATYC:
4645 024014 010046 MOV RO,-(SP) ;;PUSH RO ON STACK
4646 024016 010146 MOV R1,-(SP) ;;PUSH R1 ON STACK
4647 024020 105767 000346 TSTB $MFLG ;;SHOULD TYPE A MESSAGE?
4648 024024 001450 BEQ 5$ ;;IF NOT: BR
4649 024026 122767 000001 155170 CMPB #APTENV,$ENV ;;OPERATING UNDER APT?
4650 024034 001031 BNE 3$ ;;IF NOT: BR
4651 024036 132767 000100 155161 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4652 024044 001425 BEQ 3$ ;;IF NOT: BR
4653 024046 017600 000004 MOV @4(SP),RO ;;GET MESSAGE ADDR.
4654 024052 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.
4655 024060 005767 155120 1$: TST $MSGTYPE ;;SEE IF DONE W/ LAST XMISSION?
4656 024064 001375 BNE 1$ ;;IF NOT: WAIT
4657 024066 010067 155126 MOV RO,$MSGAD ;;PUT ADDR IN MAILBOX
4658 024072 105720 2$: TSTB (RO)+ ;;FIND END OF MESSAGE
4659 024074 001376 BNE 2$
4660 024076 166700 155116 SUB $MSGAD,RO ;;SUB START OF MESSAGE
4661 024102 006200 ASR RO ;;GET MESSAGE LNTH IN WORDS
4662 024104 010067 155112 MOV RO,$MSGLGT ;;PUT LENGTH IN MAILBOX
4663 024110 012767 000004 155066 MOV #4,$MSGTYPE ;;TELL APT TO TAKE MSG.
4664 024116 000413 BR 5$
4665 024120 017667 000004 000016 3$: MOV @4(SP),4$ ;;PUT MSG ADDR IN JSR LINKAGE
4666 024126 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDRESS
4667 024134 016746 153636 MOV 177776,-(SP) ;;PUSH 177776 ON STACK
4668 024140 004767 177340 JSR PC,$TYPE ;;CALL TYPE MACRO
4669 024144 000000 4$: .WORD 0
4670 024146 5$:
4671 024146 105767 000221 TSTB $LFLG ;;SHOULD LOG AN ERROR?
4672 024152 001422 BEQ 10$ ;;IF NOT: BR
4673 024154 017600 000004 MOV @4(SP),RO ;;GET ERROR #
4674 024160 062766 000002 000004 ADD #2,4(SP) ;;BUMP RETURN ADDR.

```



```

4675 024166 012701 001344
4676 024172 005711
4677 024174 100404
4678 024176 020021
4679 024200 001406
4680 024202 005721
4681 024204 000772
4682 024206 026701 155300
4683 024212 001402
4684 024214 010021
4685 024216 005211
4686 024220 105767 000150
4687 024224 001416
4688 024226 005767 154772
4689 024232 001413
4690 024234 005767 154744
4691 024240 001375
4692 024242 017667 000004 154736
4693 024250 062766 000002 000004
4694 024256 005267 154722
4695 024262 105067 000106
4696 024266 105067 000101
4697 024272 105067 000074
4698 024276 012601
4699 024300 012600
4700 024302 000207
4701 024304
4702 024304 010046
4703 024306 016700 155200
4704 024312 162700 001344
4705 024316 005767 154662
4706 024322 001375
4707 024324 010067 154672
4708 024330 012767 001344 154662
4709 024336 012767 000002 154640
4710 024344 012600
4711 024346 000207
4712 024350
4713 024350 010046
4714 024352 012701 001344
4715 024356 005721
4716 024360 100402
4717 024362 005021
4718 024364 000774
4719 024366
4720 024366 012600
4721 024370 000207
4722 024372 000
4723 024373 000
4724 024374 000
4725 024376
4726 000200
4727 000001
4728 000100
4729 000040
4730

6$: MOV #SASTAT,R1 ;; POINT TO TABLE START
TST (R1) ;; END OF TABLE?
BMI B$ ;; IF SO: BR
CMP RO,(R1)+ ;; PROPER ENTRY?
BEQ 9$ ;; IF SO: BR
TST (R1)+ ;; MOVE PAST COUNTER WORD
BR B$ ;; KEEP LOOKING
8$: CMP $APTR,R1 ;; TABLE FULL?
BEQ 10$ ;; IF SO: BR -- NO MORE ROOM
MOV RO,(R1)+ ;; SET UP NEW ENTRY
INC (R1) ;; BUMP ERROR COUNT
10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
BEQ 12$ ;; IF NOT: BR
TST $ENV ;; RUNNING UNDER APT?
BEQ 12$ ;; IF NOT: BR
11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
BNE 11$ ;; IF NOT: WAIT
MOV @4(SP),$FATAL ;; GET ERROR #
ADD #2,4(SP) ;; BUMP RETURN ADDR.
INC $MSGTYPE ;; TELL APT TO TAKE ERROR
12$: CLRB $FFLG ;; CLEAR FATAL FLAG
CLRB $LFLG ;; CLEAR LOG FLAG
CLRB $MFLG ;; CLEAR MESSAGE FLAG
MOV (SP)+,R1 ;; POP STACK INTO R1
MOV (SP)+,RO ;; POP STACK INTO RO
RTS PC ;; RETURN

$ATY6: MOV RO,-(SP) ;; PUSH RO ON STACK
MOV $APTR,RO
SUB #SASTAT,RO
1$: TST $MSGTY ;; GET SIZE OF STAT TABLE
BNE 1$ ;; SEE IF DONE LAST COMMUNICATION
MOV RO,$MSGLG ;; IF NOT: WAIT
MOV #SASTAT,$MSGAD ;; SET MESSAGE LENGTH
MOV #2,$MSGTY ;; SET MESSAGE ADDR.
MOV (SP)+,RO ;; TELL APT TO TAKE STATS.
RTS PC ;; POP STACK INTO RO
;; RETURN

$ATY7: MOV RO,-(SP) ;; PUSH RO ON STACK
MOV #SASTAT,R1 ;; GET START OF TABLE
1$: TST (R1)+ ;; END OF TABLE?
BMI 2$ ;; IF SO: BR
CLR (R1)+ ;; CLEAR ERROR COUNT
BR 1$ ;; KEEP CLEARING

2$: MOV (SP)+,RO ;; POP STACK INTO RO
RTS PC ;; RETURN
$MFLG: .BYTE 0 ;; MESSG. FLAG
$LFLG: .BYTE 0 ;; LOG FLAG
$FFLG: .BYTE 0 ;; FATAL FLAG
.EVEN
APTSIZE=200
APTENV=001
APTSPool=100
APTCSUP=040
;*****

```

```

4731
4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743 024376
4744 024376 010046
4745 024400 010146
4746 024402 010246
4747 024404 010346
4748 024406 010546
4749 024410 012746 020200
4750 024414 016605 000020
4751 024420 100004
4752 024422 005405
4753 024424 112766 000055 000001
4754 024432 016700 154142
4755 024436 012703 024620
4756 024442 060003
4757 024444 112723 000040
4758 024450 005002
4759 024452 016001 024610
4760 024456 160105
4761 024460 002402
4762 024462 005202
4763 024464 000774
4764 024466 060105
4765 024470 005702
4766 024472 001002
4767 024474 105716
4768 024476 100407
4769 024500 106316
4770 024502 103003
4771 024504 116663 000001 177777
4772 024512 052702 000060
4773 024516 052702 000040
4774 024522 110223
4775 024524 005720
4776 024526 020067 155110
4777 024532 103746
4778 024534 101002
4779 024536 010502
4780 024540 000764
4781 024542 105726
4782 024544 100003
4783 024546 116663 177777 177776
4784 024554 105013
4785 024556 012605
4786 024560 012603

```

```

.SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*CALL:
;*
;*      MOV      NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS                    ;;GO TO THE ROUTINE

$TYPDS:
MOV      R0,-(SP)      ;;PUSH R0 ON STACK
MOV      R1,-(SP)      ;;PUSH R1 ON STACK
MOV      R2,-(SP)      ;;PUSH R2 ON STACK
MOV      R3,-(SP)      ;;PUSH R3 ON STACK
MOV      R5,-(SP)      ;;PUSH R5 ON STACK
MOV      #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
MOV      20(SP),R5      ;;GET THE INPUT NUMBER
BPL      1$             ;;BR IF INPUT IS POS.
NEG      R5             ;;MAKE THE BINARY NUMBER POS.
MOVB    #'-(SP)        ;;MAKE THE ASCII NUMBER NEG.
MOV      RELOC,R0      ;;GET RELOCATION FACTOR.
MOV      #SDBLK,R3     ;;SETUP THE OUTPUT POINTER
ADD      R0,R3         ;;ADD IN RELOCATION FACTOR.
MOVB    #' ,(R3)+      ;;SET THE FIRST CHARACTER TO A BLANK
CLR      R2            ;;CLEAR THE BCD NUMBER
MOV      $DTBL(R0),R1  ;;GET THE CONSTANT
SUB      R1,R5         ;;FORM THIS BCD DIGIT
BLT     4$             ;;BR IF DONE
INC      R2            ;;INCREASE THE BCD DIGIT BY 1
BR      3$

4$:
ADD      R1,R5         ;;ADD BACK THE CONSTANT
TST     R2             ;;CHECK IF BCD DIGIT=0
BNE     5$            ;;FALL THROUGH IF 0
TSTB   (SP)           ;;STILL DOING LEADING 0'S?
BMI     7$            ;;BR IF YES
ASLB   (SP)           ;;MSD?
BCC     6$            ;;BR IF NO
MOVB   1(SP),-1(R3)   ;;YES--SET THE SIGN
BIS    #'0,R2         ;;MAKE THE BCD DIGIT ASCII
BIS    #' ,R2        ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
MOVB   R2,(R3)+      ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
TST    (R0)+         ;;JUST INCREMENTING
CMP    R0, .EIGHT    ;;CHECK THE TABLE INDEX
BLO    2$            ;;GO DO THE NEXT DIGIT
BHI    8$            ;;GO TO EXIT
MOV    R5,R2         ;;GET THE LSD
BR     6$            ;;GO CHANGE TO ASCII
TSTB  (SP)+         ;;WAS THE LSD THE FIRST NON-ZERO?
BPL    9$            ;;BR IF NO
MOVB  -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
CLRB  (R3)           ;;SET THE TERMINATOR
MOV   (SP)+,R5      ;;POP STACK INTO R5
MOV   (SP)+,R3     ;;POP STACK INTO R3

```

4787	024562	012602		
4788	024564	012601		
4789	024566	012600		
4790	024570	004567	176662	
4791	024574	024620		
4792	024576	016666	000002	000004
4793	024604	012616		
4794	024606	000002		
4795	024610	023420		
4796	024612	001750		
4797	024614	000144		
4798	024616	000012		
4799	024620	000004		
4800				
4801				
4802				
4803				
4804				
4805				
4806				
4807				
4808				
4809				
4810				
4811				
4812				
4813				
4814				
4815				
4816				
4817				
4818				
4819				
4820				
4821				
4822				
4823				
4824				
4825	024630	017646	000000	
4826	024634	116667	000001	000213
4827	024642	112667	000211	
4828	024646	062716	000002	
4829	024652	000406		
4830	024654	112767	000001	000173
4831	024662	112767	000006	000167
4832	024670	112767	000005	000156
4833	024676	010346		
4834	024700	010446		
4835	024702	010546		
4836	024704	116704	000147	
4837	024710	005404		
4838	024712	062704	000006	
4839	024716	110467	000134	
4840	024722	116704	000127	
4841	024726	016605	000012	
4842	024732	005003		

```

MOV (SP)+,R2 ;;POP STACK INTO R2
MOV (SP)+,R1 ;;POP STACK INTO R1
MOV (SP)+,R0 ;;POP STACK INTO R0
JSR R5, SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD $DBLK ;;ADDRESS OF MESSAGE TO BE TYPED
MOV 2(SP),4(SP) ;;ADJUST THE STACK
MOV (SP)+,(SP)
RTI ;;RETURN TO USER

$DTBL: 10000.
      1000.
      100.
      10.

$DBLK: .BLKW 4
.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;      MOV NUM,-(SP) ;;NUMBER TO BE TYPED
;      TYPOS ;;CALL FOR TYPEOUT
;      .BYTE N ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;      .BYTE M ;;M=1 OR 0
;      ;;1=TYPE LEADING ZEROS
;      ;;0=SUPPRESS LEADING ZEROS
;$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;$TYPOS OR $TYPOC
;CALL:
;      MOV NUM,-(SP) ;;NUMBER TO BE TYPED
;      TYPON ;;CALL FOR TYPEOUT
;$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;CALL:
;      MOV NUM,-(SP) ;;NUMBER TO BE TYPED
;      TYPOC ;;CALL FOR TYPEOUT

$TYPOS: MOV 2(SP),-(SP) ;;PICKUP THE MODE
        MOV 1(SP),$OFILL ;;LOAD ZERO FILL SWITCH
        MOV (SP)+,$OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD #2,(SP) ;;ADJUST RETURN ADDRESS
        BR $TYPON

$TYPOC: MOV #1,$OFILL ;;SET THE ZERO FILL SWITCH
        MOV #6,$OMODE+1 ;;SET FOR SIX(6) DIGITS
$TYPON: MOV #5,$OCNT ;;SET THE ITERATION COUNT
        MOV R3,-(SP) ;;SAVE R3
        MOV R4,-(SP) ;;SAVE R4
        MOV R5,-(SP) ;;SAVE R5
        MOV $OMODE+1,R4 ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG R4
        ADD #6,R4 ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV R4,$OMODE ;;SAVE IT FOR USE
        MOV $OFILL,R4 ;;GET THE ZERO FILL SWITCH
        MOV 12(SP),R5 ;;PICKUP THE INPUT NUMBER
        CLR R3 ;;CLEAR THE OUTPUT WORD

```

```

4843 024734 006105      1$:  ROL  R5      ;; ROTATE MSB INTO "C"
4844 024736 000404      BR  3$      ;; GO DO MSB
4845 024740 006105      2$:  ROL  R5      ;; FORM THIS DIGIT
4846 024742 006105      ROL  R5
4847 024744 006105      ROL  R5
4848 024746 010503      MOV  R5,R3
4849 024750 006103      3$:  ROL  R3      ;; GET LSB OF THIS DIGIT
4850 024752 105367 000100  DECB $OMODE  ;; TYPE THIS DIGIT?
4851 024756 100017      BPL  7$      ;; BR IF NO
4852 024760 042703 177770  BIC  #177770,R3  ;; GET RID OF JUNK
4853 024764 001002      BNE  4$      ;; TEST FOR 0
4854 024766 005704      TST  R4      ;; SUPPRESS THIS 0?
4855 024770 001403      BEQ  5$      ;; BR IF YES
4856 024772 005204      4$:  INC  R4      ;; DON'T SUPPRESS ANYMORE 0'S
4857 024774 052703 000060  BIS  #'0,R3    ;; MAKE THIS DIGIT ASCII
4858 025000 052703 000040  5$:  BIS  #' ,R3  ;; MAKE ASCII IF NOT ALREADY
4859 025004 110367 000042  MOVB R3,8$    ;; SAVE FOR TYPING
4860 025010 004567 176442  JSR  R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4861 025014 025052      .WORD 8$      ;; ADDRESS OF MESSAGE TO BE TYPED
4862 025016 105367 000032  7$:  DECB $OCNT  ;; COUNT BY 1
4863 025022 003346      BGT  2$      ;; BR IF MORE TO DO
4864 025024 002402      BLT  6$      ;; BR IF DONE
4865 025026 005204      INC  R4      ;; INSURE LAST DIGIT ISN'T A BLANK
4866 025030 000743      BR  2$      ;; GO DO THE LAST DIGIT
4867 025032 012605      6$:  MOV  (SP)+,R5  ;; RESTORE R5
4868 025034 012604      MOV  (SP)+,R4  ;; RESTORE R4
4869 025036 012603      MOV  (SP)+,R3  ;; RESTORE R3
4870 025040 016666 000002 000004  MOV  2(SP),4(SP)  ;; SET THE STACK FOR RETURNING
4871 025046 012616      MOV  (SP)+,(SP)
4872 025050 000002      RTI
4873 025052 000      8$:  .BYTE 0      ;; RETURN
4874 025053 000      .BYTE 0      ;; STORAGE FOR ASCII DIGIT
4875 025054 000      $OCNT: .BYTE 0  ;; TERMINATOR FOR TYPE ROUTINE
4876 025055 000      $OFILL: .BYTE 0  ;; OCTAL DIGIT COUNTER
4877 025056 000000      $OMODE: .WORD 0  ;; ZERO FILL SWITCH
4878      .ERROR TRAP SERVICE ROUTINE  ;; NUMBER OF DIGITS TO TYPE
4879 025060 005727      ERRTRP: TST (PC)+  ;; CHECK IF PREV TRAP TO 4 REPORTED
4880 025062 000000      1$:  .WORD 0      ;; CONTAINS ERROR REPORTED FLAG
4881 025064 001010      BNE  2$      ;; BRANCH IF NOT REPORTED
4882 025066 005267 177770  INC  1$      ;; SET DOUBLE TRAP FLAG
4883 025072 011667 154070  MOV  (SP), $TMP3  ;; SAVE THE BAD PC FOR TYPING
4884 025076 004767 174502  JSR  PC, $ERROR  ;; *** ERROR *** (GO TYPE A MESSAGE)
4885 025102 000031      .WORD 31      ;; ERROR TYPE CODE.
4886 025104 000401      BR  3$      ;; SKIP HALT
4887 025106 000000      2$:  HALT      ;; ERROR! SECOND TRAP TO 4 OCCURRED
4888      ;; BEFORE FIRST WAS PRINTED
4889 025110 005067 177746  3$:  CLR  1$
4890 025114 000002      RTI      ;; RETURN TO PROGRAM AND TRY TO RECOVER
4891
4892      .SBTTL PHYSICAL ADDRESS TYPE ROUTINE
4893      ;* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4894      $TYPAD:
4895 025116 010046      MOV  R0,-(SP)  ;; PUSH R0 ON STACK
4896 025120 010146      MOV  R1,-(SP)  ;; PUSH R1 ON STACK
4897 025122 010246      MOV  R2,-(SP)  ;; PUSH R2 ON STACK
4898 025124 010346      MOV  R3,-(SP)  ;; PUSH R3 ON STACK
    
```

```

4899 025126 016602 000012      MOV      12(SP), R2      ;GET BASE ADDRESS
4900 025132 005003      CLR      R3             ;WORKING & INDEX REGISTER
4901 025134 005767 153446      TST      MMAVA         ;CHECK FOR MEM MGMT AVAILABLE
4902 025140 001430      BEQ      1$            ;BRANCH IF NO MEM MGMT
4903 025142 032737 000001 177572    BIT      #1,          2#SRO ;CHECK IF MEM MGMT ENABLED
4904 025150 001424      BEQ      1$            ;BRANCH IF MEM MGMT NOT ENABLED
4905 025152 010201      MOV      R2,          R1 ;COPY VIRTUAL ADR
4906 025154 006101      ROL      R1            ;SHUFFLE BITS 13,14,15 INTO 1,2,3
4907 025156 006101      ROL      R1
4908 025160 006101      ROL      R1
4909 025162 006101      ROL      R1
4910 025164 006101      ROL      R1
4911 025166 042701 177761      BIC      #177761, R1    ;CLR ALL EXCEPT BITS 1,2,3
4912 025172 062701 172340      ADD      #KIPARO, R1   ;SET TO APPROPRIATE PAR
4913 025176 011101      MOV      (R1), R1      ;GET CONTENTS OF PAR
4914 025200 012700 000006      MOV      #6,          R0 ;SET UP COUNTER
4915 025204 006301 4$:      ASL      R1            ;SHIFT PAR
4916 025206 006103      ROL      R3            ;SAVE OVERFLOW BITS
4917 025210 077003      SOB      R0,          4$ ;COUNT SIX SHIFTS
4918 025212 042702 160000      BIC      #160000, R2   ;SAVE BANK BITS
4919 025216 060102      ADD      R1,          R2 ;COMPUTE PHYSICAL ADDRESS
4920 025220 005503      ADC      R3            ;MAKE SURE CARRY ISN'T LOST!
4921 025222 006302 1$:      ASL      R2            ;FIRST DIGIT TO R3
4922 025224 006103      ROL      R3
4923 025226 012700 000006      MOV      #6, R0        ;DIGIT COUNT
4924 025232 000404      BR       3$, R0        ;PRINT FIRST DIGIT
4925 025234 006302 2$:      ASL      R2            ;
4926 025236 006103      ROL      R3            ;
4927 025240 005301      DEC      R1            ;
4928 025242 001374      BNE      2$,          2$ ;
4929 025244 012701 000003 3$:      MOV      #3, R1        ;DIGIT SHIFT COUNT
4930 025250 062703 000060      ADD      #60, R3       ;MAKE IT AN ASCII DIGIT
4931 025254 110367 000036      MOV      R3,          R5 ;LOAD DIGIT INTO MESSAGE
4932 025260 004567 176172      JSR      R5,          $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4933 025264 025316      .WORD   8$,          $PRINT ;ADDRESS OF MESSAGE TO BE TYPED
4934 025266 005003      CLR      R3            ;CLEAR INDEX
4935 025270 005300      DEC      R0            ;DEC DIGIT COUNT
4936 025272 001360      BNE      2$,          2$ ;
4937 025274 012603      MOV      (SP)+, R3     ;; POP STACK INTO R3
4938 025276 012602      MOV      (SP)+, R2     ;; POP STACK INTO R2
4939 025300 012601      MOV      (SP)+, R1     ;; POP STACK INTO R1
4940 025302 012600      MOV      (SP)+, R0     ;; POP STACK INTO R0
4941 025304 012616      MOV      (SP)+, (SP)   ;ADJUST THE STACK TO CLEAR DATA
4942 025306 004567 176144      JSR      R5,          $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4943 025312 026751      .WORD   $PRINT        ;ADDRESS OF MESSAGE TO BE TYPED
4944 025314 000207      RTS      PC            ;RETURN
4945 025316 000      .BYTE   0             ;ONE DIGIT MESSAGE BUFFER
4946 025317 000      .BYTE   0             ;MESSAGE TERMINATOR

```

```

4947
4948      .SBTTL  STANDARD PROGRAM MESSAGES
4949      ;*****
4950      ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
4951      ;THE PROGRAM
4952      ;*****
4953 025320 005015 052113 030461 MMAMES: .ASCIZ <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
4954 025326 024040 042515 047515

```

4955	025334	054522	046440	047101	
4956	025342	043501	046505	047105	
4957	025350	024524	040440	040526	
4958	025356	046111	041101	042514	
4959	025364	000			
4960	025365	015	046412	046505	MEMMES: .ASCIZ <15><12>'MEMORY MAP:'
4961	025372	051117	020131	040515	
4962	025400	035120	000		
4963	025403	015	041012	052131	BYTMES: .ASCIZ <15><12>'BYTE MEMORY MAP:'
4964	025410	020105	042515	047515	
4965	025416	054522	046440	050101	
4966	025424	000072			
4967	025426	005015	040520	044522	MTMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'
4968	025434	054524	046440	046505	
4969	025442	051117	020131	040515	
4970	025450	035120	000		
4971	025453	015	043012	047522	FROM: .ASCIZ <15><12>'FROM '
4972	025460	020115	000		
4973	025463	040	047524	000040	TO: .ASCIZ ' TO '
4974	025470	005015	047111	052523	INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
4975	025476	043106	041511	042511	
4976	025504	052116	046440	046505	
4977	025512	051117	027131	027056	
4978	025520	044506	051522	020124	
4979	025526	033061	020113	047516	
4980	025534	020124	046101	020114	
4981	025542	044124	051105	020505	
4982	025550	000			
4983	025551	015	047012	020117	MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'
4984	025556	040520	044522	054524	
4985	025564	051040	043505	051511	
4986	025572	042524	051522	043040	
4987	025600	052517	042116	000	
4988	025605	015	051012	051505	PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
4989	025612	040524	052122	047111	
4990	025620	020107	043101	042524	
4991	025626	020122	020101	047520	
4992	025634	042527	020122	040506	
4993	025642	046111	051125	006505	
4994	025650	000012			
4995	025652	005015	047516	050040	NOPEs: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
4996	025660	051101	052111	020131	
4997	025666	051105	047522	051522	
4998	025674	043040	052517	042116	
4999	025702	047440	020116	042515	
5000	025710	047515	054522	051440	
5001	025716	040503	006516	000012	
5002	025724	005015	051120	043517	PROREL: .ASCII <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO BK'
5003	025732	040522	020115	047516	
5004	025740	020127	042522	044523	
5005	025746	042504	020123	040502	
5006	025754	045503	040440	020124	
5007	025762	020060	047524	034040	
5008	025770	113			
5009	025771	015	044012	052111	.ASCIZ <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
5010	025776	041440	047117	044524	

5011	026004	052516	020105	047506		
5012	026012	020122	047516	046522		
5013	026020	046101	051040	047125		
5014	026026	044516	043516	005015		
5015	026034	000				
5016	026035	015	051012	043505	MX1:	.ASCIZ <15><12>'REGISTER AT '
5017	026042	051511	042524	020122		
5018	026050	052101	000040			
5019	026054	041440	047117	051124	MX2:	.ASCIZ ' CONTROLS '
5020	026062	046117	020123	000		
5021	026067	015	041412	051117	MX3:	.ASCIZ <15><12>'CORE PARITY '
5022	026074	020105	040520	044522		
5023	026102	054524	000040			
5024	026106	005015	047515	020123	MX4:	.ASCIZ <15><12>'MOS PARITY '
5025	026114	040520	044522	054524		
5026	026122	000040				
5027	026124	005015	051515	030461	MX5:	.ASCIZ <15><12>'MS11-K CSR '
5028	026132	045455	041440	051123		
5029	026140	000040				
5030	026142	051515	030461	045455	MX6:	.ASCIZ 'MS11-K MEMORY PRESENT!! TO COMPLETELY TEST RUN DZMML...'
5031	026150	046440	046505	051117		
5032	026156	020131	051120	051505		
5033	026164	047105	020524	020041		
5034	026172	047524	041440	046517		
5035	026200	046120	052105	046105		
5036	026206	020131	042524	052123		
5037	026214	051040	047125	042040		
5038	026222	046532	046115	027056		
5039	026230	000056				
5040	026232	005015	047516	046440	NOMEM:	.ASCIZ <15><12>'NO MEMORY FOUND.'
5041	026240	046505	051117	020131		
5042	026246	047506	047125	027104		
5043	026254	000				
5044	026255	015	005012	044412	FADMES:	.ASCII <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5045	026262	050116	052125	040440		
5046	026270	046114	050040	051101		
5047	026276	046501	052105	051105		
5048	026304	020123	047111	047440		
5049	026312	052103	046101	056		
5050	026317	015	043012	051111		.ASCIZ <15><12>'FIRST ADDRESS: '
5051	026324	052123	040440	042104		
5052	026332	042522	051523	020072		
5053	026340	000040				
5054	026342	005015	040514	052123	LADMES:	.ASCIZ <15><12>'LAST ADDRESS: '
5055	026350	040440	042104	042522		
5056	026356	051523	020072	020040		
5057	026364	000				
5058	026365	015	037412	042101	BADADR:	.ASCIZ <15><12>'?ADDRESS IN UNMAPPED BANK?'
5059	026372	051104	051505	020123		
5060	026400	047111	052440	046516		
5061	026406	050101	042520	020104		
5062	026414	040502	045516	000077		
5063	026422	005015	042523	042514	CONST:	.ASCIZ <15><12>'SELECT CONSTANT: '
5064	026430	052103	041440	047117		
5065	026436	052123	047101	035124		
5066	026444	000				

5067	026445	015	052412	042516	UNEXPT: .ASCIZ (15)(12)'UNEXPECTED MEMORY PARITY ERROR'
5068	026450	050130	041505	042524	
5069	026460	020104	042515	047515	
5070	026466	054522	050040	051101	
5071	026474	052111	020131	051105	
5072	026502	047522	000122		
5073	026506	005015	051120	043517	PRELOC: .ASCIZ (15)(12)'PROGRAM RELOCATED TO '
5074	026514	040522	020115	042522	
5075	026522	047514	040503	042524	
5076	026530	020104	047524	000040	
5077	026536	005015	047515	042522	MTOE: .ASCIZ (15)(12)'MORE THAN ONE PARITY ERROR FOUND.'
5078	026544	052040	040510	020116	
5079	026552	047117	020105	040520	
5080	026560	044522	054524	042440	
5081	026566	051122	051117	043040	
5082	026574	052517	042116	000056	
5083	026602	005015	041523	047101	SCANM: .ASCIZ (15)(12)'SCANNING MEMORY FOR BAD PARITY.'
5084	026610	044516	043516	046440	
5085	026616	046505	051117	020131	
5086	026624	047506	020122	040502	
5087	026632	020104	040520	044522	
5088	026640	054524	000056		
5089	026644	005015	040520	044522	PEWNC: .ASCIZ (15)(12)'PARITY ERROR WILL NOT CLEAR.'
5090	026652	054524	042440	051122	
5091	026660	051117	053440	046111	
5092	026666	020114	047516	020124	
5093	026674	046103	040505	027122	
5094	026702	000			
5095	026703	015	047012	020117	NOMTST: .ASCIZ (15)(12)'NO MEMORY TESTED.'
5096	026710	042515	047515	054522	
5097	026716	052040	051505	042524	
5098	026724	027104	000		
5099	026727	015	051412	044513	SKPMES: .ASCIZ (15)(12)'SKIPPING TEST #'
5100	026734	050120	047111	020107	
5101	026742	042524	052123	021440	
5102	026750	000			
5103	026751	377	000377		FILL2: .ASCIZ (377)(377)
5104					
5105					.SBTTL ERROR REPORTING MESSAGES AND TABLES.
5106					*****
5107					;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
5108					*****
5109	026754	040520	044522	054524	DM1: .ASCIZ 'PARITY REGISTER DATA ERROR.'
5110	026762	051040	043505	051511	
5111	026770	042524	020122	040504	
5112	026776	040524	042440	051122	
5113	027004	051117	000056		
5114	027010	042101	051104	051505	DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'
5115	027016	020123	042524	052123	
5116	027024	042440	051122	051117	
5117	027032	052050	052123	026461	
5118	027040	024465	000056		
5119	027044	047503	051516	040524	DM4: .ASCIZ 'CONSTANT DATA ERROR(TST6-10).'
5120	027052	052116	042040	052101	
5121	027060	020101	051105	047522	
5122	027066	024122	051524	033124	



5123	027074	030455	024460	000056		
5124	027102	047522	040524	044524	DM5:	.ASCIZ 'ROTATING BIT ERROR(TST11-12).'
5125	027110	043516	041040	052111		
5126	027116	042440	051122	051117		
5127	027124	052050	052123	030461		
5128	027132	030455	024462	000056		
5129	027140	047515	020123	042522	DM6:	.ASCIZ 'MOS REFRESH TEST ERROR (TST 30-31).'
5130	027146	051106	051505	020110		
5131	027154	042524	052123	042440		
5132	027162	051122	051117	024040		
5133	027170	051524	020124	030063		
5134	027176	031455	024461	000056		
5135	027204	020063	047530	020122	DM7:	.ASCIZ '3 XOR 9 PATTERN ERROR(TST13-16).'
5136	027212	020071	040520	052124		
5137	027220	051105	020116	051105		
5138	027226	047522	024122	051524		
5139	027234	030524	026463	033061		
5140	027242	027051	000			
5141	027245	115	051101	044103	DM10:	.ASCIZ "MARCHING 1'S AND 0'S ERROR(TST 27)."
5142	027252	047111	020107	023461		
5143	027260	020123	047101	020104		
5144	027266	023460	020123	051105		
5145	027274	047522	024122	051524		
5146	027302	020124	033462	027051		
5147	027310	000				
5148	027311	120	051101	052111	DM11:	.ASCIZ 'PARITY MEMORY ADDRESS ERROR(TST17).'
5149	027316	020131	042515	047515		
5150	027324	054522	040440	042104		
5151	027332	042522	051523	042440		
5152	027340	051122	051117	052050		
5153	027346	052123	033461	027051		
5154	027354	000				
5155	027355	104	052101	050111	DM12:	.ASCIZ "DATIP WITH WRONG PARITY DIDN'T TRAP(TST17)."
5156	027362	053440	052111	020110		
5157	027370	051127	047117	020107		
5158	027376	040520	044522	054524		
5159	027404	042040	042111	023516		
5160	027412	020124	051124	050101		
5161	027420	052050	052123	033461		
5162	027426	027051	000			
5163	027431	127	047522	043516	DM13:	.ASCIZ 'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
5164	027436	050040	051101	052111		
5165	027444	020131	051124	050101		
5166	027452	042520	026104	041040		
5167	027460	052123	047040	020117		
5168	027466	042522	044507	052123		
5169	027474	051105	051440	047510		
5170	027502	051527	042440	051122		
5171	027510	051117	043040	040514		
5172	027516	027107	000			
5173	027521	120	051101	052111	DM14:	.ASCIZ 'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
5174	027526	020131	042522	044507		
5175	027534	052123	051105	047040		
5176	027542	052117	046440	050101		
5177	027550	042520	020104	051501		
5178	027556	041440	047117	051124		

S179	027564	046117	044514	043516
S180	027572	052040	044510	020123
S181	027600	042101	051104	051505
S182	027606	024123	051524	030524
S183	027614	024467	000056	
S184	027620	047515	042522	052040
S185	027626	040510	020116	047117
S186	027634	020105	042522	044507
S187	027642	052123	051105	044440
S188	027650	042116	041511	052101
S189	027656	042105	050040	051101
S190	027664	052111	020131	051105
S191	027672	047522	027122	000
S192	027677	104	052101	020101
S193	027704	044123	052517	042114
S194	027712	023516	020124	040510
S195	027720	042526	041440	040510
S196	027726	043516	042105	053440
S197	027734	042510	020116	040520
S198	027742	044522	054524	042440
S199	027750	051122	051117	052040
S200	027756	040522	050120	042105
S201	027764	052050	052123	033461
S202	027772	027051	000	
S203	027775	122	047101	047504
S204	030002	020115	040504	040524
S205	030010	042440	051122	051117
S206	030016	052050	052123	030062
S207	030024	027051	000	
S208	030027	111	051516	051124
S209	030034	041525	044524	047117
S210	030042	042440	042530	052503
S211	030050	044524	047117	042440
S212	030056	051122	051117	052050
S213	030064	052123	030462	031055
S214	030072	024466	000056	
S215	030076	051120	043517	040522
S216	030104	020115	047503	042504
S217	030112	041440	040510	043516
S218	030120	042105	053440	042510
S219	030126	020116	042522	047514
S220	030134	040503	042524	027104
S221	030142	000		
S222	030143	124	040522	050120
S223	030150	042105	020054	052502
S224	030156	020124	047516	051040
S225	030164	043505	051511	042524
S226	030172	020122	040510	020104
S227	030200	051105	047522	020122
S228	030206	044502	020124	042523
S229	030214	027124	000	
S230	030217	124	040522	050120
S231	030224	042105	052040	020117
S232	030232	030461	027064	000
S233	030237	106	044501	042514
S234	030244	020104	047524	052040

DM16: .ASCIZ 'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'

DM17: .ASCIZ "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17)."

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.'

```

5235 030252 040522 027120 000
5236 030257 050 041501 044524 DM27: .ASCIZ "(ACTION ENABLE WASN'T SET)."
5237 030264 047117 042440 040516
5238 030272 046102 020105 040527
5239 030300 047123 052047 051440
5240 030306 052105 027051 000
5241 030313 015 052012 040522 DM31: .ASCIZ '<15><12>'TRAPPED TO 4 '
5242 030320 050120 042105 052040
5243 030326 020117 020064 000
5244
5245
5246
5247
5248
5249 030333 120 004503 042522 DH1: .ASCIZ 'PC REG S/B WAS'
5250 030340 004507 027523 004502
5251 030346 040527 000123
5252 030352 027526 041520 050011 DH2: .ASCIZ 'V/PC P/PC MA S/B WAS'
5253 030360 050057 004503 040515
5254 030366 051411 041057 053411
5255 030374 051501 000
5256 030377 126 050057 004503 DH12: .ASCIZ 'V/PC P/PC MA S/B'
5257 030404 027520 041520 046411
5258 030412 004501 027523 000102
5259 030420 027526 041520 050011 DH14: .ASCIZ 'V/PC P/PC REG MA'
5260 030426 050057 004503 042522
5261 030434 004507 040515 000
5262 030441 126 050057 004503 DH15: .ASCIZ 'V/PC P/PC MAUT REG S/B WAS'
5263 030446 027520 041520 046411
5264 030454 052501 004524 042522
5265 030462 004507 027523 004502
5266 030470 040527 000123
5267 030474 027526 041520 050011 DH21: .ASCIZ 'V/PC P/PC IUT MA S/B WAS'
5268 030502 050057 004503 052511
5269 030510 004524 040515 051411
5270 030516 041057 053411 051501
5271 030524 000
5272 030525 126 050057 004503 DH23: .ASCIZ 'V/PC P/PC SRC MA DST MA S/B WAS'
5273 030532 027520 041520 051411
5274 030540 041522 046440 004501
5275 030546 051504 020124 040515
5276 030554 051411 041057 053411
5277 030562 051501 000
5278 030565 126 050057 004503 DH24: .ASCIZ 'V/PC P/PC TRP/PC'
5279 030572 027520 041520 052011
5280 030600 050122 050057 000103
5281 030606 027526 041520 050011 DH25: .ASCIZ 'V/PC P/PC TRP/PC REG WAS'
5282 030614 050057 004503 051124
5283 030622 027520 041520 051011
5284 030630 043505 053411 051501
5285 030636 000
5286 030637 126 050057 004503 DH26: .ASCIZ 'V/PC P/PC REG WAS'
5287 030644 027520 041520 051011
5288 030652 043505 053411 051501
5289 030660 000
5290 030661 122 043505 053411 DH30: .ASCIZ 'REG WAS MA WAS'

```

```

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

```

```

5291 030666 051501 046411 004501
5292 030674 040527 000123
5293
5294
5295
5296
5297 030700 000 377 000
5298 030703 000
5299 030704 000 377 377
5300 030707 000 000
5301 030711 000 377 377
5302 030714 376 376
5303 030716 000 377 377
5304 030721 377 000 000
5305 030724 000 377 000
5306 030727 377 000 000
5307 030732 377 000 377
5308 030735 376
5309
5310
5311 032110
5312
5313 000001

```

```

*****
;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
*****
DF1: .BYTE 0,-1,0,0
DF2: .BYTE 0,-1,-1,0,0
DF3: .BYTE 0,-1,-1,-2,-2
DF14: .BYTE 0,-1,-1,-1,0,0
DF21: .BYTE 0,-1,0,-1,0,0
DF30: .BYTE -1,0,-1,-2
.EVEN
. = 32110
.END

```

;THE LOADERS ARE SAVE HERE TO END OF 8K



































ERRORS DETECTED: 0

CZQMCE.BIN,CZQMCE.LST/CRF/SOL/NL:TOC=CZQMCE.SML,CZQMCE.P11  
RUN-TIME: 22 29 2 SECONDS  
RUN-TIME RATIO: 325/54=5.9  
CORE USED: 39K (77 PAGES)

J16

