

# MS11

0-124K MEMORY EXERCISER 16K  
CZQMCFO

AH-9047F-MC

COPYRIGHT ©75-78

FICHE 1 OF 1

APR 1978

**digital**

MADE IN USA



EOP10278ED566:

00010000 780330  
\*\*\*\*\*

IDENTIFICATION:

HDR10ZQMCFS9

00010000

780330  
SER 0001

PRODUCT CODE: AC-9045F-MC  
 PRODUCT NAME: CZQMCFO 0-124K MEM EXER 16K  
 PRODUCT DATE: 15-FEBRUARY-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 PDF 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
Revision F:	February 1978



## 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 BK 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. The applicable Memory System Maintenance Manual
- D. The applicable Circuit Schematics

### 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-11/20 - 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 SWOS=1, then:

Load Address 214,  
Press START.

- B. If running without memory management, then:

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

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

Load Address 777707 (PC)  
Deposit 214  
Press CONTINUE

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

Load Address 772340 (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 8K of memory and the program code was not able to be transferred properly. -
- C. In the case of error reporting and there is no terminal to allow the information transfer.

- D. In the power fail routine if the power up sequence was started before the power down sequence had a chance to complete itself.
- E. In the Memory mapping routine or any of the address control routines, failures to find a meaningful map.

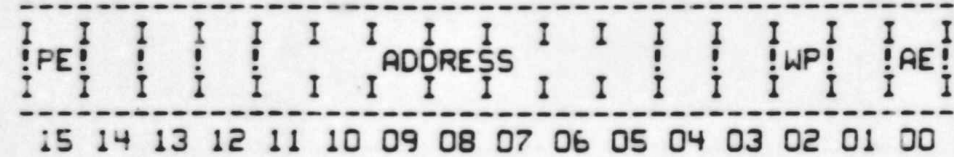
4.0 PERFORMANCE AND PROGRESS REPORTS

Not applicable

5.0 DEVICE INFORMATION TABLES

The following is a picture view of a parity control status registers, which will show bit assignments and definitions, to provide a handy reference:

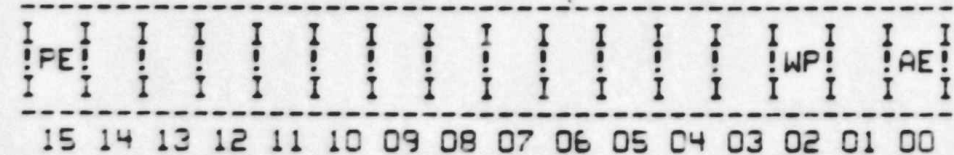
5.1 CORE PARITY REGISTER



Bit assignments are defined as follows:

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

5.2 MOS PARITY REGISTER

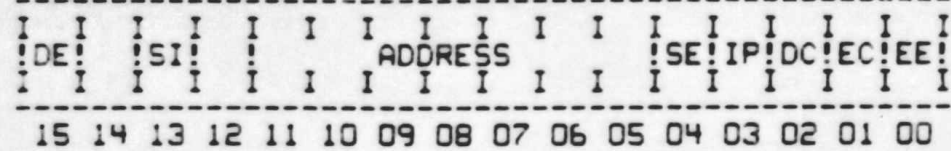




BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

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

5.3 MS11-K CSR



BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

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

## 5.0 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.

### 5.3 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.

### 5.4 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	BEG 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	BEG 1\$	:BRANCH IF DONE
40	024000	CMP -(R0),R0	:CHECK DATA WRITTEN
42	001774	BEG 3\$	:BRANCH IF OK
44	000000	HALT	:ERROR
46	000772	BR 3\$	:LOOP BACK

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

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

### 6.5.2 Toggle-in-Program #2

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



lower limit address and continuing to the upper limit.  
Toggle in the following patches to the program above.

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

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

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

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

7.0 PROGRAM FUNCTIONAL FLOW CHARTS  
Attached

8.0 PROGRAM LISTING  
Attached

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



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

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

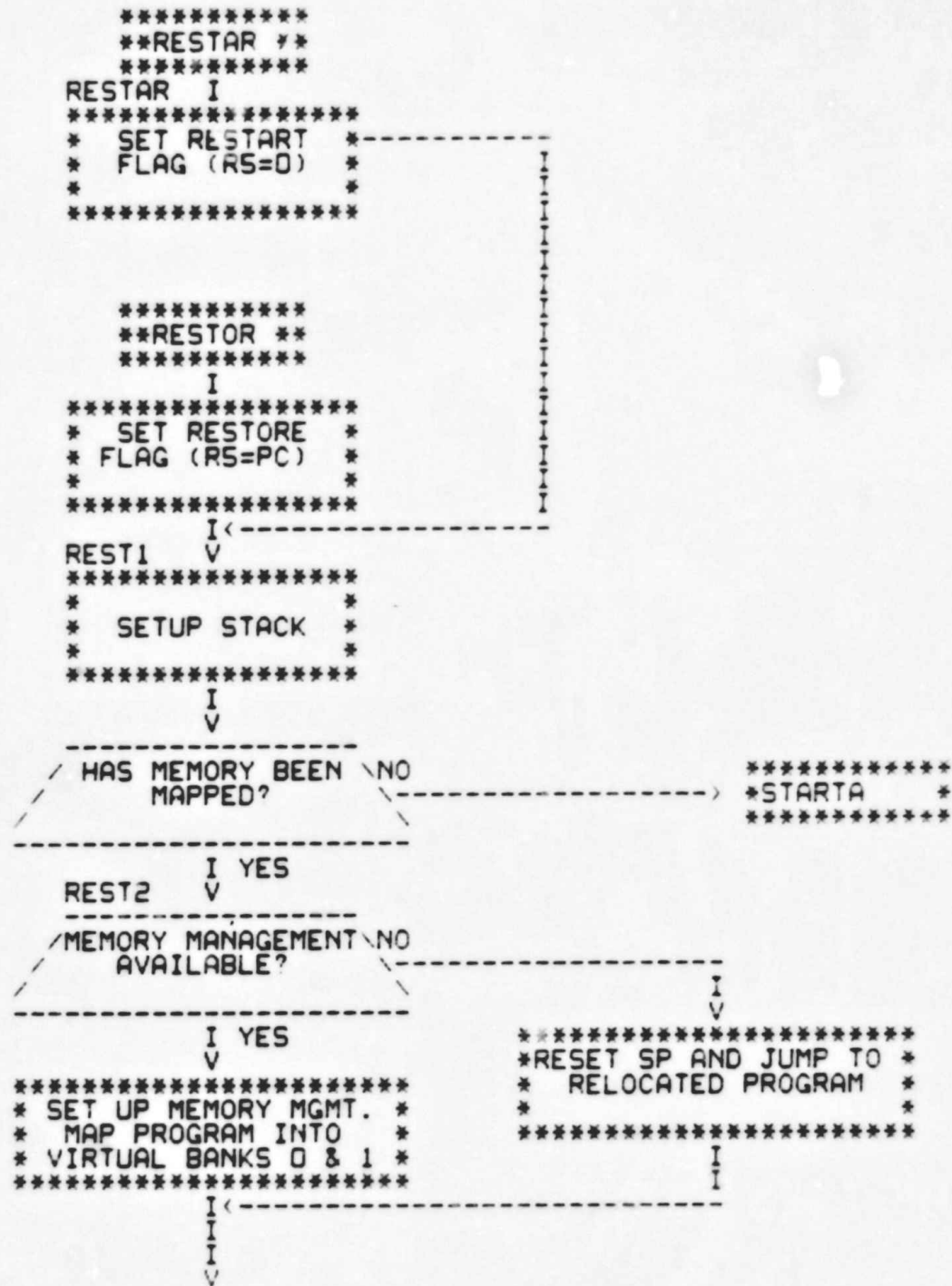
. = 0

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

CZQMCFO 0-124K MEM EXER 16K  
RESTART AND RESTORE ROUTINES

SA=210 . =300

SA=214





```
-----  
PROGRAM MAP \YES  
POINTING TO BANKS 0 \-----  
      & 1?  
-----  
      I NO  
      V RELO(59)  
*****  
** RELOCATE PROGRAM TO **  
**   BANKS 0 & 1   **  
**                   **  
*****  
      I<-----  
      V  
-----  
RESTART FLAG \YES  
(RS=0)? -----> *START1(16)*  
-----  
      I NO  
      V RESLDR(59)  
*****  
**   RESTORE LOADERS   **  
**                   **  
*****  
      I  
      V  
*****  
**HALT **  
*****
```

. =572

CZQMCFO 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 $PRINT(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

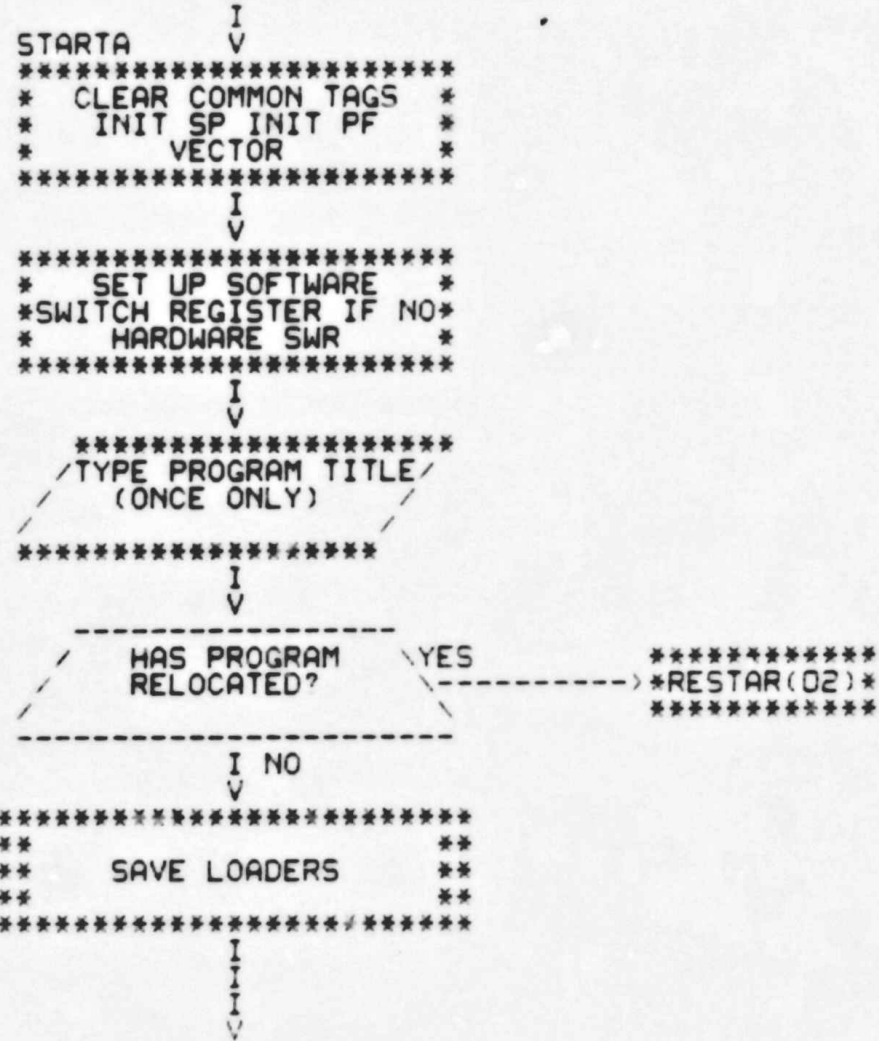
```

SA=200

```

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

```



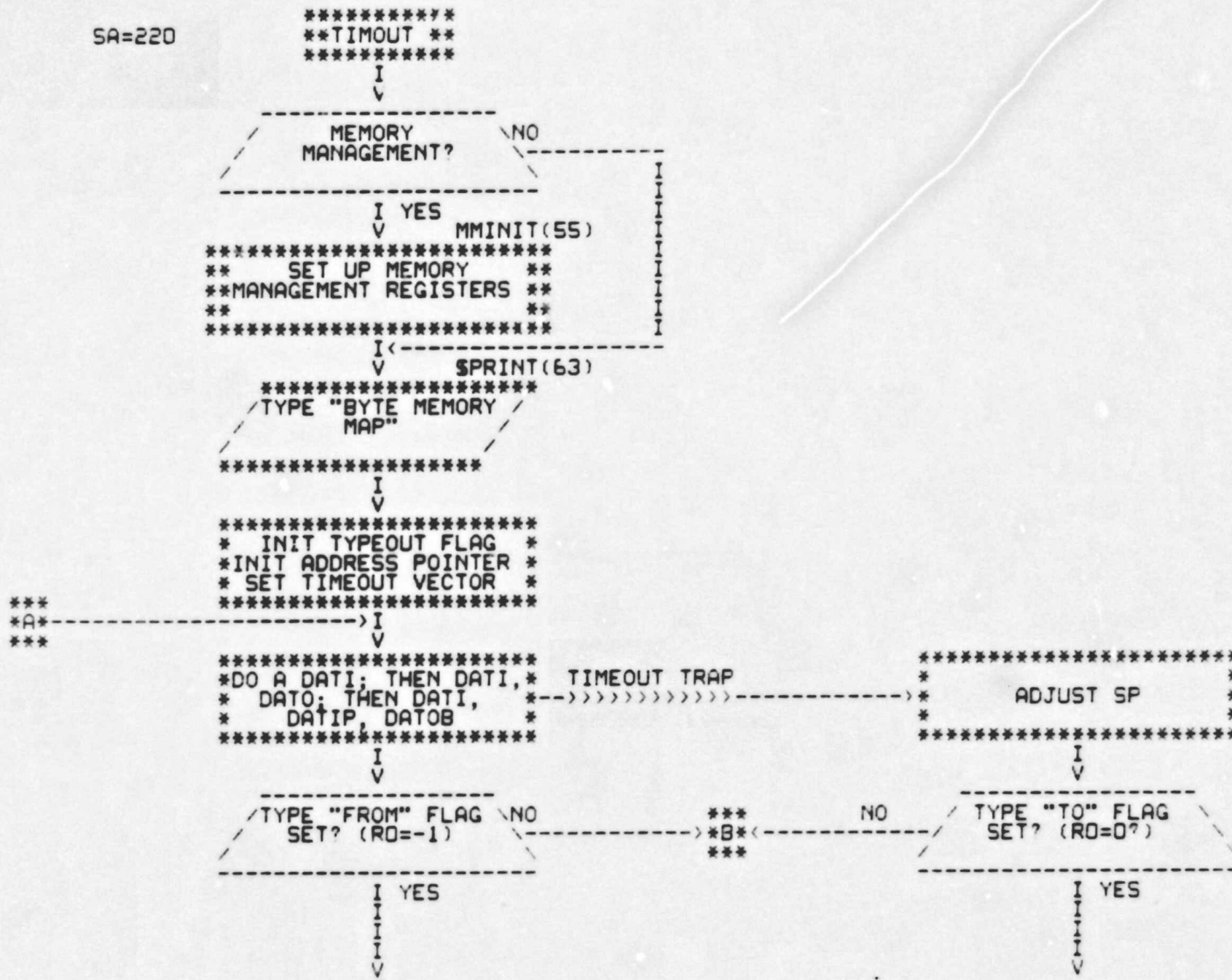


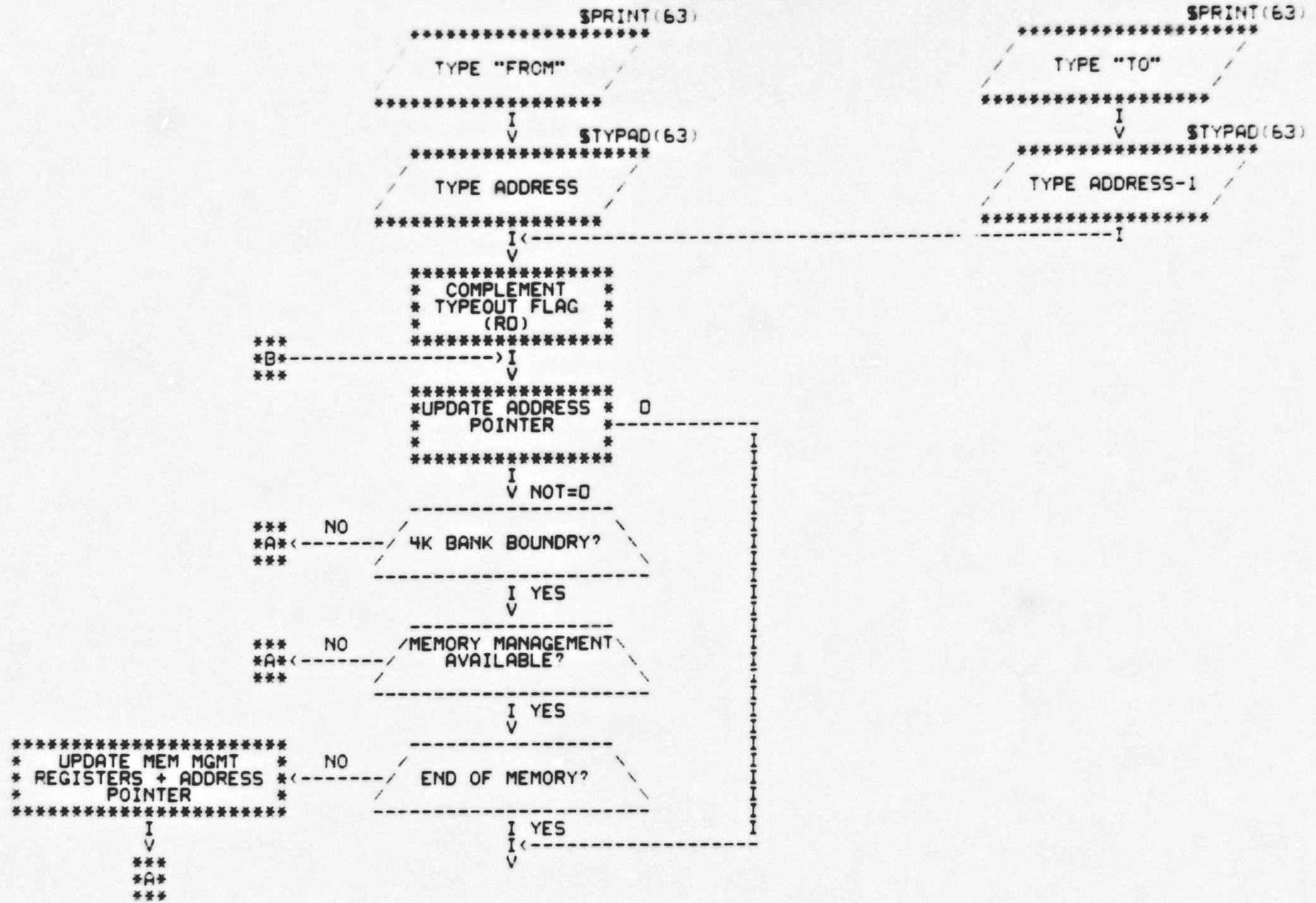






SA=220





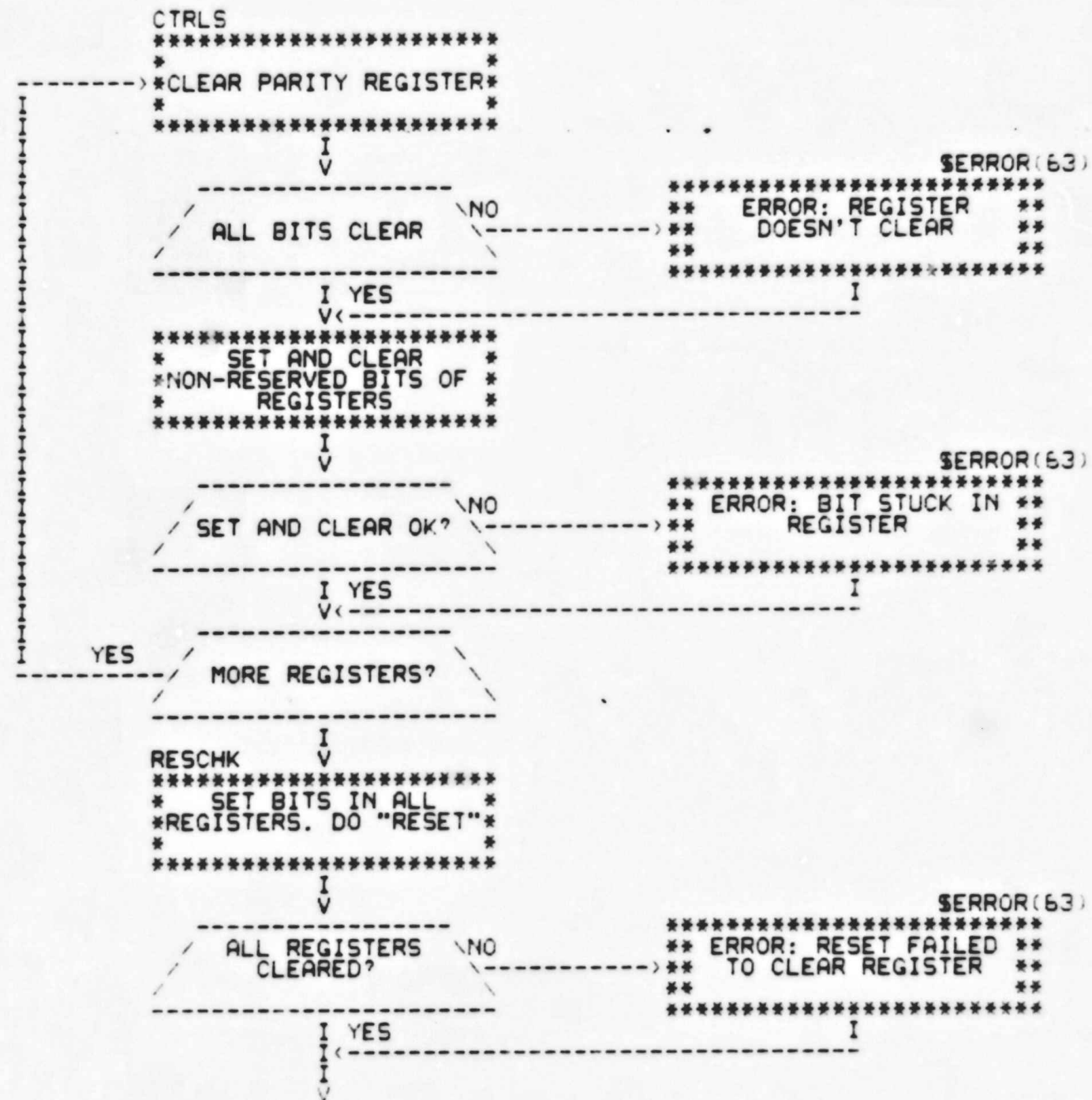




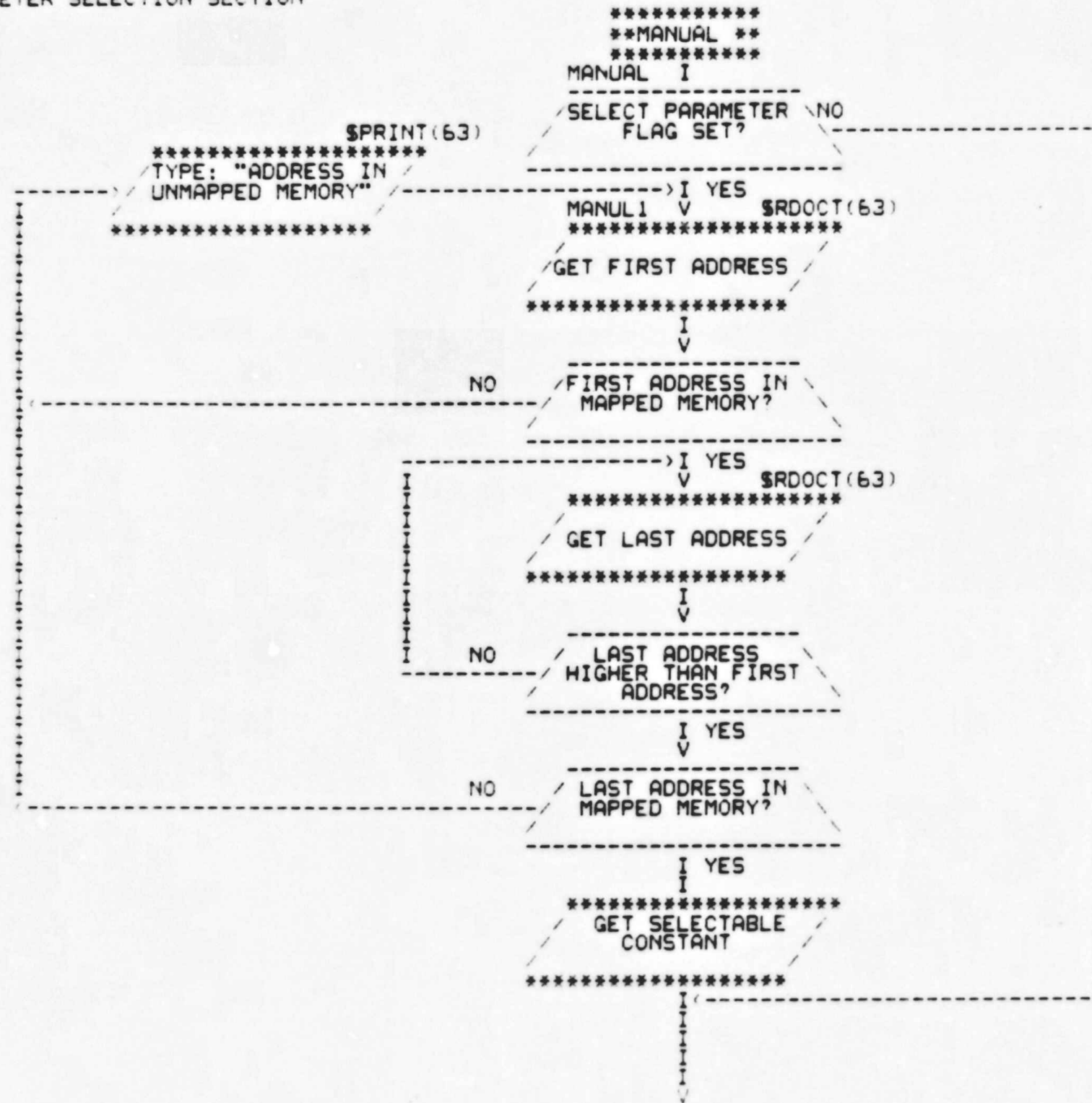




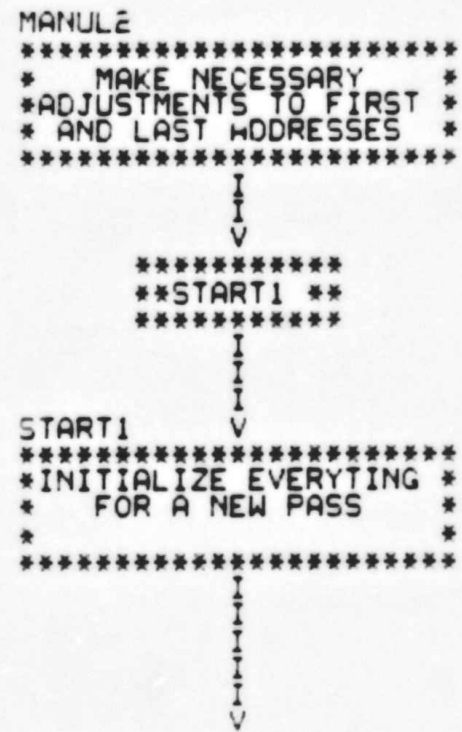






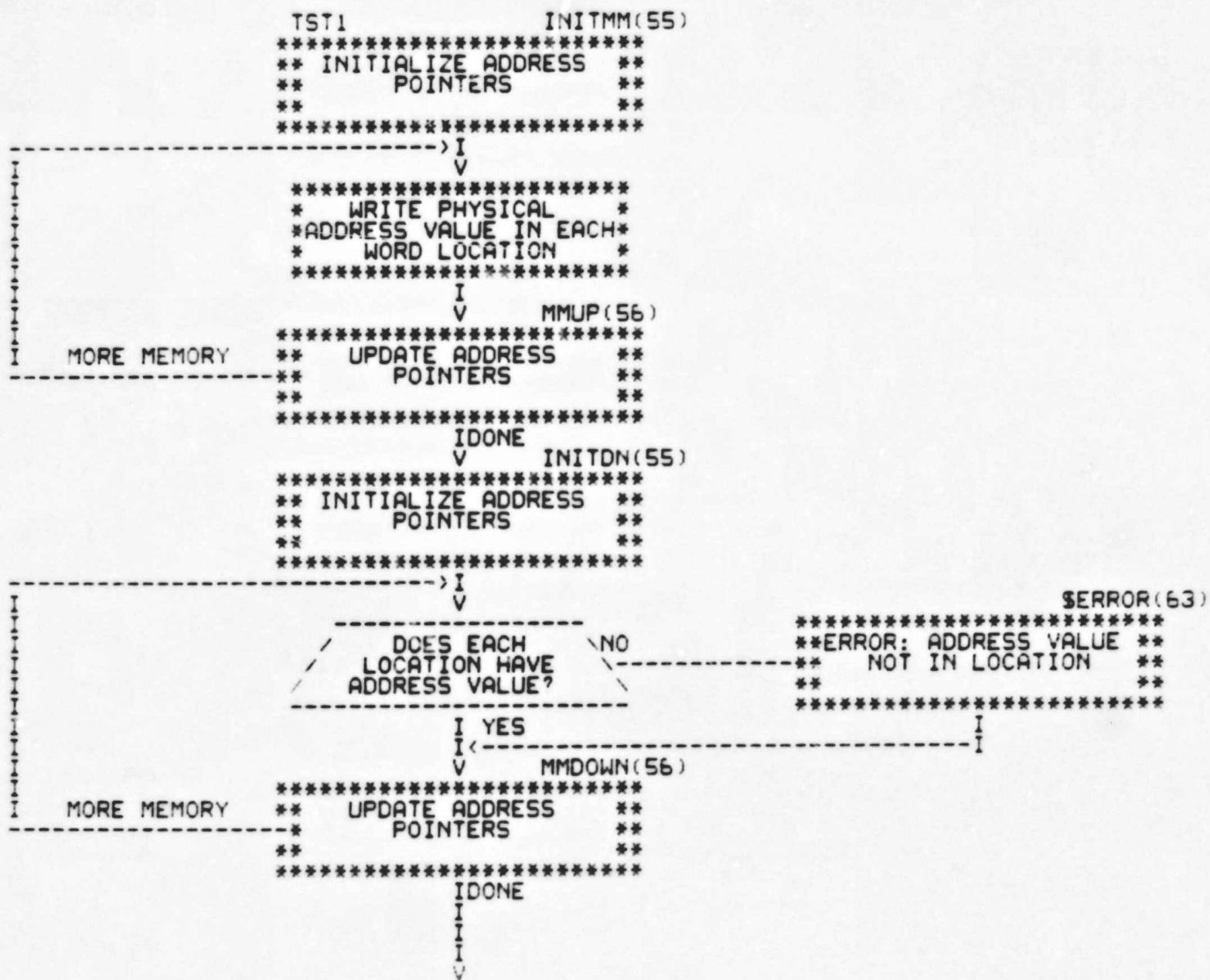


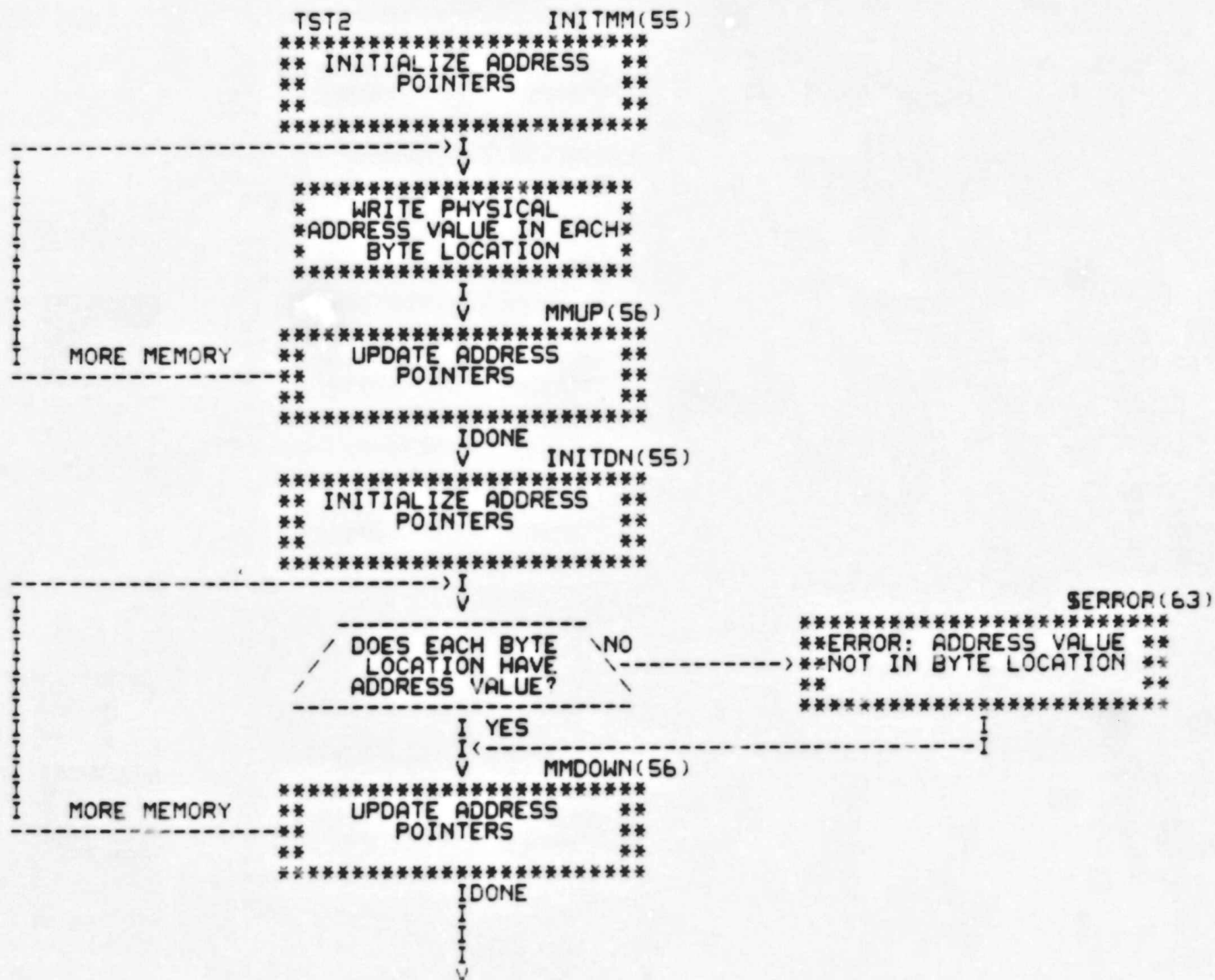
CZQMCFO 0-124K MEM EXER 16K  
START1: START OF PASS





CZQMCFO 0-124K MEM EXER 16K  
SECTION 1: ADDRESS TESTS. TEST 1







```

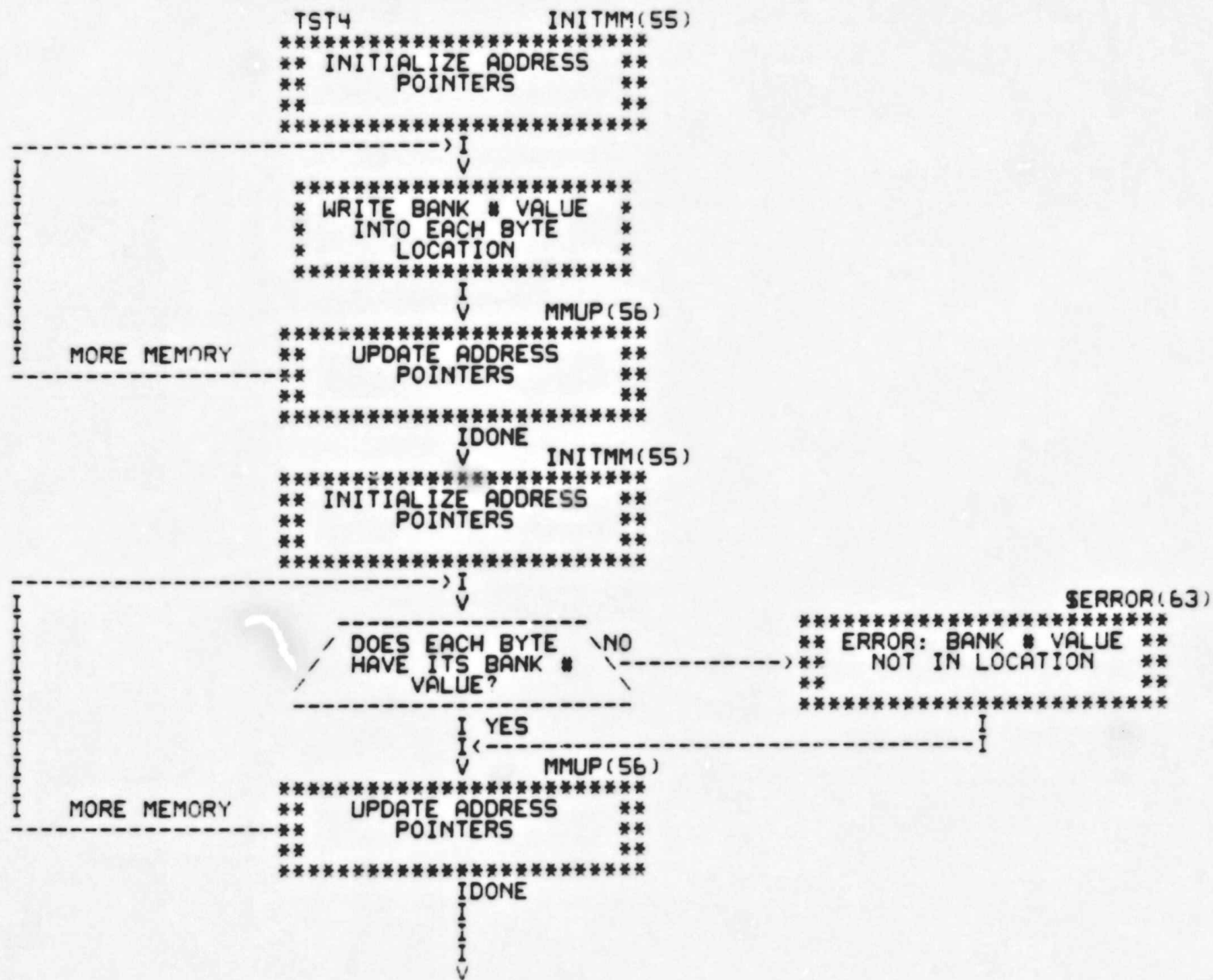
TST3          INITDN(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
----->I
          V
*****
**   WRITE ONE'S     **
** COMPLEMENT OF ADR **
** INTO WORD LOCATION **
*****
          I
          V          MMDOWN(56)
*****
MORE MEMORY ** UPDATE ADDRESS **
-----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

```

```

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

```

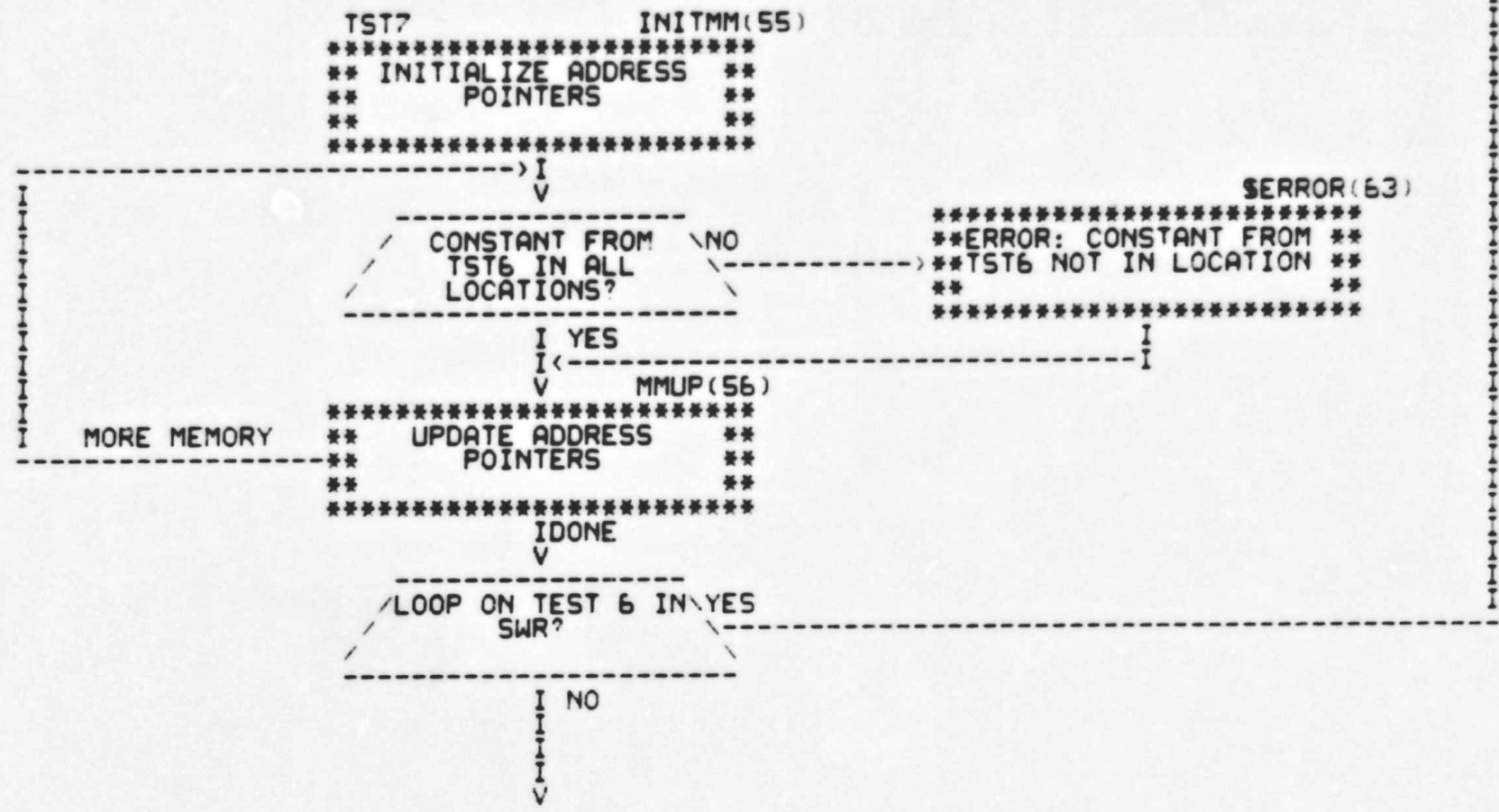


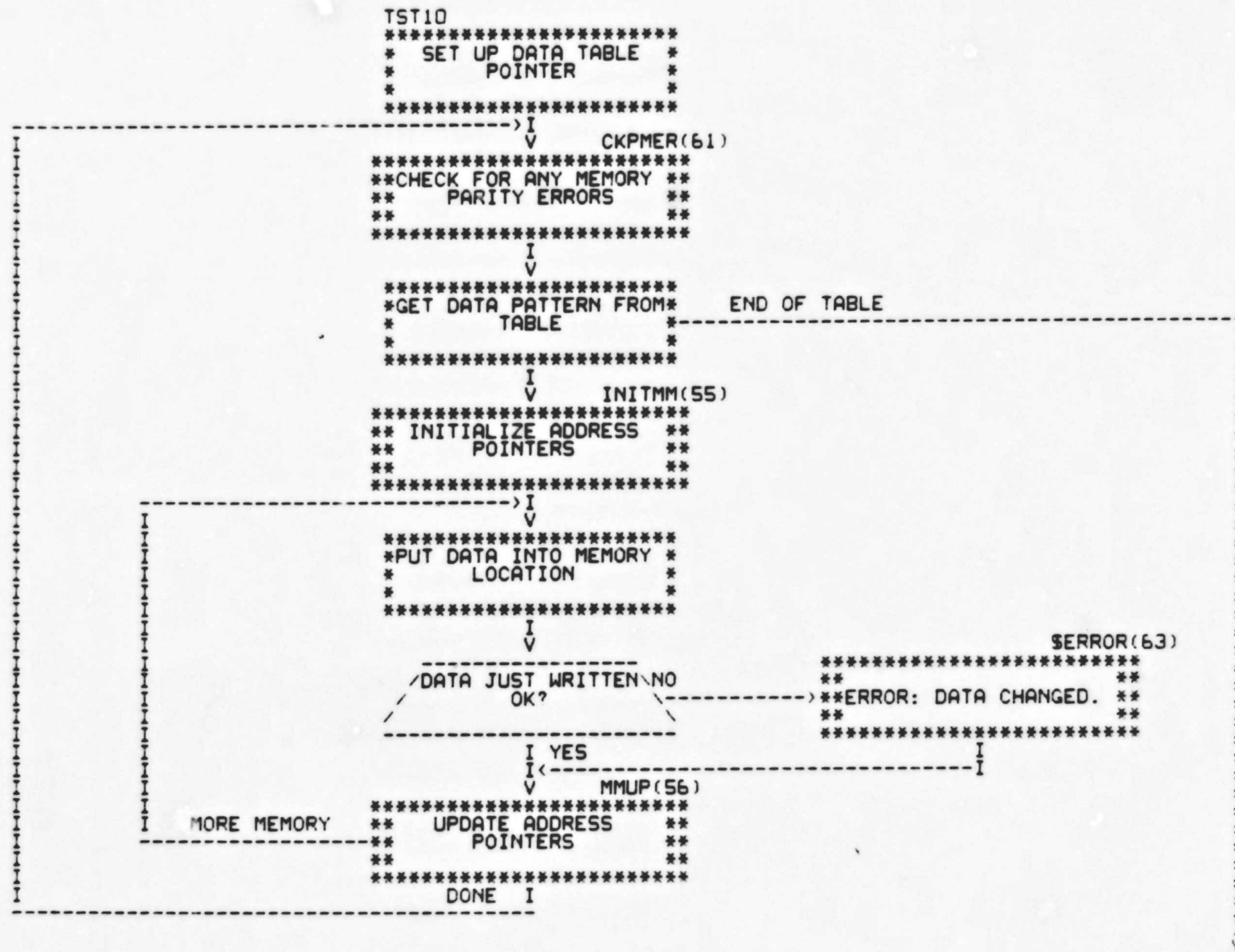






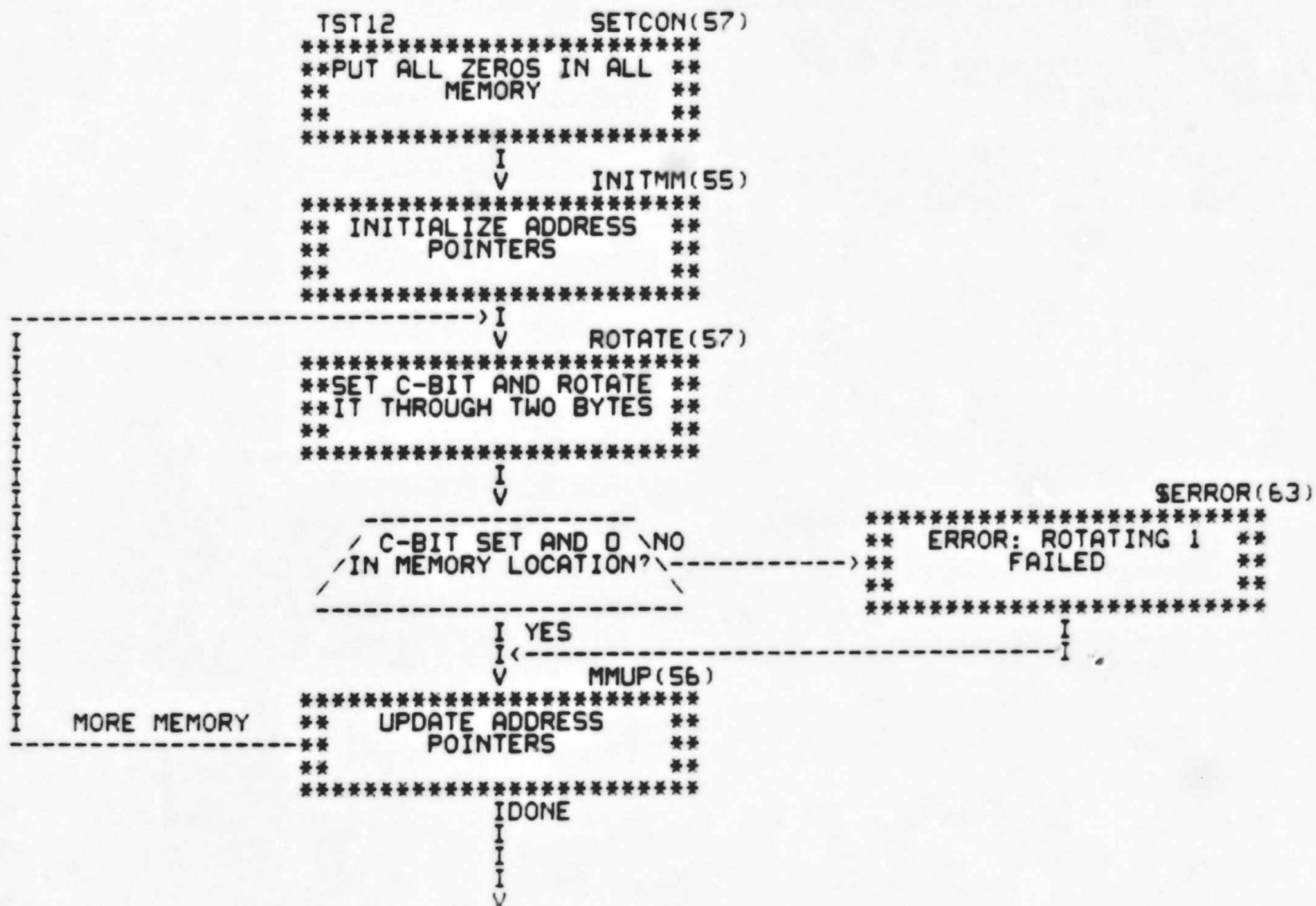




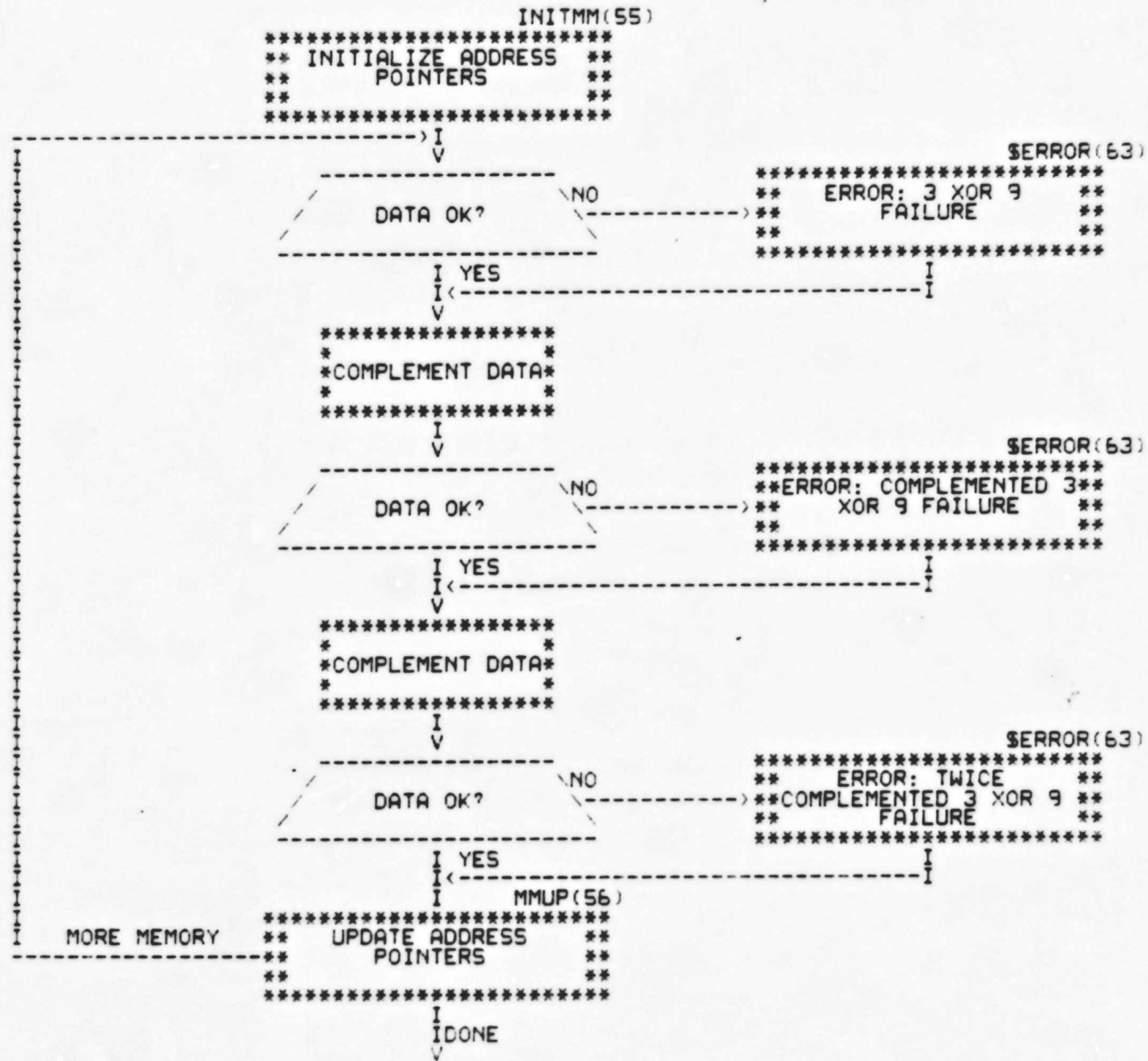














CZQMCFO 0-124K MEM EXER 16K  
TEST 14: 3 XOR 9

```

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)
*****
MORE MEMORY **   UPDATE ADDRESS   **
----- **   POINTERS             **
*****
          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 <----- I
          V      MMUP(56)
*****
MORE MEMORY **   UPDATE ADDRESS   **
----- **   POINTERS             **
*****
          IDONE
          I
          I
          I
          V

```

```

***** ERROR(63) *****
** ERROR: 3 XOR 9 **
** PATTERN FAILURE **
*****

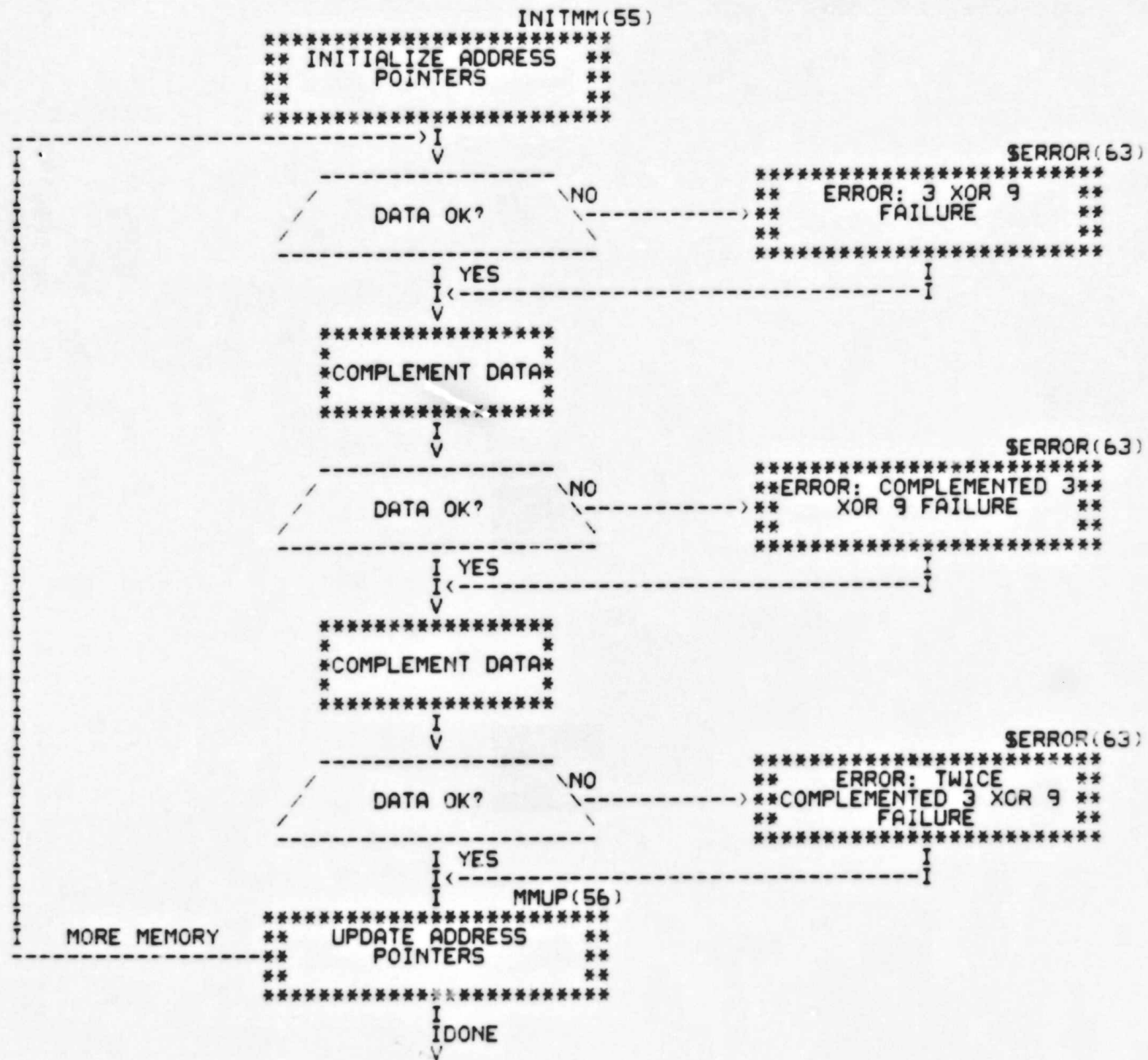
```







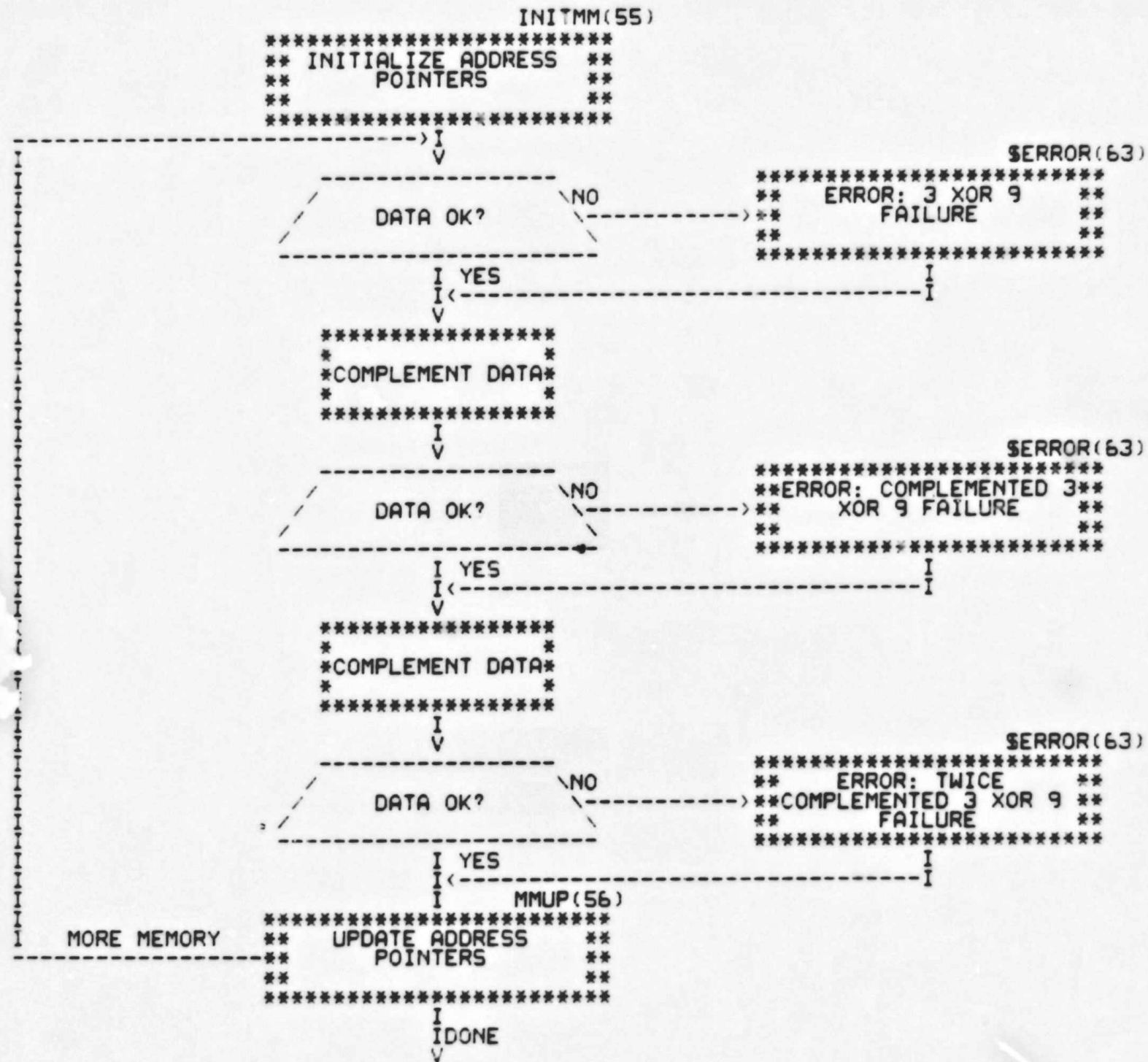
CZQMCFO 0-124K MEM EXER 16K  
TEST 15: 3 XOR 9 (FOR PARITY)



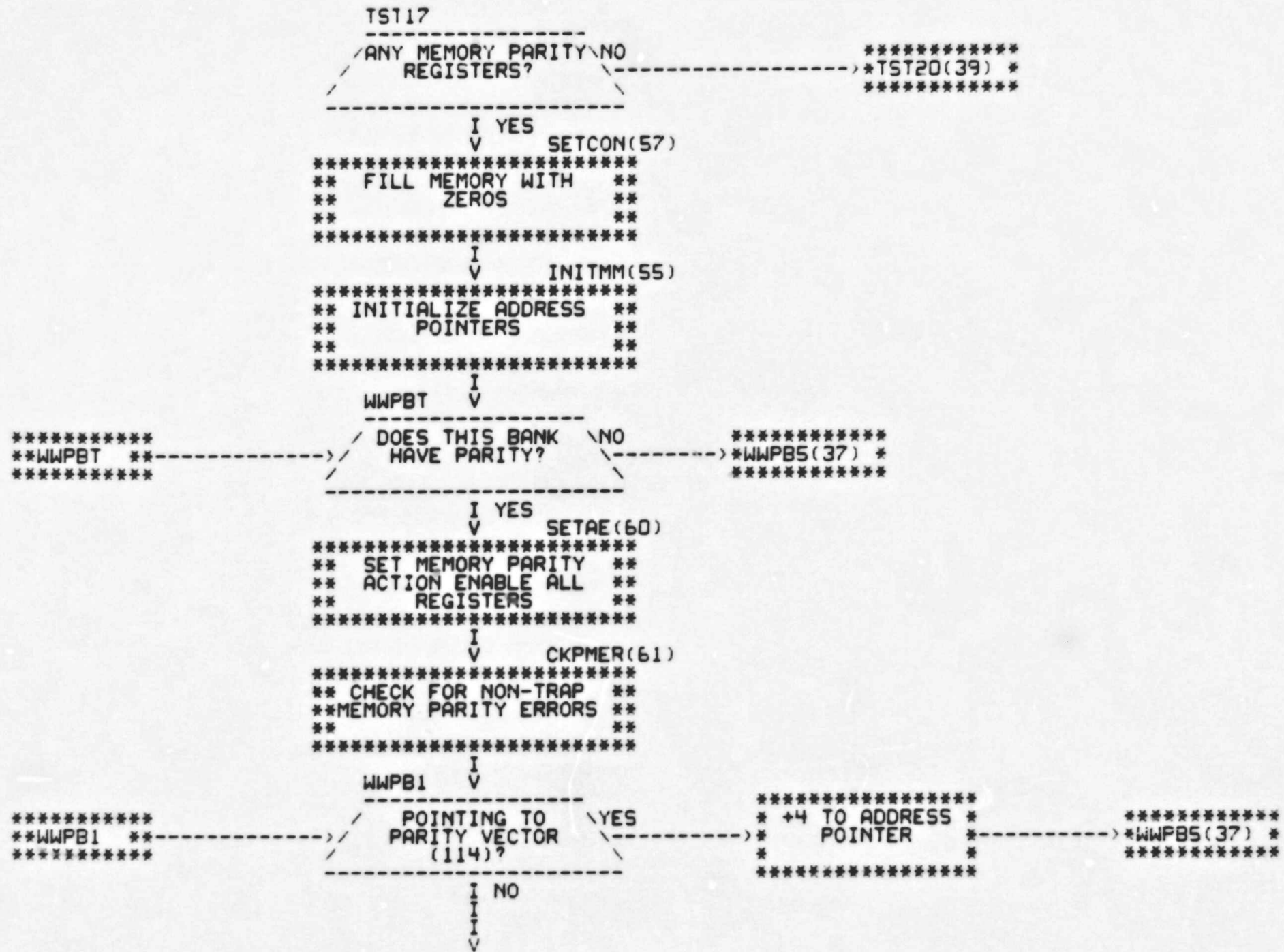
```

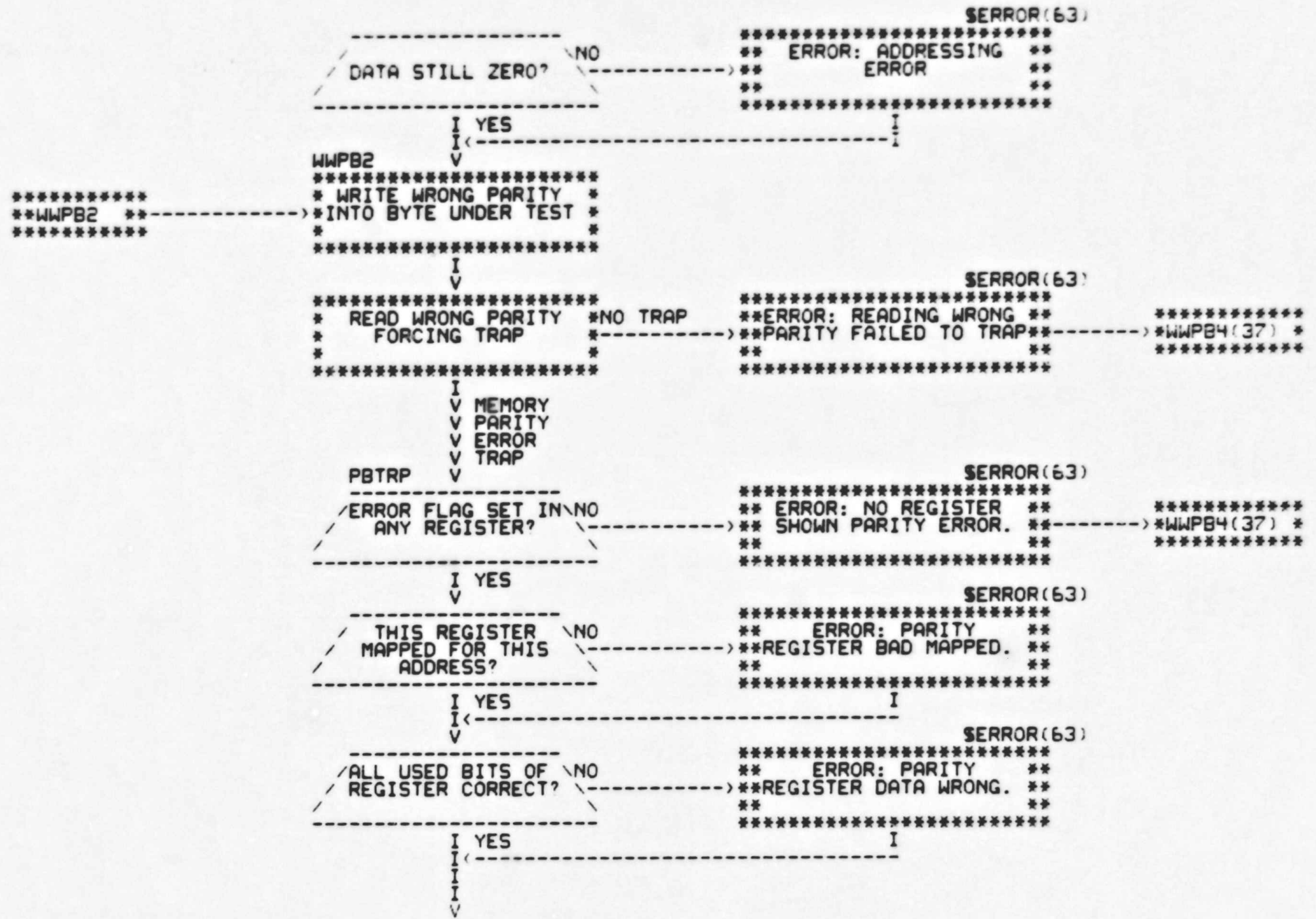
TST16          INITMM(55)
*****
** INITIALIZE ADDRESS **
**   POINTERS         **
**                   **
*****
> I
  V          W3X9(57)
*****
** WRITE 256. WORD   **
** BLOCKS WITH -1 AND **
**   401             **
*****
  I
  V          MMUP(56)
*****
** UPDATE ADDRESS   **
**   POINTERS      **
**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

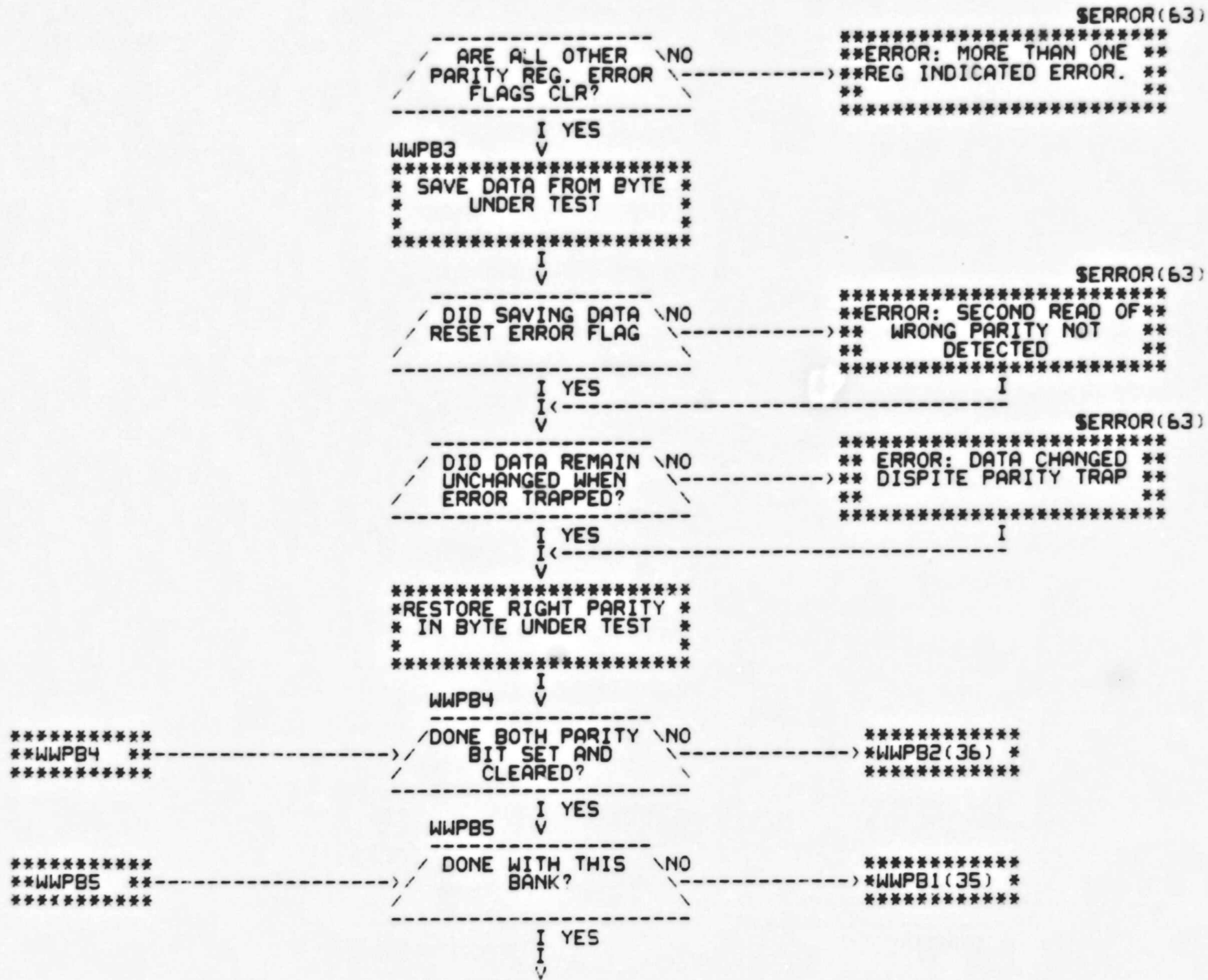
```











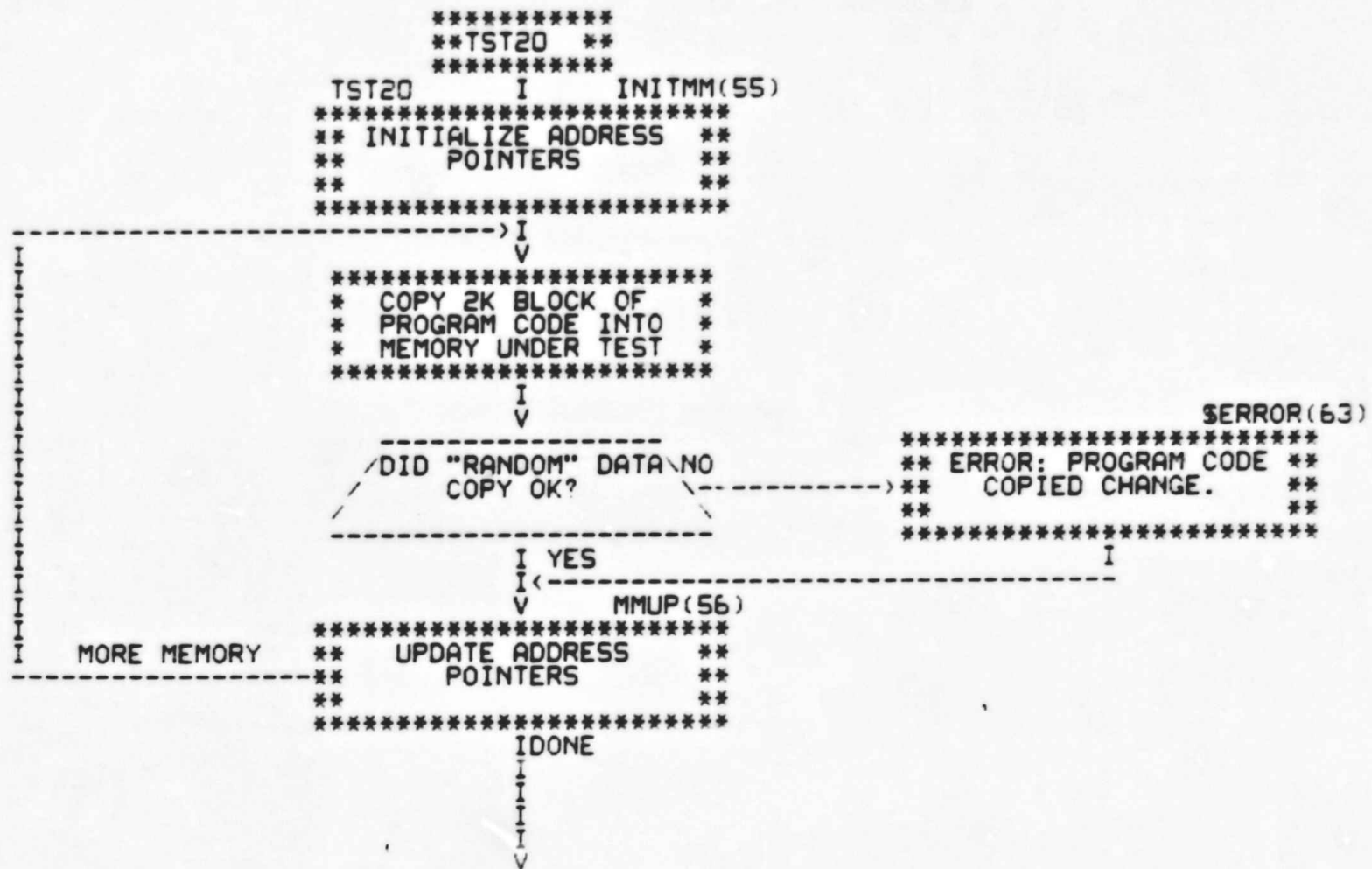


CZQMCFO 0-124K MEM EXER 16K  
TEST 17: PARITY BYTE TEST

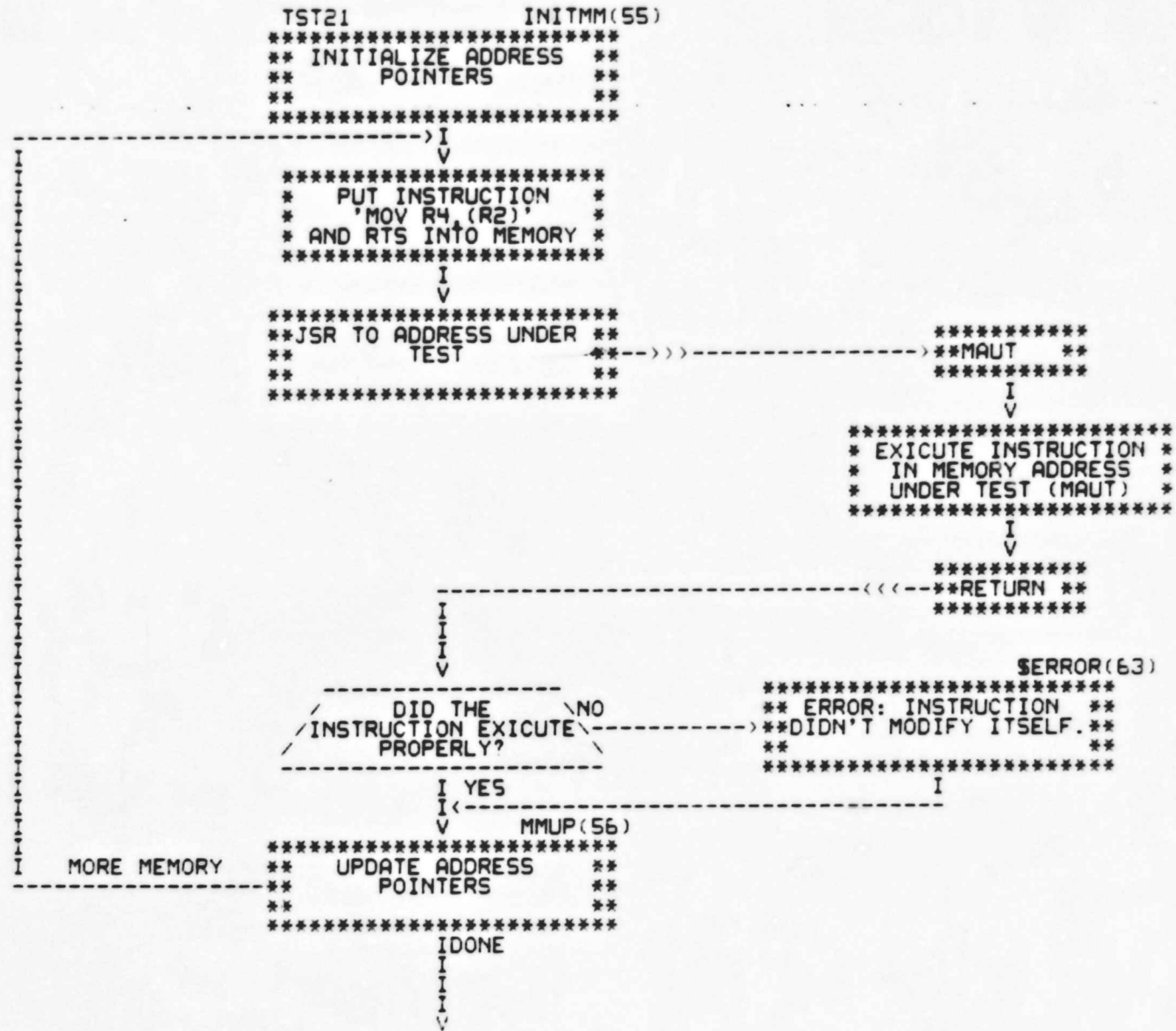
```

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

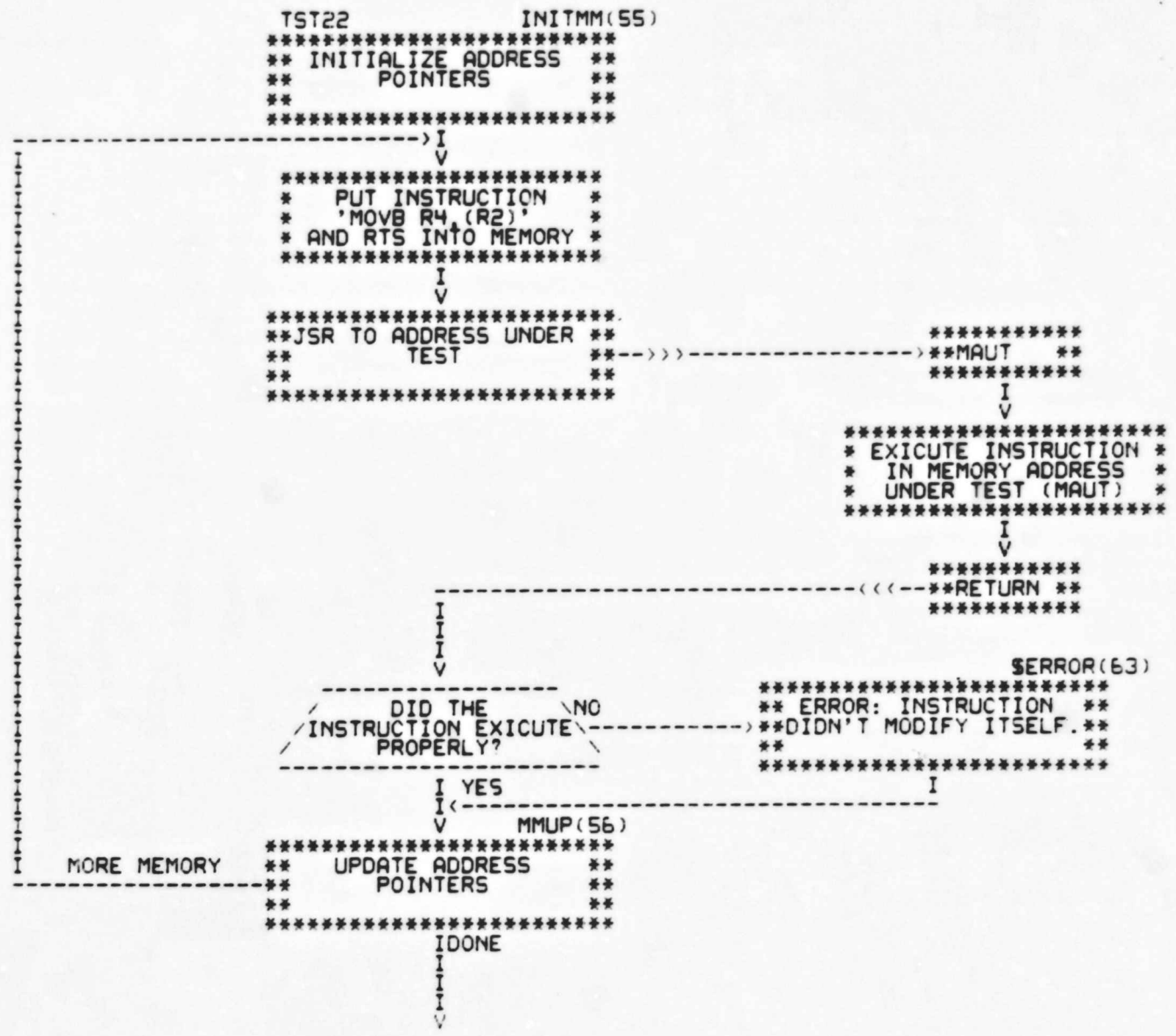
```

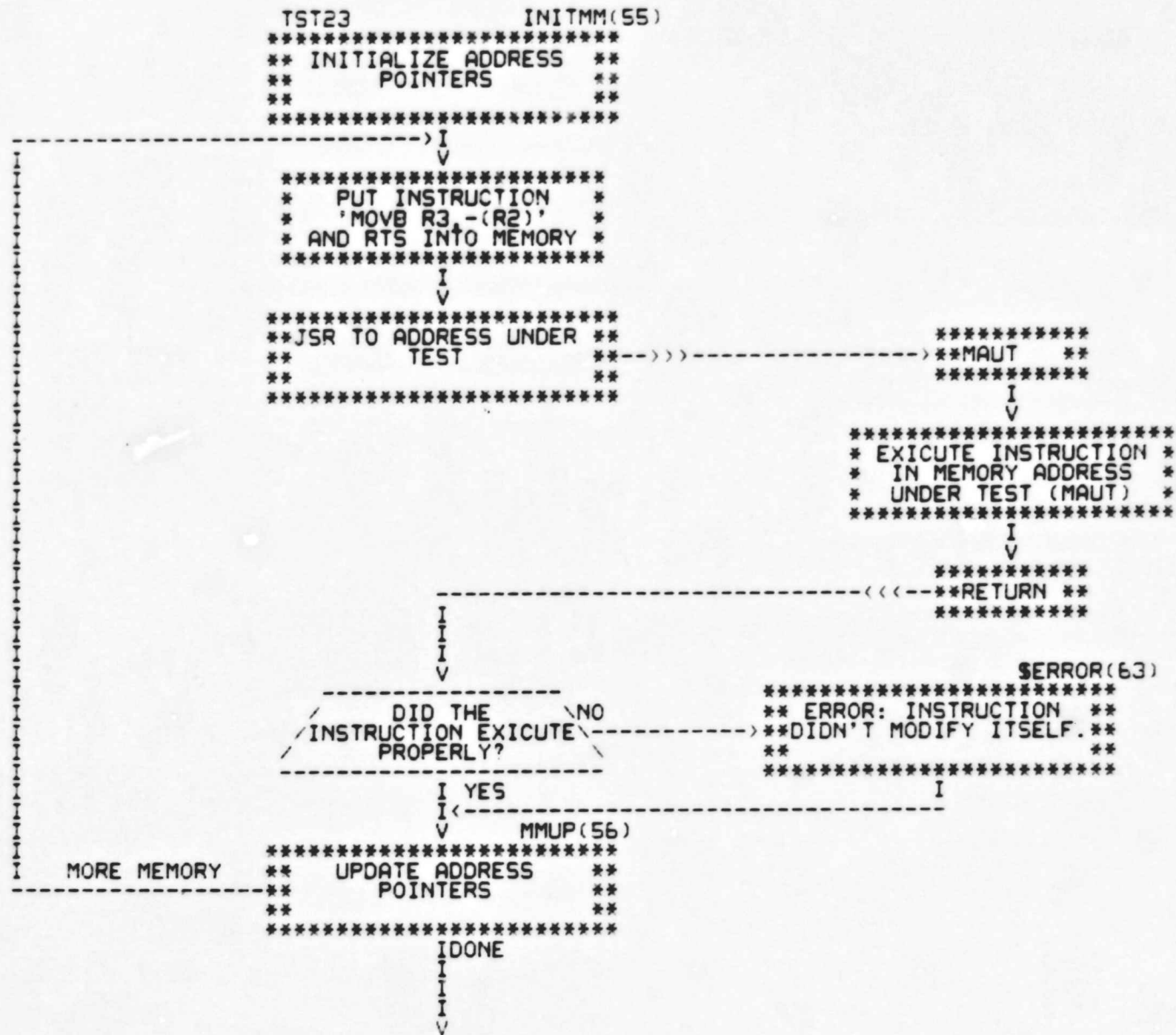


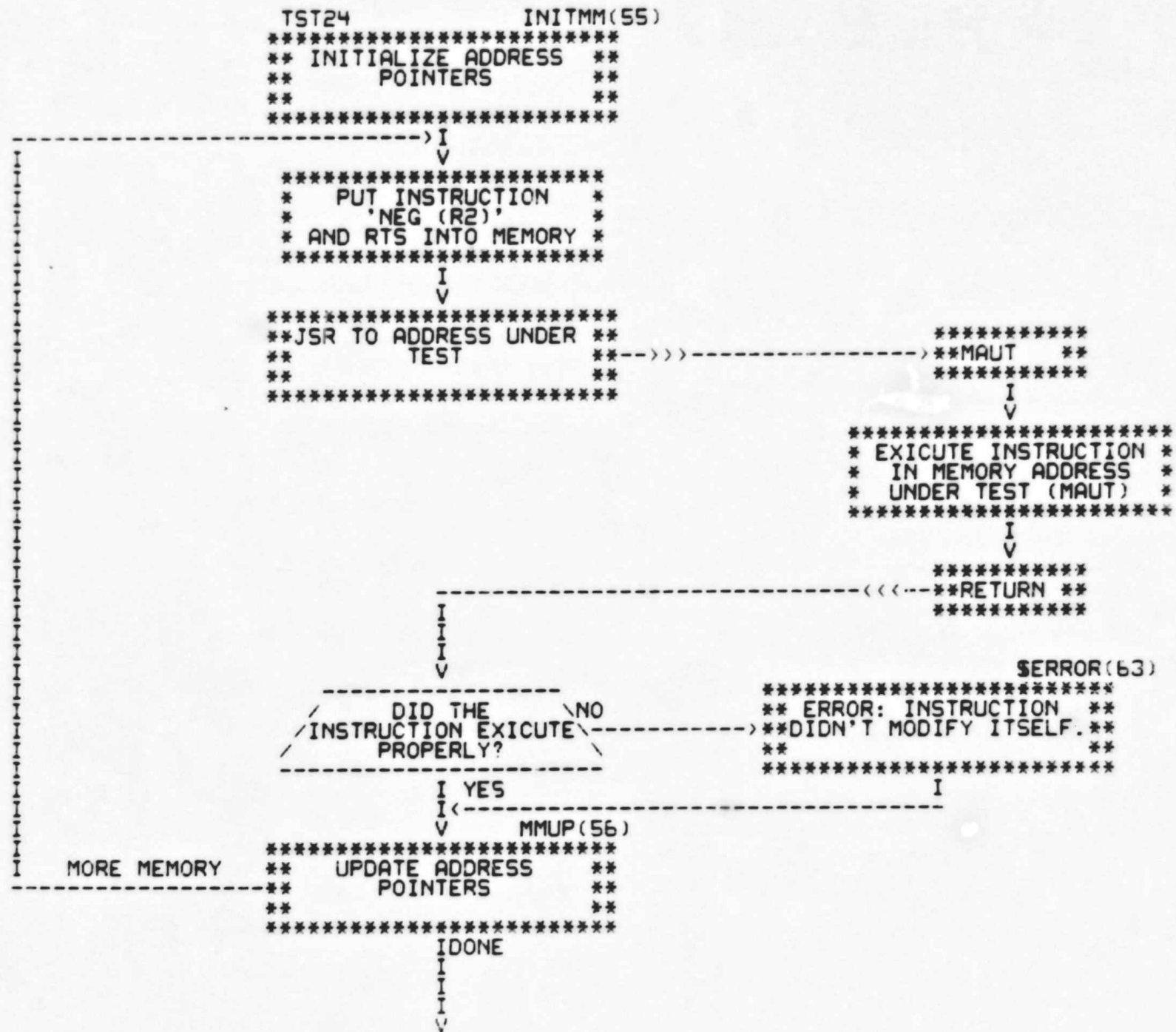
CZQMCFO 0-124K MEM EXER 16K  
TEST 21: EXICUTE DATI, DATO





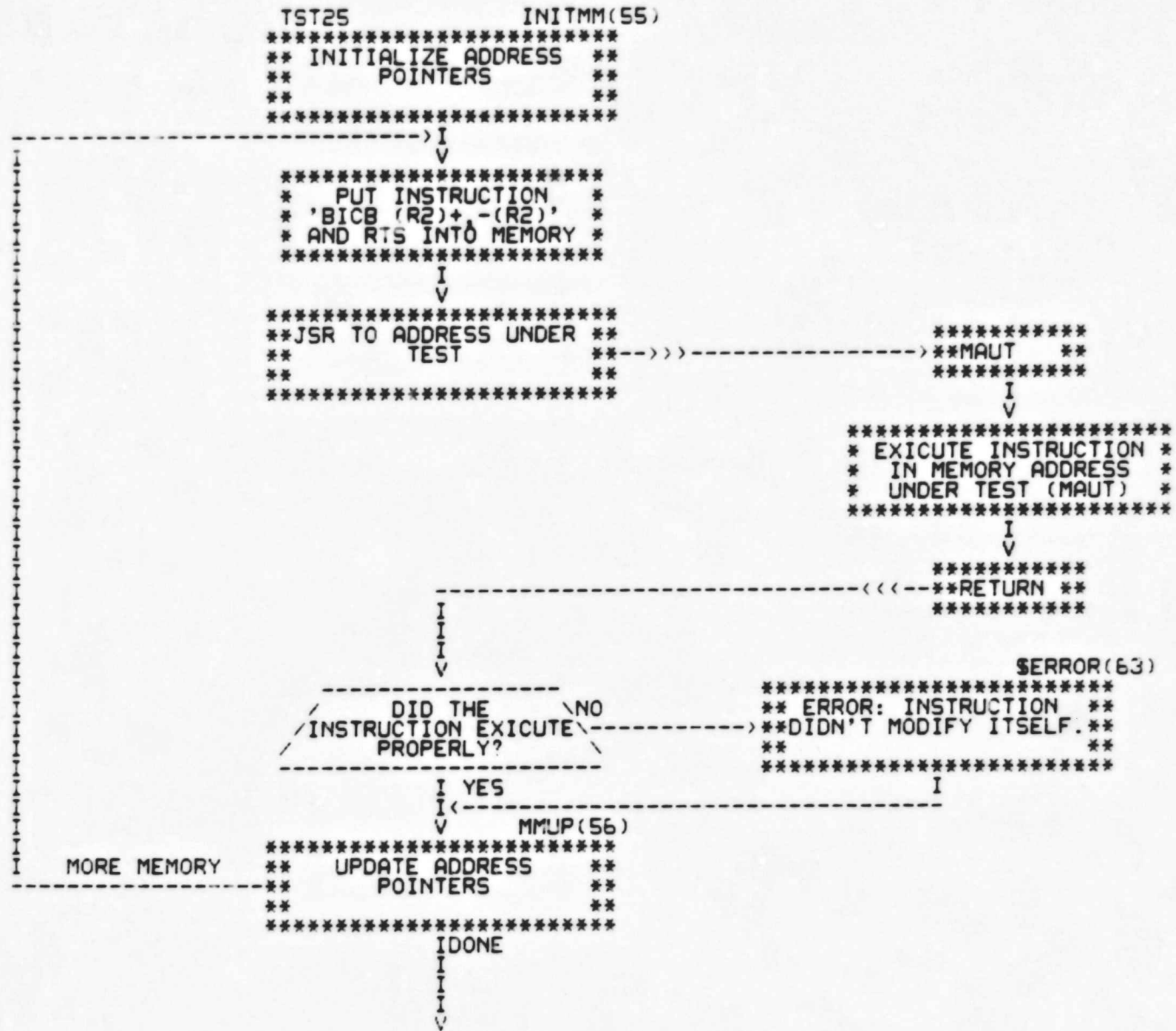




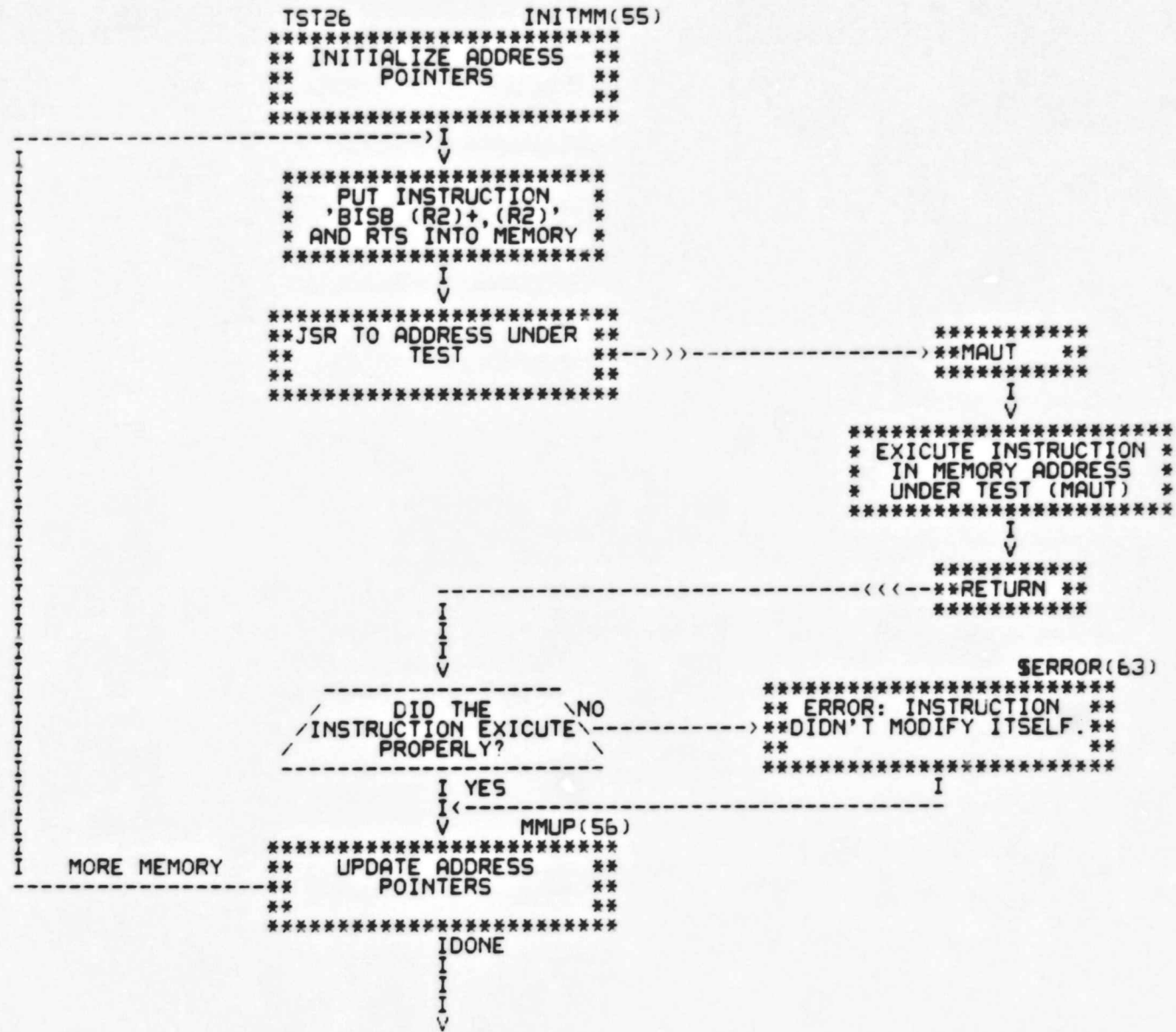


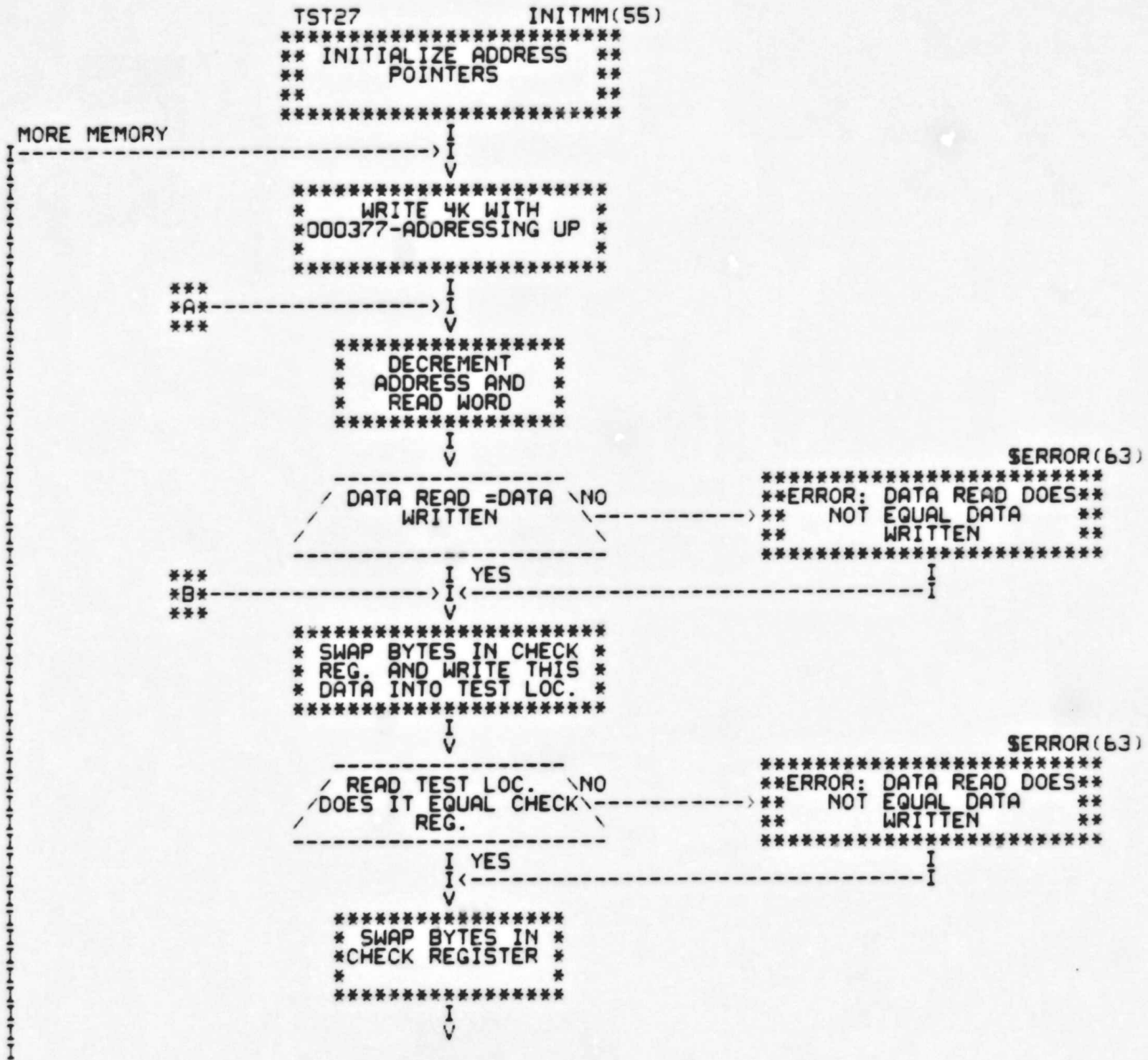


CZQMCFO 0-124K MEM EXER 16K  
TEST 25: EXICUTE DATIP, DATOB (LO BYTE)



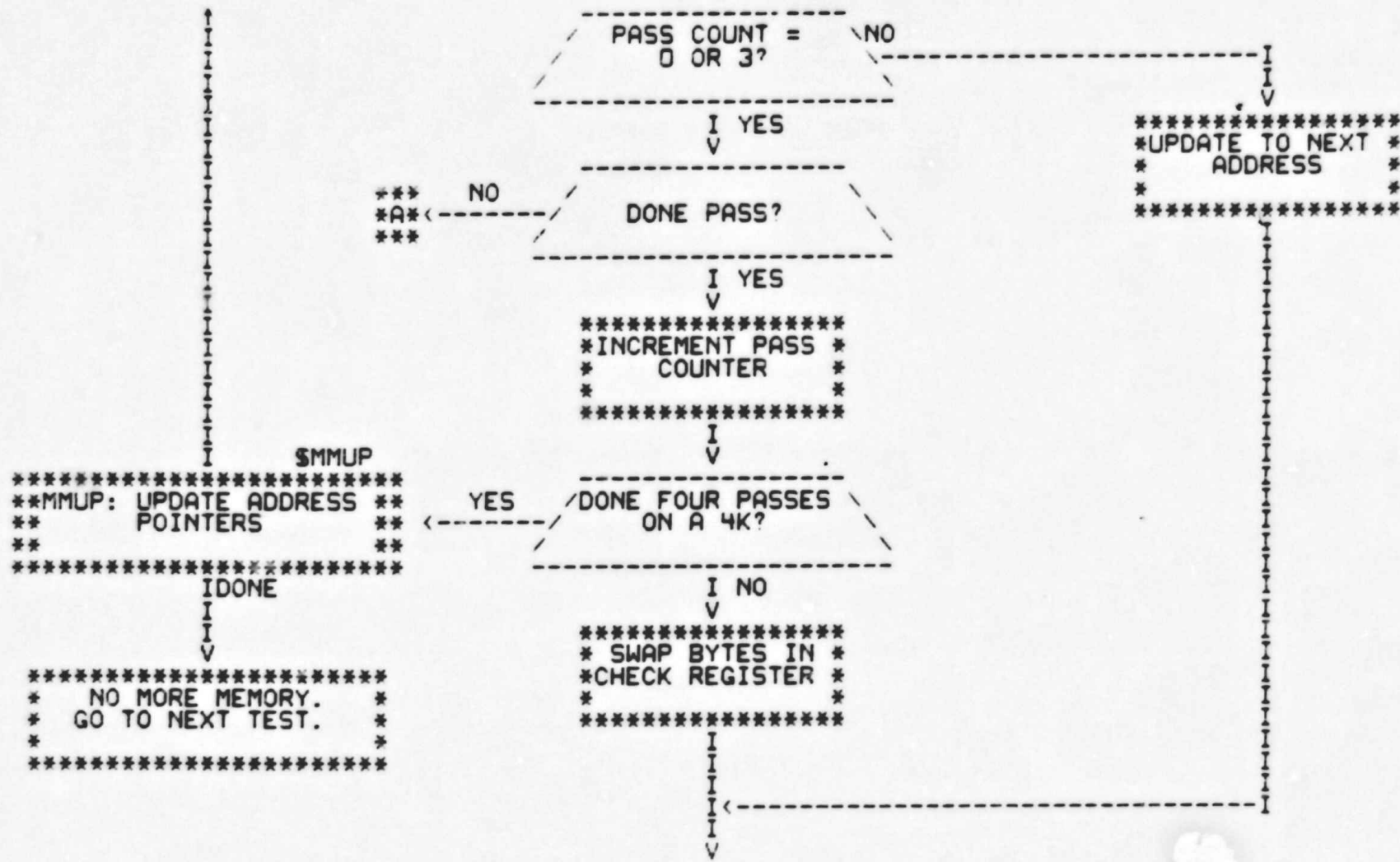
CZQMCFO 0-124K MEM EXER 16K  
TEST 26: EXICUTE DATIP, DATOB (HI BYTE)



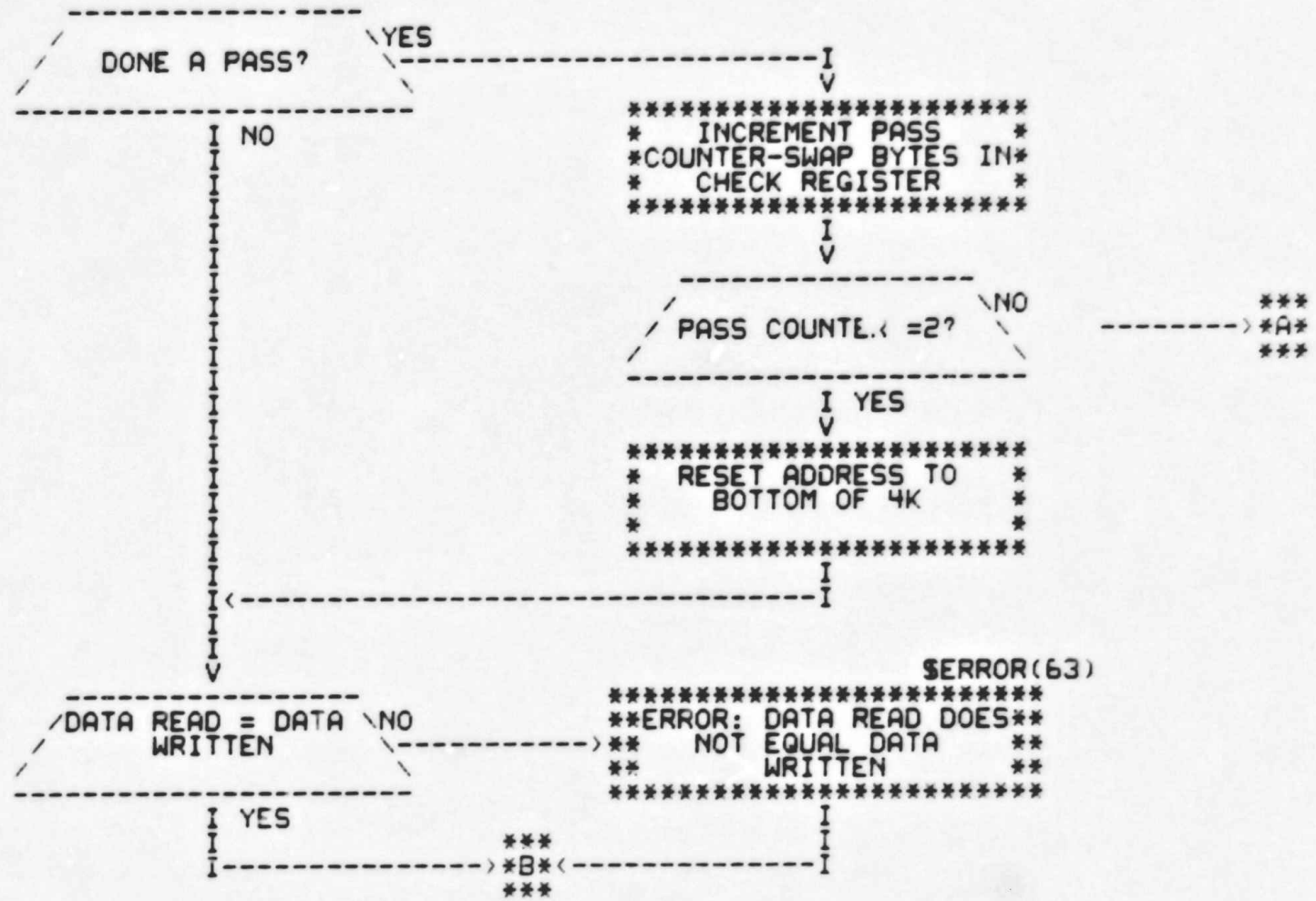


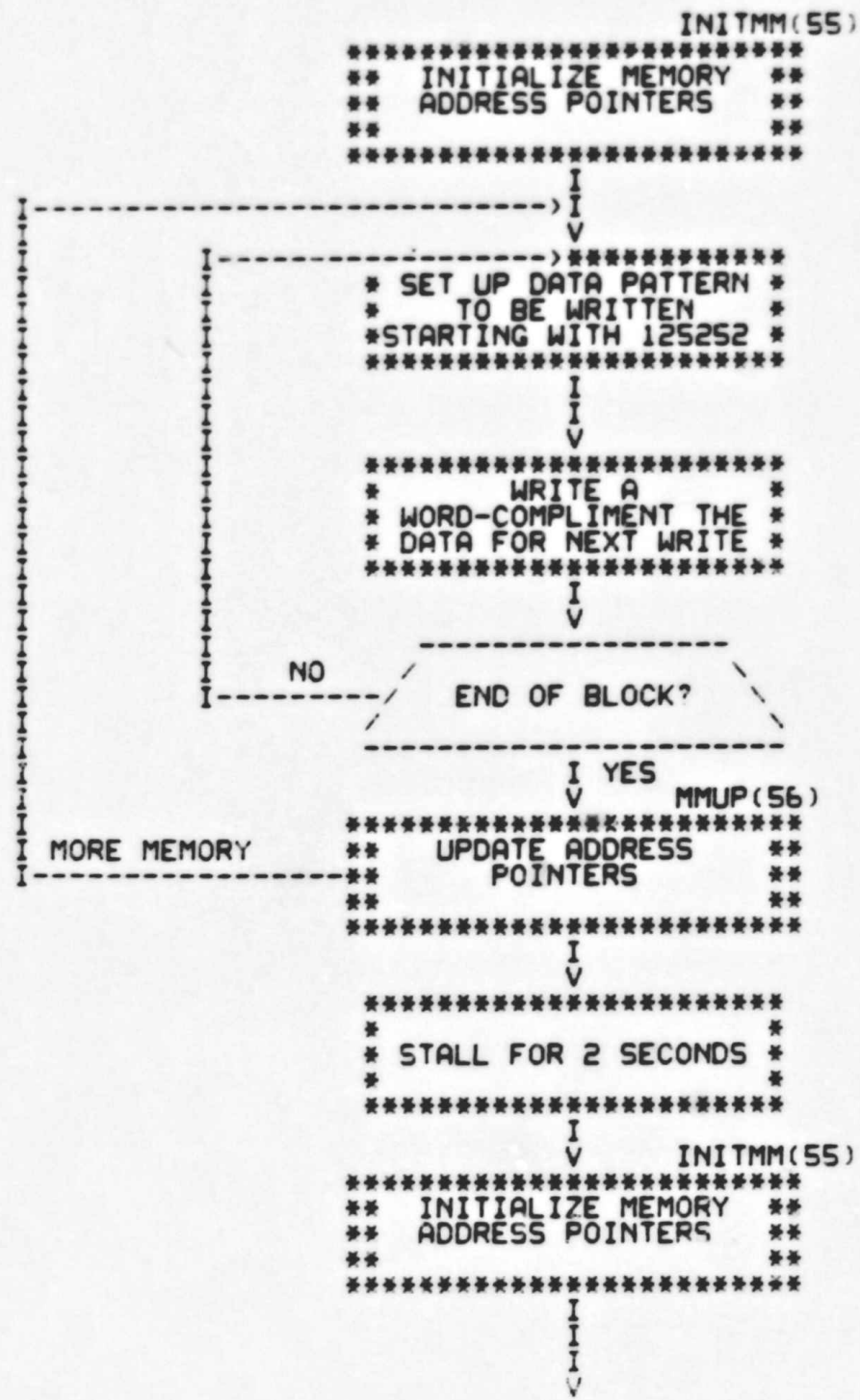


CZQMCFO 0-124K MEM EXER 16K  
TEST 27: MARCHING 1'S AND 0'S

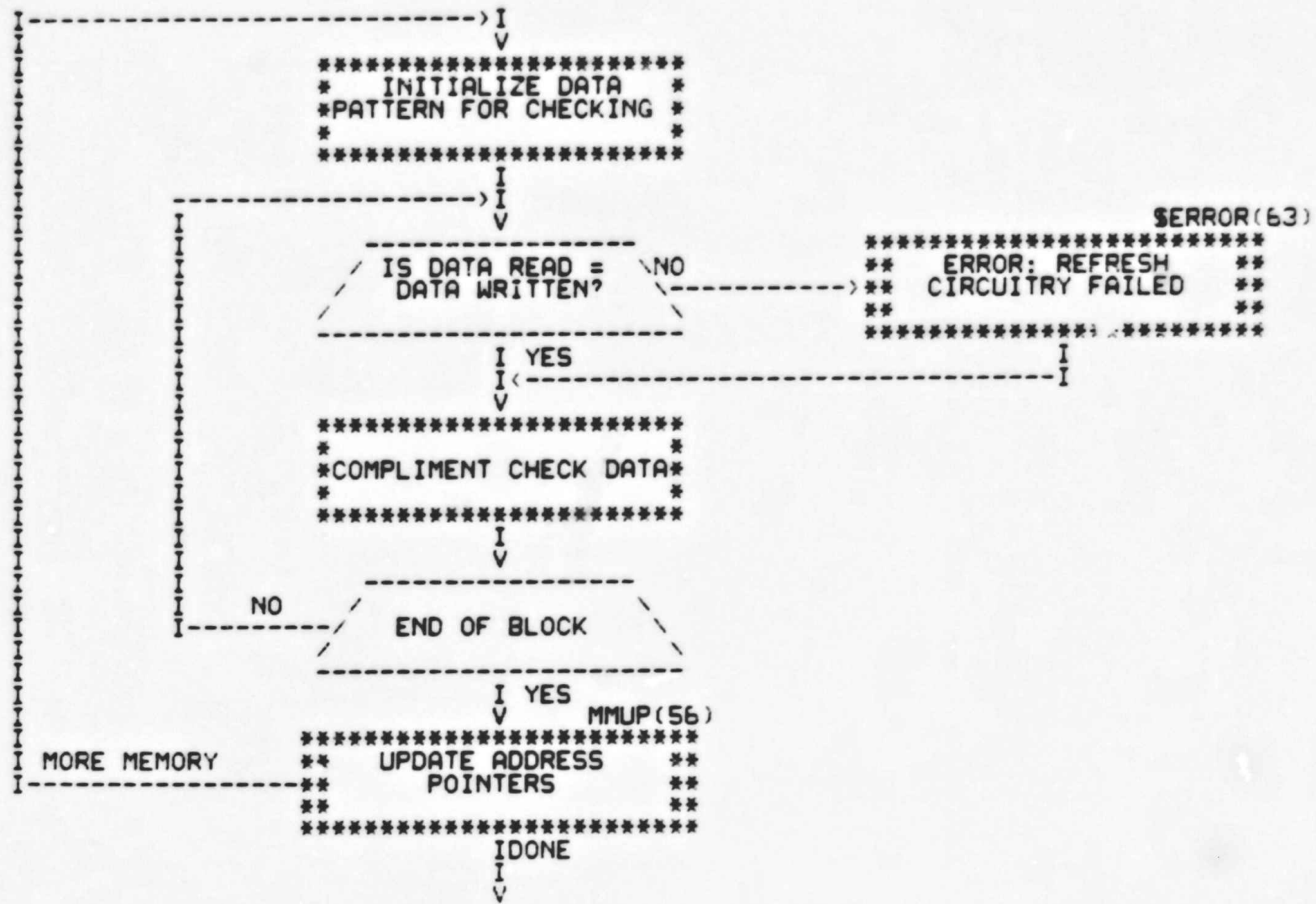


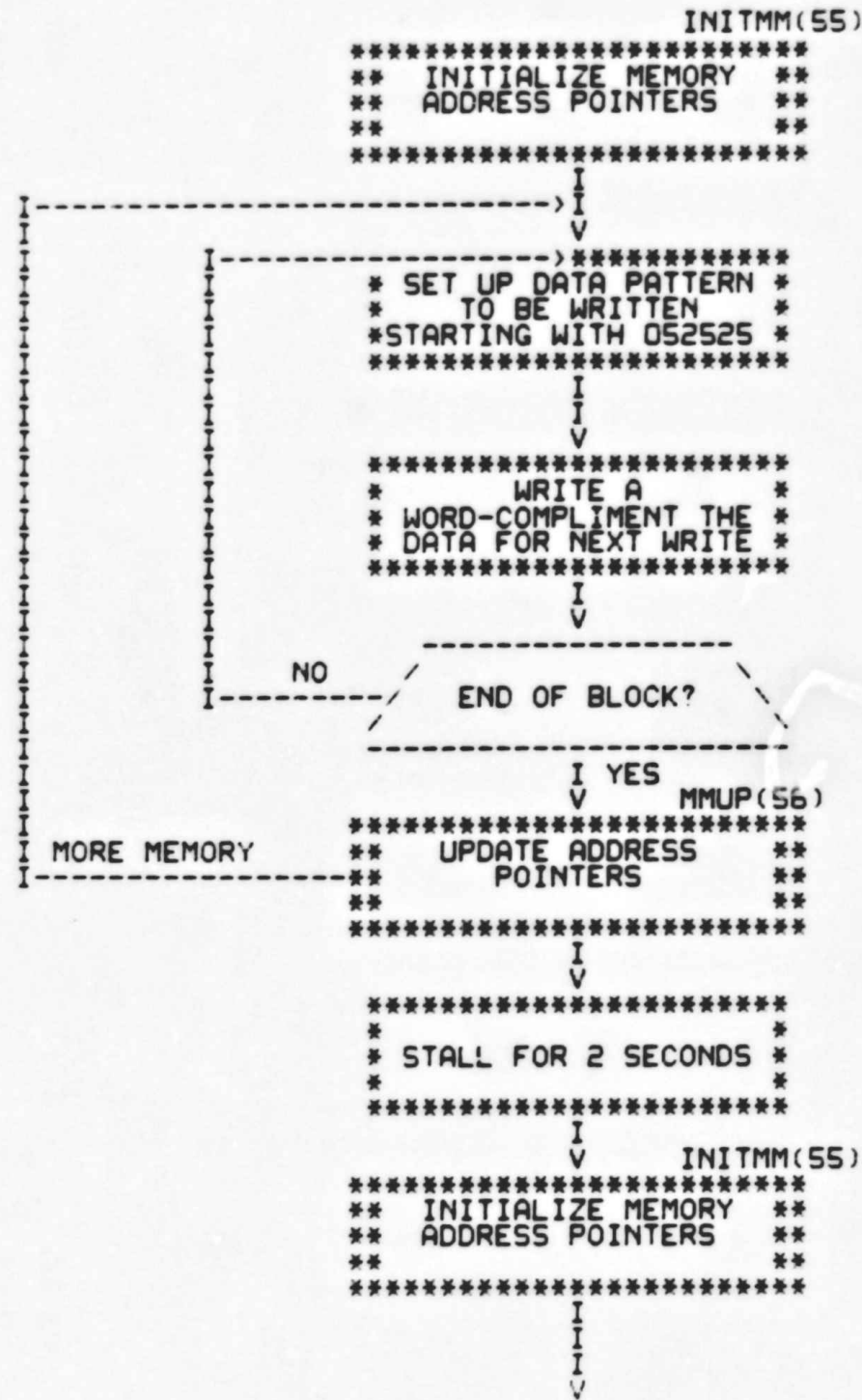
CZQMCFO 0-124K MEM EXER 16K  
TEST 27: MARCHING 1'S AND 0'S

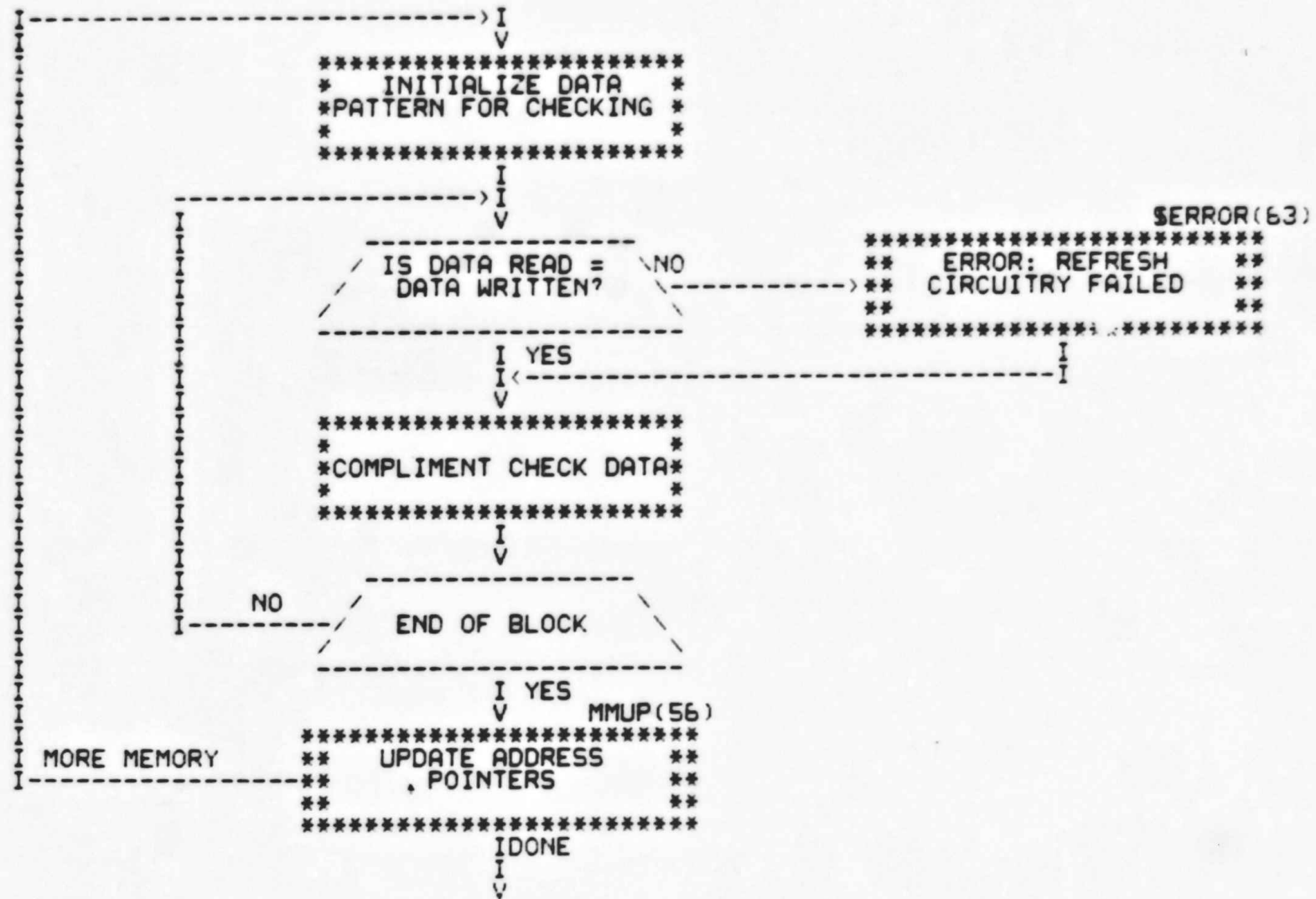
















CZQMCFO 0-124K MEM EXER 16K  
END OF PASS

SEOP  
\*\*\*\*\*  
\*STANDARD 'SYSMAC' END\*  
\* OF PASS ROUTINE \*  
\*  
\*\*\*\*\*

I  
V

\*\*\*\*\*  
/ TYPE END OF PASS /  
/ MESSAGE /  
\*\*\*\*\*

I  
V

\*\*\*\*\*  
\*START1(16)\*  
\*\*\*\*\*

NO

-----  
/ MONITOR PRESENT? /  
-----

YES

\*\*\*\*\*  
\*\*EXIT \*\*  
\*\*\*\*\*









CZQMCFO 0-124K MEM EXER 16K  
RELOCATION SUBROUTINES

```

*****
**RELOC **
*****
      I
      V
RELOC
*****
* MOVE BK BLOCK OF *
* MEMORY FROM SRC TO *
* DST *
*****
      I
      V
-----
DATA OK AFTER MOVE? \ NO
-----
      I YES
      V
*****
TYPE PROGRAM
RELOCATION MESSAGE
*****
      I
      V
*****
**RETURN **
*****

```

```

$ERROR(63)
*****
** ERROR: RELOCATION **
** FAILURE **
*****
      I
      V
*****
**HALT **
*****

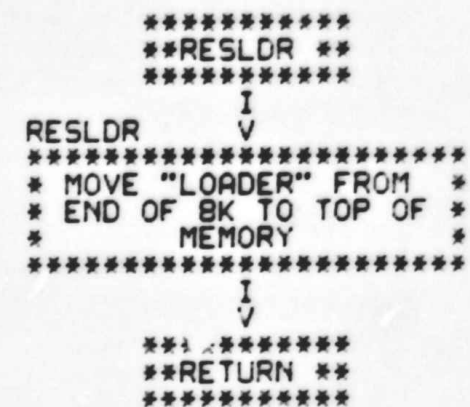
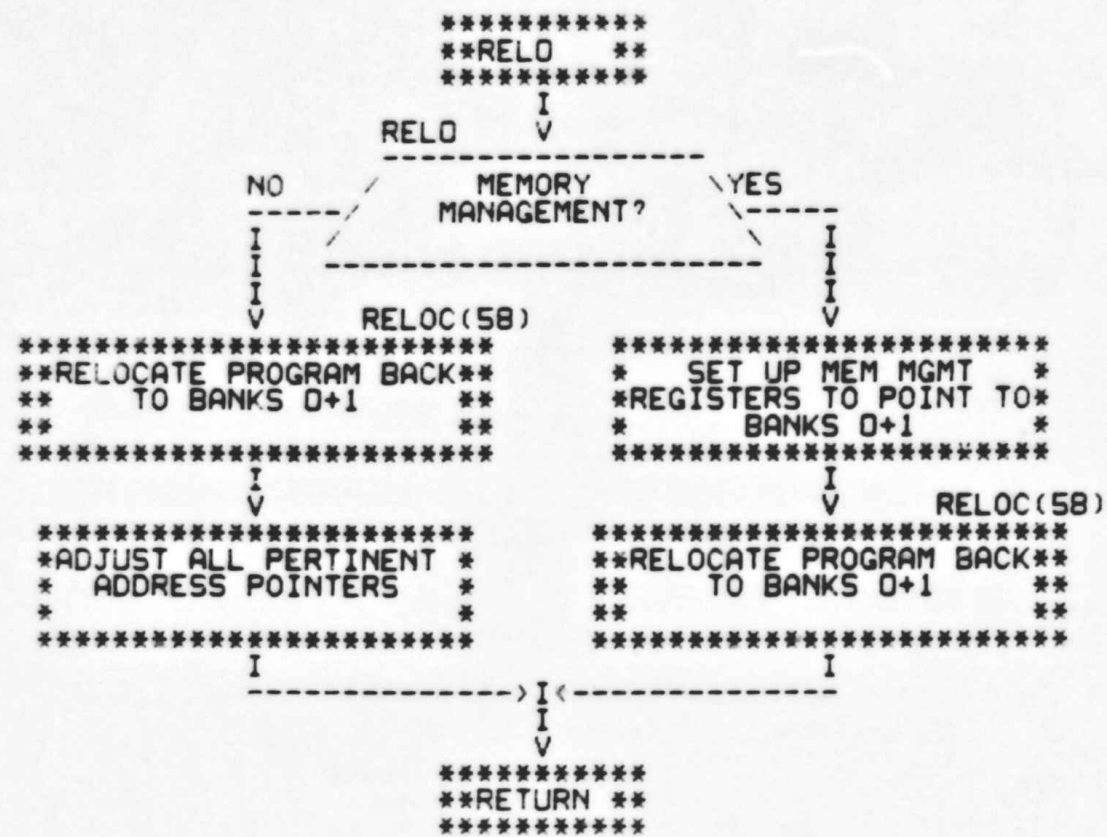
```

```

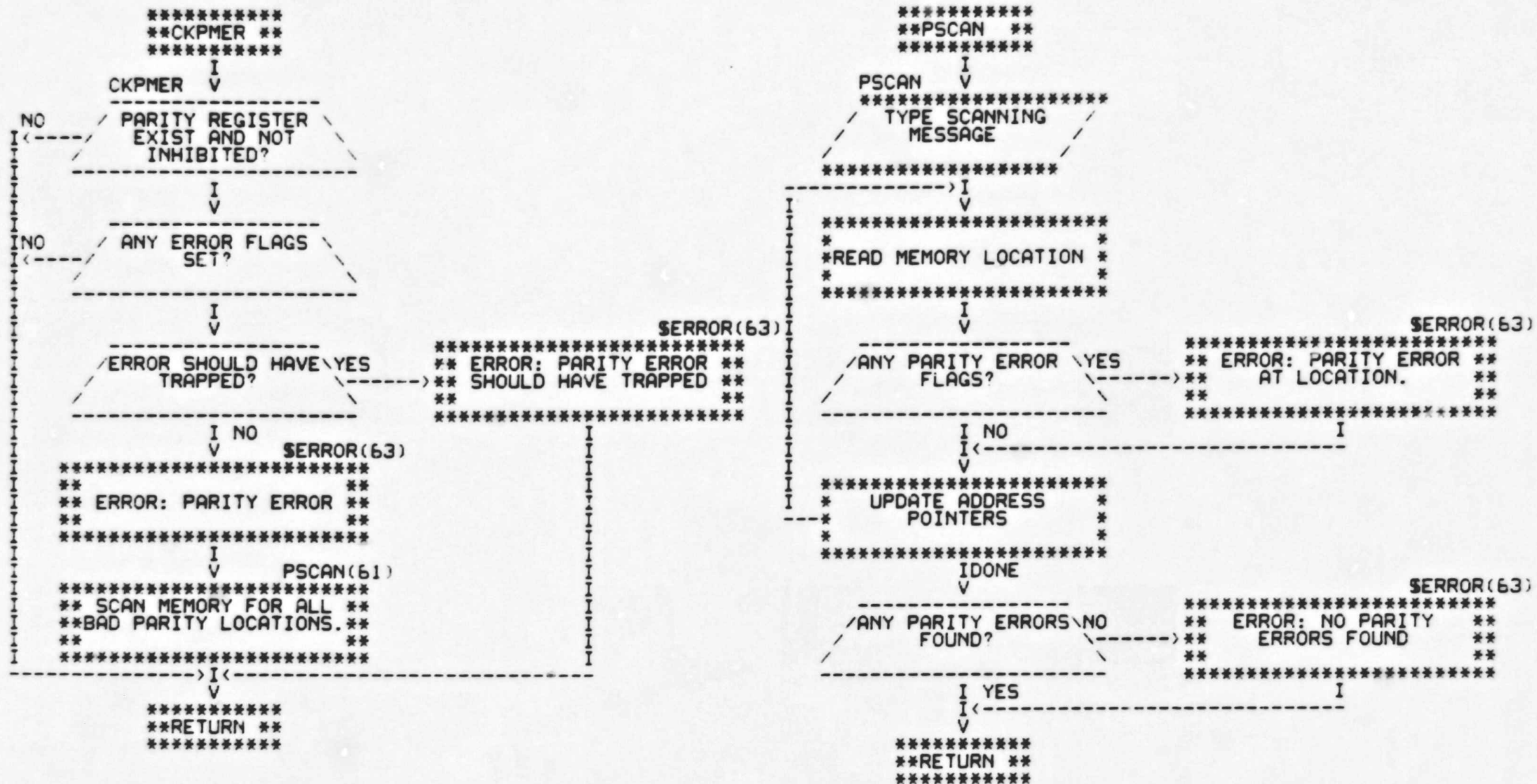
*****
**RELTOP **
*****
      I
      V
RELTOP
-----
MEMORY MANAGEMENT? \ YES
-----
      I NO
      V
*****
* SET UP DESTINATION *
* PART OF 'RELOC' TO *
* POINT TO LAST 2 BANK *
*****
      I
      V
RELOC(58)
*****
** RELOCATE PROGRAM TO **
** LAST 2 BANKS **
*****
      I
      V
*****
*ADJUST ALL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

```











CZQMCFD 0-124K MEM EXER 16K  
SPECIAL PRINTOUT ROUTINES

```

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

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

*****
**SPRNTR **--->
*****

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

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

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

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

```

```

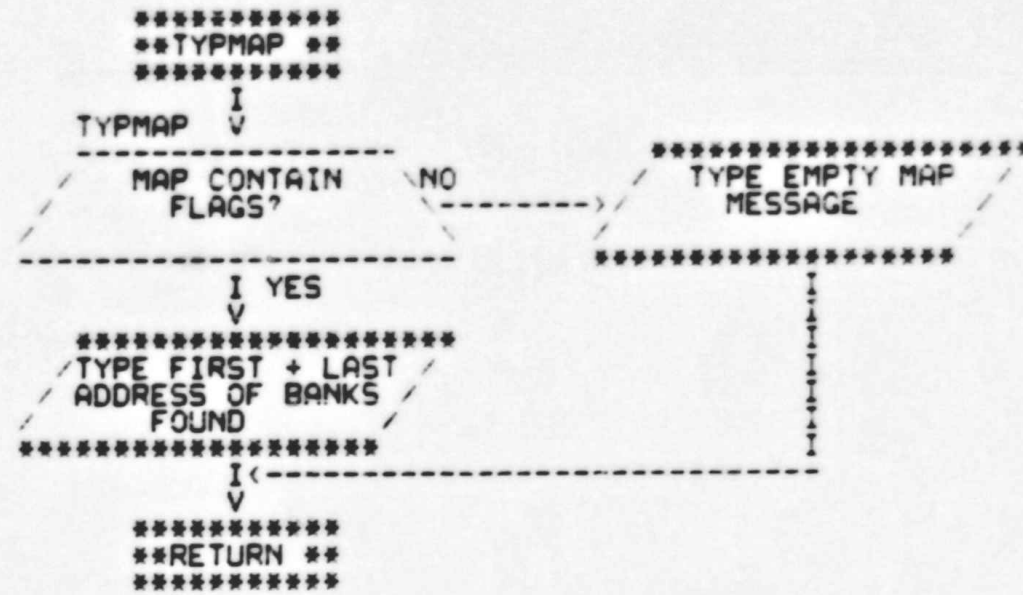
*****
* ROUTINES TO SET UP *
* DATA FOR ERROR *
* TYPEOUTS. *
*****

```

```

*****
**RETURN **
*****

```



```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```

```

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

```







F07

DECFL0 VER 00.07 20-FEB-78 07:58 PAGE 66

SEQ 0083

CZQMCFO 0-124K MEM EXER 16K  
FLOW CHART CROSS REFERENCE LIST

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

```

.TITLE CZQMCFO 0-124K MEMORY EXERCISER, 16K VER
:*COPYRIGHT (C) 1975,1978
:*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-DZGAC-C3), JAN 19, 1977.
:*

```

```

0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
G
H
I
J
K
L
M
N
O
P
Q
R
S
T
U
V
W
X
Y
Z

```

```

.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

```

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

```

\*MISCELLANEOUS DEFINITIONS

```

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

```

\*GENERAL PURPOSE REGISTER DEFINITIONS

```

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

```



100000  
100001  
100002  
100003  
100004  
100005  
100006  
100007  
100008  
100009  
100010  
100011  
100012  
100013  
100014  
100015  
100016  
100017  
100018  
100019  
100020  
100021  
100022  
100023  
100024  
100025  
100026  
100027  
100028  
100029  
100030  
100031  
100032  
100033  
100034  
100035  
100036  
100037  
100038  
100039  
100040  
100041  
100042  
100043  
100044  
100045  
100046  
100047  
100048  
100049  
100050  
100051  
100052  
100053  
100054  
100055  
100056  
100057  
100058  
100059  
100060  
100061  
100062  
100063  
100064  
100065  
100066  
100067  
100068  
100069  
100070  
100071  
100072  
100073  
100074  
100075  
100076  
100077  
100078  
100079  
100080  
100081  
100082  
100083  
100084  
100085  
100086  
100087  
100088  
100089  
100090  
100091  
100092  
100093  
100094  
100095  
100096  
100097  
100098  
100099  
100100

000007  
  
000000  
000040  
000100  
000140  
000200  
000240  
000300  
000340  
  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002  
000001  
  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002

PC= %7 ;:PROGRAM COUNTER

:\*PRIORITY LEVEL DEFINITIONS

PRO= 0 ;:PRIORITY LEVEL 0  
PR1= 40 ;:PRIORITY LEVEL 1  
PR2= 100 ;:PRIORITY LEVEL 2  
PR3= 140 ;:PRIORITY LEVEL 3  
PR4= 200 ;:PRIORITY LEVEL 4  
PR5= 240 ;:PRIORITY LEVEL 5  
PR6= 300 ;:PRIORITY LEVEL 6  
PR7= 340 ;:PRIORITY LEVEL 7

:\*"SWITCH REGISTER" SWITCH DEFINITIONS

SW15= 100000  
SW14= 40000  
SW13= 20000  
SW12= 10000  
SW11= 4000  
SW10= 2000  
SW09= 1000  
SW08= 400  
SW07= 200  
SW06= 100  
SW05= 40  
SW04= 20  
SW03= 10  
SW02= 4  
SW01= 2  
SW00= 1  
.EQUIV SW09,SW9  
.EQUIV SW08,SW8  
.EQUIV SW07,SW7  
.EQUIV SW06,SW6  
.EQUIV SW05,SW5  
.EQUIV SW04,SW4  
.EQUIV SW03,SW3  
.EQUIV SW02,SW2  
.EQUIV SW01,SW1  
.EQUIV SW00,SW0

:\*DATA BIT DEFINITIONS (BIT00 TO BIT15)

BIT15= 100000  
BIT14= 40000  
BIT13= 20000  
BIT12= 10000  
BIT11= 4000  
BIT10= 2000  
BIT09= 1000  
BIT08= 400  
BIT07= 200  
BIT06= 100  
BIT05= 40  
BIT04= 20  
BIT03= 10  
BIT02= 4  
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      .=174
196      000176      000000      DISPREG: .WORD 0          ;;SOFTWARE DISPLAY REGISTER
197      000176      000000      SWREG: .WORD 0          ;;SOFTWARE SWITCH REGISTER
198
199      000200      000137      002640      .SBTTL STARTING ADDRESS(ES)
200      000204      000167      002436      JMP @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
201      000210      000167      000064      JMP RESTAR      ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
202      000214      000167      000064      JMP RESTOR      ;SELECT VARIOUS PARAMETERS.
203      000220      000167      003376      JMP TIMOUT      ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
204      ;RESTORE LOADERS TO END OF MEMORY AND HALT.
205      ;TYPE OUT MEMORY MAP, BYTE BY BYTE.
206
207      000004      000004      .=ERRVEC
208      000006      025114      .WORD ERRTRP
209      000006      000000      .WORD 0
210
211      .SBTTL ACT11 HOOKS
212
213      ;*****
214      ;HOOKS REQUIRED BY ACT11
215      000046      000010      $SVPC=.          ;SAVE PC
216      000046      000046      .=46
217      000052      014222      $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
218      000052      000052      .=52
219      000052      040000      .WORD BIT14      ;;2)SET LOC.52 TO BIT14
220      000010      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, R0 ;POINT TO FIRST MEM MGMT DDATA REG.
238 000346 012701 000010          MOV #8, R1 ;SET UP COUNTER.
239 000352 012720 077406 1$: MOV #077406, (R0)+ ;MAP FIRST 28K 1-FOR-1.
240 000356 005301          DEC R1 ;COUNT REGESTERS.
241 000360 001374          BNE 1$ ;BR IF MORE REG.
242 000362 012700 172340          MOV #KIPAR0, R0 ;POINT TO FIRST MEM MGMT ADDRESS REG.
243 000366 005020          CLR (R0)+ ;PAR0 MAPPED INTO BANK0.
244 000370 012720 000200          MOV #200, (R0)+ ;PAR1 MAPPED INTO BANK1.
245 000374 012720 000400          MOV #400, (R0)+ ;PAR2 MAPPED INTO BANK2.
246 000400 012720 000600          MOV #600, (R0)+ ;PAR3 MAPPED INTO BANK3.
247 000404 012720 001000          MOV #1000, (R0)+ ;PAR4 MAPPED INTO BANK4.
248 000410 012720 001200          MOV #1200, (R0)+ ;PAR5 MAPPED INTO BANK5.
249 000414 012720 001400          MOV #1400, (R0)+ ;PAR6 MAPPED INTO BANK6.
250 000420 012720 007600          MOV #7600, (R0)+ ;PAR7 MAPPED INTO BANK37.
251 000424 012737 000001 177572          MOV #BIT0, @#SRO ;ENABLE MEM MGMT.
252 000432 005000 2$: CLR R0 ;INIT TEMP PAR REG.
253 000434 016701 000142          MOV PRGMAP, R1 ;GET THE PROGRAM MAP...LO 64K.
254 000440 016702 000140          MOV PRGMAP+2, R2 ;...HI 64K.
255 000444 006202 3$: 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, R0 ;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$: MOV R0, @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
262 000466 000137 000472          JMP @#5$ ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
263 000472 062700 000200 5$: ADD #200, R0 ;KEEP UPDATING TEMP PAR REG.
264 000476 006202          ASR R2 ;SHIFT POINTER...HI
265 000500 006001          ROR R1 ;...LO
266 000502 103373          BCC 5$ ;BR IF TOP BANK NOT YET FOUND.
267 000504 010037 172342          MOV R0, @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
268 000510 000410          BR 20$ ;BR TO RELOCATE SECTION.
269 000512 016700 000062 10$: MOV RELOCF, R0 ;GET RELOCATION FACTOR.
270 000516 062700 001100          ADD #STACK, R0 ;SET UP STACK POINTER.
271 000522 010006          MOV R0, SP ;SET STACK TO RELOCATE PROGRAM.
272 000524 062700 177432          ADD #20$-STACK, R0 ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
273 000530 000110          JMP (R0) ;GO TO "20$" (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	016314	JSR	PC,	RELO			:RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
277	000546	005705		21\$: 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	005316	JMP	START1				:RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
282	000566	004767	016476	22\$: JSR	PC,	RESLDR			:RESTORE THE LOADERS TO THE "TOP" OF MEMORY;
283	000572	000000		HALT					:HALT AFTER RESTORING THE LOADERS.
284	000574	000167	002054	JMP	STARTA				:CONTINUE WILL RESTART THE PROGRAM.
285				:* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED					
286				:* BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF					
287				:* 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.

POWER DOWN AND UP ROUTINES

.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 022544
326 000746 025641
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 RO, -(SP) ;; PUSH RO 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
        JSR @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
        IS: INC $SAVR6 ;; WAIT FOR THE INC
        BNE IS ;; 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.
        SPWRMG: .WORD PWRMSG ;; POWER FAIL MESSAGE POINTER
        MOV (PC)+, (SP) ;; RESTART AT RESTART
        SPWRAD: .WORD RESTART ;; RESTART ADDRESS
        RTI
        $SILLUP: HALT ;; THE POWER UP SEQUENCE WAS STARTED
        BR -.2 ;; BEFORE THE POWER DOWN WAS COMPLETE
        $SAVR6: 0 ;; PUT THE SP HERE

```



```

333      .SBTTL COMMON TAGS
334
335      ;*****
336      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
337      ;*USED IN THE PROGRAM.
338
339      001100      .=1100
340      001100      SCMTAG:      ;; START OF COMMON TAGS
341      001100      000000      .WORD      0      ;; CONTAINS THE TEST NUMBER
342      001102      000      $TSTNM: .BYTE      0      ;; CONTAINS ERROR FLAG
343      001103      000      $ERFLG: .BYTE      0      ;; CONTAINS SUBTEST ITERATION COUNT
344      001104      000000      $ICNT:  .WORD      0      ;; CONTAINS SCOPE LOOP ADDRESS
345      001106      000000      $LPADR: .WORD      0      ;; CONTAINS SCOPE RETURN FOR ERRORS
346      001110      000000      $LPERR: .WORD      0      ;; CONTAINS TOTAL ERRORS DETECTED
347      001112      000000      $ERTTL: .WORD      0      ;; CONTAINS ITEM CONTROL E.TE
348      001114      000      $ITEMB: .BYTE      0      ;; CONTAINS MAX. ERRORS PER TEST
349      001115      001      $ERMAX: .BYTE      1      ;; CONTAINS PC OF LAST ERROR INSTRUCTION
350      001116      000000      $ERRPC: .WORD      0      ;; CONTAINS ADDRESS OF 'GOOD' DATA
351      001120      000000      $GDADR: .WORD      0      ;; CONTAINS ADDRESS OF 'BAD' DATA
352      001122      000000      $BDADR: .WORD      0      ;; CONTAINS 'GOOD' DATA
353      001124      000000      $GDDAT: .WORD      0      ;; CONTAINS 'BAD' DATA
354      001126      000000      $BDDAT: .WORD      0      ;; RESERVED--NOT TO BE USED
355      001130      000000      .WORD      0
356      001132      000000      .WORD      0
357      001134      000      $AUTOB: .BYTE      0      ;; AUTOMATIC MODE INDICATOR
358      001135      000      $INTAG: .BYTE      0      ;; INTERRUPT MODE INDICATOR
359      001136      000000      .WORD      0
360      001140      177570      $SWR:   .WORD      DSWR      ;; ADDRESS OF SWITCH REGISTER
361      001142      177570      $DISPLAY: .WORD      DDISP      ;; ADDRESS OF DISPLAY REGISTER
362      001144      177560      $TKS:   177560      ;; TTY KBD STATUS
363      001146      177562      $TKB:   177562      ;; TTY KBD BUFFER
364      001150      177564      $STPS: 177564      ;; TTY PRINTER STATUS REG. ADDRESS
365      001152      177566      $STPB: 177566      ;; TTY PRINTER BUFFER REG. ADDRESS
366      001154      000      $NULL: .BYTE      0      ;; CONTAINS NULL CHARACTER FOR FILLS
367      001155      002      $FILLS: .BYTE      2      ;; CONTAINS # OF FILLER CHARACTERS REQUIRED
368      001156      012      $FILLC: .BYTE      12      ;; INSERT FILL CHARS. AFTER A "LINE FEED"
369      001157      000      $STPFLG: .BYTE      0      ;; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
370      001160      000000      $STMP0: .WORD      0      ;; USER DEFINED
371      001162      000000      $STMP1: .WORD      0      ;; USER DEFINED
372      001164      000000      $STMP2: .WORD      0      ;; USER DEFINED
373      001166      000000      $STMP3: .WORD      0      ;; USER DEFINED
374      001170      000000      $TIMES: 0      ;; MAX. NUMBER OF ITERATIONS
375      001172      000000      $ESCAPE: 0      ;; ESCAPE ON ERROR ADDRESS
376      001174      177607      $BELL:  .ASCIZ <207><377><377>      ;; CODE FOR BELL
377      001200      077      $QUES:  .ASCIZ /?/      ;; QUESTION MARK
378      001201      015      $CRLF:  .ASCIZ <15>      ;; CARRIAGE RETURN
379      001202      000012      $LF:   .ASCIZ <12>      ;; LINE FEED
380      ;*****
381      .SBTTL APT MAILBOX-ETABLE
382
383      ;*****
384      .EVEN
385      001204      $MAIL:      ;; APT MAILBOX
386      001204      000000      $MSGTY: .WORD      MSGTY      ;; MESSAGE TYPE CODE
387      001206      000000      $FATAL: .WORD      AFATAL      ;; FATAL ERROR NUMBER
388      001210      000000      $TESTN: .WORD      ATESTN      ;; TEST NUMBER

```

389	001212	000000	\$PASS: .WORD	APASS	:: PASS COUNT
390	001214	000000	\$DEVCT: .WORD	ADEVCT	:: DEVICE COUNT
391	001216	000000	\$UNIT: .WORD	AUNIT	:: I/O UNIT NUMBER
392	001220	000000	\$MSGAD: .WORD	AMSGAD	:: MESSAGE ADDRESS
393	001222	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
394	001224		\$ETABLE:		:: APT ENVIRONMENT TABLE
395	001224	000	\$ENV: .BYTE	AENV	:: ENVIRONMENT BYTE
396	001225	000	\$ENVM: .BYTE	AENVM	:: ENVIRONMENT MODE BITS
397	001226	000000	\$SWREG: .WORD	ASWREG	:: APT SWITCH REGISTER
398	001230	000000	\$USWR: .WORD	AUSWR	:: USER SWITCHES
399	001232	000000	\$CPUOP: .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	\$MTYP1: .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	\$MTYP2: .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	\$MTYP3: .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	\$MTYP4: .BYTE	AMTYP4	:: MEM. TYPE, BLK#4
422	001252	000000	\$MADR4: .WORD	AMADR4	:: MEM. LAST ADDRESS, BLK#4
423	001254	000000	\$VECT1: .WORD	AVECT1	:: INTERRUPT VECTOR#1, BUS PRIORITY#1
424	001256	000000	\$VECT2: .WORD	AVECT2	:: INTERRUPT VECTOR#2, BUS PRIORITY#2
425	001260	000000	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
426	001262	000000	\$DEVN: .WORD	ADEVN	:: DEVICE MAP
427	001264	000000	\$CDW1: .WORD	ACDW1	:: CONTROLLER DESCRIPTION WORD#1
428	001266	000000	\$CDW2: .WORD	ACDW2	:: CONTROLLER DESCRIPTION WORD#2
429	001270	000000	\$DDW0: .WORD	ADDW0	:: DEVICE DESCRIPTOR WORD#0
430	001272	000000	\$DDW1: .WORD	ADDW1	:: DEVICE DESCRIPTOR WORD#1
431	001274	000000	\$DDW2: .WORD	ADDW2	:: DEVICE DESCRIPTOR WORD#2
432	001276	000000	\$DDW3: .WORD	ADDW3	:: DEVICE DESCRIPTOR WORD#3
433	001300	000000	\$DDW4: .WORD	ADDW4	:: DEVICE DESCRIPTOR WORD#4
434	001302	000000	\$DDW5: .WORD	ADDW5	:: DEVICE DESCRIPTOR WORD#5
435	001304	000000	\$DDW6: .WORD	ADDW6	:: DEVICE DESCRIPTOR WORD#6
436	001306	000000	\$DDW7: .WORD	ADDW7	:: DEVICE DESCRIPTOR WORD#7
437	001310	000000	\$DDW8: .WORD	ADDW8	:: DEVICE DESCRIPTOR WORD#8
438	001312	000000	\$DDW9: .WORD	ADDW9	:: DEVICE DESCRIPTOR WORD#9
439	001314	000000	\$DDW10: .WORD	ADDW10	:: DEVICE DESCRIPTOR WORD#10
440	001316	000000	\$DDW11: .WORD	ADDW11	:: DEVICE DESCRIPTOR WORD#11
441	001320	000000	\$DDW12: .WORD	ADDW12	:: DEVICE DESCRIPTOR WORD#12
442	001322	000000	\$DDW13: .WORD	ADDW13	:: DEVICE DESCRIPTOR WORD#13
443	001324	000000	\$DDW14: .WORD	ADDW14	:: DEVICE DESCRIPTOR WORD#14
444	001326	000000	\$DDW15: .WORD	ADDW15	:: DEVICE DESCRIPTOR WORD#15







```

501      ;:*****
502      ;*THE FOLLOWING TAGS ARE USER DEFINED
503      ;:*****
504      001514 000000 $VERPC: .WORD 0 ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE (SERTYP).
505      001516 070032 RESRVD: .WORD 070032 ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
506      ;NOTE: FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
507      001520 000000 LMAD: .WORD 0 ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
508      001522 000000 LDDISP: .WORD 0 ;CONTAINS DISPLAY REGISTER IMAGE
509      001524 000000 MEMMAP: .WORD 0 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
510      001524 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
511      001526 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
512      001530 000000 TSTMAP: .WORD 0 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
513      001530 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
514      001532 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
515      001534 000000 SAVTST: .WORD 0 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
516      ;TEST EACH BANK ONCE.
517      001534 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
518      001536 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
519      001540 000000 PMEMAP: .WORD 0 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
520      001540 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
521      001542 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
522      001544 000000 BITPT: .WORD 0 ;POINTER TO CURRENT 4K BANK OF MEMORY
523      001544 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
524      001546 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
525      001550 000000 TMPPT: .WORD 0 ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
526      001550 000000 .WORD 0 ;FIRST WORD CONTAINS LOW (0-64K) MAP
527      001552 000000 .WORD 0 ;SECOND WORD CONTAINS HIGH (64-128K) MAP
528      001554 000000 MMORE: .WORD 0 ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
529      ;SET UP BY "INITMM" AND "INITDN" ROUTINES.
530      ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
531      001556 000 SELFFLG: .BYTE 0 ;OPERATOR SELECTED PARAMETERS FLAG. (SA=204)
532      001557 000 FLAGBK: .BYTE 0 ;BK BLOCK INDICATOR. USED IN "INITMM" AND "MMUP".
533      001560 000 OEFLG: .BYTE 0 ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
534      001562 000 FSTADR: .WORD 0 ;FIRST VIRTUAL ADDRESS TO BE TESTED.
535      001562 000000 .WORD 0 ;FIRST ADDRESS IS USER SELECTABLE.
536      001564 000000 TMPFAD: .WORD 0 ;ADJUSTED FIRST ADDRESS.
537      001566 000000 FADMSK: .WORD 0 ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
538      001570 000000 000000 FADMAP: .WORD 0,0 ;TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
539      001574 000000 LSTADR: .WORD 0 ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
540      001576 000000 .WORD 0 ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
541      001600 000000 TEMPLAD: .WORD 0 ;LAST ADDRESS IS USER SELECTABLE.
542      001600 000000 LADMSK: .WORD 0 ;ADJUSTED LAST ADDRESS.
543      001602 000000 000000 LADMAP: .WORD 0,0 ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
544      001606 000000 BLKMSK: .WORD 0 ;TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
545      001610 000000 .CONST: .WORD 0 ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
546      001612 000004 WWP: .WORD 4 ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
547      001614 000000 TEMP: .WORD 0 ;USER SELECTABLE CONSTANT DATA.
548      001616 000000 CASFLG: .WORD 0 ;WRITE WRONG PARITY COMMAND
549      001620 177746 CASREG: .WORD 177746 ;TEMPORARY STORAGE
550      ;CACHE PRESENT FLAG
551      ;CACHE CONTROL REGISTER
552      ;:*****
553      ;* RELATIVE ADDRESSING TABLE.
554      ;* THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
555
556

```

```

557
558
559 001622
560 001622 001100
561 001624 001516
562 001626 002076
563 001630 002276
564 001632 012052
565 001634 002050
566 001636 017430
567 001640 002340
568 001642 000010
569 001644 014004
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,$BDDAT,0
DT2: $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
DT12: $VERPC,$ERRPC,$GDADR,$GDDAT,0
DT14: $VERPC,$ERRPC,$TMP0,$GDADR,0
DT15: $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
DT21: $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
DT23: $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
DT24: $VERPC,$ERRPC,$BDADR,0
DT25: $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
DT26: $VERPC,$ERRPC,$TMP0,$TMP1,0
DT30: $TMP0,$TMP1,$GDADR,$BDDAT,0
DT31: $TMP3,0
.WORD -1 ;TABLE TERMINATOR.

```

```

.SBTTL MEMORY PARITY PATTERNS TABLE
;*****
;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
;*****
MPPATS: 125325 ;EVEN,ODD
        152652 ;ODD,EVEN
        052452 ;EVEN,ODD
        025125 ;ODD,EVEN
        102070 ;EVEN,EVEN
        072527 ;ODD,ODD
        177777 ;EVEN,EVEN

```







669 002210 000000  
 670 002212 000000  
 671 002214 000000  
 672 002216 172125  
 673 002220 000000  
 674 002222 000000  
 675 002224 000000  
 676 002226 172127  
 677 002230 000000  
 678 002232 000000  
 679 002234 000000  
 680 002236 172131  
 681 002240 000000  
 682 002242 000000  
 683 002244 000000  
 684 002246 172133  
 685 002250 000000  
 686 002252 000000  
 687 002254 000000  
 688 002256 172135  
 689 002260 000000  
 690 002262 000000  
 691 002264 000000  
 692 002266 172137  
 693 002270 000000  
 694 002272 000000  
 695 002274 000000  
 696  
 697 002276 000021  
 698  
 699

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

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

:THIS IS THE END OF THE TABLE !  
 MPRX: .BLKW 17.

:TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.  
 : (THE EXTRA WORD IS FOR A TERMINATOR.)

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:

Index	Code	Pointer	Item	Pointer	Description
700					
701					
702					
703					
704					
705					
706					
707					
708					
709					
710					
711					
712					
713					
714	002340				
715			ITEM 1		
716	002340	027010	DM1		:PARITY REGISTER DATA ERROR.
717	002342	030367	DH1		:PC REG, S/B WAS
718	002344	001646	DT1		:\$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
719	002346	030734	DF1		:16, 18, 16, 16
720			ITEM 2		
721	002350	027044	DM2		:ADDRESS TEST ERROR(TST1-5).
722	002352	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
723	002354	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
724	002356	030740	DF2		:16, 18, 18, 16, 16
725			ITEM 3		
726	002360	027044	DM2		:ADDRESS TEST ERROR(TST1-5).
727	002362	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
728	002364	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
729	002366	030745	DF3		:16, 18, 18, 8, 8
730			ITEM 4		
731	002370	027100	DM4		:CONSTANT DATA ERROR(TST6-10).
732	002372	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
733	002374	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
734	002376	030740	DF2		:16, 16, 18, 16, 16
735			ITEM 5		
736	002400	027136	DM5		:ROTATING BIT ERROR(TST11-12).
737	002402	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
738	002404	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
739	002406	030740	DF2		:16, 18, 18, 16, 16
740			ITEM 6		
741	002410	027174	DM6		:MOS REFRESH TEST ERROR (TST30-31).
742	002412	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
743	002414	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
744	002416	030740	DF2		:16, 18, 18, 16, 16
745			ITEM 7		
746	002420	027240	DM7		:3 XOR 9 PATTERN ERROR(TST13-16).
747	002422	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
748	002424	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
749	002426	030740	DF2		:16, 18, 18, 16, 16
750			ITEM 10		
751	002430	027301	DM10		:MARCHING 1'S AND 0'S ERROR(TST27).
752	002432	030406	DH2		:V/PC, P/PC, MA, S/B, WAS
753	002434	001660	DT2		:\$VERPC, \$ERRPC, \$GDADR, \$GDDAT, \$BDDAT
754	002436	030740	DF2		:16, 18, 18, 16, 16
755			ITEM 11		

756	002440	027345	DM11	: PARITY MEMORY ADDRESS ERROR(TST17).
757	002442	030406	DH2	: V/PC P/PC MA S/B WAS
758	002444	001660	DT2	: \$VERPC \$EARPC \$GDADR \$GDDAT \$BDDAT
759	002446	030745	DF3	: 16, 18, 18, 8, 8
760			: * ITEM 12	
761	002450	027411	DM12	: DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
762	002452	030433	DH12	: V/PC P/PC MA S/B
763	002454	001674	DT12	: \$VERPC \$EARPC \$GDADR \$GDDAT
764	002456	030745	DF3	: 16, 18, 18, 8
765			: * ITEM 13	
766	002460	027465	DM13	: WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
767	002462	030433	DH12	: V/PC P/PC MA S/B
768	002464	001674	DT12	: \$VERPC \$EARPC \$GDADR \$GDDAT
769	002466	030745	DF3	: 16, 18, 18, 8
770			: * ITEM 14	
771	002470	027555	DM14	: PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
772	002472	030454	DH14	: V/PC P/PC REG MA
773	002474	001706	DT14	: \$VERPC \$EARPC \$TMPO \$GDADR
774	002476	030752	DF14	: 16, 18, 18, 18
775			: * ITEM 15	
776	002500	027010	DM1	: PARITY REGISTER DATA ERROR.
777	002502	030475	DH15	: V/PC P/PC MAUT REG S/B WAS
778	002504	001720	DT15	: \$VERPC \$EARPC \$GDADR \$TMPO \$GDDAT \$BDDAT
779	002506	030752	DF14	: 16, 18, 18, 18, 16, 16
780			: * ITEM 16	
781	002510	027654	DM16	: MORE THAN ONE REGISTER INDICATED PARITY ERROR.
782	002512	030454	DH14	: V/PC P/PC REG MA
783	002514	001706	DT14	: \$VERPC \$EARPC \$TMPO \$GDADR
784	002516	030752	DF14	: 16, 18, 18, 18
785			: * ITEM 17	
786	002520	027733	DM17	: DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
787				: TRAPPED(TST21).
788	002522	030406	DH2	: V/PC P/PC MA S/B WAS
789	002524	001660	DT2	: \$VERPC \$EARPC \$GDADR \$GDDAT \$BDDAT
790	002526	030745	DF3	: 16, 18, 18, 8, 8
791			: * ITEM 20	
792	002530	030031	DM20	: RANDOM DATA ERROR(TST20).
793	002532	030406	DH2	: V/PC P/PC MA S/B WAS
794	002534	001660	DT2	: \$VERPC \$EARPC \$GDADR \$GDDAT \$BDDAT
795	002536	030740	DF2	: 16, 18, 18, 16, 16
796			: * ITEM 21	
797	002540	030063	DM21	: INSTRUCTION EXECUTION ERROR(TST21-26).
798	002542	030530	DH21	: V/PC P/PC IUT MA S/B WAS
799	002544	001736	DT21	: \$VERPC \$EARPC \$TMPO \$GDADR \$GDDAT \$BDDAT
800	002546	030760	DF21	: 16, 18, 16, 18, 16, 16
801			: * ITEM 22	
802			: * ITEM 23	
803	002550	030132	DM23	: PROGRAM CODE CHANGED WHEN RELOCATED.
804	002552	030561	DH23	: V/PC P/PC SRC MA DST MA S/B WAS
805	002554	001754	DT23	: \$VERPC \$EARPC \$GDADR \$BDADR \$GDDAT \$BDDAT
806	002556	030752	DF14	: 16, 18, 18, 18, 16, 16
807			: * ITEM 24	
808	002560	030177	DM24	: TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
809	002562	030621	DH24	: V/PC P/PC TRP/PC
810	002564	001772	DT24	: \$VERPC \$EARPC \$BDADR
811	002566	030752	DF14	: 16, 18, 18



# JOB

812			;* ITEM 25	
813	002570	030253	DM25	: TRAPPED TO 114.
814	002572	030642	DH25	: V/PC, P/PC, TRP/PC, REG, WAS
815	002574	002002	DT25	: \$VERPC, \$ERRPC, \$BDADR, \$TMPO, \$TMP1
816	002576	030752	DF14	: 16, 18, 18, 18, 16
817			;* ITEM 26	
818	002600	030273	DM26	: FAILED TO TRAP.
819	002602	030673	DH26	: V/PC, P/PC, REG, WAS
820	002604	002016	DT26	: \$VERPC, \$ERRPC, \$TMPO, \$TMP1
821	002606	030740	DF2	: 16, 18, 18, 16
822			;* ITEM 27	
823	002610	030313	DM27	: (ACTION ENABLE WASN'T SET).
824	002612	030673	DH26	: V/PC, P/PC, REG, WAS
825	002614	002016	DT26	: \$VERPC, \$ERRPC, \$TMPO, \$BDADR
826	002616	030740	DF2	: 16, 18, 18, 16
827			;* ITEM 30	
828	002620	000000	0	: NO MESSAGE.
829	002622	030715	DH30	: REG, WAS, MA, WAS
830	002624	002030	DT30	: \$TMPO, \$TMP1, \$GDADR, \$BDADR
831	002626	030766	DF30	: 18, 16, 18, 8
832			;* ITEM 31	
833	002630	030347	DM31	: TRAPPED TO 4
834	002632	000000	0	: NO HEADER
835	002634	002042	DT31	: \$TMP3
836	002636	030766	DF30	: 18

K08

837

.SBTTL START: SETUP AND MAP MEMORY

```

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

```

844 002640 105067 176712 START: CLR B SELFLG ; CLEAR SELECT PARAMETER FLAG.
845 002644 000403 BR STARTA ; GO DO SETUP AND MEMORY MAP.
846 002646 112767 177777 176702 SELECT: MOV B #-1, SELFLG ; SET THE SELECT PARAMETERS FLAG.
847 002654 STARTA:
848 .SBTTL INITIALIZE THE COMMON TAGS
849 ;; CLEAR THE COMMON TAGS ($CMTAG) AREA
850 002654 012706 001100 MOV # $CMTAG, R6 ; FIRST LOCATION TO BE CLEARED
851 002660 005026 CLR (R6)+ ; CLEAR MEMORY LOCATION
852 002662 022706 001140 CMP # SWR, R6 ;; DONE?
853 002666 001374 BNE .-6 ; LOOP BACK IF NO
854 002670 012706 001100 MOV # STACK, SP ; SETUP THE STACK POINTER
855 ;; INITIALIZE A FEW VECTORS
856 002674 012737 000610 000024 MOV # SPWRDN, # PWRVEC ; POWER FAILURE VECTOR
857 002702 012737 000340 000026 MOV # 340, # PWRVEC+2 ; LEVEL 7
858 002710 016767 011242 011232 MOV # SENDCT, # SEOPCT ; SETUP END-OF-PROGRAM COUNTER
859 ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
860 ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
861 002716 013746 000004 MOV # ERRVEC, -(SP) ; SAVE ERROR VECTOR
862 002722 012737 002756 000004 MOV # 64$, # ERRVEC ; SET UP ERROR VECTOR
863 002730 012767 177570 176202 MOV # DSWR, SWR ; SETUP FOR A HARDWARE SWITCH REGISTER
864 002736 012767 177570 176176 MOV # DDISP, DISPLAY ; AND A HARDWARE DISPLAY REGISTER
865 002744 022777 177777 176166 CMP #-1, # SWR ; TRY TO REFERENCE HARDWARE SWR
866 002752 001012 BNE 66$ ; BRANCH IF NO TIMEOUT TRAP OCCURRED
867 ; AND THE HARDWARE SWR IS NOT = -1
868 002754 000403 BR 65$ ; BRANCH IF NO TIMEOUT
869 002756 012716 002764 64$: MOV # 65$, (SP) ; SET UP FOR TRAP RETURN
870 002762 000002 RTI
871 002764 012767 000176 176146 65$: MOV # SWREG, SWR ; POINT TO SOFTWARE SWR
872 002772 012767 000174 176142 MOV # DISPRG, DISPLAY
873 003000 012637 000004 66$: MOV (SP)+, # ERRVEC ; RESTORE ERROR VECTOR
874
875 003004 005067 176202 CLR $PASS ; CLEAR PASS COUNT
876 003010 132767 000200 176207 BITB # APTSIZE, $ENVM ; TEST USER SIZE UNDER APT
877 003016 001403 BEQ 67$ ; YES, USE NON-APT SWITCH
878 003020 012767 001226 176112 MOV # $SWREG, SWR ; NO, USE APT SWITCH REGISTER
879 003026 67$:
880 003026 005067 176470 CLR LDDISP ; CLEAR DISPLAY REGISTER STORAGE LOCN
881 003032 005077 176104 CLR # DISPLAY ; CLEAR DISPLAY REGISTER
882 .SBTTL TYPE PROGRAM NAME
883 ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
884 003036 005227 177777 INC #-1 ; FIRST TIME?
885 003042 001040 BNE 68$ ; BRANCH IF NO
886 003044 022737 014222 000042 CMP # $ENDAD, # 42 ; ACT-11?
887 003052 001434 BEQ 68$ ; BRANCH IF YES
888 003054 004567 020432 JSR R5, $PRINT ; GO PRINT OUT THE FOLLOWING MESSAGE.
889 003060 003132 .WORD 69$ ; ADDRESS OF MESSAGE TO BE TYPED
890 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
891 003062 005737 000042 TST # 42 ; ARE WE RUNNING UNDER XXDP/ACT?
892 003066 001015 BNE 70$ ; BRANCH IF YES
    
```





M08

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

949
950
951
952 003324 012706 001100
953 003330 012700 001524
954 003334 012701 001526
955 003340 005010
956 003342 005011
957 003344 005002
958 003346 012703 000001
959 003352 005004
960 003354 005067 175606
961 003360 004567 020126
962 003364 025421
963
964 003366 012737 003502 000004
965 003374 011222
966 003376 032702 017777
967 003402 001374
968 003404 050310
969 003406 050411
970 003410 010267 175550
971 003414 005367 175544
972 003420 005767 175162
973 003424 001432
974 003426 042767 160000 175530
975 003434 013705 172344
976 003440 005067 175522
977 003444 006305
978 003446 006305
979 003450 006305
980 003452 006305
981 003454 006305
982 003456 006167 175504
983 003462 006305
984 003464 006167 175476
985 003470 060567 175470
986 003474 005567 175466
987 003500 000404
988
989
990 003502 022626
991 003504 052702 017777
992 003510 005202
993 003512 005767 175070
994 003516 001411
995 003520 062737 000200 172344
996 003526 012702 040000
997 003532 006303
998 003534 006104
999 003536 100316
1000 003540 000402
1001
1002 003542 106303
1003 003544 100313
1004 003546 012737 025114 000004

```

```

R4 = BANK POINTER... HI 64K.
RS = SCRATCH REGISTER.
*****
MAPMEM: MOV #STACK, SP ; RESET THE STACK
MOV #MEMMAP, R0 ; SET UP MEMORY MAP POINTER...LO 64K.
MOV #MEMMAP+2, R1 ; HI 64K.
CLR (R0) ; CLR MEMORY MAP...LO 64K.
CLR (R1) ; ...HI 64K.
CLR R2 ; SET ADDRESS POINTER TO 0
MOV #1, R3 ; SETUP 4K BANK POINTER...LO 64K.
CLR R4 ; ...HI 64K.
CLR STMP3 ; INIT TEMPORARY HIGH ADDRESS BITS.
JSR R5, SPRINT ; GO PRINT OUT THE FOLLOWING MESSAGE.
.WORD MEMMES ; ADDRESS OF MESSAGE TO BE TYPED
"MEMORY MAP:"
MOV #2$, @#ERRVEC ; SET UP TIMEOUT VECTOR
MOV (R2), (R2)+ ; READ+WRITE ALL MEMORY
BIT #MASK4K, R2 ; CHECK FOR 4K BOUNDARY
BNE 1$ ; BRANCH IF MORE IN BANK
BIS R3, (R0) ; SET FLAG FOR BANK...LO 64K.
BIS R4, (R1) ; ...HI 64K.
MOV R2, STMP2 ; SAVE ADDRESS POINTER.
DEC STMP2 ; ADJUST TO LAST ADR, LAST BANK.
TST MMAVA ; CHECK FOR MEM MGMT.
BEQ 3$ ; BR IF NO MEM MGMT.
BIC #160000, STMP2 ; CLEAR BANK BITS ON RELATIVE ADDRESS.
MOV @#KIPAR2, R5 ; SAVE KIPAR2.
CLR STMP3 ; MAKE SURE HI BITS ARE INIT.
ASL R5 ; SHIFT IT 6 PLACES.
ASL R5
ASL R5
ASL R5
ASL R5
ROL STMP3
ASL R5
ROL STMP3
ADD R5, STMP2 ; MAKE LAST ADR PHYSICAL.
ADC STMP3
BR 3$ ; GO TO UPDATE POINTERS.

;* TIMEOUT TRAPS TO HERE
2$: CMP (SP)+, (SP)+ ; RESTORE THE STACK POINTER
BIS #MASK4K, R2 ; LAST ADDRESS OF 4K BANK
INC R2 ; FIRST ADDRESS OF NEXT BANK.
TST MMAVA ; CHECK FOR MEM MGMT
BEQ 4$ ; BRANCH IF NO MEM MGMT
ADD #200, @#KIPAR2 ; UPDATE THIRD PAR
MOV #40000, R2 ; POINT TO START OF THIRD PAR
ASL R3 ; UPDATE LO BANK POINTER.
ROL R4 ; UPDATE HI BANK POINTER
BPL 1$ ; BRANCH IF MORE MEMORY TO MAP.
BR 5$ ; EXIT WHEN DONE.

4$: ASLB R3 ; UPDATE MAP POINTER
BPL 1$ ; BRANCH IF NOT YET DONE
5$: MOV #ERRTRP, @#ERRVEC ; RESET TIMEOUT VECTOR

```

N08

CZQMCFD 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCF.P11 14-FEB-78 08:19

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 22  
 GET VALUE FOR SOFTWARE SWITCH REGISTER

SEQ 0104

1005	003554	004767	014632		JSR	PC,	TYPMAP	:GO TYPE THE MAP.
1006	003560	004567	017726		JSR	RS,	SPRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1007	003564	001201			.WORD	SCRLF		:ADDRESS OF MESSAGE TO BE TYPED
1008	003566	011067	175742		MOV	(R0),	SAVTST	:SET UP TEST MAP...LO 64K.
1009	003572	011167	175740		MOV	(R1),	SAVTST+2	:...HI 64K.
1010	003576	011000			MOV	(R0),	RO	:GET LOW MEM MAP
1011	003600	042700	177760		BIC	#177760, R0		:MASK ALL BUT BOTTOM 4 BANKS
1012	003604	020027	000017		CMF	R0	#17	:CHECK THAT BOTTOM 16K IS ALL THERE!
1013	003610	001530			BEQ	GMPR		:BRANCH IF BOTTOM 16K EXISTS
1014	003612	004567	017674		JSR	RS,	SPRINT	:GO PRINT OUT THE FOLLOWING MESSAGE.
1015	003616	025524			.WORD	INSUFF		:ADDRESS OF MESSAGE TO BE TYPED
1016								: "FIRST 16K OF MEMORY NOT ALL THERE!"
1017	003620	000000			6\$:	HALT		: FATAL ERROR HALT.
1018								: MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
1019								:*****
1020								:* SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
1021								:* DATI, DATIP, DATO, AND DATOB.
1022								:*****
1023	003622	012706	001100		†TIMEOUT:	MOV	#STACK, SP	:SET UP THE STACK POINTER.
1024	003626	005067	174754			CLR	MMAVA	:CLEAR MEM MGMT AVAILABLE FLAG.
1025	003632	032777	010000	175300		BIT	#SW12, 2SWR	:CHECK IF MEM MGMT TO BE INHIBITED.
1026	003640	001011				BNE	1\$	:BR IF NO MEM MGMT.
1027	003642	012737	003664	000004		MOV	#1\$, 2#ERRVEC	:SET TIMEOUT FOR MEM MGMT CHECK.
1028	003650	005037	177572			CLR	2#SR0	:CHECK FOR MEM MGMT...TIMES OUT IF NONE.
1029	003654	004767	010420			JSR	PC	MMINIT
1030	003660	005267	174722			INC	MMAVA	:SET MEM MGMT AVAILABLE FLAG.
1031	003664				1\$:			
1032	003664	004567	017622			JSR	RS,	SPRINT
1033	003670	025437				.WORD	BYTMS	
1034								:GO PRINT OUT THE FOLLOWING MESSAGE.
1035	003672	005000						:ADDRESS OF MESSAGE TO BE TYPED
1036	003674	005002						: "BYTE MEMORY MAP:"
1037	003676	012737	003742	000004		CLR	R0	:SET UP TYPE OUT FLAG.
1038	003704	105712				CLR	R2	:SET ADDRESS POINTER TO ZERO.
1039	003706	032702	000001		10\$:	MOV	#20\$, 2#ERRVEC	:SET TIME OUT VEC TO SERVICE NON-EX MEM.
1040	003712	001001				TSTB	(R2)	:DO DATI ONLY.
1041	003714	011212				BIT	#BIT0, R2	:CHECK FOR WORD ADDRESS.
1042	003716	151212				BNE	11\$	:BR IF ODD BYTE ADDRESS.
1043	003720	005700			11\$:	MOV	(R2), (R2)	:DO DATI, DATO...NOP FOR READ ONLY MAP.
1044	003722	001023				BISB	(R2), (R2)	:DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
1045	003724	004567	017562			TST	R0	:CHECK FOR PREVIOUS TYPED.
1046	003730	025507				BNE	30\$	:BR IF ALREADY TYPED "FROM".
1047						JSR	RS,	SPRINT
1048	003732	010246				.WORD	FROM	
1049	003734	004767	021212					:GO PRINT OUT THE FOLLOWING MESSAGE.
1050	003740	000413						:ADDRESS OF MESSAGE TO BE TYPED
1051								: "FROM"
1052	003742	022626				MOV	R2, -(SP)	:PUT THE DATA ON THE STACK.
1053	003744	005700				JSR	PC,	STYPAD
1054	003746	001411				BR	29\$	:DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1055	003750	004567	017536					:GO TO ADDRESS POINTER UPDATE.
1056	003754	025517						
1057								
1058	003756	005302						
1059	003760	010246						
1060	003762	004767	021164					



```

1061 003766 005202          INC      R2          ;RESET ADDRESS POINTER.
1062 003770 005100          COM      R0          ;RESET PREVIOUS TYP0UT FLAG.
1063 003772 005202          INC      R2          ;UPDATE ADDRESS POINTER TO NEXT BYTE.
1064 003774 001423          BEQ     31$          ;EXIT IF ZERO REACHED.
1065 003776 032702 017777  BIT     @MASK4K,R2   ;CHECK FOR 4K BANK BOUNDRY.
1066 004002 001340          BNE     10$          ;BR IF MORE THIS 4K BANK.
1067 004004 005767 174576  TST     MMAVA        ;CHECK IF MEM MGMT IS AVAILABLE.
1068 004010 001735          BEQ     10$          ;BR IF NO MEM MGMT.
1069 004012 022737 007600 172346  CMP     @7600, @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
1070 004020 001411          BEQ     31$          ;EXIT WHEN ALL DONE.
1071 004022 012702 060000          MOV     @60000, R2   ;RESET VIRTUAL ADDRESS POINTER.
1072 004026 013737 172346 172344  MOV     @#KIPAR3, @#KIPAR2 ;SAVE MEM MGMT REG FOR TYP0UT.
1073 004034 062737 000200 172346  ADD     @200, @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
1074 004042 000720          BR     10$          ;BR BACK TO DO NEXT BANK
1075 004044 005700          TST     R0          ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
1076 004046 001407          BEQ     32$          ;BR TO EXIT IF TYPING ALL DONE.
1077 004050 004567 017436  JSR     R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1078 004054 025517          .WORD  TO          ;ADDRESS OF MESSAGE TO BE TYPED
1079                                     ;"TO"
1080 004056 005302          DEC     R2          ;BACK ADDRESS POINTER UP ONE BYTE.
1081 004060 010246          MOV     R2, -(SP)   ;PUT THE DATA ON THE STACK.
1082 004062 004767 021064  JSR     PC, $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
1083 004066 000000          HALT                                     ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
1084                                     ;* TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
1085 004070 000654          BR     TIMEOUT      ;LOOP BACK AND DO AGAIN UPON CONTINUE.
1086
1087 .SBTTL MAP PARITY REGISTERS
1088 ;*****
1089 ;* SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
1090 ;* THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
1091 ;*****
1092
1093 004072 012704 002276 175034  GMPR:  MOV     @MPRX, R4          ;SET UP POINTER TO PARITY REG EXIST TABLE.
1094 004076 032777 000100          BIT     @SW06, @SWR   ;CHECK FOR INHIBIT PARITY SWITCH.
1095 004104 001036          BNE     GMPRD        ;BR IF INHIBIT PARITY.
1096 004106 012703 002076          MOV     @MPRO, R3    ;SET UP TABLE POINTER
1097 004112 012737 004134 000004  MOV     @GMPRB, @ERRVEC ;SET UP TIMEOUT TRAP SERVICE
1098 004120 042713 000001  GMPRA: BIC     #1, (R3) ;CLEAR FLAG BIT IN TABLE
1099 004124 005773 000000          TST     @R3         ;DOES THIS MEMORY PARITY REGISTER EXIST.
1100 ;* IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
1101 004130 012324          MOV     (R3)+, (R4)+ ;SAVE IT IN THE PARITY REG EXIST TABLE.
1102 004132 000403          BR     GMPRC        ;SKIP TIMEOUT SERVICE CODE
1103 ;* TIMEOUT COMES HERE
1104 004134 022626          GMPRB: CMP     (SP)+, (SP)+ ;RESTORE STACK POINTER
1105 004136 052723 000001          BIS     #1, (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
1106 004142 005023          GMPRC: CLR     (R3)+ ;CLEAR THE MAP...LO 64K.
1107 004144 005023          CLR     (R3)+      ;...HI 64K.
1108 004146 005023          CLR     (R3)+      ;...AND THE MASK.
1109 004150 020327 002276          CMP     R3, @MPRX   ;HAVE WE CHECKED ALL REGISTERS?
1110 004154 103761          BLO     GMPRA        ;NO - GO BACK TO CHECK NEXT ONE
1111 004156 005014          CLR     (R4)        ;SET TERMINATOR IN PARITY REG EXIST TABLE.
1112 004160 012737 025114 000004  MOV     @ERRTRP, @ERRVEC ;RESTORE TRAPCATCHER
1113 004166 005767 176104          TST     MPRX        ;ANY PARITY REGISTERS PRESENT?
1114 004172 001006          BNE     MPAMEM      ;YES - GO TEST CONTROLS PRESENT
1115 004174 004567 017312  JSR     R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1116 004200 025605          .WORD  MTR         ;ADDRESS OF MESSAGE TO BE TYPED

```



C09

CZQMCFO 0-124K MEMORY EXERCISER. 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 24  
CZQMCF.P11 14-FEB-78 09:19 MAP PARITY REGISTERS

SEQ 0106

1117  
1118 004202 005014  
1119 004204 000167 001156  
1120

GMPRD: CLR (R4)  
JMP MANUAL

:"NO MEMORY PARITY REGISTERS FOUND"  
:MAKE SURE TABLE IS CLEAR.  
;AND SKIP ALL CONTROLS TESTING

1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133 004210 004767 014054  
1134 004214 012767 000001 175322  
1135 004222 005067 175320  
1136 004226 012702 014000  
1137 004232 005767 174350  
1138 004236 001404  
1139 004240 012702 054000  
1140 004244 004767 010030  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148 004250 005067 175264  
1149 004254 005067 175262  
1150 004260 012703 002076  
1151 004264 032713 000001  
1152 004270 001052  
1153 004272 013773 001612 000000  
1154  
1155 004300 011212  
1156 004302 005712  
1157 004304 043773 001612 000000  
1158 004312 005773 000000  
1159  
1160  
1161 004316 100014  
1162 004320 032773 007740 000000  
1163 004326 001404  
1164 004330 012763 070032 000006  
1165 004336 000413  
1166 004340 012763 077772 000006 5\$:  
1167 004346 000407  
1168 004350 032773 007740 000000 6\$:  
1169 004356 001417  
1170 004360 012763 070000 000006  
1171 004366 056763 175152 000002 7\$:  
1172 004374 056763 175146 000004  
1173 004402 056767 175136 175130  
1174 004410 056767 175132 175124  
1175 004416 062703 000010  
1176 004422 020327 002276

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

MPAMEM: JSR PC, CLRPAR ; INITIALIZE ALL PARITY REGISTERS
MOV #1, BITPT ; INITIALIZE 4K POINTER
CLR BITPT+2 ; CLEAR HI 64K POINTER
MOV #14000, R2 ; SET ADR POINTER TO 14000.
TST MAVA ; CHECK FOR MEM MGMT
BEQ MAPRB ; BRANCH IF NO MEM MGMT
MOV #54000, R2 ; SET ADR POINTER TO PAR2
JSR PC, MMINIT ; SET UP ALL MEMORY MGMT REGISTERS.

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

MAPRB: CLR PMEMAP ; CLEAR THE PARITY MEMORY MAP
CLR PMEMAP+2
1$: MOV #MPRO, R3 ; INITIALIZE TABLE ADDRESS
2$: BIT #1, (R3) ; IS THIS REGISTER PRESENT?
BNE 3$ ; NO - GET THE NEXT ONE
MOV @#WWP, @ (R3) ; YES - SET WRITE WRONG PARITY
; AND CLEAR REST OF REGISTER
MOV (R2), (R2) ; WRITE WRONG PARITY
TST (R2) ; READ WRONG PARITY
BIC @#WWP, @ (R3) ; CLEAR WRITE WRONG PARITY
TST @ (R3) ; OTHERWISE, CHECK TO SEE IF THIS
; CONTROL REGISTER GOT A PARITY
; ERROR
BPL 6$ ; BRANCH IF IT DIDN'T AND CHECK
BIT #7740, @ (R3) ; IS IT A CORE PAR. REG.
BEQ 5$ ; BRANCH IF NOT.
MOV #70032, @ (R3) ; IF IT IS SET UP MASK
BR 7$ ; AND BRANCH TO SET BITS.
5$: MOV #77772, @ (R3) ; IF MOS SET UP MASK
BR 7$ ; AND BRANCH TO SET BIT.
6$: BIT #7740, @ (R3) ; IF ANY BITS ARE SET
BEQ 3$ ; THEN CSR IS MS11-K.
MOV #70000, @ (R3) ; IF MS11-K SET MASK.
7$: BIS BITPT, 2 (R3) ; SET FLAG IN MAP FOR THIS PARITY REGISTER
BIS BITPT+2, 4 (R3)
BIS BITPT, PMEMAP ; SET FLAG IN PARITY MAP
BIS BITPT+2, PMEMAP+2
3$: ADD #10, R3 ; STEP UP TO NEXT REGISTER
CMP R3, #MPRX ; ARE WE DONE WITH TABLE?
```

```

1177 004426 103716          BLO      2$          ;GO BACK TO CHECK FOR ANY MORE!
1178 004430 011212          MOV      (R2)      (R2)      ;CLEAR BAD PARITY
1179 004432 005767 174150      TST      MMAVA      ;CHECK FOR MEM MGMT
1180 004436 001444          BEQ      10$          ;BR IF NO MEM MGMT
1181 004440 062737 000200 172344 4$:  ADD      #200, @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1182 004446 006367 175072          ASL      BITPT      ;UPDATE BANK POINTER...LO 64K.
1183 004452 006167 175070          ROL      BITPT+2    ;...HI 64K.
1184 004456 100441          BMI      TMAP      ;BR IF ALL DONE.
1185 004460 023727 172344 001000  CMP      @#KIPAR2, #1000 ;THIS CODE TESTS IF MS11-K IS
1186 004466 001013          BNE      12$          ;PRESENT AND IF IT IS I SET
1187 004470 032737 000003 002260  BIT      #3, @#MPR14+2 ;THE BIT TO DISABLE ECC IN
1188 004476 001004          BNE      13$          ;THE LOCATION WWP THAT IS
1189 004500 032737 000003 002270  BIT      #3, @#MPR15+2 ;USED AS THE COMMAND TO
1190 004506 001400          BEQ      13$          ;WRITE WRONG PARITY.
1191 004510 012737 020004 001612 13$:  MOV      #20004, @#WWP
1192 004516 036767 175022 175000 12$:  BIT      BITPT, MEMMAP ;CHECK IF BANK EXISTS...LJ 64K.
1193 004524 001255          BNE      1$          ;BR IF BANK EXISTS.
1194 004526 036767 175014 174772  BIT      BITPT+2, MEMMAP+2 ;...HI 64K.
1195 004534 001251          BNE      1$          ;BR IF BANK EXISTS.
1196 004536 000740          BR       4$          ;BR IF BANK DOESN'T EXIST.
1197 004540 036767 175000 174756 11$:  BIT      BITPT, MEMMAP ;CHECK IF BANK EXISTS.
1198 004546 001244          BNE      1$          ;BR IF BANK EXISTS.
1199 004550 062702 020000          ADD      #20000, R2 ;UPDATE ADDRESS POINTER TO NEXT BANK.
1200 004554 106367 174764          ASLB    BITPT      ;MOVE POINTER TO NEXT BANK.
1201 004560 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
*****

```

```

1202
1203
1204
1205
1206
1207
1208 004562 004767 013502  TMAP:  JSR      PC, CLPAR ;INITIALIZE ALL PARITY REGISTERS PRESENT
1209 004566 004567 016720  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1210 004572 025462          .WORD   MMAP      ;ADDRESS OF MESSAGE TO BE TYPED
1211          ;"PARITY MEMORY MAP:"
1212 004574 012703 002076          MOV      #MPRO, R3 ;INITIALIZE TABLE POINTER
1213 004600 032713 000001 1$:  BIT      #BIT0, (R3) ;CHECK IF THIS REGISTER IS PRESENT.
1214 004604 001046          BNE      2$          ;BR IF NOT PRESENT.
1215 004606 022763 070032 000006  CMP      #70032, 6(R3)
1216 004614 001004          BNE      3$          ;
1217 004616 004567 016670  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1218 004622 026123          .WORD   MX3      ;ADDRESS OF MESSAGE TO BE TYPED
1219          ;"CORE PARITY"
1220 004624 000417          BR       5$          ;
1221 004626 022763 077772 000006 3$:  CMP      #77772, 6(R3)
1222 004634 001004          BNE      4$          ;
1223 004636 004567 016650  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1224 004642 026142          .WORD   MX4      ;ADDRESS OF MESSAGE TO BE TYPED
1225          ;"MOS PARITY"
1226 004644 000407          BR       5$          ;
1227 004646 022763 070000 000006 4$:  CMP      #70000, 6(R3)
1228 004654 001003          BNE      5$          ;
1229 004656 004567 016630  JSR      R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1230 004662 026160          .WORD   MX5      ;ADDRESS OF MESSAGE TO BE TYPED
1231          ;"MS11-K CSR"
1232 004664          5$:

```



```

1233 004664 004567 016622 JSR R5 $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1234 004670 026071 .WORD MX1 ;ADDRESS OF MESSAGE TO BE TYPED
1235 ;"REGISTER AT"
1236 004672 011346 MOV (R3),-(SP) ;SAVE (R3) FOR TYPEOUT
1237 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
1238 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
1239 004674 013746 177776 MOV @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1240 004700 004767 020004 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
1241 004704 004567 016602 JSR R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1242 004710 026110 .WORD MX2 ;ADDRESS OF MESSAGE TO BE TYPED
1243 ;"CONTROLS"
1244 004712 010300 MOV R3, RC ;SET UP R0 FOR TYPMAP ROUTINE.
1245 004714 005720 TST (R0)+ ;UPDATE POINTER TO MAP.
1246 004716 004767 013470 JSR PC, TYPMAP ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1247 004722 062703 000010 ADD #10, R3 ;UPDATE TO NEXT REGISTER IN TABLE.
1248 004726 020327 002276 CMP R3, #MPRX ;ARE WE ALL DONE WITH TABLE?
1249 004732 103722 BLO 1$ ;BRANCH IF MORE REGISTERS
1250 004734 004567 016552 JSR R5, $SPRINT ;THE REASON I'M OUTPUTTING THIS CRLF
1251 004740 001201 $CRLF ;IS TO GIVE THE PRINTER ENOUGH TIME TO
1252 ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
1253 004742 022737 070000 002264 CMP #70000,@#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
1254 004750 001006 BNE 7$ ;IF NO BRANCH
1255 004752 043727 002260 001540 BIC @#MPR14+2,@#PMEMAP ;IF YES THEN CLEAR THE BITS IN
1256 004760 043737 002262 001540 BIC @#MPR14+4,@#PMEMAP ;THE PARITY MEMORY MAP.
1257 004766 022737 070000 002274 7$: CMP #70000,@#MPR15+6 ;DO WE HAVE A MS11-K
1258 004774 001031 BNE 9$ ;IF NO GO TO TESTS NOW.
1259 004776 043737 002270 001540 BIC @#MPR15+2,@#PMEMAP ;IF YES I AM GOING TO
1260 005004 043737 002272 001542 BIC @#MPR15+4,@#PMEMAP+2 ;CLEAR THE PARITY INDICATORS
1261 005012 012705 002276 MOV #MPRX, R5 ;FOR THAT PORTION OF MEMORY.
1262 005016 021537 002256 6$: CMP (R5),@#MPR14 ;SEARCH FOR THIS MS11-K CSR IN
1263 005022 001004 BNE 8$ ;AND IF ITS THERE DELETE IT
1264 005024 005015 CLR (R5)
1265 005026 052737 000001 002256 BIS #1,@#MPR14
1266 005034 022537 002266 3$: CMP (R5)+,@#MPR15 ;SEARCH FOR MS11-K CSR IN
1267 005040 001366 BNE 6$ ;THE AVAILABILITY TABLE.
1268 005042 005045 CLR -(R5) ;AND CLEAR ITS ADDRESS FROM THE TABLE
1269 005044 052737 000001 002266 BIS #1,@#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
1270 005052 004567 016434 JSR R5, $SPRINT ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1271 005056 026176 .WORD MX6
1272 005060 005737 002276 9$: TST @#MPRX ;ARE THERE ANY PARITY REGISTERS TO TEST?
1273 005064 001002 BNE CTRLS ;IF SO TEST THE BITS IN THE REGISTERS.
1274 005066 000167 000274 JMP MANUAL ;IF NO JUMP OVER REGISTER TESTS.
1275
1276 ;SBTTL TEST PARITY REGISTERS
1277 ;*****
1278 ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
1279 ;* CAN BE SET AND CLEARED.
1280 ;* THIS IS A ONCE ONLY TEST.
1281 ;*****
1282
1283 005072 012703 002076 CTRLS: MOV #MPRO, R3 ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1284 005076 011302 1$: MOV (R3), R2 ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
1285 005100 062703 000010 ADD #10, R3 ;UPDATE POINTER TO NEXT PAR. REG. ADD.
1286 005104 032702 000001 BIT #1, R2 ;IS THIS REGISTER BEING USED?
1287 005110 001372 BNE 1$ ;GO TO NEXT IF NOT
1288 005112 020327 002276 CMP R3, #MPRX ;ARE WE AT END OF TABLE
    
```

```

1289 005116 003055          BGT  RESCHK          ;GO TO NEXT TEST IF YES
1290 005120 005763 177776  TST  -2(R3)          ;TEST MASK FOR PARITY REGISTER
1291 005124 001764          BEQ  1$              ;IF = 0 THEN DO NOT TEST
1292 005126 016367 177776 174362 MOV  -2(R3), RESRVD ;GET MASK FOR REGISTER WE ARE WORKING ON
1293 005134 012700 000001    MOV  #1, R0          ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
1294 005140 005012          CLR  (R2)           ;INITIALIZE THE PARITY REGISTER
1295 005142 011201          MOV  (R2), R1       ;READ THE CONTENTS OF THE PARITY REGISTER
1296 005144 046701 174346    BIC  RESRVD, R1     ;CLEAR BITS WHICH ARE RESERVED
1297 005150 001405          BEQ  2$              ;CHECK OTHER BITS - BRANCH IF OK
1298 005152 004767 013134    JSR  PC, SPRNT      ;SET UP VALUES FOR ERROR PRINTING.
1299 005156 004767 014456    JSR  PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1300 005162 000001          .WORD 1             ;ERROR TYPE CODE.
1301 005164 030067 174326    2$: BIT  R0, RESRVD  ;IS THIS BIT RESERVED?
1302 005170 001025          BNE  3$              ;YES - DON'T TEST IT
1303 005172 010012          MOV  R0, (R2)       ;NO - SET THIS BIT IN THE PARITY REGISTER
1304 005174 011201          MOV  (R2), R1       ;READ & SAVE CONTENTS OF THE PARITY REGISTER
1305 005176 005012          CLR  (R2)           ;CLEAR THE PARITY REGISTER
1306 005200 046701 174312    BIC  RESRVD, R1     ;CLEAR BIT LOCATIONS THAT ARE RESERVED
1307 005204 020001          CMP  R0, R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
1308 005206 001405          BEQ  66$            ;BRANCH OVER ERROR CALL IF GOOD DATA.
1309 005210 004767 013126    JSR  PC, SPRNT      ;SET UP VALUES FOR ERROR PRINTING.
1310 005214 004767 014420    JSR  PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1311 005220 000001          .WORD 1             ;ERROR TYPE CODE.
1312 005222          66$:
1313 005222 011201          MOV  (R2), R1       ;READ THE CONTENTS OF THE PARITY REGISTER
1314 005224 046701 174266    BIC  RESRVD, R1     ;CLEAR BITS WHICH ARE RESERVED
1315 005230 001405          BEQ  3$              ;CHECK OTHER BITS - BRANCH IF OK
1316 005232 004767 013054    JSR  PC, SPRNT      ;SET UP VALUES FOR ERROR PRINTING.
1317 005236 004767 014376    JSR  PC, $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
1318 005242 000001          .WORD 1             ;ERROR TYPE CODE.
1319 005244 006300          3$: ASL  R0           ;ROTATE TO GET NEXT BIT TO BE TESTED
1320 005246 103346          BCC  2$              ;BRANCH IF NOT DONE WITH ALL BITS
1321 005250 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.
;*****

```

```

1328 005252 012704 002076  RESCHK: MOV  #MPRO, R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
1329 005256 010403          1$: MOV  R4, R3
1330 005260 062704 000010    ADD  #10, R4
1331 005264 032713 000001    BIT  #1, (R3)        ;IS THIS REGISTER BEING USED
1332 005270 001372          BNE  1$              ;BRANCH IF NO
1333 005272 012773 177777 000000    MOV  #-1, @ (R3)    ;SET ALL BITS TO A 1
1334 005300 022704 002276    CMP  #MPRX, R4      ;ARE WE AT THE END OF THE TABLE
1335 005304 002764          BLT  1$              ;IF YES THEN WE ARE READY TO TEST
1336 005306 000005          RESET              ;RESET THE WORLD
1337 005310 012703 002076    MOV  #MPRO, R3      ;LOAD INITIAL ADDRESS FOR POINTER
1338 005314 011302          2$: MOV  (R3), R2      ;STORE PARITY REGISTER ADDRESS
1339 005316 062703 000010    ADD  #10, R3
1340 005322 032702 000001    BIT  #1, R2
1341 005326 001372          BNE  2$              ;
1342 005330 022703 002276    CMP  #MPRX, R3
1343 005334 002014          BGE  MANUAL
1344 005336 011201          MOV  (R2), R1      ;GET CONTENTS OF REGISTER

```



1345	005340	005012			CLR	(R2)			
1346	005342	042701	077772		BIC	#77772, R1			:CLEAR BITS NOT EFFECTED BY RESET
1347	005346	005701			TST	R1			:CHECK IF REST WERE CLEARED BY RESET
1348	005350	001405			BEQ	65\$			:BRANCH OVER ERROR CALL IF GOOD DATA.
1349	005352	004767	012734	64\$:	JSR	PC,	SPRNT		:SET UP VALUES FOR ERROR PRINTING.
1350	005356	004767	014256		JSR	PC,	\$ERROR		:*** ERROR *** (GO TYPE A MESSAGE)
1351	005362	000001			.WORD	1			:ERROR TYPE CODE.
1352	005364			65\$:					
1353	005364	000753			BR	2\$			:BRANCH BACK TO CHECK NEXT REGISTER
1354									
1355									
1356	005366	012700	000014	MANUAL:	MOV	#12, R0			:SET COUNTER TO CLEAR 12 WORDS.
1357	005372	012701	001562		MOV	#FSTADR, R1			:STARTING AT FSTADR.
1358	005376	005021		1\$:	CLR	(R1)+			:CLEAR THE LOCATIONS.
1359	005400	005300			DEC	R0			:COUNT.
1360	005402	001375			BNE	1\$			:BR IF MORE.
1361	005404	105767	174146		TSTB	SELFLG			:CHECK FOR SELECT PARAMETERS STARTUP.
1362	005410	001005			BNE	MANUL1			:BR IF PARAMETERS TO BE SELECTED.
1363	005412	016767	173546	174154	MOV	\$TMP2	LSTADR		:SET UP VIRTUAL LAST ADDRESS.
1364	005420	000167	000402		JMP	MANUL2			:SKIP PARAMETER SELECTION SECTION.



1365  
1366  
1367  
1368  
1369 005424 012700 000001  
1370 005430 005001  
1371 005432 005002  
1372 005434 005003  
1373 005436 004567 016050  
1374 005442 026311  
1375  
1376  
1377  
1378 005444 013746 177776  
1379 005450 004767 015664  
1380 005454 042716 000001  
1381 005460 005067 174050  
1382 005464 005067 174046  
1383 005470 062702 020000  
1384 005474 005503  
1385 005476 020367 016006  
1386 005502 103403  
1387 005504 101006  
1388 005506 020216  
1389 005510 101004  
1390 005512 006300  
1391 005514 006101  
1392 005516 100364  
1393 005520 000507  
1394 005522 030067 173776  
1395 005526 001003  
1396 005530 030167 173772  
1397 005534 001501  
1398 005536 016704 015746  
1399 005542  
1400 005542 004567 015744  
1401 005546 026376  
1402  
1403  
1404  
1405 005550 013746 177776  
1406 005554 004767 015560  
1407 005560 005716  
1408 005562 001010  
1409 005564 005767 015720  
1410 005570 001005  
1411 005572 016716 173366  
1412 005576 016767 173364 015704  
1413 005604 012667 173764  
1414 005610 020467 015674  
1415 005614 101352  
1416 005616 103403  
1417 005620 021667 173750  
1418 005624 101346  
1419 005626 032716 017777  
1420 005632 001404

```

.SBTTL USER PARAMETER SELECTION SECTION
:*****
:* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
:*****
MANUL1: MOV    #BIT0, R0      ;SET UP BANK POINTER.
        CLR    R1             ;...HI 64K.
        CLR    R2             ;CLEAR ADDRESS POINTER.
        CLR    R3             ;...HI ADDRESS BITS.
        JSR    R5, $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD  FADMES        ;ADDRESS OF MESSAGE TO BE TYPED
        ;"FIRST ADDRESS:"

:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV    @#PSW, -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR    PC, $RDOCT    ;GO TO THE SUBROUTINE
        BIC    #BIT0, (SP)   ;MAKE SURE ADDRESS IS ON .4 WORD BOUNDARY.
        CLR    SAVTST        ;INIT TEST MAP...LO 64K.
        CLR    SAVTST+2      ;...HI 64K.
1$: ADD    #20000, R2        ;UPDATE ADDRESS POINTER TO NEXT BANK.
        ADC    R3
        CMP    R3, $HIOCT    ;CHECK HI ADDRESS BITS.
        BLO   2$             ;BR IF NOT HI ENOUGH YET.
        BHI   3$             ;BR IF PAST SELECTED ADDRESS.
        CMP    R2, (SP)      ;CHECK THE LO ADDRESS BITS.
        BHI   3$             ;BR IF PAST SELECTED ADDRESS.
2$: ASL    R0                ;UPDATE POINTER...LO 64K.
        ROL    R1            ;...HI 64K.
        BPL   1$             ;BR BACK TO CHECK NEXT BANK.
        BR    17$            ;BR IF OVERFLOW.
3$: BIT    R0, MEMMAP        ;CHECK IF BANK EXISTS.
        BNE   4$             ;BR IF BANK EXISTS.
        BIT    R1, MEMMAP+2  ;CHECK HI 64K.
        BEQ   17$            ;BR IF ADDRESS IN UN-MAPPED BANK.
4$: MOV    $HIOCT, R4        ;SAVE FIRST ADR HI BITS.
10$: JSR    R5, $PRINT       ;GO PRINT OUT THE FOLLOWING MESSAGE.
        .WORD  LAOMES        ;ADDRESS OF MESSAGE TO BE TYPED
        ;"LAST ADDRESS:"

:* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
:* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
        MOV    @#PSW, -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
        JSR    PC, $RDOCT    ;GO TO THE SUBROUTINE
        TST    (SP)          ;CHECK IF ADR 0 SELECTED (DEFAULT).
        BNE   11$            ;BR IF NOT 0 (DEFAULT)
        TST    $HIOCT        ;CHECK HI BITS.
        BNE   11$            ;BR IF NOT 0 (DEFAULT).
        MOV    $TMP2, (SP)    ;SET UP DEFAULT LAST ADR.
        MOV    $TMP3, $HIOCT
11$: MOV    (SP)+, LSTADR    ;GET THE DATA.
        CMP    R4, $HIOCT    ;CHECK FOR LAST ADR BELOW FIRST ADR.
        BHI  10$             ;BB IF LAST BELOW FIRST.
        BLO  12$             ;BR IF LAST ABOVE 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.

```

```

1421 005634 010067 173730      MOV      R0,      FADMAP ;SET UP FIRST ADDRESS MAP.
1422 005640 010167 173726      MOV      R1,      FADMAP+2
1423 005644 050067 173664      13$:    BIS      R0,      SAVTST ;SET FLAG IN TEST MAP...LO 64K.
1424 005650 050167 173662      BIS      R1,      SAVTST+2 ;...HI 64K.
1425 005654 020367 015630      14$:    CMP      R3,      $HI0CT ;CHECK FOR PAST LAST ADR.
1426 005660 103404      RLO      15$      ;BR IF BELOW LAST ADR.
1427 005662 101020      BHI      16$      ;BR IF GONE PAST LAST ADR.
1428 005664 020267 173704      CMP      R2,      LSTADR ;CHECK FOR PAST LAST ADR.
1429 005670 101015      BHI      16$      ;BR IF GONE PAST LAST ADR.
1430 005670 062702 020000      15$:    ADD      #20000, R2 ;UPDATE ADDRESS POINTER.
1431 005676 005503      ADC      R3 ;...HI BITS.
1432 005700 006300      ASL      R0 ;UPDATE BANK POINTER...LO 64K.
1433 005702 006101      ROL      R1 ;...HI 64K.
1434 005704 100415      BMI      17$      ;BR IF OVERFLOW.
1435 005706 030067 173612      BIT      R0,      MEMMAP ;CHECK IF THIS BANK EXISTS.
1436 005712 001354      BNE      13$      ;BR IF BANK EXISTS.
1437 005714 030167 173606      BIT      R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1438 005720 001351      BNE      13$      ;BR IF BANK EXISTS.
1439 005722 000754      BR       14$      ;BR IF BANK DOESN'T EXIST.
1440 005724 030067 173574      16$:    BIT      R0,      MEMMAP ;CHECK IF THIS BANK EXISTS.
1441 005730 001010      BNE      20$      ;BR IF IT EXISTS.
1442 005732 030167 173570      BIT      R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
1443 005736 001005      BNE      20$      ;BR IF IT EXISTS.
1444 005740 005726      17$:    TST      (SP)+ ;ADJUST THE STACK.
1445 005742 004567 015544      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1446 005746 026421      .WORD   BADADR ;ADDRESS OF MESSAGE TO BE TYPED
1447 ;"ADDRESS IN UNMAPPED BANK?"
1448 ;LOOP BACK TO THE BEGINNING.
1448 005750 000606      BR       MANUAL
1449 005752 010067 173624      20$:    MOV      R0,      LADMAP ;SET UP MAP FOR LAST ADDRESS.
1450 005756 010167 173622      MOV      R1,      LADMAP+2
1451 005762 005767 172620      21$:    TST      MMAVA ;CHECK FOR MEMORY MANAGEMENT.
1452 005766 001404      BEQ      22$      ;BR IF NO MEM MGMT.
1453 005770 042716 160000      BIC      #16000, (SP) ;ADJUST FSTADR TO VIRTUAL BANK 0.
1454 005774 062716 040000      ADD      #40000, (SP) ;...TO VIRTUAL BANK 2.
1455 006000 012667 173556      22$:    MOV      (SP)+, FSTADR ;SAVE FIRST ADDRESS OFF THE STACK.
1456 006004      30$:
1457 006004 004567 015502      JSR      R5,      $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
1458 006010 026456      .WORD   CONST ;ADDRESS OF MESSAGE TO BE TYPED
1459 ;"SELECT CONSTANT:"
1460 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
1461 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
1462 006012 013746 177776      MOV      @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
1463 006016 004767 015316      JSR      PC,      $RDOCT ;GO TO THE SUBROUTINE
1464 006022 012667 173562      MOV      (SP)+, .CONST ;SAVE THE CONSTANT
1465 006026 005767 172554      MANUL2: TST      MMAVA ;CHECK IF MEM MGMT IS AVAILABLE.
1466 006032 001406      BEQ      31$      ;BR IF NO MEM MGMT.
1467 006034 042767 160000 173532      BIC      #16000, LSTADR ;ADJUST LSTADR TO VIRTUAL BANK 0.
1468 006042 062767 040000 173524      ADD      #40000, LSTADR ;...VIRTUAL BANK 2.
1469 006050 062767 000002 173516      31$:    ADD      #2,      LSTADR ;ADJUST LAST ADDRESS UP ONE WORD.
1470 006056 042767 000001 173510      BIC      #BIT0, LSTADR ;MAKE SURE IT IS A WORD ADDRESS.
1471 006064 032767 017777 173502      BIT      #MASK4K, LSTADR ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
1472 006072 001004      BNE      START1 ;BR IF NOT ON BOUNDRY.
1473 006074 005067 173502      CLR      LADMAP ;CLEAR OUT THE LAST ADDRESS MAP.
1474 006100 005067 173500      CLR      LADMAP+2
1475

```







```

1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504 006166
1505 006166 004567 012436
1506 006172 000001
1507
1508 006174 000167 005604
1509
1510
1511 006200 004467 006222
1512 006204 004767 007644
1513 006210 010012
1514 006212 012201
1515 006214 020001
1516 006216 001405
1517 006220 004767 012142
1518 006224 004767 013410
1519 006230 000002
1520 006232
1521 006232 062700 000002
1522 006236 030502
1523 006240 001363
1524 006242 004767 006736
1525
1526
1527
1528 006246 004467 006612
1529 006252 004767 007576
1530 006256 162700 000002
1531 006262 014201
1532 006264 020001
1533 006266 001405
1534 006270 004767 012046
1535 006274 004767 013340
1536 006300 000002
1537 006302
1538 006302 030502
1539 006304 001364
1540 006306 004767 007362

```

```

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

;* UPWARDS WORD ADDRESSING.
      JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$: MOV R0, (R2) ;WRITE VALUE OF ADDRESS INTO ADDRESS
      MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRT2 ;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, SPRT0 ;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.

```

```

1541
1542
1543
1544
1545
1546
1547
1548
1549
1550 006312
1551 006312 004567 012312
1552 006316 000000
1553
1554 006320 004467 006102
1555 006324 004767 007524
1556 006330 110022
1557 006332 005200
1558 006334 030502
1559 006336 001374
1560 006340 004767 006640
1561
1562
1563
1564 006344 004467 006514
1565 006350 004767 007500
1566 006354 005300
1567 006356 114201
1568 006360 120001
1569 006362 001405
1570 006364 004767 011752
1571 006370 004767 013244
1572 006374 000003
1573 006376
1574 006376 030502
1575 006400 001365
1576 006402 004767 007266
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587 006406
1588 006406 004567 012216
1589 006412 000000
1590
1591 006414 004467 006444
1592 006420 004767 007430
1593 006424 005100
1594 006426 062700 000002
1595 006432 010042
1596 006434 030502

```

```

*****
; *TEST 2 WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
; *
; * R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
; * R1 = DATA READ FROM MEMORY (WAS)
; * R2 = VIRTUAL ADDRESS
; * R3 = NOT USED
; * R4 = NOT USED
; * R5 = BLOCK BOUNDRY BIT MASK.
*****
TST2:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * UPWARDS BYTE ADDRESSING.
      JSR      R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   MOVB    RO,      (R2)+ ;WRITE VALUE OF ADDRESS INTO ADDRESS
      INC     RO        ;ADD ONE TO PHYSICAL ADDRESS
      BIT     R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE    2$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,      MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
; * CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
; * DOWNWARDS BYTE ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
4$:   DEC     RO        ;DEC DATA BY 1
      MOVB    -(R2),   R1     ;GET THE DATA FROM MEMORY
      CMPB   RO,      R1     ;CHECK THE DATA...LO BYTE ONLY VALID.
      BEQ    65$       ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR      PC,      SPRNTO ;SET UP VALUES FOR ERROR PRINTING.
      JSR    PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   3          ;ERROR TYPE CODE.
65$:  BIT     R5,      R2     ;CHECK FOR END OF A BLOCK.
      BNE    4$        ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,      MMDOWN ;FIND NEXT BLOCK AND LOOP TO $TAG1.
*****
; *TEST 3 WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
; *
; * R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
; * R1 = DATA READ FROM MEMORY (WAS)
; * R2 = VIRTUAL ADDRESS
; * R3 = NOT USED
; * R4 = NOT USED
; * R5 = BLOCK BOUNDRY BIT MASK.
*****
TST3:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
; * DOWNWARDS WORD ADDRESSING.
      JSR      R4,      INITDN ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   JSR      PC,      PHYADR ;GET PHYSICAL ADDRESS INTO R0
2$:   COM     RO        ;COMPLEMENT THE ADR
      ADD     #2,      RO     ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
      MOV     RO,      -(R2) ;PUT DATA INTO MEMORY
      BIT     R5,      R2     ;CHECK FOR END OF A BLOCK.

```



```

1597 006436 001373          BNE 2$          ;BRANCH IF MORE IN CURRENT BLOCK.
1598 006440 004767 007230 JSR PC, MMDOWN ;FIND NEXT BLOCK AND LOOP TO 1$.
1599
1600 ;* CHECK COMPLEMENT DATA WRITTEN DOWN
1601 ;* UPWARDS WORD ADDRESSING.
1602 006444 004467 005756 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1603 006450 004767 007400 3$: JSR PC, PHYADR ;GET PHYSICAL ADDRESS INTO RO
1604 006454 005100          COM RO          ;COMPLEMENT IT
1605 006456
1606 006456 012201          MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
1607 006460 020001          CMP RO, R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
1608 006462 001405          BEQ 65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
1609 006464 004767 011676 64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1610 006470 004767 013144 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1611 006474 000002          .WORD 2     ;ERROR TYPE CODE.
1612 006476
1613 006476 162700 000002 65$: SUB #2, RO   ;COUNT DOWN WITH ADDRESS
1614 006502 030502          BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1615 006504 001364          BNE 4$      ;BRANCH IF MORE IN CURRENT BLOCK.
1616 006506 004767 006472 JSR PC, MMUP  ;FIND NEXT BLOCK AND LOOP TO 3$.
1617
1618 ;*****
1619 ;*TEST 4 WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
1620 ;* RO = DATA WRITTEN INTO MEMORY (SHOULD BE)
1621 ;* R1 = DATA READ FROM MEMORY (WAS)
1622 ;* R2 = VIRTUAL ADDRESS
1623 ;* R3 = NOT USED
1624 ;* R4 = NOT USED
1625 ;* R5 = BLOCK BOUNDARY BIT MASK.
1626 ;*****
1627 006512
1628 006512 004567 012112 1ST4: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
1629 006516 000000          .WORD 0     ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
1630 ;* UPWARDS BYTE ADDRESSING.
1631 006520 004467 005702 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1632 006524 004767 007400 1$: JSR PC, BANKNO ;GET THE BANK NUMBER INTO RO
1633 006530 110022          2$: MOVB RO, (R2)+ ;WRITE BANK # INTO ALL ADDRESSES
1634 006532 030502          BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1635 006534 001375          BNE 2$      ;BRANCH IF MORE IN CURRENT BLOCK.
1636 006536 004767 006442 JSR PC, MMUP  ;FIND NEXT BLOCK AND LOOP TO 1$.
1637
1638 ;* CHECK THAT DATA WRITTEN ABOVE CAN BE READ
1639 ;* UPWARDS BYTE ADDRESSING.
1640 006542 004467 005660 JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1641 006546 004767 007356 3$: JSR PC, BANKNO ;GET THE BANK NUMBER INTO RO
1642 006552 112201          4$: MOVB (R2)+, R1 ;READ THE DATA OUT OF MEMORY
1643 006554 020001          CMP RO, R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
1644 006556 001405          BEQ 65$     ;BRANCH OVER ERROR CALL IF GOOD DATA.
1645 006560 004767 011564 64$: JSR PC, SPRNT1 ;SET UP VALUES FOR ERROR PRINTING.
1646 006564 004767 013050 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
1647 006570 000003          .WORD 3     ;ERROR TYPE CODE.
1648 006572
1649 006572 030502          65$: BIT R5, R2   ;CHECK FOR END OF A BLOCK.
1650 006574 001366          BNE 4$      ;BRANCH IF MORE IN CURRENT BLOCK.
1651 006576 004767 006402 JSR PC, MMUP  ;FIND NEXT BLOCK AND LOOP TO 3$.
1652

```



```

1653
1654
1655
1656
1657
1658
1659
1660
1661
1662 006602
1663 006602 004567 012022
1664 006606 000000
1665
1666 006610 004467 006250
1667 006614 004767 007310
1668 006620 005100
1669 006622 110042
1670 006624 030502
1671 006626 001375
1672 006630 004767 007040
1673
1674
1675
1676 006634 004467 006224
1677 006640 004767 007264
1678 006644 005100
1679 006646 114201
1680 006650 020001
1681 006652 001405
1682 006654 004767 011462
1683 006660 004767 012754
1684 006664 000003
1685 006666
1686 006666 030502
1687 006670 001366
1688 006672 004767 006776
    
```

```

*****
:TEST 5 WRITE 1'S COMPLEMENT OF BANK #.
:RU = DATA WRITTEN INTO MEMORY (SHOULD BE)
:RI = DATA READ FROM MEMORY (WAS)
:R2 = VIRTUAL ADDRESS
:R3 = NOT USED
:R4 = NOT USED
:R5 = BLOCK BOUNDARY BIT MASK.
*****
*ST5:
      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$:   MCVB   R0,      -(R2)        ;PUT 1'S COM OF BANK # INTO MEMORY
      BIT     R5,      R2          ;CHECK FOR END OF A BLOCK.
      BNE    2$,      $          ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,      MNDOWN      ;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$:  JSR      PC,      SPRNTO      ;SET UP VALUES FOR ERROR PRINTING.
      JSR      PC,      SERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   3          ;ERROR TYPE CODE.
65$:  BIT     R5,      R2          ;CHECK FOR END OF A BLOCK.
      BNE    4$,      $          ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR    PC,      MNDOWN      ;FIND NEXT BLOCK AND LOOP TO $TAG1.
    
```

```

1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704 006676
1705 006676 004567 011726
1706 006702 000000
1707 006704 016700 172700
1708 006710 004467 005512
1709 006714 010022
1710 006716 030502
1711 006720 001375
1712 006722 004767 006256
1713
1714
1715
1716
1717
1718 006726
1719 006726 004567 011676
1720 006732 000000
1721 006734 016700 172650
1722 006740 004467 005462
1723 006744
1724 006744 012201
1725 006746 020001
1726 006750 001405
1727 006752 004767 011410
1728 006756 004767 012656
1729 006762 000004
1730 006764
1731 006764 030502
1732 006766 001366
1733 006770 004767 006210
1734
1735
1736
1737 006774 032777 000400 172136
1738 007002 001416
1739 007004 017746 172130
1740 007010 042716 177740
1741 007014 022726 000006
1742 007020 001007
1743 007022 162767 000001 172052
1744 007030 162767 000030 172050

```

```

.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).
* RO = 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.
*****
TST6: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
TST6A: MOV .CONST, R0 ;GET USER CONSTANT
JSR R4, INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R0, (R2)+ ;WRITE CONSTANT INTO MEMORY.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
*****
*TEST 7 READ MEMORY AND COMPARE TO CONSTANT.
* IMPORTANT: THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
*****
TST7: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MOV .CONST, R0 ;GET USER CONSTANT
JSR R4, INITM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, SERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 4 ;ERROR TYPE CODE.
65$: BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
* SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
* ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
* BY SIMPLY "TOGGLING" SW00 WHEN SW01, SW02, AND SW08 ARE SET.
BIT #SW08, $SWR ;CHECK THAT LOOP ON TEST BIT SET
BEQ TST10 ;BRANCH IF NOT LOOP ON TEST
MOV $SWR, -(SP) ;GET SWITCH REGISTER DATA.
BIC #177740, (SP) ;CLEAR NON-TEST-NUMBER SWITCHES.
CMP #6, (SP)+ ;CHECK IF TEST 6 IN SWITCHES.
BNE TST10 ;BRANCH IF NOT TEST 6
SUB #1, $STNM ;RESET TEST NUM
SUB #TST7-TST6,$LPADR ;RESET LOOP ADR

```



D10

CZQMCFO 0-124K MEMORY EXERCISER, 16K VER  
CZQMC.F11 14-FEB-78 09:19 T7

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 38  
READ MEMORY AND COMPARE TO CONSTANT.

SEQ 0120

1745	007036	000722	
1746			
1747			
1748			
1749			
1750			
1751	007040		
1752	007040	004567	011564
1753	007044	000000	
1754	007046	016704	172562
1755	007052	004767	010560
1756	007056	012400	
1757	007060	001420	
1758	007062	004467	005340
1759	007066	010012	
1760	007070	012201	
1761	007072	020001	
1762	007074	001405	
1763	007076	004767	011264
1764	007102	004767	012532
1765	007106	000004	
1766	007110		
1767	007110	030502	
1768	007112	001365	
1769	007114	004767	006064
1770	007120	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,      $SCOPE      ;GO TO SCOPE ROUTINE.
      .WORD   0          ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
      JSR     PC,      CKPMER      ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
1$:    MOV     (R4)+,   R0        ;GET THE DATA PATTERN.
      BEQ    TST11        ;BR IF END OF TABLE.
      JSR     R4,      INITMM      ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2$:    MOV     R0,      (R2)       ;PUT DATA PATTERN INTO MEMORY.
      MOV     (R2)+,   R1        ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP    R0,      R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:   JSR     PC,      SPRT2      ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   4          ;ERROR TYPE CODE.
65$:   BIT     R5,      R2        ;CHECK FOR END OF A BLOCK.
      BNE    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

```



E10

```

1771
1772
1773
1774 007122
1775 007122 004567 011502
1776 007126 000000
1777 007130 012700 177777
1778 007134 004767 007030
1779 007140 004467 005262
1780 007144 000241
1781 007146 004767 007036
1782 007152 016201 177776
1783 007156 103402
1784 007160 020001
1785 007162 001405
1786 007164 004767 011176
1787 007170 004767 012444
1788 007174 000005
1789 007176
1790 007176 030502
1791 007200 001361
1792 007202 004767 005776
1793
1794
1795
1796
1797 007206
1798 007206 004567 011416
1799 007212 000000
1800 007214 005000
1801 007216 004767 006746
1802 007222 004467 005200
1803 007226 000261
1804 007230 004767 006754
1805 007234 016201 177776
1806 007240 103002
1807 007242 020001
1808 007244 001405
1809 007246 004767 011114
1810 007252 004767 012362
1811 007256 000005
1812 007260
1813 007260 030502
1814 007262 001361
1815 007264 004767 005714
  
```

```

*****
*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$,     R1      ;BRANCH IF 'C' BIT WAS SET
      CMP     R0,      R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     64$,     R1      ;BRANCH OVER ERROR CALL IF GOOD DATA.
63$:  JSR     PC,      SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    5            ;ERROR TYPE CODE.
64$:  BIT     R5,      R2      ;CHECK FOR END OF A BLOCK.
      BNE     1$,     R1      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
*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$,     R1      ;GET RESULT
      CMP     R0,      R1      ;BRANCH IF 'C' IS CLEAR
      BEQ     64$,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
63$:  JSR     PC,      SPRINT2 ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR     PC,      $ERROR  ;SET UP VALUES FOR ERROR PRINTING.
      .WORD    5            ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD    5            ;ERROR TYPE CODE.
64$:  BIT     R5,      R2      ;CHECK FOR END OF A BLOCK.
      BNE     1$,     R1      ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  
```

F10

```

1816
1817
1818
1819 007270
1820 007270 004567 011334
1821 007274 000777
1822
1823 007276 000167 000312
1824
1825 007302 005000
1826 007304 012703 177777
1827 007310 004467 005112
1828 007314 004767 006736
1829 007320 030502
1830 007322 001374
1831 007324 004767 005654
1832
1833
1834
1835
1836 007330 005000
1837 007332 004467 005070
1838 007336 012704 000100
1839 007342
1840 007342 012201
1841 007344 020001
1842 007346 001405
1843 007350 004767 011012
1844 007354 004767 012260
1845 007360 000007
1846 007362
1847 007362 012201
1848 007364 020001
1849 007366 001405
1850 007370 004767 010772
1851 007374 004767 012240
1852 007400 000007
1853 007402
1854 007402 012201
1855 007404 020001
1856 007406 001405
1857 007410 004767 010752
1858 007414 004767 012220
1859 007420 000007
1860 007422
1861 007422 012201
1862 007424 020001
1863 007426 001405
1864 007430 004767 010732
1865 007434 004767 012200
1866 007440 000007
1867 007442
1868 007442 005100
1869 007444 005304
1870 007446 001335
1871 007450 005100

*****
*TEST 13 3 XOR 9 TEST PATTERN.
*****
TST13:
JSR R5 $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 777 ;MINIMUM BLOCK SIZE OF 256. WORDS
;REQUIRED FOR THIS TEST.
JMP TST14 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
;AVAILABLE FOR TEST.
3X9: CLR R0 ;SET UP TEST DATA
MOV #-1, R3 ;SET COM DATA REG
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: JSR PC, W3X9 ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
BIT R5, R2 ;CHECK FOR END OF A BLOCK.
BNE 1$ ;BRANCH IF MORE IN CURRENT BLOCK.
JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

*****
* CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
*****
CLR R0 ;SET CHECK WORD
JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
11$: MOV #64, R4 ;SET 256. WORD COUNTER
12$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
65$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 67$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
66$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
67$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
68$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
69$:
MOV (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 71$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
70$: JSR PC, SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
.WORD 7 ;ERROR TYPE CODE.
71$:
COM R0 ;COMPLEMENT CHECK WORD
DEC R4 ;DECREMENT 256. WORD COUNTER
BNE 12$
COM R0 ;COMPLEMENT CHECK WORD

```



# G10

CZQMCFO 0-124K MEMORY EXERCISER. 16K VER  
 CZQMC.F.P11 14-FEB-78 06:19 T13

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 41  
 3 XOR 9 TEST PATTERN.

SEQ 0123

1872	007452	030502		BIT	R5	R2	:CHECK FOR END OF A BLOCK.
1873	007454	001330		BNE	11\$		:BRANCH IF MORE IN CURRENT BLOCK.
1874	007456	004767	005522	JSR	PC,	MMUP	:FIND NEXT BLOCK AND LOOP TO 11\$.
1875							
1876				:*****			
1877				:* CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.			
1878				:*****			
1879	007462	005000		CLR	R0		
1880	007464	004467	004736	JSR	R4,	INITMM	:INITIALIZE THE MEMORY ADDRESS POINTERS.
1881	007470	012704	000100	21\$: MOV	#64.,	R4	:SET 256. WORD COUNTER
1882	007474	012703	000004	22\$: MOV	#4,	R3	:SET 4 WORD COUNTER
1883	007500			23\$:			
1884	007500	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
1885	007502	020001		CMP	R0,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
1886	007504	001405		BEQ	73\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
1887	007506	004767	010654	72\$: JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
1888	007512	004767	012122	JSR	PC,	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
1889	007516	000007		.WORD	7		:ERROR TYPE CODE.
1890	007520			73\$:			
1891	007520	005100		COM	R0		:COMPLEMENT CHECK WORD
1892	007522	005142		COM	-(R2)		:COMPLEMENT TEST DATA
1893	007524	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
1894	007526	020001		CMP	R0,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
1895	007530	001405		BEQ	75\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
1896	007532	004767	010630	74\$: JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
1897	007536	004767	012076	JSR	PC,	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
1898	007542	000007		.WORD	7		:ERROR TYPE CODE.
1899	007544			75\$:			
1900	007544	005100		COM	R0		:COMPLEMENT CHECK WORD
1901	007546	005142		COM	-(R2)		:COMPLEMENT TEST DATA
1902	007550	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
1903	007552	020001		CMP	R0,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
1904	007554	001405		BEQ	77\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
1905	007556	004767	010604	76\$: JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
1906	007562	004767	012052	JSR	PC,	\$ERROR	:*** ERROR *** (GO TYPE A MESSAGE)
1907	007566	000007		.WORD	7		:ERROR TYPE CODE.
1908	007570			77\$:			
1909	007570	005303		DEC	R3		:DECREMENT 4 WORD COUNTER
1910	007572	001342		BNE	23\$		:BR IF NOT DONE.
1911	007574	005100		COM	R0		:COMPLEMENT CHECK WORD
1912	007576	005304		DEC	R4		:DECREMENT 256. WORD COUNTER
1913	007600	001335		BNE	22\$		:BR IF NOT DONE.
1914	007602	005100		COM	R0		:COMPLEMENT CHECK WORD
1915	007604	030502		BIT	R5	R2	:CHECK FOR END OF A BLOCK.
1916	007606	001330		BNE	21\$		:BRANCH IF MORE IN CURRENT BLOCK.
1917	007610	004767	005370	JSR	PC,	MMUP	:FIND NEXT BLOCK AND LOOP TO 21\$.



# H10

CZQMCFD 0-124K MEMORY EXERCISER, 16K VER  
CZQMCF.P11 14-FEB-78 09:19 T14

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 42  
COMPLEMENT 3 XOR 9 TEST PATTERN

SEQ 0124

```
1918 ::*****
1919 ::*TEST 14 COMPLEMENT 3 XOR 9 TEST PATTERN
1920 ::*****
1921 †ST14: JSR R5, $SCOPE ; GO TO SCOPE ROUTINE.
1922 .WORD 777 ; MINIMUM BLOCK SIZE OF 256. WORDS
1923 ; REQUIRED FOR THIS TEST.
1924 JMP TST15 ; SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
1925 ; AVAILABLE FOR TEST.
1926 MOV #-1, R0 ; SET UP TEST DATA
1927 CLR R3 ; SET COM DATA REG
1928 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
1929 1$: JSR PC, W3X9 ; WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
1930 BIT R5, R2 ; CHECK FOR END OF A BLOCK.
1931 BNE 1$, ; BRANCH IF MORE IN CURRENT BLOCK.
1932 JSR PC, MMUP ; FIND NEXT BLOCK AND LOOP TO 1$.
1933
1934
1935 ::*****
1936 ::* CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
1937 ::*****
1938 MOV #-1, R0 ; SET CHECK WORD
1939 JSR R4, INITMM ; INITIALIZE THE MEMORY ADDRESS POINTERS.
1940 11$: MOV #64., R4 ; SET 256. WORD COUNTER
1941 12$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
1942 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
1943 BEQ 65$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
1944 JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
1945 JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1946 .WORD 7 ; ERROR TYPE CODE.
1947 65$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
1948 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
1949 BEQ 67$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
1950 JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
1951 JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1952 .WORD 7 ; ERROR TYPE CODE.
1953 67$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
1954 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
1955 BEQ 69$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
1956 JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
1957 JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1958 .WORD 7 ; ERROR TYPE CODE.
1959 69$: MOV (R2)+, R1 ; GET THE DATA FROM MEMORY UNDER TEST.
1960 CMP R0, R1 ; COMPARE THE CHECK WORD WITH THE DATA READ.
1961 BEQ 71$ ; BRANCH OVER ERROR CALL IF GOOD DATA.
1962 JSR PC, SPRNT2 ; SET UP VALUES FOR ERROR PRINTING.
1963 JSR PC, $ERROR ; *** ERROR *** (GO TYPE A MESSAGE)
1964 .WORD 7 ; ERROR TYPE CODE.
1965 71$: COM R0 ; COMPLEMENT CHECK WORD
1966 DEC R4 ; DECREMENT 256. WORD COUNTER
1967 BNE 12$
```

```

1974 007776 005100          COM      R0          ;COMPLEMENT CHECK WORD
1975 010000 030502          BIT      R5          ;CHECK FOR END OF A BLOCK.
1976 010002 001330          BNE     11$         ;BRANCH IF MORE IN CURRENT BLOCK.
1977 010004 004767 005174    JSR     PC,        MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
1978
1979
1980
1981
1982 010010 012700 177777          MOV     #-1,       R0          ;SET UP CHECK WORD.
1983 010014 004467 004406    JSR     R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1984 010020 012704 000100    21$:   MOV     #64.,  R4          ;SET 256. WORD COUNTER
1985 010024 012703 000004    22$:   MOV     #4,     R3          ;SET 4 WORD COUNTER
1986 010030
1987 010030 012201          MOV     (R2)+,    R1          ;GET THE DATA FROM MEMORY UNDER TEST.
1988 010032 020001          CMP     R0,       R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
1989 010034 001405          BEQ    73$         ;BRANCH OVER ERROR CALL :. GOOD DATA.
1990 010036 004767 010324    72$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
1991 010042 004767 011572    JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
1992 010046 000007          .WORD  7              ;ERROR TYPE CODE.
1993
1994
1995
1996 010050 005100          COM     R0          ;COMPLEMENT CHECK WORD
1997 010052 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
1998 010054 012201          MOV     (R2)+,    R1          ;GET THE DATA FROM MEMORY UNDER TEST.
1999 010056 020001          CMP     R0,       R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
2000 010060 001405          BEQ    74$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2001 010062 004767 010300    74$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2002 010066 004767 011546    JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2003 010072 000007          .WORD  7              ;ERROR TYPE CODE.
2004
2005
2006
2007 010074 005100          COM     R0          ;COMPLEMENT CHECK WORD
2008 010076 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2009 010100 012201          MOV     (R2)+,    R1          ;GET THE DATA FROM MEMORY UNDER TEST.
2010 010102 020001          CMP     R0,       R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
2011 010104 001405          BEQ    75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2012 010106 004767 010254    75$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2013 010112 004767 011522    JSR     PC,        $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2014 010116 000007          .WORD  7              ;ERROR TYPE CODE.
2015
2016
2017
2018 010120 005303          DEC     R3          ;DECREMENT 4 WORD COUNTER
2019 010122 001342          BNE    23$         ;BR IF NOT DONE.
2020 010124 005100          COM     R0          ;COMPLEMENT CHECK WORD
2021 010126 005304          DEC     R4          ;DECREMENT 256. WORD COUNTER
2022 010130 001335          BNE    22$         ;BR IF NOT DONE.
2023 010132 005100          COM     R0          ;COMPLEMENT CHECK WORD
2024 010134 030502          BIT     R5          ;CHECK FOR END OF A BLOCK.
2025 010136 001330          BNE    21$         ;BRANCH IF MORE IN CURRENT BLOCK.
2026 010140 004767 005040    JSR     PC,        MMUP ;FIND NEXT BLOCK AND LOOP TO 21$.

```



# J10

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T15

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 44  
 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

SEQ 0126

```

2021 ::*****
2022 ::*TEST 15      MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
2023 ::*****
2024 †T15:
2025   010144   004567   010460   JSR    R5,    $SCOPE ;GO TO SCOPE ROUTINE.
2026   010150   000777           .WORD  777          ;MINIMUM BLOCK SIZE OF 256. WORDS
2027                                     ;REQUIRED FOR THIS TEST.
2028   010152   000167   000610   JMP    TST16        ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2029                                     ;AVAILABLE FOR TEST.
2030   010156   012700   000401   MOV    #401,   R0    ;SET UP PARITY "ALL ZEROS" PATTERN
2031   010162   012703   177777   MOV    #-1,   R3    ;SET COM DATA REG
2032   010166   004467   004234   JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2033   010172   004767   006060   1$:   JSR    PC,    W3X9   ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2034   010176   030502           BIT    R5,    R2    ;CHECK FOR END OF A BLOCK.
2035   010200   001374           BNE   1$,    ;BRANCH IF MORE IN CURRENT BLOCK.
2036   010202   004767   004776   JSR    PC,    MMUP   ;FIND NEXT BLOCK AND LOOF TO 1$.
2037
2038 ::*****
2039 ::* CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2040 ::*****
2041   010206   012700   000401   MOV    #401,   R0    ;RESET PARITY "ALL ZEROS" PATTERN.
2042   010212   012703   177777   MOV    #-1,   R3    ;RESET PARITY ALL ONES PATTERN.
2043   010216   004467   004204   JSR    R4,    INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2044   010222   012704   000100   11$:  MOV    #64.,   R4    ;SET 256. WORD COUNTER
2045   010226           12$:
2046   010226   012201           MOV    (R2)+,  R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2047   010230   020001           CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2048   010232   001405           BEQ   65$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2049   010234   004767   010126   64$:  JSR    PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2050   010240   004767   011374   JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2051   010244   000007           .WORD  7          ;ERROR TYPE CODE.
2052   010246           65$:
2053   010246   012201           MOV    (R2)+,  R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2054   010250   020001           CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2055   010252   001405           BEQ   67$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2056   010254   004767   010106   66$:  JSR    PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2057   010260   004767   011354   JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2058   010264   000007           .WORD  7          ;ERROR TYPE CODE.
2059   010266           67$:
2060   010266   012201           MOV    (R2)+,  R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2061   010270   020001           CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2062   010272   001405           BEQ   69$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2063   010274   004767   010066   68$:  JSR    PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2064   010300   004767   011334   JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2065   010304   000007           .WORD  7          ;ERROR TYPE CODE.
2066   010306           69$:
2067   010306   012201           MOV    (R2)+,  R1    ;GET THE DATA FROM MEMORY UNDER TEST.
2068   010310   020001           CMP    R0,    R1    ;COMPARE THE CHECK WORD WITH THE DATA READ.
2069   010312   001405           BEQ   71$,    ;BRANCH OVER ERROR CALL IF GOOD DATA.
2070   010314   004767   010046   70$:  JSR    PC,    SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2071   010320   004767   011314   JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2072   010324   000007           .WORD  7          ;ERROR TYPE CODE.
2073   010326           71$:
2074   010326   010046           MOV    R0,    -(SP) ;SAVE R0
2075   010330   010300           MOV    R3,    R0    ;PUT R3 INTO R0
2076   010332   012603           MOV    (SP)+,  R3   ;PUT SAVED R0 INTO R3
  
```



# K10

CZQMCFD 0-124K MEMORY EXERCISER. 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T15

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 45  
 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

SEQ 0127

```

2077 010334 005304          DEC      R4          ;COUNT 256. WORDS
2078 010336 001333          BNE     12$         ;BRANCH IF MORE
2079 010340 010046          MOV     R0,        -(SP) ;SAVE R0
2080 010342 010300          MOV     R3,        R0   ;PUT R3 INTO R0
2081 010344 012603          MOV     (SP)+,    R3   ;PUT SAVED R0 INTO R3
2082 010346 030502          BIT     R5,        R2   ;CHECK FOR END OF 2 BLOCK.
2083 010350 001324          BNE     11$         ;BRANCH IF MORE IN CURRENT BLOCK.
2084 010352 004767 004626   JSR     PC,        MMUP ;FIND NEXT BLOCK AND LOOP TO 11$.
2085
2086
2087
2088
2089 010356 012700 000401     MOV     #401,     R0   ;SET UP PARITY "ALL ZEROS" PATTERN.
2090 010362 012703 177777     MOV     #-1,     R3   ;SET UP ALL ONES PATTERN.
2091 010366 004467 004034     JSR     R4,        INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2092 010372 012704 000100     21$:   MOV     #64.,R4    ;SET 256. WORD COUNTER
2093 010376
2094 010376 012201          MOV     (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2095 010400 020001          CMP     R0,        R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2096 010402 001405          BEQ     73$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2097 010404 004767 007756     72$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2098 010410 004767 011224     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2099 010414 000007          .WORD   7           ;ERROR TYPE CODE.
2100 010416
2101 010416 005100          COM     R0          ;COMPLEMENT CHECK WORD
2102 010420 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2103 010422 012201          MOV     (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2104 010424 020001          CMP     R0,        R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2105 010426 001405          BEQ     75$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2106 010430 004767 007732     74$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2107 010434 004767 011200     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2108 010440 000007          .WORD   7           ;ERROR TYPE CODE.
2109 010442
2110 010442 005100          COM     R0          ;COMPLEMENT CHECK WORD
2111 010444 005142          COM     -(R2)       ;RESTORE DATA
2112 010446 012201          MOV     (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2113 010450 020001          CMP     R0,        R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2114 010452 001405          BEQ     77$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2115 010454 004767 007706     76$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2116 010460 004767 011154     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2117 010464 000007          .WORD   7           ;ERROR TYPE CODE.
2118 010466
2119 010466 012201          MOV     (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2120 010470 020001          CMP     R0,        R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2121 010472 001405          BEQ     79$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2122 010474 004767 007666     78$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2123 010500 004767 011134     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2124 010504 000007          .WORD   7           ;ERROR TYPE CODE.
2125 010506
2126 010506 005100          COM     R0          ;COMPLEMENT CHECK WORD
2127 010510 005142          COM     -(R2)       ;COMPLEMENT TEST DATA
2128 010512 012201          MOV     (R2)+,    R1   ;GET THE DATA FROM MEMORY UNDER TEST.
2129 010514 020001          CMP     R0,        R1   ;COMPARE THE CHECK WORD WITH THE DATA READ.
2130 010516 001405          BEQ     81$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2131 010520 004767 007642     80$:   JSR     PC,        SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2132 010524 004767 011110     JSR     PC,        $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)

```

2133	010530	000007			.WORD	7		; ERROR TYPE CODE.
2134	010532		81\$:		COM	RO		; COMPLEMENT CHECK WORD
2135	010532	005100			COM	-(R2)		; RESTORE DATA
2136	010534	005142			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2137	010536	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2138	010540	020001			BEQ	83\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2139	010542	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2140	010544	004767	007616		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2141	010550	004767	011064		.WORD	7		; ERROR TYPE CODE.
2142	010554	000007						
2143	010556		83\$:		MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2144	010556	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2145	010560	020001			BEQ	85\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2146	010562	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2147	010564	004767	007576		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE F. MESSAGE)
2148	010570	004767	011044		.WORD	7		; ERROR TYPE CODE.
2149	010574	000007						
2150	010576		85\$:		COM	RO		; COMPLEMENT CHECK WORD
2151	010576	005100			COM	-(R2)		; COMPLEMENT TEST DATA
2152	010600	005142			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2153	010602	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2154	010604	020001			BEQ	87\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2155	010606	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2156	010610	004767	007552		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2157	010614	004767	011020		.WORD	7		; ERROR TYPE CODE.
2158	010620	000007						
2159	010622		87\$:		COM	RO		; COMPLEMENT CHECK WORD
2160	010622	005100			COM	-(R2)		; RESTORE DATA
2161	010624	005142			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2162	010626	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2163	010630	020001			BEQ	89\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2164	010632	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2165	010634	004767	007526		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2166	010640	004767	010774		.WORD	7		; ERROR TYPE CODE.
2167	010644	000007						
2168	010646		89\$:		MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2169	010646	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2170	010650	020001			BEQ	91\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2171	010652	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2172	010654	004767	007506		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2173	010660	004767	010754		.WORD	7		; ERROR TYPE CODE.
2174	010664	000007						
2175	010666		91\$:		COM	RO		; COMPLEMENT CHECK WORD
2176	010666	005100			COM	-(R2)		; COMPLEMENT TEST DATA
2177	010670	005142			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2178	010672	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2179	010674	020001			BEQ	93\$		; BRANCH OVER ERROR CALL IF GOOD DATA.
2180	010676	001405			JSR	PC,	SPRNT2	; SET UP VALUES FOR ERROR PRINTING.
2181	010700	004767	007462		JSR	PC,	\$ERROR	; *** ERROR *** (GO TYPE A MESSAGE)
2182	010704	004767	010730		.WORD	7		; ERROR TYPE CODE.
2183	010710	000007						
2184	010712		93\$:		COM	RO		; COMPLEMENT CHECK WORD
2185	010712	005100			COM	-(R2)		; RESTORE DATA
2186	010714	005142			MOV	(R2)+,	R1	; GET THE DATA FROM MEMORY UNDER TEST.
2187	010716	012201			CMP	RO,	R1	; COMPARE THE CHECK WORD WITH THE DATA READ.
2188	010720	020001						



M10

CZQMCF0 0-124K MEMORY EXERCISER, 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T15

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 47  
 MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY

SEQ 0129

```

2189 010722 001405          BEQ      95$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2190 010724 004767 007436 94$: JSR      PC,          SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2191 010730 004767 010704 JSR      PC,          $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2192 010734 000007          .WORD    7           ;ERROR TYPE CODE.
2193 010736          95$:
2194 010736 010046          MOV      R0,         -(SP)     ;SAVE R0
2195 010740 010300          MOV      R3,         R0       ;PUT R3 INTO R0
2196 010742 012603          MOV      (SP)+,     R3       ;PUT SAVED R0 INTO R3
2197 010744 005304          DEC      R4           ;DECREMENT 256. WORD COUNTER
2198 010746 001213          BNE     22$          ;BRANCH IF MORE.
2199 010750 010046          MOV      R0,         -(SP)     ;SAVE R0
2200 010752 010300          MOV      R3,         R0       ;PUT R3 INTO R0
2201 010754 012603          MOV      (SP)+,     R3       ;PUT SAVED R0 INTO R3
2202 010756 030502          BIT      R5,         R2       ;CHECK FOR END OF A BLOCK.
2203 010760 001204          BNE     21$          ;BRANCH IF MORE IN CURRENT BLOCK.
2204 010762 004767 004216 JSR      PC,          MMUP     ;FIND NEXT BLOCK AND LOOP TO 21$.
2205
2206 ;*****
2207 ;*TEST 16 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
2208 ;*****
2209 010766          TST16:
2210 010766 004567 007636 JSR      R5,          $SCOPE   ;GO TO SCOPE ROUTINE.
2211 010772 000777          .WORD    777         ;MINIMUM BLOCK SIZE OF 256. WORDS
2212          ;REQUIRED FOR THIS TEST.
2213 010774 000167 000610 JMP      TST17         ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
2214          ;AVAILABLE FOR TEST.
2215 011000 012700 177777 MOV      #-1,        R0       ;SET UP ALL ONES PATTERN
2216 011004 012703 000401 MOV      #401,       R3       ;SET UP PARITY "ALL ZEROS" PATTERN
2217 011010 004467 003412 JSR      R4,          INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2218 011014 004767 005236 1$: JSR      PC,          W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
2219 011020 030502          BIT      R5,         R2       ;CHECK FOR END OF A BLOCK.
2220 011022 001374          BNE     1$          ;BRANCH IF MORE IN CURRENT BLOCK.
2221 011024 004767 004154 JSR      PC,          MMUP     ;FIND NEXT BLOCK AND LOOP TO 1$.
2222
2223 ;*****
2224 ;* CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
2225 ;*****
2226 011030 012700 177777 MOV      #-1,        R0       ;SET UP ALL ONES PATTERN
2227 011034 012703 000401 MOV      #401,       R3       ;SET UP PARITY "ALL ZEROS" PATTERN
2228 011040 004467 003362 JSR      R4,          INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2229 011044 012704 000100 MOV      #64.,       R4       ;SET 256. WORD COUNTER
2230 011050          11$:
2231 011050 012201          12$: MOV      (R2)+,     R1       ;GET THE DATA FROM MEMORY UNDER TEST.
2232 011052 020001          CMP      R0,         R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
2233 011054 001405          BEQ     65$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2234 011056 004767 007304 64$: JSR      PC,          SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2235 011062 004767 010552 JSR      PC,          $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2236 011066 000007          .WORD    7           ;ERROR TYPE CODE.
2237 011070          65$:
2238 011070 012201          MOV      (R2)+,     R1       ;GET THE DATA FROM MEMORY UNDER TEST.
2239 011072 020001          CMP      R0,         R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
2240 011074 001405          BEQ     67$          ;BRANCH OVER ERROR CALL IF GOOD DATA.
2241 011076 004767 007264 66$: JSR      PC,          SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2242 011102 004767 010532 JSR      PC,          $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
2243 011106 000007          .WORD    7           ;ERROR TYPE CODE.
2244 011110          67$:
    
```





2301	011302	004767	010332	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2302	011306	000007		.WORD	7		:ERROR TYPE CODE.
2303	011310					77\$:	
2304	011310	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2305	011312	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2306	011314	001405		BEQ	79\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2307	011316	004767	007044	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2308	011322	004767	010312	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2309	011326	000007		.WORD	7		:ERROR TYPE CODE.
2310	011330					79\$:	
2311	011330	005100		COM	RO		:COMPLEMENT CHECK WORD
2312	011332	005142		COM	-(R2)		:COMPLEMENT TEST DATA
2313	011334	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2314	011336	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2315	011340	001405		BEQ	81\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2316	011342	004767	007020	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2317	011346	004767	010266	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2318	011352	000007		.WORD	7		:ERROR TYPE CODE.
2319	011354					81\$:	
2320	011354	005100		COM	RO		:COMPLEMENT CHECK WORD
2321	011356	005142		COM	-(R2)		:RESTORE DATA
2322	011360	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2323	011362	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2324	011364	001405		BEQ	83\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2325	011366	004767	006774	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2326	011372	004767	010242	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2327	011376	000007		.WORD	7		:ERROR TYPE CODE.
2328	011400					83\$:	
2329	011400	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2330	011402	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2331	011404	001405		BEQ	85\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2332	011406	004767	006754	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2333	011412	004767	010222	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2334	011416	000007		.WORD	7		:ERROR TYPE CODE.
2335	011420					85\$:	
2336	011420	005100		COM	RO		:COMPLEMENT CHECK WORD
2337	011422	005142		COM	-(R2)		:COMPLEMENT TEST DATA
2338	011424	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2339	011426	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2340	011430	001405		BEQ	87\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2341	011432	004767	006730	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2342	011436	004767	010176	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2343	011442	000007		.WORD	7		:ERROR TYPE CODE.
2344	011444					87\$:	
2345	011444	005100		COM	RO		:COMPLEMENT CHECK WORD
2346	011446	005142		COM	-(R2)		:RESTORE DATA
2347	011450	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2348	011452	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2349	011454	001405		BEQ	89\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2350	011456	004767	006704	JSR	PC,	SPRINT2	:SET UP VALUES FOR ERROR PRINTING.
2351	011462	004767	010152	JSR	PC,	SERROR	*** ERROR *** (GO TYPE A MESSAGE)
2352	011466	000007		.WORD	7		:ERROR TYPE CODE.
2353	011470					89\$:	
2354	011470	012201		MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2355	011472	020001		CMP	RO,	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2356	011474	001405		BEQ	91\$		:BRANCH OVER ERROR CALL IF GOOD DATA.



## C11

CZQMCFD 0-124K MEMORY EXERCISER. 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T16

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 50  
 COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.

SEG 0132

2357	011476	004767	006664	90\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2358	011502	004767	010132		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2359	011506	000007			.WORD	7		:ERROR TYPE CODE.
2360	011510			91\$:				
2361	011510	005100			COM	RO		:COMPLEMENT CHECK WORD
2362	011512	005142			COM	-(R2)		:COMPLEMENT TEST DATA
2363	011514	012201			MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2364	011516	020001			CMP	RO	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2365	011520	001405			BEQ	93\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2366	011522	004767	006640	92\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2367	011526	004767	010106		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2368	011532	000007			.WORD	7		:ERROR TYPE CODE.
2369	011534			93\$:				
2370	011534	005100			COM	RO		:COMPLEMENT CHECK WORD
2371	011536	005142			COM	-(R2)		:RESTORE DATA
2372	011540	012201			MOV	(R2)+,	R1	:GET THE DATA FROM MEMORY UNDER TEST.
2373	011542	020001			CMP	RO	R1	:COMPARE THE CHECK WORD WITH THE DATA READ.
2374	011544	001405			BEQ	95\$		:BRANCH OVER ERROR CALL IF GOOD DATA.
2375	011546	004767	006614	94\$:	JSR	PC,	SPRNT2	:SET UP VALUES FOR ERROR PRINTING.
2376	011552	004767	010062		JSR	PC,	SERROR	:*** ERROR *** (GO TYPE A MESSAGE)
2377	011556	000007			.WORD	7		:ERROR TYPE CODE.
2378	011560			95\$:				
2379	011560	010046			MOV	RO,	-(SP)	:SAVE RO
2380	011562	010300			MOV	R3,	RO	:PUT R3 INTO RO
2381	011564	012603			MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
2382	011566	005304			DEC	R4		:DECREMENT 256. WORD COUNTER
2383	011570	001213			BNE	22\$		:BRANCH IF MORE.
2384	011572	010046			MOV	RO,	-(SP)	:SAVE RO
2385	011574	010300			MOV	R3,	RO	:PUT R3 INTO RO
2386	011576	012603			MOV	(SP)+,	R3	:PUT SAVED RO INTO R3
2387	011600	030502			BIT	R5	R2	:CHECK FOR END OF A BLOCK.
2388	011602	001204			BNE	21\$		:BRANCH IF MORE IN CURRENT BLOCK.
2389	011604	004767	003374		JSR	PC,	MMUP	:FIND NEXT BLOCK AND LOOP TO 21\$.



```

2390
2391
2392
2393
2394
2395
2396
2397
2398 011610
2399 011610 004567 007014
2400 011614 000000
2401 011616 005767 170454
2402 011622 001404
2403 011624 032777 000100 167306
2404 011632 001402
2405 011634 000167 000622
2406 011640 005000
2407 011642 004767 004322
2408 011646 004467 002554
2409 011652 036767 167666 167660
2410 011660 001010
2411 011662 036767 167660 167652
2412 011670 001004
2413 011672 050502
2414 011674 005202
2415 011676 000167 000540
2416 011702 004767 005674
2417 011706 004767 005724
2418 011712 020227 000114
2419 011716 001004
2420 011720 062702 000004
2421 011724 000167 000512
2422 011730 111201
2423 011732 001405
2424 011734 004767 006352
2425 011740 004767 007674
2426 011744 000011
2427 011746
2428 011746 105067 167606
2429 011752 112700 000252
2430 011756 110012
2431 011760 016703 167644
2432 011764 056773 167622 000000
2433 011772 052733 000001
2434 011776 005713
2435 012000 001371
2436
2437 012002 110012
2438 012004 016703 167620
2439 012010 046733 167576
2440 012014 005713
2441 012016 001374
2442 012020 016737 167606 000114
2443
2444 012026 105412
2445

```

```

*****
*TEST 17 WORSE CASE NOISE PARITY BYTE TESTING
* CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
* 1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
* 2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
* 3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
* 4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
*****
TST17:
JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
MWPB0: TST MPRX ;CHECK FOR ANY PARITY MEMORY.
BEQ 1$ ;BR IF NO PARITY MEMORY.
BIT #SW06, 2SWR ;CHECK FORINHIBIT PARITY SWITCH.
BEQ 2$ ;BR IF NOT SET.
1$: JMP TST20 ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2$: CLR R0 ;ZERO TO BE PUT IN ALL MEMORY.
JSR PC, SETCON ;ROUTINE TO LOAD ALL MEMORY.
MWPBYT: BIT BITPT, PMEMAP ;INITIALIZE THE MEMORY ADDRESS POINTERS.
BNE 2$ ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
BIT BITPT+2, PMEMAP+2 ;BR IF PARITY MEM.
BNE 2$ ;...HI 64K.
BIS R5, R2 ;BR IF PARITY MEM.
INC R2 ;POINT TO END OF BLOCK.
JMP WWPB5 ;FIRST ADR OF NEXT BLOCK.
2$: JSR PC, SETAE ;BR TO FIND NEXT BLOCK.
JSR PC, CKPMER ;SET ACTION ENABLE (EVEN IF BANK0.)
MWPB1: CMP R2, #114 ;CHECK FOR ANY NON TRAP PARITY ERRORS.
BNE 3$ ;CHECK IF POINTING TO PARITY ERROR VECTOR.
ADD #4, R2 ;BR IF NOT AT VECTOR.
JMP WWPB5 ;SKIP PARITY VECTOR.
3$: MOV (R2), R1 ;CHECK FOR BLOCK END.
BEQ 65$ ;CHECK IF BYTE STILL CLEARED.
64$: JSR PC, SPRNT ;BRANCH OVER ERROR CALL IF GOOD DATA.
JSR PC, SERROR ;SET UP VALUES FOR ERROR PRINTING.
WORD 11 ;*** ERROR *** (GO TYPE A MESSAGE)
65$: CLRB OEFLG ;CLEAR ODD/EVEN FLAG.
MOV #252, R0 ;SET UP DATA...EVEN, SETS PARITY BIT.
MWPB2: MOV R0, (R2) ;MOV DATA INTO TEST LOCATION.
MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
10$: BIS WWP, 2(R3) ;SET WRITE WRONG PARITY.
BIS #AE, 2(R3)+
TST (R3) ;CHECK FOR TABLE TERMINATOR.
BNE 10$ ;BR IF MORE REGS IN TABLE.
;* SET WRONG PARITY IN LOCATION UNDER TEST.
MOV R0, (R2) ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
11$: MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
BIC WWP, 2(R3)+ ;CLEAR WRITE WRONG PARITY.
TST (R3) ;CHECK FOR TABLE TERMINATOR.
BNE 11$ ;BR IF MORE PARITY REGISTERS.
MOV .PBTRP, 2#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
;* DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
NEGB (R2) ;DATIP (DATOB AND COM PARITY BIT.)
;* SHOULD HAVE TRAPPED TO PBTRP.

```

# E11

CZQMCF0 0-124K MEMORY EXERCISER. 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T17

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 52  
 WORSE CASE NOISE PARITY BYTE TESTING

SEQ 0134

```

2446 012030 016737 167602 000114      MOV    .PESRV, 2#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
2447 012036 004767 006300      JSR    PC,    SPRNTD ;SET UP VALUES FOR ERROR PRINTING.
2448 012042 004767 007572      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2449 012046 000012      .WORD 12 ;ERROR TYPE CODE.
2450 012050 000562      BR     WWPB4 ;SKIP TRAP SERVICE.
2451
2452
2453 012052 016737 167560 000114  ;* EXPECTED PARITY MEMORY TRAPS COME HERE.
2454 012060 022626      PBTRP: MOV    .PESRV, 2#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
2455 012062 016703 167540      CMP    (SP)+, (SP)+ ;RESET THE STACK POINTER AFTER TRAP.
2456 012066 032713 000001      MOV    .MPRO, R3 ;GET PARITY REG AND MAP TABLE POINTER.
2457 012072 001003      BIT    #BIT0, (R3) ;CHECK IF THIS REGISTER EXISTS.
2458 012074 017301 000000      BNE    22$, ;BR IF IT DOESN'T EXIST.
2459 012100 100413      MOV    2(R3), R1 ;GET THE CONTENTS.
2460 012102 062703 000010      BMI    23$, ;BR IF ERROR FLAG SET.
2461 012106 020367 167516      ADD    #10, R3 ;MOVE POINTER TO NEXT REG.
2462 012112 103765      CMP    R3, .MPRX ;CHECK FOR END OF TABLE.
2463 012114 004767 006222      BLO    21$, ;BR IF MORE REGISTERS.
2464 012120 004767 007514      JSR    PC,    SPRNTD ;SET UP VALUES FOR ERROR PRINTING.
2465 012124 000013      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2466 012126 000533      .WORD 13 ;ERROR TYPE CODE.
2467 012130 036763 167410 000002 23$: BR     WWPB4 ;EXIT AFTER ERROR.
2468 012136 001011      BIT    BITPT, 2(R3) ;CHECK THE MAP FOR THIS REGISTER.
2469 012140 036763 167402 000004 24$: BNE    24$, ;BR IF THIS REGISTER CONTROLS THIS BANK.
2470 012146 001005      BIT    BITPT+2,4(R3) ;CHECK THE HI 64K.
2471 012150 004767 006162 65$: BNE    24$, ;BR IF THIS REGISTER CONTROLS THIS BANK.
2472 012154 004767 007460      JSR    PC,    SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2473 012160 000014      JSR    PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2474 012162      .WORD 14 ;ERROR TYPE CODE.
2475 012162 010046      24$: MOV    R0,-(SP) ;:PUSH R0 ON STACK
2476 012164 010200      MOV    R2,R0 ;GET THE ADDRESS POINTER.
2477 012166 042700 003777      BIC    #3777, R0 ;CLEAR LOW ADDRESS BITS.
2478 012172 000300      SWAB  R0 ;SHIFT 6 PLACES RIGHT.
2479 012174 006300      ASL   R0
2480 012176 006300      ASL   R0
2481 012200 005767 166402      TST   MMAVA ;CHECK FOR MEM MGMT.
2482 012204 001404      BEQ   25$, ;BR IF NO MEM MGMT.
2483 012206 042700 177600      BIC   #177600,R0 ;CLEAR BANK BITS
2484 012212 063700 172344      ADD   2#KIPAR2,R0 ;ADD MEM MGMT OFFSET.
2485 012216 052700 100001 25$: BIS   #BIT15+BIT0,R0 ;SET ERROR AND AE BIT IN CHECK WORD.
2486 012222 016367 000006 167266      MOV   6(R3), RESRVD ;GET APPROPRIATE MASK.
2487 012230 046700 167262      BIC   RESRVD, R0 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2488 012234 046701 167256      BIC   RESRVD, R1 ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
2489 ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
2490 012240 020001      CMP   R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2491 012242 001405      BEQ   67$, ;BRANCH OVER ERROR CALL IF GOOD DATA.
2492 012244 004767 006066 66$: JSR   PC,    SPRNTP ;SET UP VALUES FOR ERROR PRINTING.
2493 012250 004767 007364      JSR   PC,    $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2494 012254 000015      .WORD 15 ;ERROR TYPE CODE.
2495 012256      67$:
2496 012256 005073 000000      CLR   2(R3) ;CLEAR REG INCLUDING ACTION ENABLE.
2497 012262 010346      MOV   R3,-(SP) ;:PUSH R3 ON STACK
2498 012264 062703 000010 26$: ADD   #10, R3 ;UPDATE POINTER TO NEXT PARITY REG + MAP.
2499 012270 020367 167334      CMP   R3, .MPRX ;CHECK FOR END OF TABLE.
2500 012274 101014      BHI   WWPB3 ;BR IF END OF TABLE REACHED.
2501 012276 032713 000001      BIT   #BIT0, (R3) ;CHECK IF NEXT REG EXISTS.
  
```







# G11

CZQMCFD 0-124K MEMORY EXERCISER. 16K VER  
 CZQMCF.P11 14-FEB-78 08:19 T20

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 54  
 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.

SEQ 0136

```

2547
2548
2549
2550 012462
2551 012462 004567 006142
2552 012466 000000
2553 012470 010703
2554 012472 042703 007777
2555 012476 004467 001724
2556 012502 010246
2557 012504 010346
2558 012506 012322
2559 012510 032703 007777
2560 012514 001002
2561 012516 162703 010000
2562 012522 030502
2563 012524 001370
2564 012526 012603
2565 012530 012602
2566 012532 012300
2567 012534 012201
2568 012536 020001
2569 012540 001405
2570 012542 004767 005620
2571 012546 004767 007066
2572 012552 000020
2573 012554
2574 012554 032703 007777
2575 012560 001002
2576 012562 162703 010000
2577 012566
2578 012566 030502
2579 012570 001360
2580 012572 004767 002406

```

```

*****
*TEST 20 RANDOM DATA TESTING THRU PROGRAM CODE RELOCATION.
*****
↑ST20:
      JSR      R5,      $SCOPE ;GO TO SCOPE ROUTINE.
      .WORD   0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
RANTST: MOV     PC,      R3 ;GET CURRENT PROGRAM COUNTER.
      BIC     #7777, R3 ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
      JSR     R4,      INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:   MOV     R2,      -(SP) ;SAVE MEMORY POINTER.
      MOV     R3,      -(SP) ;SAVE "DATA" POINTER.
2$:   MOV     (R3)+, (R2)+ ;MOV CODE INTO TEST MEMORY.
      BIT     #7777, R3 ;CHECK FOR END OF "DATA TABLE"
      BNE    3$ ;BRANCH IF MORE
      SUB     #10000, R3 ;RESET POINTER TO START OF "RANDOM DATA"
3$:   BIT     R5,      R2 ;CHECK FOR END OF BLOCK
      BNE    2$ ;BRANCH IF MORE.
      MOV     (SP)+, R3 ;RESET "DATA" POINTER.
      MOV     (SP)+, R2 ;RESET MEMORY POINTER.
4$:   MOV     (R3)+, R0 ;GET S/B DATA.
      MOV     (R2)+, R1 ;GET THE DATA FROM MEMORY UNDER TEST.
      CMP     R0,      R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ    65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:  JSR     PC,      SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   20 ;ERROR TYPE CODE.
65$:  BIT     #7777, R3 ;CHECK FOR END OF "DATA TABLE"
      BNE    5$ ;BR IF MORE.
      SUB     #10000, R3 ;RESET POINTER TO TOP OF "DATA TABLE".
5$:   BIT     R5,      R2 ;CHECK FOR END OF A BLOCK.
      BNE    4$ ;BRANCH IF MORE IN CURRENT BLOCK.
      JSR     PC,      MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.

```

.SBTTL SECTION 3: INSTRUCTION EXECUTION TESTS.

\*\*\*\*\*  
\*TEST 21 EXECUTE DATA, DATA 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 PLACED THERE	CONTENTS OF MEMORY LOCATION 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.

```

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 012576
2608 012576 004567 006026
2609 012502 000003
2610
2611 012604 000167 000056
2612
2613 012610 012703 010412
2614 012614 012704 000205
2615 012620 010400
2616 012622 004467 001600
2617 012626 010322
2618 012630 010412
2619 012632 004542
2620 012634 012201
2621 012636 020001
2622 012640 001405
2623 012642 004767 005514
2624 012646 004767 006766
2625 012652 000021
2626 012654
2627 012654 010322
2628 012656 030502
2629 012660 001363
2630 012662 004767 002316

```

```

*****
TST21: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
        .WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
        ;REQUIRED FOR THIS TEST.
        JMP TST22 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
        ;AVAILABLE FOR TEST.
DIDO: MOV #010412, R3 ;GET 'MOV R4, (R2)' INSTRUCTION (IUT).
        MOV #205, R4 ;GET 'RTS R5'
        MOV R4, R0 ;SET UP S/B DATA AFTER EXECUTION.
        JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: 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$.

```

```

2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656 012666
2657 012666 004567 005736
2658 012672 000003
2659
2660 012674 000167 000060
2661
2662 012700 012703 110412
2663 012704 012704 000205
2664 012710 012700 110605
2665 012714 004467 001506
2666 012720 010322
2667 012722 010412
2668 012724 004542
2669 012726 012201
2670 012730 020001
2671 012732 001405
2672 012734 004767 005422
2673 012740 004767 006674
2674 012744 000021
2675 012746
2676 012746 010322
2677 012750 030502
2678 012752 001363
2679 012754 004767 002224

```

```

*****
*TEST 22 EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
* EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
* AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
* CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
* THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
*
* MEMORY LOCATION INSTRUCTION CONTENTS OF MEMORY LOCATION
* LOCATION PLACED THERE AFTER INSTRUCTION EXECUTION
*
* 1ST PASS / 40000 110412 110605
* THRU TEST / 40002 000205 000205
*
* 2ND PASS / 40002 110412 110605
* THRU TEST / 40004 000205 000205
*
* ETC., ETC., ETC.
*
* R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
* R1 = DATA READ FROM MEMORY (WAS).
* R2 = ADDRESS OF IUT/DATA.
* R3 = INSTRUCTION UNDER TEST (IUT).
* R4 = RTS R5 (CODE 205).
* R5 = BLOCK BOUNDARY BIT MASK.
*****
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$.

```



2680  
2681  
2682  
2683  
2684  
2685  
2686  
2687  
2688  
2689  
2690  
2691  
2692  
2693  
2694  
2695  
2696  
2697  
2698  
2699  
2700  
2701  
2702  
2703  
2704  
2705  
2706  
2707  
2708  
2709  
2710  
2711  
2712  
2713  
2714  
2715  
2716  
2717  
2718  
2719  
2720  
2721  
2722  
2723  
2724  
2725  
2726  
2727  
2728  
2729

012760  
012760 004567 544  
012764 000003  
012766 000167 000064  
012772 012703 110342  
012776 012704 000205  
013002 012700 161342  
013006 004467 001414  
013012 010322  
013014 010412  
013016 004562 177776  
013022 005302  
013024 012201  
013026 020001  
013030 001405  
013032 004767 005324  
013036 004767 006576  
013042 000021  
013044  
013044 010322  
013046 030502  
013050 001361  
013052 004767 002126

```
*****
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$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
        JSR R5, -(R2) ;GO EXECUTE THE IUT.
        DEC R2 ;ADJUST R2 TO POINT TO MAUT.
        MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, 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$.
```

# K11

2730  
2731  
2732  
2733  
2734  
2735  
2736  
2737  
2738  
2739  
2740  
2741  
2742  
2743  
2744  
2745  
2746  
2747  
2748  
2749  
2750  
2751  
2752  
2753  
2754  
2755  
2756  
2757  
2758  
2759  
2760  
2761  
2762  
2763  
2764  
2765  
2766  
2767  
2768  
2769  
2770  
2771  
2772  
2773  
2774  
2775  
2776  
2777  
2778

013056			005546
013056	004567		
013062	000003		
013064	000167	000060	
013070	012703	005412	
013074	012704	000205	
013100	012700	172366	
013104	004467	001316	
013110	010322		
013112	010412		
013114	004542		
013116	012201		
013120	020001		
013122	001405		
013124	004767	005232	
013130	004767	006504	
013134	000021		
013136			
013136	010322		
013140	030502		
013142	001363		
013144	004767	002034	

```

*****
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 LOCATION      INSTRUCTION      CONTENTS OF MEMORY LOCATION
      LOCATION            PLACED THERE      AFTER INSTRUCTION EXECUTION

1ST PASS / 40000          005412           172366
THRU TEST / 40002          000205           000205

2ND PASS / 40002          005412           172366
THRU TEST / 40004          000205           000205

      ETC., ETC., ETC.

R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
R1 = DATA READ FROM MEMORY (WAS).
R2 = ADDRESS OF IUT/DATA.
R3 = INSTRUCTION UNDER TEST (IUT).
R4 = RTS R5 (CODE 205).
R5 = BLOCK BOUNDARY BIT MASK.
*****
  
```

```

†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, 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$.
  
```



```

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 013150
2805 013150 004567 005454
2806 013154 000003
2807
2808 013156 000167 000060
2809
2810 013162 012703 142242
2811 013166 012704 000205
2812 013172 012700 142000
2813 013176 004467 001224
2814 013202 010322
2815 013204 010412
2816 013206 004542
2817 013210 012201
2818 013212 020001
2819 013214 001405
2820 013216 004767 005140
2821 013222 004767 006412
2822 013226 000021
2823 013230
2824 013230 010322
2825 013232 030502
2826 013234 001363
2827 013236 004767 001742

```

```

*****
*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 PLACED THERE	CONTENTS OF MEMORY LOCATION AFTER INSTRUCTION EXECUTION
1ST PASS / THRU TEST	40000 / 40002	142242 / 000205	142000 / 000205
2ND PASS / THRU TEST	40002 / 40004	142242 / 000205	142000 / 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, 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$.

```



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  
2871  
2872  
2873  
2874  
2875  
2876  
2877

013242  
013242 004567 005362  
013246 000003  
013250 000167 000062  
013254 012703 152212  
013260 012704 000205  
013264 012700 157212  
013270 004467 001132  
013274 010322  
013276 010412  
013300 004542  
013302 005302  
013304 012201  
013306 020001  
013310 001405  
013312 004767 005044  
013316 004767 006316  
013322 000021  
013324  
013324 010322  
013326 030502  
013330 001362  
013332 004767 001646

```
*****
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
PLACED THERE AFTER INSTRUCTION EXECUTION
1ST PASS / 40000 152212 157212
THRU TEST / 40002 000205 000205
2ND PASS / 40002 152212 157212
THRU TEST / 40004 000205 000205
ETC., ETC., ETC.
R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
R1 = DATA READ FROM MEMORY (WAS).
R2 = ADDRESS OF IUT/DATA.
R3 = INSTRUCTION UNDER TEST (IUT).
R4 = RTS R5 (CODE 205).
R5 = BLOCK BOUNDARY BIT MASK.
*****
```

```
TST26: JSR R5, $SCOPE ;GO TO SCOPE ROUTINE.
.WORD 3 ;MINIMUM BLOCK SIZE OF 2 WORDS
REQUIRED FOR THIS TEST.
JMP TST27 ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
AVAILABLE FOR TEST.
DPDBH: MOV #152212, R3 ;GET 'BISB (R2)+, (R2)' INSTRUCTION (IUT).
MOV #205, R4 ;GET 'RTS R5'
MOV #157212, R0 ;SET UP S/B DATA AFTER EXECUTION.
R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$: MOV R3, (R2)+ ;PUT IUT INTO FIRST LOC OF BLOCK.
2$: MOV R4, (R2) ;PUT 'RTS R5' FOLLOWING IUT.
JSR R5, -(R2) ;GO EXECUTE THE IUT.
DEC R2 ;RESET R2 TO POINT TO IUT.
MOV (R2)+, R1 ;GET THE DATA FROM THE MEM ADR UNDER TEST.
CMP R0, R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
BEQ 65$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$: JSR PC, 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$.
```

```

2878
2879
2880
2881
2882
2883
2884
2885
2886
2887
2888
2889
2890
2891
2892
2893
2894
2895
2896
2897
2898
2899
2900
2901
2902
2903 013336
2904 013336 004567 005256
2905 013342 000000
2906 013344 004467 001056
2907 013350 010267 166240
2908 013354 005003
2909 013356 012700 000377
2910 013362 010022
2911 013364 030502
2912 013366 001375
2913 013370 014201
2914 013372 020001
2915 013374 001405
2916 013376 004767 004764
2917 013402 004767 006232
2918 013406 000010
2919 013410
2920 013410 000300
2921 013412 010012
2922 013414 011201
2923 013416 020001
2924 013420 001405
2925 013422 004767 004740
2926 013426 004767 006206
2927 013432 000010
2928 013434
2929 013434 000300
2930 013436 005703
2931 013440 001403
2932 013442 020327 000003
2933 013446 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.
*****

```

```

†ST27:
      JSR      R5,      $SCOPE      ;GO TO SCOPE ROUTINE.
      .WORD   0              ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      JSR      R4,      INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      MOV      R2,TEMP        ;SAVE BANK STARTING ADDRESS
      CLR      R3              ;CLEAR PASS COUNTER
      MOV      #000377,R0      ;SETUP TO WRITE PATTERN
      MOV      R0,(R2)+        ;WRITE PATTERN
      BIT      R5,R2          ;END OF 4K?
      BNE     2$              ;CONTINUE WRITING IF NO.
      MOV      -(R2),R1       ;GET DATA WRITTEN
      CMP      R0, R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR     PC,      SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   10              ;ERROR TYPE CODE.

      SWAB    R0              ;SWAP BYTES OF DATA
      MOV     R0,(R2) ;WRITE SWAPPED WORD
      MOV     (R2),R1         ;GET DATA WRITTEN
      CMP     R0, R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
      BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
      JSR     PC,      SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
      JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
      .WORD   10              ;ERROR TYPE CODE.

      SWAB    R0              ;PUT DATA BACK TO ORIGINAL
      TST     R3              ;IF ON PASS 0 OR PASS 3
      BEQ     5$              ;WE ARE ADDRESSING DOWN
      CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
      BNE     6$              ;UPWARD

```



```

2934 013450 030502 58: BIT R5,R2 ;DONE A PASS?
2935 013452 001346 BNE 38 ;IF NO CONTINUE
2936 013454 005203 INC R3 ;IF YES INCREMENT PASS COUNTER
2937 013456 022703 000004 CMP #4,R3 ;ARE WE DONE ALL PASSES FOR THIS 4K?
2938 013462 001427 BEQ 98 ;IF YES BRANCH
2939 013464 000300 SWAB R0 ;ELSE SET UP NEW READ WORD
2940 013466 000404 BR 78 ;GO TO START OF ADDRESS UP
2941 013470 062702 000002 68: ADD #2,R2 ;UPDATE TO NEXT ADDRESS
2942 013474 030502 BIT R5,R2 ;DONE A PASS
2943 013476 001411 BEQ 88 ;IF YES BRANCH
2944 013500 011201 78: MOV (R2),R1 ;GET DATA WRITTEN
2945 013502 020001 CMP R0,R1 ;COMPARE THE CHECK WORD WITH THE DATA READ.
2946 013504 001405 BEQ 69$ ;BRANCH OVER ERROR CALL IF GOOD DATA.
2947 013506 004767 004654 68$: JSR PC,SPRNT2 ;SET UP VALUES FOR ERROR PRINTING.
2948 013512 004767 006122 JSR PC,$ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
2949 013516 000010 .WORD 10 ;ERROR TYPE CODE.
2950 013520 69$:
2951 013520 000733 BR 48
2952 013522 005203 88: INC R3 ;INCREMENT PASS COUNTER
2953 013524 000300 SWAB R0 ;SET UP NEW READ WORD
2954 013526 020327 000002 CMP R3,#2 ;ADDRESSING UP?
2955 013532 001316 BNE 38 ;IF NO GO TO DOWN SEQUENCE
2956 013534 016702 166054 MOV TEMP,R2 ;IF YES RESET ADDRESS TO START
2957 013540 000757 BR 78 ;GO TO UP SEQUENCE
2958 013542 004467 000660 98: JSR R4,INITMM ;INITIALIZE MEMORY ADDRESS POINTERS
2959 013546 004767 001432 JSR PC,MMUP ;UPDATE TO NEW BANK IF EXISTS
2960
2961 *****
2962 *TEST 30 WRITE CHECKERBOARD STARTING WITH '125252' DATA.
2963 * THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
2964 * FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
2965 * DETERIORATE BETWEEN REFRESH CYCLES.
2966 *
2967 * RO=DATA WRITTEN INTO MEMORY(SHOULD BE)
2968 * R1=DATA READ FROM MEMORY(WAS)
2969 * R2=VIRTUAL ADDRESS
2970 * R3=SMALL LOOP COUNTER FOR STALL
2971 * R4=NUMBER OF TIMES SMALL LOOP DONE
2972 * R5=BLOCK BOUNDARY BIT MASK.
2973 *****
2974 013552 †ST30:
2975 013552 004567 005052 JSR R5,$SCOPE ;GO TO SCOPE ROUTINE.
2976 013556 000000 .WORD 0 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2977 013560 004467 000642 JSR R4,INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2978 013564 012700 125252 MOV #125252,R0 ;SETUP DATA PATTERN
2979 013570 010022 18: MOV R0,(R2)+ ;WRITE A WORD
2980 013572 005100 COM R0 ;COMPLEMENT DATA
2981 013574 030502 BIT R5,R2 ;CHECK FOR END OF A BLOCK.
2982 013576 001374 BNE 18 ;BRANCH IF MORE IN CURRENT BLOCK.
2983 013600 004767 001400 JSR PC,MMUP ;FIND NEXT BLOCK AND LOOP TO 18.
2984 013604 005003 CLR R3 ;SET UP COUNTER FOR STALL
2985 013606 012704 000046 28: MOV #46,R4 ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
2986 013612 005303 DEC R3
2987 013614 001376 BNE 28
2988 013616 005304 DEC R4
2989 013620 001374 BNE 28

```



```

2990 013622 004467 000600      JSR   R4,   INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2991 013626 012700 125252      MOV   #125252,R0    ;INIT DATA FOR CHECKING
2992 013632                    3$:
2993 013632 012201                    MOV   (R2)+, R1     ;GET THE DATA FROM MEMORY UNDER TEST.
2994 013634 020001                    CMP   R0,   R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
2995 013636 001405                    BEQ   65$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
2996 013640 004767 004522      JSR   PC,   SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
2997 013644 004767 005770      JSR   PC,   $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2998 013650 000006                    .WORD 6            ;ERROR TYPE CODE.
2999 013652                    65$:
3000 013652 005100                    COM   R0
3001 013654 030502                    BIT   R5,   R2     ;CHECK FOR END OF A BLOCK.
3002 013656 001365                    BNE   3$         ;BRANCH IF MORE IN CURRENT BLOCK.
3003 013660 004767 001320      JSR   PC,   MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
3004
3005 ::*****
3006 :*TEST 31 WRITE CHECKERBOARD STARTING WITH 052525 L.ATA
3007 :*****
3008 †ST31:
3009 JSR   R5,   $SCOPE ;GO TO SCOPE ROUTINE.
3010 .WORD 0           ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3011 JSR   R4,   INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3012 MOV   #052525,R0   ;SETUP DATA PATTERN
3013 MOV   R0,   (R2)+ ;WRITE A WORD
3014                    1$:
3015 COM   R0
3016 BIT   R5,   R2     ;CHECK FOR END OF A BLOCK.
3017 BNE   1$         ;BRANCH IF MORE IN CURRENT BLOCK.
3018 JSR   PC,   MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
3019 CLR   R3
3020 MOV   #46,   R4    ;SET COUNTER FOR LOOP
3021                    2$:
3022 DEC   R3
3023 BNE   2$
3024 DEC   R4
3025 BNE   2$
3026 JSR   R4,   INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3027 MOV   #052525,R0   ;INIT PATTERN FOR CHECKING
3028                    3$:
3029 MOV   (R2)+, R1     ;GET THE DATA FROM MEMORY UNDER TEST.
3030 CMP   R0,   R1     ;COMPARE THE CHECK WORD WITH THE DATA READ.
3031 BEQ   65$         ;BRANCH OVER ERROR CALL IF GOOD DATA.
3032 JSR   PC,   SPRINT2 ;SET UP VALUES FOR ERROR PRINTING.
3033 JSR   PC,   $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3034 .WORD 6            ;ERROR TYPE CODE.
3035                    65$:
3036 COM   R0
3037 BIT   R5,   R2     ;CHECK FOR END OF A BLOCK.
3038 BNE   3$         ;BRANCH IF MORE IN CURRENT BLOCK.
3039 JSR   PC,   MMUP   ;FIND NEXT BLOCK AND LOOP TO 1$.
    
```

```

3037 .SBTTL DONE: RELOCATE PROGRAM AND REPEAT ALL TESTS.
3038 013776 DONE:
3039 013776 004567 004626 JSR R5, $SCOPE :GO TO SCOPE ROUTINE.
3040 014002 000000 .WORD 0 :NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
3041 014004 005067 165160 TST32: CLR $TIMES :RESET ITERATION COUNTER FOR RESTARTING TEST.
3042 014010 105067 165066 CLR $STNM :RESET TEST NUMBER.
3043 014014 036767 164562 165512 1$: BIT PRGMAP, SAVTST :CHECK IF PROGRAM IS IN TEST AREA.
3044 014022 001004 BNE 2$ :BR IF IT PROG IN MEM TO BE TESTED.
3045 014024 036767 164554 165504 BIT PRGMAP+2, SAVTST+2 :CHECK HI 64K
3046 014032 001435 BEQ $EOP :BR IF PROG NOT IN MEM TO BE TESTED.
3047 014034 032777 000200 165076 2$: BIT $SW07, $SWR :CHECK FOR INHIBIT RELOCATION SWITCH.
3048 014042 001031 BNE $EOP :SKIP RELOCATION IF SWITCH SET.
3049 014044 022767 000003 164530 CMP #3, PRGMAP :CHECK IF PROGRAM IN FIRST BK.
3050 014052 001013 BNE 4$ :BR IF NOT IN FIRST BK.
3051 014054 023737 000042 000046 CMP #42, #46 :CHECK FOR ACT11
3052 014062 001416 BEQ 6$ :BR IF ACT11.
3053 014064 105737 001224 TSTB $ENV :CHECK FOR APT11
3054 014070 001013 BNE 6$ :IF APT11 DO NOT RELOCATE
3055 :MUST BE XXDP OR STANDALONE
3056 014072 004767 002362 JSR PC RELTOP :RELOCATE PROGRAM TO TOP OF MEMORY.
3057 014076 000167 172002 3$: JMP START1 :LOOP BACK AND RUN ALL TESTS AGAIN.
3058
3059 014102 004767 002754 4$: JSR PC RELO :RELOCATE PROGRAM BACK TO FIRST BK.
3060 014106 005737 000042 TST #42 :TEST FOR XXDP
3061 014112 001402 BEQ 6$ :IF NOT RUNNING UNDER MON. DONT
3062 014114 004767 003150 5$: JSR PC, RESLDR :RESTORE LOADERS.
3063 014120
3064 014120 004567 007366 6$: JSR R5 $PRINT :GO PRINT OUT THE FOLLOWING MESSAGE.
3065 014124 001201 .WORD $CALF :ADDRESS OF MESSAGE TO BE TYPED

```



3066  
3067  
3068  
3069  
3070  
3071  
3072  
3073  
3074  
3075 014126  
3076 014126 000240  
3077 014130 005067 165034  
3078 014134 005267 165052  
3079 014140 042767 100000 165044  
3080 014146 005327  
3081 014150 000001  
3082 014152 003040  
3083 014154 012737  
3084 014156 000001  
3085 014160 014150  
3086 014162 004567 007324  
3087 014166 014260  
3088 014170 016746 165016  
3089  
3090  
3091 014174 013746 177776  
3092 014200 004767 010226  
3093 014204 004567 007302  
3094 014210 014275  
3095 014212  
3096  
3097 014212 016700 163624  
3098 014216 001416  
3099 014220 000005  
3100 014222 004710  
3101 014224 000240  
3102 014226 000240  
3103 014230 000240  
3104 014232 023737 000042 000046  
3105 014240 001405  
3106 014242 105737 001224  
3107 014246 001002  
3108 014250 004767 003074  
3109 014254  
3110 014254 000167 171624  
3111 014260 005015 047105 020104  
3112 014266 040520 051523 021440  
3113 014274 000  
3114 014275 377 000  
3115  
3116  
3117  
3118  
3119  
3120  
3121

```

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

$EOP:
      NOP
      CLR $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
      INC $PASS          ;;INCREMENT THE PASS NUMBER
      BIC #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
      DEC (PC)+         ;;LOOP?
$EOPCT: .WORD 1
      BGT $DOAGN        ;;YES
      MOV (PC)+,$(PC)+  ;;RESTORE COUNTER
$ENDCT: .WORD 1
      $EOPCT
      JSR R5, $SPRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD $ENDMG     ;ADDRESS OF MESSAGE TO BE TYPED
      MOV $PASS,-(SP)  ;SAVE $PASS FOR TYPEOUT
;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**
      MOV $#PSW,-(SP)  ;PUT THE PROCESSOR STATUS ON THE STACK
      JSR PC, $TYPDS   ;GO TO THE SUBROUTINE
      JSR R5, $SPRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
      .WORD $ENULL     ;ADDRESS OF MESSAGE TO BE TYPED

$GET42:
      MOV 42, R0        ;GET MONITOR ADDRESS
      BEQ $DOAGN        ;BRANCH IF NO MONITOR
      RESET             ;CLEAR THE WORLD
$ENDAD: JSR PC,(R0)     ;GO TO MONITOR
      NOP              ;SAVE ROOM
      NOP              ;FOR
      NOP              ;ACT11
      CMP $#42,$#46    ;ARE WE UNDER ACT11 OR XXDP
      BEQ $DOAGN       ;IF ACT11 THEN RESTART
      TSTB $#ENV       ;CHECK FOR APT11
      BNE $DOAGN       ;IF APT11 THEN RESTART
      JSR PC, SAVLDR   ;IF XXDP FIRST SAVE MONITOR

$DOAGN: JMP START1     ;RETURN*****
$ENDMG: .ASCIZ '<15><12>/END PASS #/'

$ENULL: .BYTE -1,-1,0 ;:NULL CHARACTER STRING
.SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
.SBTTL MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
*****
;* SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
;* THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
;* THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
;* THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.

```



```

3122
3123
3124 014300
3125 014300 012737 077406 172300
3126 014306 012737 077406 172302
3127 014314 012737 077406 172304
3128 014322 012737 077406 172306
3129 014330 005037 172310
3130 014334 005037 172312
3131 014340 005037 172314
3132 014344 012737 077406 172316
3133 014352 005037 172340
3134 014356 012737 000200 172342
3135 014364 005037 172344
3136 014370 005037 172346
3137 014374 005037 172350
3138 014400 005037 172352
3139 014404 005037 172354
3140 014410 012737 007600 172356
3141 014416 012737 000001 177572
3142 014424 000207
3143
3144
3145
3146
3147
3148 014426 012767 000001 165110
3149 014434 005067 165106
3150 014440 005002
3151 014442 016705 165140
3152 014446 005767 164134
3153 014452 001514
3154 014454 005037 172344
3155 014460 012702 040000
3156 014464 036767 165054 165036 1$:
3157 014472 001015
3158 014474 036767 165046 165030
3159 014502 001011
3160 014504 062737 000200 172344
3161 014512 006367 165026
3162 014516 006167 165024
3163 014522 100360
3164 014524 000000
3165 014526 036767 165012 165046 2$:
3166 014534 001004
3167 014536 036767 165004 165040
3168 014544 001405
3169 014546 016705 165026 3$:
3170 014552 042767 020000 165016
3171 014560 013737 172344 172346 4$:
3172 014566 016767 164752 164754
3173 014574 016767 164746 164750
3174 014602 032705 020000
3175 014606 001505
3176 014610 062737 000200 172346 5$:
3177 014616 006367 164726

```

```

:* PARS 4, 5, AND 6 ARE UNUSED.
:*****
MMINIT:
MOV #200-1*400+UP+RW, @#KIPDR0 ;SET KIPDR0 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW, @#KIPDR1 ;SET KIPDR1 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW, @#KIPDR2 ;SET KIPDR2 = RW UP 200 BLOCKS
MOV #200-1*400+UP+RW, @#KIPDR3 ;SET KIPDR3 = RW UP 200 BLOCKS
CLR @#KIPDR4
CLR @#KIPDR5
CLR @#KIPDR6
MOV #200-1*400+UP+RW, @#KIPDR7 ;SET KIPDR7 = RW UP 200 BLOCKS
CLR @#KIPAR0 ;MAP PAR0 INTO BANK0
MOV #200, @#KIPAR1 ;MAP PAR1 INTO BANK1
CLR @#KIPAR2 ;MAP PAR2 INTO BANK0
CLR @#KIPAR3
CLR @#KIPAR4
CLR @#KIPAR5
CLR @#KIPAR6
MOV #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
MOV #1, @#SRO ;ENABLE MEMORY MANAGEMENT
RTS PC ;RETURN

```

```

:*****
:* MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
:*****
INITMM: MOV #BITC, BITPT ;SET POINTER TO BANK0
CLR BITPT+2 ;CLEAR HI 64K BANK POINTERS
CLR R2 ;SET ADDRESS POINTER TO 0
MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
BEQ 10$ ;BRANCH IF NO MEM MGMT
CLR @#KIPAR2 ;SET UP 3RD PAR TO BANK0
MOV #4000, R2 ;RESET VIRTUAL ADR POINTER
BIT BITPT, TSTMAP ;CHECK IF THIS BANK TO BE TESTED
BNE 2$ ;BRANCH IF MATCH
BIT BITPT+2, TSTMAP+2 ;CHECK IN HI MAP
BNE 2$ ;BRANCH IF MATCH
ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
ROL BITPT+2 ;...HI POINTER.
BPL 1$ ;BR IF MORE.
HALT ;FATAL ERROR!!! NO 4K BANK FOUND?
BIT BITPT, LADMAP ;CHECK IF LAST BANK.
BNE 3$ ;BR IF LAST BANK.
BIT BITPT+2, LADMAP+2 ;CHECK IF LAST BANK.
BEQ 4$ ;BR IF NOT LAST BANK.
MOV LADMSK, R5 ;SET MASK TO FIND LAST ADR.
BIC #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
MOV @#KIPAR2, @#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
BITPT, TMPPT ;COPY BITPT...LO 64K.
MOV BITPT+2, TMPPT+2 ;...HI 64K.
BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF 8K.
BEQ 21$ ;BRANCH IF NOT 8K.
ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
ASL TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.

```

3178	014622	006167	164724		ROL	TMPPT+2	...	HI POINTER.
3179	014626	100473			BMI	20\$		:BR IF NO MORE.
3180	014630	036767	164714	164672	BIT	TMPPT, TSTMAP		:CHECK IF BANK TO BE TESTED.
3181	014636	001004			BNE	6\$		:BRANCH IF A MATCH.
3182	014640	036767	164706	164664	BIT	TMPPT+2, TSTMAP+2		:CHECK FOR HI 64K BANKS.
3183	014646	001760			BEQ	5\$		:BRANCH IF NO MEMORY
3184	014650	036767	164674	164724	6\$: BIT	TMPPT, LADMAP		:CHECK IF LAST BANK.
3185	014656	001004			BNE	7\$		:BRANCH IF A MATCH
3186	014660	036767	164666	164716	BIT	TMPPT+2, LADMAP+2		:CHECK HI 64K
3187	014666	001455			BEQ	21\$		:BR IF NOT LAST BANK.
3188	014670	016705	164704		7\$: MOV	LADMSK, R5		:RESET MASK TO FIND LAST ADR.
3189	014674	052767	020000	164674	BIS	#20000, TEMPLAD		:MAKE SURE LAST ADDRESS IS IN BANK 3.
3190	014702	000447			BR	21\$		:BR TO FINISH UP.
3191								
3192	014704	036767	164634	164616	10\$: BIT	BITPT, TSTMAP		:CHECK IF THIS BANK TO BE TESTED.
3193	014712	001006			BNE	11\$		:BR IF MATCH.
3194	014714	062702	020000		ADD	#20000, R2		:UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3195	014720	106367	164620		ASLB	BITPT		:UPDATE BANK POINTER TO NEXT BANK.
3196	014724	100367			BPL	10\$		:BR IF MORE BANKS.
3197	014726	000000			HALT			:FATAL ERROR!!! NO 4K BANK FOUND?
3198	014730	016767	164610	164612	11\$: MOV	BITPT, TMPPT		:COPY BANK POINTER.
3199	014736	036767	164602	164636	BIT	BITPT, LADMAP		:CHECK IF LAST BANK.
3200	014744	001021			BNE	12\$		:BR IF LAST BANK.
3201	014746	032705	020000		BIT	#BIT13, R5		:CHECK FOR 8K BLOCK SIZE.
3202	014752	001423			BEQ	21\$		:BRANCH IF SMALLER BLOCK SIZE.
3203	014754	106367	164570		ASLB	TMPPT		:POINT TO NEXT BANK.
3204	014760	100416			BMI	20\$		:BRANCH IF OVERFLOW.
3205	014762	036767	164562	164540	BIT	TMPPT, TSTMAP		:CHECK IF BANK TO BE TESTED.
3206	014770	001412			BEQ	20\$		:BRANCH IF NOT TO BE TESTED.
3207	014772	112767	000011	164557	MOVB	#11, FLAG8K		:SET 8K BLOCK SIZE FLAG.
3208	015000	036767	164544	164574	BIT	TMPPT, LADMAP		:CHECK FOR LAST BANK.
3209	015006	001403			BEQ	20\$		:BR IF NOT LAST BANK.
3210	015010	016705	164564		12\$: MOV	LADMSK, R5		:RESET MASK TO FIND LAST ADR.
3211	015014	000402			BR	21\$		:SKIP MASK RESET.
3212	015016	012705	017777		20\$: MOV	#MASK4K, R5		:RESET MASK TO 4K BLOCK SIZE.
3213	015022	056767	164516	164520	21\$: BIS	BITPT, TMPPT		:SET TMPPT FOR FLAGING LAST BANK.
3214	015030	056767	164512	164514	BIS	BITPT+2, TMPPT+2		
3215	015036	036767	164502	164524	BIT	BITPT, FADMAP		:CHECK IF FIRST ADDRESS NEEDS TO BE SET.
3216	015044	001004			BNE	22\$		:BR IF FIRST BANK.
3217	015046	036767	164474	164516	BIT	BITPT+2, FADMAP+2		:CHECK HI 64K.
3218	015054	001450			BEQ	INITEX		:BR IF NOT FIRST BANK.
3219	015056	016702	164502		22\$: MOV	TMPFAD, R2		:RESET ADDRESS POINTER TO FIRST ADR.
3220	015062	000445			BR	INITEX		
3221								
3222	015064	016705	164516		INITDN: MOV	BLKMSK, R5		:RESET R5 TO CURRENT BLOCK MASK.
3223	015070	005002			CLR	R2		:INIT ADDRESS POINTER.
3224	015072	005767	163510		TST	MMAVA		:CHECK FOR MEM MGMT
3225	015076	001411			BEQ	31\$		:BRANCH IF NO MEM MGMT
3226	015100	012767	100000	164440	MOV	#BIT15, BITPT+2		:SET POINTER TO TOP BIT
3227	015106	005067	164432		CLR	BITPT		
3228	015112	012737	007600	172344	MOV	#7600, #KIPAR2		:SET PAR TO TOP OF MEM
3229	015120	000403			BR	32\$		:BRANCH TO COMMON AREA
3230								
3231	015122	012767	000400	164414	31\$: MOV	#BIT8, BITPT		:SET UP BANK POINTER
3232	015130	012767	015152	164416	32\$: MOV	#33\$, MMORE		:SET "MMDOWN" EXIT ADDRESS.
3233	015136	066767	163436	164410	ADD	RELOCF, MMORE		:ADD OFFSET



```

3234 015144 004767 000524 JSR PC, MMDOWN ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
3235 015150 000000 HALT ;FATAL ERROR!!! NO MEM INDICATED IN MEM MAP ABOVE BK!
3236 015152 036767 164366 164422 33$: BIT BITPT, LADMAP ;CHECK FOR NON BOUNDARY LAST ADDR.
3237 015160 001004 BNE 34$ ;BR IF LAST BANK FLAG FOUND.
3238 015162 036767 164360 164414 BIT BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDARY LAST ADDR.
3239 015170 001402 BEQ INITEX ;BR IF NO LAD FLG FOUND.
3240 015172 016702 164376 34$: MOV LSTADR, R2 ;SET UP R2.
3241 015176 010467 164352 INITEX: MOV R4, MMORE ;PUT RETURN PC INTO "MMORE"
3242 015202 000204 RTS R4 ;RETURN

```

```

*****
* COMMON UPWARDS ADDRESSING ROUTINE
* FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
* GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
* DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
*****

```

```

3243
3244
3245
3246
3247
3248
3249
3250 015204 036767 164340 164370 MMUP: BIT TMPPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3251 015212 001122 BNE 10$ ;BR IF LAST BANK.
3252 015214 036767 164332 164362 BIT TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
3253 015222 001116 BNE 10$ ;BR IF LAST BANK.
3254 015224 016705 164356 MOV BLKMSK, R5 ;RESET R5 TO BLOCK MASK.
3255 015230 005767 163352 TST MMAVA ;CHECK FOR MEM MGMT AVAILABLE
3256 015234 001515 BEQ 20$ ;BRANCH IF NO MEM MGMT
3257 015236 012702 040000 MOV #40000, R2 ;RESET VIRTUAL ADR POINTER
3258 015242 062737 000200 172344 1$: ADD #200, @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
3259 015250 006367 164270 ASL BITPT ;UPDATE LO POINTER TO NEXT BANK.
3260 015254 006167 164266 ROL BITPT+2 ;...HI POINTER.
3261 015260 100577 BMI 32$ ;BR IF ALL DONE.
3262 015262 036767 164256 164240 BIT BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS
3263 015270 001004 BNE 2$ ;BRANCH IF MATCH
3264 015272 036767 164250 164232 BIT BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
3265 015300 001760 BEQ 1$ ;BRANCH IF NO MATCH
3266 015302 036767 164236 164272 2$: BIT BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3267 015310 001004 BNE 3$ ;BRANCH IF LAST BANK FLAG.
3268 015312 036767 164230 164264 BIT BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
3269 015320 001405 BEQ 4$ ;BR IF NOT LAST BANK.
3270 015322 016705 164252 3$: MOV LADMSK, R5 ;RESET MASK.
3271 015326 042767 020000 164242 BIC #20000, TEMPLAD ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
3272 015334 016767 164204 164206 4$: MOV BITPT, TMPPT ;COPY BITPT...LO 64K.
3273 015342 016767 164200 164202 MOV BITPT+2,TMPPT+2 ;...HI 64K.
3274 015350 032705 020000 BIT #BIT13, R5 ;CHECK FOR A BLOCK SIZE OF BK.
3275 015354 001530 BEQ 31$ ;BRANCH IF NOT.
3276 015356 013737 172344 172346 MOV @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
3277 015364 062737 000200 172346 5$: ADD #200, @#KIPAR3 ;UP DATE FORTH PAR.
3278 015372 006367 164152 TMPPT ;UPDATE LO POINTER TO NEXT 4K BANK.
3279 015376 006167 164150 ROL TMPPT+2 ;...HI POINTER.
3280 015402 100513 BMI 30$ ;BR IF NO MORE.
3281 015404 036767 164140 164116 6$: BIT TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3282 015412 001004 BNE 7$ ;BRANCH IF A MATCH.
3283 015414 036767 164132 164110 BIT TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
3284 015422 001760 BEQ 5$ ;BRANCH IF NO MEMORY
3285 015424 036767 164120 164150 7$: BIT TMPPT,LADMAP ;CHECK FOR LAST BANK FLAG.
3286 015432 001004 BNE 8$ ;BRANCH IF A MATCH
3287 015434 036767 164112 164142 BIT TMPPT+2,LADMAP+2 ;CHECK HI 64K
3288 015442 001475 BEQ 31$ ;BR IF NO LAST BANK FLAG.
3289 015444 016705 164130 8$: MOV LADMSK, R5 ;RESET MASK TO FIND LAST ADDRESS.

```



```

3290 015450 052767 020000 164120      BIS      #20000, TEMPLAD ;SET VIRTUAL ADR TO BANK 3.
3291 015456 000467      BR       31$
3292
3293 015460 026702 164112      10$:    CMP      TEMPLAD, R2      ;CHECK IF LAST ADR REACHED.
3294 015464 001064      BNE     31$          ;BR IF MORE.
3295 015466 000474      BR       32$          ;BR IF ALL DONE.
3296
3297 015470 106267 164063      20$:    ASRB     FLAG8K      ;SHIFT 8K FLAG
3298 015474 001407      BEQ     22$          ;BR IF NOT 8K BLOCK.
3299 015476 103455      BCS     30$          ;BR IF ANOTHER 4K.
3300 015500 105067 164053      CLRB    FLAG8K      ;CLEAR OUT ALL FLAGS.
3301 015504 162702 040000      SUB     #40000, R2    ;BACK UP 8K.
3302 015510 062702 020000      21$:    ADD     #20000, R2    ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
3303 015514 106367 164024      22$:    ASLB     BITPT      ;UPDATE POINTER.
3304 015520 100457      BMI     32$          ;BRANCH WHEN END IS REACHED.
3305 015522 036767 164016 164000      BIT     BITPT, TSTMAP ;CHECK IF THIS BANK EXISTS.
3306 015530 001767      BEQ     21$          ;BRANCH IF NO MATCH.
3307 015532 036767 164006 164042      BIT     BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3308 015540 001402      BEQ     23$          ;BR IF NO MATCH.
3309 015542 016705 164032      MOV     LADMSK, R5   ;RESET MASK TO FIND LAST ADR.
3310 015546 016767 163772 163774      23$:    MOV     BITPT, TMPPT ;SET UP TMP POINTER.
3311 015554 032705 020000      BIT     #BIT13, R5   ;CHECK FOR 8K BLOCK SIZE.
3312 015560 001426      BEQ     31$          ;BRANCH IF SMALLER BLOCK SIZE.
3313 015562 106367 163762      ASLB    TMPPT        ;POINT TO NEXT BANK.
3314 015566 100421      BMI     30$          ;BRANCH IF OVERFLOW.
3315 015570 036767 163754 163732      BIT     TMPPT, TSTMAP ;CHECK IF BANK TO BE TESTED.
3316 015576 001415      BEQ     30$          ;BRANCH IF NOT TO BE TESTED.
3317 015600 036767 163740 163774      BIT     BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3318 015606 112767 000011 163743      MOVB   #11, FLAG8K  ;SET 8K BLOCK FLAG.
3319 015614 036767 163724 163760      BIT     BITPT, LADMAP ;CHECK FOR LAST BANK FLAG.
3320 015622 001403      BEQ     30$          ;BR IF NO FLAG.
3321 015624 016705 163750      MOV     LADMSK, R5   ;RESET MASK TO FIND LAST ADR.
3322 015630 000402      BR       31$
3323 015632 012705 017777      30$:    MOV     #MASK4K, R5  ;SET MASK TO 4K.
3324 015636 056767 163702 163704      31$:    BIS     BITPT, TMPPT ;SET TMPPT FOR FINDING LAST ADR.
3325 015644 056767 163676 163700      BIS     BITPT+2, TMPPT+2
3326 015652 016716 163676      MOV     MMORE, (SP)  ;FUDGE RETURN ADDRESS TO LOOP.
3327 015656 000207      RTS     PC           ;RETURN
3328
3329 015660 005767 164412      ;* BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
3330 015664 001402      32$:    TST     MPRX        ;CHECK FOR ANY PARITY REGISTERS PRESENT.
3331 015666 004767 001744      BEQ     33$          ;BR IF NONE.
3332 015672 000207      JSR     PC, CKPMER  ;CHECK FOR PARITY MEMORY ERRORS.
3333
3334
3335
3336
3337
3338
3339
3340 015674 036767 163644 163666      MMDOWN: BIT     BITPT, FADMAP ;CHECK FOR FIRST ADR FLAG.
3341 015702 001004      BNE     1$           ;BR IF FIRST ADR IN THIS BANK.
3342 015704 036767 163636 163660      BIT     BITPT+2, FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
3343 015712 001404      BEQ     2$           ;BR IF NO FLAG
3344 015714 026702 163644      1$:    CMP     TMPFAD, R2  ;CHECK IF FIRST ADDRESS REACHED.
3345 015720 001052      BNE     9$           ;BR IF MORE.

```

```

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

```

3346	015722	000453				BR	10\$	:BR IF ALL DONE.
3347	015724	005767	162656		2\$:	TST	MMAVA	:CHECK IF MEM MGMT IS AVAILABLE
3348	015730	001425				BEQ	6\$	:BRANCH IF NOT
3349	015732	162737	000200	172344	3\$:	SUB	#200, 2#KIPAR2	:LOWER MEM MGMT PAR BY 4K
3350	015740	006067	163602			ROR	BITPT+2	:MOV POINTER TO NEXT LOWER BANK...HI MAP.
3351	015744	006067	163574			ROR	BITPT	:...LO MAP.
3352	015750	103440				BCS	10\$	:BR IF NO MORE.
3353	015752	036767	163566	163550		BIT	BITPT, TSTMAP	:CHECK FOR BANK EXISTING
3354	015760	001004				BNE	4\$	:BR IF BANK TO BE TESTED.
3355	015762	036767	163560	163542		BIT	BITPT+2, TSTMAP+2	:CHECK FOR BANK IN HI MAP.
3356	015770	001760				BEQ	3\$	:BR IF NOT THERE.
3357	015772	012702	060000		4\$:	MOV	#60000, R2	:SET ADR POINTER TO TOP OF BANK
3358	015776	000411				BR	7\$	:GO TO COMMON EXIT
3359	016000	162702	020000		5\$:	SUB	#20000, R2	:BACK POINTER DOWN ONE BANK
3360	016004	006267	163534		6\$:	ASR	BITPT	:MOVE POINTER TO NEXT LOWER BANK
3361	016010	103420				BCS	10\$	:BRANCH TO EXIT IF NO MOF. MEM
3362	016012	036767	163526	163510		BIT	BITPT, TSTMAP	:CHECK IF BANK EXISTS
3363	016020	001767				BEQ	5\$	:BRANCH IF BANK DOESN'T EXIST
3364	016022	036767	163516	163540	7\$:	BIT	BITPT, FADMAP	:CHECK IF FIRST BANK FLAG.
3365	016030	001004				BNE	8\$	:BR IF FIRST BANK.
3366	016032	036767	163510	163532		BIT	BITPT+2, FADMAP+2	:CHECK IF FIRST BANK FLAG.
3367	016040	001402				BEQ	9\$	:BR IF NO FLAG FOUND.
3368	016042	016705	163520		8\$:	MOV	FADMSK, R5	:SET UP R5 TO FIND FIRST ADDRESS.
3369	016046	016716	163502		9\$:	MOV	MORE, (SP)	:RESET RETURN ADDRESS
3370	016052	000207			10\$:	RTS	PC	:RETURN

```

3371
3372
3373
3374
3375
3376 016054 010200
3377 016056 005067 163076
3378 016062 005767 162520
3379 016066 001417
3380 016070 010146
3381 016072 013701 172344
3382 016076 006301
3383 016100 006301
3384 016102 006301
3385 016104 006301
3386 016106 006301
3387 016110 006167 163044
3388 016114 006301
3389 016116 006167 163036
3390 016122 060100
3391 016124 012601
3392 016126 000207
3393
3394
3395
3396
3397 016130 005000
3398 016132 010146
3399 016134 010246
3400 016136 016701 163402
3401 016142 016702 163400
3402 016146 006202
3403 016150 006001
3404 016152 103403
3405 016154 105200
3406 016156 100373
3407 016160 000000
3408 016162
3409 016162 012602
3410 016164 012601
3411 016166 000207
3412
3413
3414
3415
3416 016170
3417 016170 004467 176232
3418 016174 010022
3419 016176 030502
3420 016200 001375
3421 016202 004767 176776
3422 016206 000207

```

```

.SBTTL SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
*****
* SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
* BITS 16 AND 17 ARE IN $TMP0.
*****

```

```

PHYADR: MOV R2 R0 ;VITRUAL INTO R0
        CLR $TMP0 ;CLEAR TEMP SAVE OF HIGH BITS
        TST MAVA ;CHECK FOR MEM MGMT AVAILABLE
        BEQ 1$ ;BRANCH IF NO MEM MGMT
        MOV R1, -(SP) ;PUSH R1 ON STACK
        MOV @#KIPAR2, R1 ;GET PAR TO BE ADDED TO VIRTUAL
        ASL R1 ;SHIFT IT 6 TIMES
        ASL R1
        ASL R1
        ASL R1
        ASL R1
        ROL $TMP0 ;SAVE EXTRA BITS
        ASL R1
        ROL $TMP0
        ADD R1, R0 ;ADD SHIFTED PAR TO VIRTUAL
        MOV (SP)+, R1 ;POP STACK INTO R1
1$: RTS PC ;RETURN

```

```

*****
* SUBROUTINE TO PUT BANK NUMBER INTO R0.
*****

```

```

BANKNO: CLR R0 ;INIT R0
        MOV R1, -(SP) ;PUSH R1 ON STACK
        MOV R2, -(SP) ;PUSH R2 ON STACK
        MOV BITPT, R1 ;GET BANK MAP POINTER...LO 64K.
        MOV BITPT+2, R2 ;...HI 64K.
1$: ASR R2 ;SHIFT POINTER...HI
    ROR R1 ;...LO
    BCS 2$ ;BR WHEN POINTER FOUND.
    INCB R0 ;COUNT BANKS.
    BPL 1$ ;BR IF NOT OVERFLOW.
2$: HALT ;FATAL ERROR!!! NO POINTER FOUND.
        MOV (SP)+, R2 ;POP STACK INTO R2
        MOV (SP)+, R1 ;POP STACK INTO R1
        RTS PC ;RETURN

```

```

*****
* SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
*****

```

```

SETCON: JSR R4, INITMM ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2$: MOV R0, (R2)+ ;MOV CONSTANT INTO MEMORY
    BIT R5, R2 ;CHECK FOR END OF A BLOCK.
    BNE 2$ ;BRANCH IF MORE IN CURRENT BLOCK.
    JSR PC, MMUP ;FIND NEXT BLOCK AND LOOP TO 1$.
    RTS PC ;RETURN

```





RELOCATION SUBROUTINES.

```

3477 .SBTTL RELOCATION SUBROUTINES.
3478 ;*****
3479 ;* ROUTINE TO RELOCATE PROGRAM CODE
3480 ;*****
3481 RELOC:
3482   MOV     R2,-(SP)      ;; PUSH R2 ON STACK
3483   MOV     R3,-(SP)      ;; PUSH R3 ON STACK
3484   MOV     R4,-(SP)      ;; PUSH R4 ON STACK
3485   4$:    MOV     (R5)+, R2      ;; GET FIRST LOCATION.
3486   MOV     (R5)+, R3      ;; GET FIRST LOCATION OF DESTINATION.
3487   MOV     #20000, R4     ;; SET UP BK COUNTER.
3488   1$:    MOV     (R2)+, (R3)+  ;; MOV THE DATA.
3489   DEC     R4            ;; COUNT THE WORDS.
3490   BNE    1$           ;; BR IF MORE.
3491   MOV     #20000, R4     ;; RESET THE COUNTER.
3492   2$:    CMP     -(R2), -(R3)  ;; CHECK THE DATA JUST MOVED.
3493   BEQ    3$           ;; BR IF DATA OK.
3494   MOV     (R2), $GDDAT  ;; GET SOURCE DATA.
3495   MOV     (R3), $BDDAT  ;; GET DESTINATION DATA.
3496   MOV     R2, $GDADR   ;; GET SOURCE ADDRESS.
3497   MOV     R3, $BDADR   ;; GET DESTINATION ADDRESS.
3498   JSR    PC, $ERRR    ;; *** ERROR *** (GO TYPE A MESSAGE)
3499   .WORD  23           ;; ERROR TYPE CODE.
3500   HALT                    ;; FATAL ERROR!!! RELOCATION FAILED.
3501   SUB     #4, R5        ;; ADJUST RETURN POINTER.
3502   BR     4$           ;; GO BACK AND TRY AGAIN.
3503   3$:    DEC     R4        ;; COUNT WORDS.
3504   BNE    2$           ;; BR IF MORE.
3505   JSR    R5, $PRINT    ;; GO PRINT OUT THE FOLLOWING MESSAGE.
3506   .WORD  PRELOC       ;; ADDRESS OF MESSAGE TO BE TYPED
3507   MOV     R3, -(SP)     ;; PUT THE DATA ON THE STACK.
3508   JSR    PC, $STYPAD  ;; DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
3509   MOV     (SP)+, R4     ;; POP STACK INTO R4
3510   MOV     (SP)+, R3     ;; POP STACK INTO R3
3511   MOV     (SP)+, R2     ;; POP STACK INTO R2
3512   RTS     R5           ;; RETURN
3513
3514 ;*****
3515 ;* SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
3516 ;*****
3517 RELTOP: CMP     #3, PRGMAP  ;; CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
3518 BEQ     1$           ;; BR IF OK
3519 HALT                    ;; FATAL ERROR!!! PROG SHOULD BE IN BANKS 0 AND 1
3520 1$:    MOV     R0,-(SP)  ;; PUSH R0 ON STACK
3521   MOV     R1,-(SP)  ;; PUSH R1 ON STACK
3522   TST     MMABA
3523   BEQ     10$
3524   MOV     #7600, @#KIPAR3  ;; SET PAR TO TOP OF MEM
3525   CLR     R0        ;; INIT BANK POINTER...LO 64K
3526   MOV     #BIT15, R1  ;; ...HI 64K.
3527   MOV     #200, @#KIPAR3  ;; BACK DOWN ONE BANK.
3528   2$:    SUB     R1, @#KIPAR3  ;; MOVE POINTER...HI 64K.
3529   ROR     R1        ;; ...LO 64K.
3530   ROR     R0
3531   BCS    90$
3532   BIT     R1, MEMMAP+2  ;; CHECK FOR BANK EXISTS.

```



```

3533 016540 001003          BNE      3$          ;BR IF AVAILABLE
3534 016542 030067 162756  BIT      RO,      MEMMAP ;CHECK FOR BANK EXISTS.
3535 016546 001764          BEQ      2$          ;BR IF NO BANK FOUND.
3536 016550 013737 172346 172344 3$:  MOV     J#KIPAR3,J#KIPAR2 ;COPY PAR
3537 016556 010046          MOV     RO,-(SP)      ;PUSH RO ON STACK
3538 016560 010146          MOV     R1,-(SP)      ;PUSH R1 ON STACK
3539 016562 162737 000200 172344 4$:  SUB     #200, J#KIPAR2 ;BACK DOWN WITH LOW PAR.
3540 016570 006001          ROR     R1           ;SHIFT POINTER.
3541 016572 006000          ROR     RO           ;...LO 64K.
3542 016574 103457          BCS     90$         ;BR IF OVERFLOW.
3543 016576 030167 162724 5$:  BIT     R1,      MEMMAP+2 ;CHECK IF BANK EXISTS...HI 64K.
3544 016602 001003          BNE     6$          ;BR IF BANK EXISTS.
3545 016604 030067 162714  BIT     RO,      MEMMAP ;CHECK IF BANK EXISTS...LO 64K.
3546 016610 001764          BEQ     4$          ;BR IF BANK DOESN'T EXIST.
3547 016612 052601 6$:  BIS     (SP)+, R1      ;GET SECOND BANK POINTER.
3548 016614 052600          BIS     (SP)+, RO      ;...LO 64K.
3549 016616 030067 161760  BIT     RO           ;CHECK FOR CONFLICT.
3550 016622 001044          BNE     90$         ;ABORT IF DESTINATION OVERLAYS SOURCE.
3551 016624 004567 177506  JSR     R5,      RELOC  ;GO RELOCATE PROGRAM.
3552 016630 000000          .WORD  0           ;SOURCE FIRST ADDRESS.
3553 016632 040000          .WORD  40000        ;DESTINATION FIRST ADDRESS.
3554 016634 013737 172344 172340  MOV     J#KIPAR2,J#KIPAR0 ;RELOCATE LO BANK
3555 016642 013737 172346 172342  MOV     J#KIPAR3,J#KIPAR1 ;RELOCATE HI BANK.
3556                                ;* PROGRAM SHOULD NOW BE EXECUTING OUT OF LAST TWO BANKS OF MEMORY.
3557 016650 010167 161730  MOV     R1,      PRGMAP+2 ;RESET PROGRAM MAP.
3558 016654 000473          BR      30$         ;BR TO COMMON EXIT.
3559
3560 016656 012700 000400 10$:  MOV     #BIT8, RO     ;SET BANK POINTER TO TOP OF MEM.
3561 016662 005001          CLR     R1           ;SET ADDRESS POINTER TO TOP.
3562 016664 162701 020000 11$:  SUB     #20000, R1    ;BACK DOWN ONE BANK.
3563 016670 006200          ASR     RO           ;MOVE POINTER DOWN ONE BANK.
3564 016672 103420          BCS     90$         ;BR IF OVERFLOW.
3565 016674 030067 162624  BIT     RO,      MEMMAP ;CHECK IF THIS BANK EXISTS.
3566 016700 001771          BEQ     11$         ;BR IF NON-EXISTANT BANK.
3567 016702 162701 020000  SUB     #20000, R1    ;BACK DOWN TO NEXT BANK.
3568 016706 006200          ASR     RO           ;MOV POINTER DOWN ONE BANK.
3569 016710 103411          BCS     90$         ;BR IF OVERFLOW.
3570 016712 030067 162606  BIT     RO,      MEMMAP ;CHECK IF THIS BANK EXISTS.
3571 016716 001762          BEQ     11$         ;BR TO START OVER IF NO LOWER BANK.
3572 016720 010046          MOV     RO,      -(SP) ;SAVE THE POINTER.
3573 016722 006300          ASL     RO           ;RESET POINTER TO HI BANK.
3574 016724 052600          BIS     (SP)+, RO    ;SET BIT FOR LO BANK.
3575 016726 030067 161650  BIT     RO,      PRGMAP ;CHECK FOR A PROGRAM CONFLICT.
3576 016732 001401          BEQ     12$         ;BR IF NO CONFLICT.
3577 016734 90$:
3578 016734 000000          HALT                    ;FATAL ERROR!!! NOT ENOUGH MEMORY??
3579 016736 010167 000006 12$:  MOV     R1,      13$   ;SET DATA FOR RELOCATION SUBROUTINE.
3580 016742 004567 177370  JSR     R5,      RELOC  ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3581 016746 000000          .WORD  0           ;SOURCE STARTING ADDRESS.
3582 016750 000000          .WORD  0           ;DESTINATION STARTING ADDRESS.
3583 016752 010167 161622  MOV     R1,      RELOCF ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3584 016756 060107          ADD     R1,      PC    ;JUMP TO RELOCATED PROGRAM
3585                                ;* PROGRAM NOW EXECUTING OUT OF TOP OF MEMORY.
3586 016760 360106          ADD     R1,      SP    ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3587 016762 010167 161612  MOV     R1,      RELOCF ;SET THE RELOCATION FACTOR.
3588 016766 060137 000004  ADD     R1,      J#ERRVEC ;ADJUST ERROR VECTOR.
    
```





```

3645 017234 001404 BEQ 23$ ;BR IF HARDWARE SWITCH REGISTER.
3646 017236 161667 161676 SUB (SP), SWR ;ADJUST SOFTWARE SWITCH REGISTER.
3647 017242 161667 161674 SUB (SP), DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3648 017246 162616 23$: SUB (SP)+ (SP) ;ADJUST RETURN ADDRESS.
3649 017250 005067 161324 30$: CLR RELOC ;RESET RELOCATION FACTOR.
3650 017254 012767 000003 161320 MOV #3, PRGMAP ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
3651 017262 005067 161316 CLR PRGMAP+2 ; HI 64K.
3652 017266 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.
:*****

```

```

3658
3659 017270 016700 162224 RESLDR: MOV LMAD, R0 ;CHECK IF THE LOADERS WERE SAVED.
3660 017274 001001 BNE 1$ ;BR IF LOADER AREA WAS SAVED.
3661 017276 000000 HALT ;FATAL ERROR!!! CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
3662 017300 005767 161302 1$: TST MMAVA ;CHECK FOR MEM MGMT.
3663 017304 001402 BEQ 2$ ;SKIP IF NO MEM MGMT.
3664 017306 005037 177572 CLR @SRO ;DISABLE MEM MGMT.
3665 017312 012701 040000 2$: MOV #40000, R1 ;GET END OF BK, ASSUME PROG NOT RELOCATED.
3666 017316 012702 002734 MOV #1500., R2 ;GET COUNTER.
3667 017322 014140 3$: MOV -(R1), -(R0) ;MOVE THE LOADER AREA.
3668 017324 005302 DEC R2 ;COUNT HOW LONG THE AREA IS.
3669 017326 001375 BNE 3$ ;BR IF NOT MORE TO MOVE.
3670 017330 005067 162164 CLR LMAD ;CLEAR MONITOR SAVED FLAG
3671 017334 005767 161246 TST MMAVA ;CHECK FOR MEM MGMT.
3672 017340 001402 BEQ 4$ ;BR IF NO MEM MGMT.
3673 017342 005237 177572 INC @SRO ;ENABLE MEM MGMT.
3674 017346 000207 4$: RTS PC ;RETURN.

```

```

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

```

```

3675
3676
3677 017350 005767 162144 SAVLDR: TST LMAD ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3678 017354 001024 BNE 4$ ;BRANCH IF ALREADY SAVED
3679 017356 012700 040000 MOV #40000, R0 ;GET END OF BK
3680 017362 010001 R0, R1 ;GET END OF BK
3681 017364 012737 017376 000004 MOV #2$, @ERRVEC ;SET UP TIMEOUT VECTOR
3682 017372 011020 1$: MOV (R0), (R0)+ ;SEARCH FOR END OF MEMORY
3683 017374 000776 BR 1$ ;KEEP SEARCHING
3684 017376 022626 2$: CMP (SP)+ (SP)+ ;RESTORE STACK POINTER
3685 017400 012737 025114 000004 @ERRTRP, @ERRVEC ;RESET TIMEOUT VECTOR.
3686 017406 010046 MOV R0, -(SP) ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3687 017410 012702 002734 MOV #1500., R2 ;SET UP WORD COUNTER
3688 017414 014041 3$: MOV -(R0), -(R1) ;SAVE THE LOADERS
3689 017416 005302 DEC R2 ;COUNT THE WORDS
3690 017420 001375 BNE 3$ ;BRANCH IF MORE WORDS
3691 017422 012667 162072 MOV (SP)+, LMAD ;SAVE LAST MEMORY ADDRESS
3692 017426 000207 4$: RTS PC ;RETURN

```



```

3693
3694
3695
3696
3697
3698
3699 017430 011667 161466
3700 017434 004567 004052
3701 017440 026501
3702
3703 017442 010146
3704 017444 010346
3705 017446 016703 162156
3706 017452 005733
3707 017454 100415
3708 017456 005713
3709 017460 001374
3710 017462 004767 002152
3711
3712 017466 000024
3713 017470 000417
3714 017472 005713
3715 017474 001415
3716 017476 005733
3717 017500 100374
3718 017502 004567 004004
3719 017506 026572
3720
3721 017510
3722 017510 004767 000610
3723 017514 004767 002120
3724 017520 000025
3725 017522 004767 000216
3726 017526 000761
3727 017530
3728 017530 012603
3729 017532 012601
3730 017534 000002
3731
3732
3733
3734
3735
3736 017536 005767 162534
3737 017542 001434
3738 017544 032777 000100 161366
3739 017552 001030
3740 017554 005767 161020
3741 017560 001410
3742 017562 032777 000040 161350
3743 017570 001004
3744 017572 026727 161764 001000
3745 017600 103415

```

```

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

```



E13

```

3746 017602 016737 162030 000114 SETAE: MOV .PESRV, @PARVEC ;SET PARITY ERROR TRAP VECTOR
3747 017610 005037 000116 CLR @PARVEC+2 ;PRIORITY LEVEL 0 ON TRAP
3748 017614 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
3749 017616 016703 162006 MOV .MPRX, R3 ;GET PARITY REGISTER TABLE POINTER.
3750 017622 052733 000001 MAMF1: BIS @AE, @R3+ ;SET ACTION ENABLE BIT IN PARITY REG
3751 017626 005713 TST (R3) ;CHECK FOR END OF TABLE.
3752 017630 001374 BNE MAMF1 ;BR IF MORE PARITY REGISTERS.
3753 017632 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3754 017634 000207 MAMF2: RTS PC ;RETURN.
3755
3756 ;*****
3757 ;* SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
3758 ;*****
3759 017636 005767 162434 CKPMER: TST MPRX ;CHECK IF ANY PARITY REGISTERS EXIST.
3760 017642 001437 BEQ 4$ ;BR IF NO PARITY REGISTERS.
3761 017644 032777 000100 161266 BIT @SW6, @SWR ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
3762 017652 001033 BNE 4$ ;BR IF PARITY ERROR CHECKING INHIBITED.
3763 017654 010346 MOV R3, -(SP) ;PUSH R3 ON STACK
3764 017656 016703 161746 MOV .MPRX, R3 ;GET PARITY REG TABLE POINTER.
3765 017662 005733 1$: TST @R3+ ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3766 017664 100023 BPL 3$ ;BR IF NO ERROR
3767 017666 032773 000001 177776 BIT @BIT0, @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
3768 017674 001010 BNE 2$ ;BR IF NO ACTION ENABLE.
3769 017676 004767 000422 64$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
3770 017702 004767 001732 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3771 017706 000026 .WORD 26 ;ERROR TYPE CODE.
3772 017710 000411 BP 3$
3773 017712 004767 000026 JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3774 017716
3775 017716 004767 000402 2$: JSR PC, SPRTQ ;SET UP VALUES FOR ERROR PRINTING.
3776 017722 004767 001712 JSR PC, $ERROR ;*** ERROR *** (GO TYPE A MESSAGE)
3777 017726 000027 .WORD 27 ;ERROR TYPE CODE.
3778 017730 004767 000010 JSR PC, PSCAN ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3779 017734 005713 3$: TST (R3) ;CHECK FOR TABLE TERMINATOR.
3780 017736 001351 BNE 1$ ;BR IF MORE.
3781 017740 012603 MOV (SP)+, R3 ;POP STACK INTO R3
3782 017742 000207 4$: RTS PC ;RETURN.
3783
3784 ;*****
3785 ;* THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY.
3786 ;* TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
3787 ;* LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
3788 ;*****

```

3789	017744			PSCAN:	MOV	R0,-(SP)	::	PUSH R0 ON STACK
3790	017744	010046			MOV	R1,-(SP)	::	PUSH R1 ON STACK
3791	017746	010146			MOV	R2,-(SP)	::	PUSH R2 ON STACK
3792	017750	010246			MOV	R3,-(SP)	::	PUSH R3 ON STACK
3793	017752	010346			MOV	R4,-(SP)	::	PUSH R4 ON STACK
3794	017754	010446			MOV	2#114,-(SP)	::	PUSH 2#114 ON STACK
3795	017756	013746	000114		MOV	2#116,-(SP)	::	PUSH 2#116 ON STACK
3796	017762	013746	000116		JSR	R5	\$SPRINT	GO PRINT OUT THE FOLLOWING MESSAGE.
3797	017766	004567	003520		.WORD	SCANM		ADDRESS OF MESSAGE TO BE TYPED
3798	017772	026636						"SCANNING MEMORY FOR BAD PARITY."
3799								SET BIT POINTER TO FIRST BANK.
3800	017774	012700	000001		MOV	#BIT0, R0		CLR HI 64K POINTER.
3801	020000	005001			CLR	R1		INIT ADDRESS POINTER.
3802	020002	005002			CLR	R2		INIT ERROR DETECTED FLAG.
3803	020004	005004			CLR	R4		CLEAR THE PARITY REGISTER(S).
3804	020006	004767	000256		JSR	PC	CLRPAR	HALT IF ANOTHER PARITY TRAP.
3805	020012	012737	000116	000114	MOV	#116, 2#114		
3806	020020	005037	000116		CLR	2#116		
3807	020024	005767	160556		TST	MMAVA		CHECK FOR MEMORY MANAGEMENT.
3808	020030	001406			BEQ	1\$		BR IF NO MEM MGMT.
3809	020032	013746	172344		MOV	2#KIPAR2,-(SP)		PUSH 2#KIPAR2 ON STACK
3810	020036	005037	172344		CLR	2#KIPAR2		INIT MEM MGMT TO POINT TO BANK 0.
3811	020042	012702	040000		MOV	#40000, R2		SET ADR POINTER TO PAR2.
3812	020046	030067	161452		BIT	R0, MEMMAP		CHECK IF THIS BANK OF MEM EXISTS.
3813	020052	001003		1\$:	BNE	2\$		BR IF THIS BANK EXISTS.
3814	020054	030167	161446		BIT	R1, MEMMAP+2		CHECK HI 64K MAP.
3815	020060	001442			BEQ	10\$		BR IF THIS BANK DOESN'T EXIST.
3816	020062			2\$:				
3817	020062	010146			MOV	R1,-(SP)		PUSH R1 ON STACK
3818	020064	111201		3\$:	MOVB	(R2), R1		READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
3819	020066	016703	161536		MOV	.MPRX, R3		SET UP POINTER TO PARITY REGISTERS.
3820	020072	005733		4\$:	TST	2(R3)+		CHECK FOR THE ERROR FLAG.
3821	020074	100024			BPL	6\$		BR IF NO ERROR FLAG.
3822	020076	005704			TST	R4		CHECK IF FIRST ERROR, THIS SCAN.
3823	020100	001003			BNE	5\$		BR IF MORE THAN ONE ERROR FOUND.
3824	020102	005367	161004		DEC	\$ERTTL		ADJUST ERROR COUNT.
3825	020106	005204			INC	R4		SET FLAG TO INDICATE ERROR FOUND.
3826	020110			5\$:				
3827	020110	004767	000210	64\$:	JSR	PC, SPRTQ		SET UP VALUES FOR ERROR PRINTING.
3828	020114	004767	001520		JSR	PC, \$ERROR		*** ERROR *** (GO TYPE A MESSAGE)
3829	020120	000030			.WORD	30		ERROR TYPE CODE.
3830	020122	111212			MOVB	(R2), (R2)		REWRITE THE LOCATION TO CLEAR BAD PARITY.
3831	020124	005053			CLR	2-(R3)		CLEAR THE ERROR FLAG.
3832	020126	105712			TSTB	(R2)		CHECK IF THE PARITY ERROR WAS CLEARED.
3833	020130	005733			TST	2(R3)+		CHECK FOR THE ERROR FLAG.
3834	020132	100005			BPL	6\$		BR IF IT IS OK.
3835	020134	004567	003352		JSR	R5, \$SPRINT		GO PRINT OUT THE FOLLOWING MESSAGE.
3836	020140	026700			.WORD	PEWNC		ADDRESS OF MESSAGE TO BE TYPED
3837								"PARITY ERROR WILL NOT CLEAR."
3838	020142	005073	177776		CLR	2-2(R3)		CLEAR OUT THE PARITY ERROR FLAG.
3839	020146	005713		6\$:	TST	(R3)		CHECK FOR THE END OF REG ADR TABLE.
3840	020150	001350			BNE	4\$		BR IF MORE PARITY REGISTERS.
3841	020152	005202			INC	R2		GO TO NEXT MEMORY ADDRESS.
3842	020154	032702	017777		BIT	#MASK4K,R2		CHECK FOR END OF 4K BANK.
3843	020160	001341			BNE	3\$		BR IF MORE MEMORY THIS BANK.
3844	020162	012601			MOV	(SP)+,R1		POP STACK INTO R1



```

3845 020164 000402          BR      11$          ;BR TO CHECK FOR NEXT BANK.
3846 020166 062702 020000 10$: ADD    #20000, R2    ;SKIP BANKS THAT AREN'T THERE.
3847 020172 005767 160410 11$: TST    MMAVA        ;CHECK FOR MEM MGMT.
3848 020176 001413          BEQ    12$          ;BR IF NO MEM MGMT.
3849 020200 062737 000200 172344 ADD    #200, @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
3850 020206 012702 040000 MOV    #40000, R2    ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
3851 020212 006300          ASL    R0           ;UPDATE BANK POINTER.
3852 020214 006101          ROL    R1           ;...HI 64K.
3853 020216 100313          BPL    1$           ;BR IF MORE BANKS.
3854 020220 012637 172344 MOV    (SP)+, @#KIPAR2 ;:POP STACK INTO @#KIPAR2
3855 020224 000402          BR      20$        ;GO CHECK IF ANY ERRORS FOUND.
3856 020226 106300 12$: ASLB   R0           ;UPDATE POINTER TO NEXT BANK.
3857 020230 100306          BPL    1$           ;BR IF MORE BANKS.
3858 020232 005704 20$: TST    R4           ;CHECK IF ANY PARITY ERRORS DETECTED.
3859 020234 001003          BNE    21$        ;BR IF ERRORS DETECTED.
3860 020236 004567 003250 JSR    R5, $SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
3861 020242 025706          .WORD NOPE$       ;ADDRESS OF MESSAGE TO BE TYPED
3862 020244          21$:
3863 020244 012637 000116 MOV    (SP)+, @#116 ;:POP STACK INTO @#116
3864 020250 012637 000114 MOV    (SP)+, @#114 ;:POP STACK INTO @#114
3865 020254 012604          MOV    (SP)+, R4   ;:POP STACK INTO R4
3866 020256 012603          MOV    (SP)+, R3   ;:POP STACK INTO R3
3867 020260 012602          MOV    (SP)+, R2   ;:POP STACK INTO R2
3868 020262 012601          MOV    (SP)+, R1   ;:POP STACK INTO R1
3869 020264 012600          MOV    (SP)+, R0   ;:POP STACK INTO R0
3870 020266 000207          RTS     PC         ;RETURN.
3871
3872
3873 ;*****
3874 ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
3875 ;*****
3875 020270 CLRPAR:
3876 020270 010346          MOV    R3, -(SP)   ;:PUSH R3 ON STACK
3877 020272 016703 161332 MOV    .MPRX, R3   ;GET PARITY REGISTER TABLE POINTER.
3878 020276 005713 1$: TST    (R3)       ;CHECK FOR THE TABLE TERMINATOR.
3879 020300 001402          BEQ    2$          ;BR IF DONE ALL PARITY REGISTERS.
3880 020302 005033          CLR    @ (R3)+    ;CLEAR THE PARITY REGISTER.
3881 020304 000774          BR      1$         ;BR FOR MORE
3882 020306 2$:
3883 020306 012603          MOV    (SP)+, R3   ;:POP STACK INTO R3
3884 020310 000207          RTS     PC         ;RETURN.
3885
3886 .SBTTL SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
3887 ;*****
3888 ;* THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.%CMTAG)
3889 ;* FOR ERROR PRINTOUT BY .%ERROR & .%ERRTYP ROUTINES FROM **SYSTEMAC**.
3890 ;*****
3891 020312 010267 160602 SPRNT: MOV    R2, $GDADR ;SAVE THE ADDRESS UNDER TEST.
3892 020316 005067 150602 CLR    $GDADR      ;SHOULD BE DATA IS "0".
3893 020322 000430          BR      SPRNTB
3894
3895 020324 014367 160630 SPRNTQ: MOV   -(R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
3896 020330 013367 160626 MOV    @ (R3)+, $TMP1 ;GET THE CONTENTS OF THE PARITY REG.
3897 020334 000402          BR      SPRNTQ
3898
3899 020336 011367 160616 SPRNTP: MOV   (R3), $TMP0 ;GET THE PARITY REGISTER ADDRESS.
3900 020342 010267 160552 SPRNTO: MOV   R2, $GDADR ;GET THE MEMORY ADDRESS BEING TESTED

```



# H13

CZQMCFD 0-124K MEMORY EXERCISER. 16K VER  
 CZQMCF.P11 14-FEB-78 09:19

MACY11 30A(1052) 20-FEB-78 07:56 PAGE 81  
 SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.

SEQ 0163

```

3901 020346 000414          BR      SPRNTA          ;BR TO COMMON SECTION.
3902
3903 020350 010267 160544    SPRNT1: MOV     R2      $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
3904 020354 005367 160540    DEC     $GDADR  ;ADJUST IT FOR PRINTOUT.
3905 020360 000407          BR      SPRNTA          ;BR TO COMMON SECTION.
3906
3907 020362 010367 160572    SPRNT3: MOV     R3      $TMPD   ;GET THE DATA IN R3.
3908 020366 010267 160526    SPRNT2: MOV     R2      $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
3909 020372 162767 000002 160520  SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
3910 020400 010067 160520    SPRNTA: MOV     R0      $GDDAT  ;GET WHAT THE DATA SHOULD BE
3911 020404 010167 160516    SPRNTB: MOV     R1      $BDDAT  ;GET WHAT THE DATA WAS
3912 020410 000207          RTS     PC           ;RETURN TO ENTER ERROR ROUTINES
3913
3914
3915
3916
3917
3918 020412 005710          *
3919 020414 001007          * SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
3920 020416 005760 000002    * R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
3921 020422 001004          *
3922 020424 004567 003062    *
3923 020430 026266          *
3924
3925 020432 000475          *
3926 020434          *
3927 020434 010146          *
3928 020436 010246          *
3929 020440 010346          *
3930 020442 010446          *
3931 020444 012701 000001    *
3932 020450 005002          *
3933 020452 012703 177777    *
3934 020456 010304          *
3935 020460 030110          *
3936 020462 001014          *
3937 020464 030260 000002    *
3938 020470 001011          *
3939 020472 105703          *
3940 020474 001042          *
3941 020476 162703 000001    *
3942 020502 005604          *
3943 020504 004567 003002    *
3944 020510 025517          *
3945 020512 000410          *
3946 020514 105703          *
3947 020516 001431          *
3948 020520 062703 000001    *
3949 020524 005504          *
3950 020526 004567 002760    *
3951 020532 025507          *
3952 020534          *
3953 020534 010346          *
3954 020536 010446          *
3955 020540 006303          *
3956 020542 006104          *

```

```

3957 020544 006003          ROR    R3          ;RESTORE BITS 14-0.
3958 020546 010446          MOV    R4,-(SP)     ;SAVE R4 FOR TYPEOUT
3959                                ;TYPE ADDRESS BITS 21-15
3960                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3961                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**
3962 020550 013746 177776     MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3963 020554 004767 004104     JSR    PC, $TYPOS  ;GO TO THE SUBROUTINE
3964 020560 003          .BYTE 3          ;TYPE 3 DIGIT(S)
3965 020561 000          .BYTE 0          ;SUPPRESS LEADING ZEROS
3966 020562 010346          MOV    R3,-(SP)     ;SAVE R3 FOR TYPEOUT
3967                                ;TYPE ADDRESS BITS 14-0
3968                                ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
3969                                ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**
3970 020564 013746 177776     MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
3971 020570 004767 004070     JSR    PC, $TYPOS  ;GO TO THE SUBROUTINE
3972 020574 005          .BYTE 5          ;TYPE 5 DIGIT(S)
3973 020575 001          .BYTE 1          ;TYPE LEADING ZEROS
3974 020576 012604          MOV    (SP)+,R4     ;POP STACK INTO R4
3975 020600 012603          MOV    (SP)+,R3     ;POP STACK INTO R3
3976 020602 062703 020000     5$:   ADD    #20000, R3 ;UPDATE TO NEXT BANK.
3977 020606 005504          ADC    R4          ;...HI ADDRESS BITS.
3978 020610 006301          ASL    R1          ;SHIFT POINTER...LO 64K.
3979 020612 006102          ROL    R2          ;...HI 64K.
3980 020614 103321          BCC    2$         ;BR IF MORE BANKS.
3981 020616 012604          MOV    (SP)+,R4     ;POP STACK INTO R4
3982 020620 012603          MOV    (SP)+,R3     ;POP STACK INTO R3
3983 020622 012602          MOV    (SP)+,R2     ;POP STACK INTO R2
3984 020624 012601          MOV    (SP)+,R1     ;POP STACK INTO R1
3985 020626 000207     6$:   RTS    PC          ;RETURN.
3986
3987 .SBTTL SCOPE HANDLER ROUTINE
3988
3989 ;*****
3990 ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
3991 ;*AND LOAD THE TEST NUMBER($STNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
3992 ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
3993 ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
3994 ;*SW14=1 LOOP ON TEST
3995 ;*SW11=1 INHIBIT ITERATIONS
3996 ;*SW09=1 LOOP ON ERROR
3997 ;*SW08=1 LOOP ON TEST IN SWR<4:0>
3998 ;*CALL
3999 ;* SCOPE ;;SCOPE=IOT
4000
4001 020630 $SCOPE:
4002 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4003 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**
4004 020630 013746 177776     MOV    @#PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4005 020634 004767 001524     JSR    PC, $CKSWR  ;GO TO THE SUBROUTINE
4006 020640 012504          MOV    (R5)+, R4    ;SAVE MINIMUM BLOCK MASK NEXT TEST.
4007 020642 010516          MOV    R5, (SP)     ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
4008 020644 032777 040000 160266 1$:   BIT    #BIT14,@SWR ;;LOOP ON PRESENT TEST?
4009 020652 001117          BNE    $OVER       ;;YES IF SW14=1
4010 ;*****START OF CODE FOR THE XOR TESTER*****
4011 020654 000416     $XTSTR: BR    6$    ;;IF RUNNING ON THE "XOR" TESTER CHANGE
4012                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)

```



4013	020656	013746	000004		MOV	2#ERRVEC, -(SP)	:: SAVE THE CONTENTS OF THE ERROR VECTOR
4014	020662	012737	020702	000004	MOV	#5\$, 2#ERRVEC	:: SET FOR TIMEOUT
4015	020670	005737	177060		TST	2#17706C	:: TIME OUT ON XOR?
4016	020674	012637	000004		MOV	(SP)+, 2#ERRVEC	:: RESTORE THE ERROR VECTOR
4017	020700	000466			BR	SSVLAD	:: GO TO THE NEXT TEST
4018	020702	022626		5\$:	CMP	(SP)+, (SP)+	:: CLEAR THE STACK AFTER A TIME OUT
4019	020704	012637	000004		MOV	(SP)+, 2#ERRVEC	:: RESTORE THE ERROR VECTOR
4020	020710	000426			BR	7\$	:: LOOP ON THE PRESENT TEST
4021	020712			6\$::#####	END OF CODE FOR THE XOR TESTER#####		
4022	020712	032777	000400	160220	BIT	#BIT08, 2SWR	:: LOOP ON SPEC. TEST?
4023	020720	001407			BEQ	2\$	:: BR IF NO
4024	020722	017746	160212		MOV	2SWR, -(SP)	:: SET DESIRED TEST NUM. FROM SWR
4025	020726	042716	000340		BIC	#SSWRMK, (SP)	:: STRIP AWAY UNDESIRED BITS
4026	020732	122667	160144		CMPB	(SP)+, \$STSTM	:: ON THE RIGHT TEST?
4027	020736	001465			BEQ	\$OVER	:: BR IF YES
4028	020740	105767	160137	2\$:	TSTB	\$ERFLG	:: HAS AN ERROR OCCURRED?
4029	020744	001421			BEQ	3\$	:: BR IF NO
4030	020746	126767	160143	160127	CMPB	\$ERMAX, \$ERFLG	:: MAX. ERRORS FOR THIS TEST OCCURRED?
4031	020754	101015			BHI	3\$	:: BR IF NO
4032	020756	032777	001000	160154	BIT	#BIT09, 2SWR	:: LOOP ON ERROR?
4033	020764	001404			BEQ	4\$	:: BR IF NO
4034	020766	016767	160116	160112	7\$:	MOV	\$LPERR, \$LPADR
4035	020774	000446			BR	\$OVER	:: SET LOOP ADDRESS TO LAST SCOPE
4036	020776	105067	160101	4\$:	CLRB	\$ERFLG	:: ZERO THE ERROR FLAG
4037	021002	005067	160162		CLR	\$TIMES	:: CLEAR THE NUMBER OF ITERATIONS TO MAKE
4038	021006	000415			BR	1\$	:: ESCAPE TO THE NEXT TEST
4039	021010	032777	004000	160122	3\$:	BIT	#BIT11, 2SWR
4040	021016	001011			BNE	1\$	:: BR IF YES
4041	021020	005767	160166		TST	\$PASS	:: IF FIRST PASS OF PROGRAM
4042	021024	001406			BEQ	1\$	:: INHIBIT ITERATIONS
4043	021026	005267	160052		INC	\$ICNT	:: INCREMENT ITERATION COUNT
4044	021032	026767	160132	160044	CMP	\$TIMES, \$ICNT	:: CHECK THE NUMBER OF ITERATIONS MADE
4045	021040	002024			BGE	\$OVER	:: BR IF MORE ITERATION REQUIRED
4046	021042	012767	000001	160034	1\$:	MOV	#1, \$ICNT
4047	021050	016767	000552	160112	MOV	\$MXCNT, \$TIMES	:: REINITIALIZE THE ITERATION COUNTER
4048	021056	105267	160020		SSVLAD:	INCB	\$STSTM
4049	021062	116767	160014	160120	MOV	\$STSTM, \$TESTN	:: SET TEST NUMBER IN APT MAILBOX
4050	021070	011667	160012		MOV	(SP), \$LPADR	:: SAVE SCOPE LOOP ADDRESS
4051	021074	011667	160010		MOV	(SP), \$LPERR	:: SAVE ERROR LOOP ADDRESS
4052	021100	005067	160066		CLR	\$ESCAPE	:: CLEAR THE ESCAPE FROM ERROR ADDRESS
4053	021104	112767	000001	160003	MOV	#1, \$ERMAX	:: ONLY ALLOW ONE(1) ERROR ON NEXT TEST
4054	021112	016777	157764	160022	\$OVER:	MOV	\$STSTM, 2DISPLAY
4055	021120	016716	157762		MOV	\$LPADR, (SP)	:: FUDGE RETURN ADDRESS
4056	021124	020516			INSERT:	CMP	R5, (SP)
4057	021126	001402			BEQ	1\$	:: CHECK FOR LOOP ON TEST.
4058	021130	000167	000470		JMP	ENDINS	:: BR IF START NEXT TEST.
4059	021134	012767	037777	160444	1\$:	MOV	#37777, BLKMSK
4060	021142	005767	160044		TST	\$PASS	:: SET BK BOUNDARY MASK.
4061	021146	001404			BEQ	2\$	:: CHECK FOR PASS 0.
4062	021150	126727	157726	000021	CMPB	\$STSTM, #21	:: BR IF PASS 0
4063	021156	103002			BHIS	3\$	:: CHECK IF IN SECTION 3.
4064	021160	006267	160422	2\$:	ASR	BLKMSK	:: BR IF IN SECTION 3.
4065	021164	016767	160372	160372	3\$:	MOV	FSTADR, TMPFAD
4066	021172	005767	157402		TST	RELOC	:: RESET BOUNDARY TO 4K.
4067	021176	001430			BEQ	4\$	:: GET FIRST ADDRESS.
4068	021200	032777	000040	157732	BIT	#SW05, 2SWR	:: CHECK IF PRG RELOCATED.
							:: BR IF NOT RELOCATED.
							:: CHECK IF LOC 0-776 TO BE PROTECTED.



4069	021206	001424				BEQ	4\$		:BR IF SW NOT SET.
4070	021210	026727	160350	001000		CMP	TMPFAD, #1000		:CHECK IF NOT BEING TESTED.
4071	021216	103020				BHIS	4\$		:BR IF ALREADY PROTECTED.
4072	021220	012767	001000	160336		MOV	#1000, TMPFAD		:RESET FIRST ADDRESS.
4073	021226	052767	000001	160334		BIS	#BIT0, FADMAP		:SET FLAG IN FIRST BANK.
4074	021234	026727	160334	001000		CMP	LSTADR, #1000		:CHECK IF GONE PAST LAST ADR.
4075	021242	101006				BHI	4\$		:BR IF ENOUGH MEMORY.
4076	021244	004567	002242			JSR	R5, \$PRINT		:GO PRINT OUT THE FOLLOWING MESSAGE.
4077	021250	026737				.WORD	NOMTST		:ADDRESS OF MESSAGE TO BE TYPED
4078									: "NO MEMORY TESTED"
4079	021252	016716	160366			MOV	.TST32, (SP)		:ADJUST RETURN ADR FOR ABORT.
4080	021256	000207				RTS	PC		:ABORT.
4081	021260	016767	160310	160310	4\$:	MOV	LSTADR, TEMPLAD		:GET LAST ADDRESS.
4082	021266	016767	160242	160234		MOV	SAVTST, TSTMAP		:GET TEST MAP, LO 64K.
4083	021274	016767	160236	160230		MOV	SAVTST+2, TSTMAP+2		:HI 64K.
4084	021302	046767	157274	160220		BIC	PRGMAP, TSTMAP		:DON'T TEST OVER THE PROCLAM.
4085	021310	046767	157270	160214		BIC	PRGMAP+2, TSTMAP+2		
4086	021316	005767	157670			TST	\$PASS		:CHECK FOR FIRST PASS
4087	021322	001011				BNE	10\$		:BR IF NOT FIRST PASS.
4088	021324	032767	000003	160176		BIT	#3, TSTMAP		:CHECK IF FIRST TWO BANKS AVAILABLE.
4089	021332	001405				BEQ	10\$		:NOT TESTING FIRST 2 BANKS.
4090	021334	042767	177774	160166		BIC	#177774, TSTMAP		:CLR ALL BUT FIRST 2 BANKS.
4091	021342	005067	160164			CLR	TSTMAP+2		
4092	021346	005704			10\$:	TST	R4		:CHECK FOR A MINIMUM BLOCK SIZE.
4093	021350	001503				BEQ	20\$		:BR IF NO MIN BLOCK SIZE.
4094	021352	030467	160206			BIT	R4, TMPFAD		:CHECK IF FIRST ADR ON BLOCK BOUNDARY.
4095	021356	001416				BEQ	11\$		:BR IF FIRST ADR ON BLOCK BOUNDARY.
4096	021360	050467	160200			BIS	R4, TMPFAD		:ADJUST FIRST ADR TO END OF BLOCK.
4097	021364	005267	160174			INC	TMPFAD		:FIRST ADR TO FIRST ADR OF NEXT BLOCK.
4098	021370	032767	017777	160166		BIT	#MASK4K, TMPFAD		:CHECK IF FIRST ADR REACHED 4K BOUNDARY.
4099	021376	001006				BNE	11\$		:BR IF NOT ON 4K BOUNDARY.
4100	021400	046767	160164	160122		BIC	FADMAP, TSTMAP		:DON'T TEST FIRST BANK.
4101	021406	046767	160160	160116		BIC	FADMAP+2, TSTMAP+2		
4102	021414	030467	160156		11\$:	BIT	R4, TEMPLAD		:CHECK IF LAST ADR ON BLOCK BOUNDARY.
4103	021420	001414				BEQ	12\$		:BR IF ON BLOCK BOUNDARY.
4104	021422	040467	160150			BIC	R4, TEMPLAD		:ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDARY.
4105	021426	032767	017777	160142		BIT	#MASK4K, TEMPLAD		:CHECK IF ADJUSTED TO 4K BOUNDARY.
4106	021434	001006				BNE	12\$		:BR IF NOT ON 4K BOUNDARY.
4107	021436	046767	160140	160064		BIC	LADMAP, TSTMAP		:SKIP TESTING LAST BANK.
4108	021444	046767	160134	160060		BIC	LADMAP+2, TSTMAP+2		
4109	021452	036767	160112	160122	12\$:	BIT	FADMAP, LADMAP		:CHECK IF FIRST AND LAST IN SAME BANK.
4110	021460	001004				BNE	13\$		:BR IF IN SAME BANK.
4111	021462	036767	160104	160114		BIT	FADMAP+2, LADMAP+2		:... UPPER 64K.
4112	021470	001404				BEQ	14\$		:BR IF FIRST AND LAST NOT SAME BANK.
4113	021472	026767	160100	160064	13\$:	CMP	TEMPLAD, TMPFAD		:CHECK IF ANY MEMORY LEFT.
4114	021500	101406				BLOS	15\$		:BR IF NO MEMORY TO TEST.
4115	021502	005767	160022		14\$:	TST	TSTMAP		:CHECK IF ANY BANKS LEFT TO TEST!!
4116	021506	001017				BNE	16\$		:BR IF TEST MAP NOT EMPTY.
4117	021510	005767	160016			TST	TSTMAP+2		:CHECK FOR ANY BANKS.
4118	021514	001014				BNE	16\$		:BR IF TEST MAP NOT EMPTY.
4119	021516				15\$:				
4120	021516	004567	001770			JSR	R5, \$PRINT		:GO PRINT OUT THE FOLLOWING MESSAGE.
4121	021522	026763				.WORD	SKPMES		:ADDRESS OF MESSAGE TO BE TYPED
4122									: "SKIPPING TEST #"
4123	021524	005046				CLR	-(SP)		:CLEAR THE WORD ON THE STACK.
4124	021526	116716	157350			MOVB	\$TSTNM, (SP)		:PUT THE DATA ON THE STACK.

```

4125      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4126      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4127      MOV      @#PSW, -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
4128      JSR      PC, $TYPOS      ;GO TO THE SUBROUTINE
4129      .BYTE    3                ;TYPE 3 DIGITS.
4130      .BYTE    1                ;TYPE LEADING ZEROS.
4131      BR      ENDINS           ;RETURN TO SKIP TEST.
4132      16$:    ADD      #4, (SP)   ;SKIP THE SKIP ON RETURN.
4133      ADD      #4, $LPADR      ;ADJUST THE LOOP ADR PAST THE SKIP.
4134      20$:    MOV      #MASK4K, FADMSK ;GET 4K MASK.
4135      MOV      TMPFAD, R5      ;GET FIRST ADR.
4136      21$:    BIC      R5, FADMSK ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
4137      ASL      R5             ;MOVE LOWEST BIT UP ONE.
4138      BNE     21$            ;LOOP UNTIL OVERFLOW.
4139      MOV      #MASK4K, LADMSK ;SET MASK BITS
4140      MOV      TEMPLAD, R5     ;GET LAST ADR.
4141      22$:    BIC      R5, LADMSK ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
4142      ASL      R5             ;MOVE LOWEST BIT OF LAST ADR UP ONE.
4143      BNE     22$            ;LOOP UNTIL OVERFLOW.
4144      ENDINS:  RTS             ;EXIT SCOPE ROUTINE BACK TO TEST.
4145      $MXCNT: 4                ;MAX. NUMBER OF ITERATIONS
4146      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4147      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4148      MOV      @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
4149      JSR      PC, $CKSWR     ;GO TO THE SUBROUTINE
4150      .SBTTL  ERROR HANDLER ROUTINE
4151
4152      ;*****
4153      ;THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
4154      ;SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
4155      ;AND GO TO $ERRTYP ON ERROR
4156      ;THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
4157      ;$SW15=1      HALT ON ERROR
4158      ;$SW13=1      INHIBIT ERROR TYPEOUTS
4159      ;$SW10=1      BELL ON ERROR
4160      ;$SW09=1     LOOP ON ERROR
4161      ;CALL
4162      ;*      ERROR      N      ;;ERROR=EMI AND N=ERROR ITEM NUMBER
4163
4164      $ERROR:
4165      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4166      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4167      MOV      @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
4168      JSR      PC, $CKSWR     ;GO TO THE SUBROUTINE
4169      ADD      #2, (SP)        ;ADJUST POINTER PAST CODE WORD.
4170      7$:    INCB     $ERRFLG   ;SET THE ERROR FLAG
4171      BEQ     7$             ;DON'T LET THE FLAG GO TO ZERO
4172      MOV     $STNM, @DISPLAY  ;DISPLAY TEST NUMBER AND ERROR FLAG
4173      BIT     #BIT10, @SWR    ;BELL ON ERROR?
4174      BEQ     1$             ;NO - SKIP
4175      JSR     R5, $SPRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4176      .WORD  $BELL          ;ADDRESS OF MESSAGE TO BE TYPED
4177      1$:    INC      $ERRTTL  ;COUNT THE NUMBER OF ERRORS
4178      MOV     (SP), $ERRPC    ;GET ADDRESS OF ERROR INSTRUCTION
4179      SUB     #2, $ERRPC
4180      MOV     @ERRPC, $ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE

```



CZQMCF.P11 14-FEB-78 08:19

ERROR HANDLER ROUTINE

```

4181 021732 032777 020000 157200 BIT #BIT13,DSWR ;;SKIP TYPEOUT IF SET
4182 021740 001005 BNE 20$ ;;SKIP TYPEOUTS
4183 021742 004767 000116 JSR PC,$ERRTYP ;;GO TO USER ERROR ROUTINE
4184 021746 004567 001540 JSR R5,$SPRINT ;;GO PRINT OUT THE FOLLOWING MESSAGE.
4185 021752 001201 .WORD $CRLF ;;ADDRESS OF MESSAGE TO BE TYPED
4186 021754 20$: CMPB #APTENV,$ENV ;;RUNNING IN APT MODE
4187 021754 122767 000001 157242 BNE 2$ ;;NO SKIP APT ERROR REPORT
4188 021762 001007 MOVB $ITEMB,21$ ;;SET ITEM NUMBER AS ERROR NUMBER
4189 021764 116767 157124 000004 JSR PC,$ATY4 ;;REPORT FATAL ERROR TO APT
4190 021772 004767 002044 21$: .BYTE 0
4191 021776 000 .BYTE 0
4192 021777 000 22$: BR 22$ ;;APT ERROR LOOP
4193 022000 000777 157132 2$: TST DSWR ;;HALT ON ERROR
4194 022002 005777 BPL 3$ ;;SKIP IF CONTINUE
4195 022006 100005 HALT ;;HALT ON ERROR!
4196 022010 000000
4197 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
4198 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4199 022012 013746 177776 MOV @PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4200 022016 004767 000342 JSR PC,$CKSWR ;GO TO THE SUBROUTINE
4201 022022 032777 001000 157110 3$: BIT #BIT09,DSWR ;;LOOP ON ERROR SWITCH SET?
4202 022030 001402 BEQ 4$ ;;BR IF NO
4203 022032 016716 157052 MOV $LPERR,(SP) ;;FUDGE RETURN FOR LOOPING
4204 022036 005767 157130 4$: TST $ESCAPE ;;CHECK FOR AN ESCAPE ADDRESS
4205 022042 001402 BEQ 5$ ;;BR IF NONE
4206 022044 016716 157122 5$: MOV $ESCAPE,(SP) ;;FUDGE RETURN ADDRESS FOR ESCAPE
4207 022050 6$: CMP #SENDAD,@#42 ;;ACT-11 AUTO-ACCEPT?
4208 022050 022737 014222 000042 BNE 6$ ;;BRANCH IF NO
4209 022056 001001 HALT ;;YES
4210 022060 000000
4211 022062 6$: RTS PC
4212 022062 000207
4213 ;;*****
4214
4215 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
4216
4217 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4218 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),
4219 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4220
4221 $ERRTYP:
4222 022064 004567 001422 JSR R5,$SPRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4223 022070 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4224 022072 010046 MOV RO,-(SP) ;SAVE RO
4225 022074 005000 CLR RO ;PICKUP THE ITEM INDEX
4226 022076 156700 157012 BISB $ITEMB,RO
4227 022102 001007 BNE 1$ ;IF ITEM NUMBER IS ZERO, JUST
4228 ;TYPE THE PC OF THE ERROR
4229 022104 016746 157006 MOV $ERRPC,-(SP) ;;SAVE $ERRPC FOR TYPEOUT
4230 ;ERROR ADDRESS
4231 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4232 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4233 022110 013746 177776 MOV @PSW,-(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4234 022114 004767 002570 JSR PC,$TYPOC ;GO TO THE SUBROUTINE
4235 022120 000513 BR 10$ ;GET OUT
4236 022122 016767 156770 157364 1$: MOV $ERRPC,$VERPC ;SET UP VIRTUAL PC FOR TYPEOUT.

```



```

4237 022130 166767 56444 157356 SUB RELOCF, $VERPC ;MAKE VIRTUAL IF NOT ALREADY.
4238 022136 005300 DEC RO ;ADJUST THE INDEX SO THAT IT WILL
4239 022140 006300 ASL RO ; WORK FOR THE ERROR TABLE
4240 022142 006300 ASL RO
4241 022144 006300 ASL RO
4242 022146 066700 157466 ADD .ERRTB, RO ;FORM TABLE POINTER
4243 022152 012067 000006 MOV (RO)+,2$ ;PICKUP "ERROR MESSAGE" POINTER
4244 022156 001406 BEQ 3$ ;SKIP TYPEOUT IF NO POINTER
4245 022160 004567 001326 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4246 022164 000000 2$: .WORD 0 ;"ERROR MESSAGE" POINTER GOES HERE
4247 022166 004567 001320 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4248 022172 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4249 022174 012067 000006 3$: MOV (RO)+,4$ ;PICKUP "DATA HEADER" POINTER
4250 022200 001406 BEQ 5$ ;SKIP TYPEOUT IF 0
4251 022202 004567 001304 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4252 022206 000000 4$: .WORD 0 ;"DATA HEADER" PC.NTER GOES HERE
4253 022210 004567 001276 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
4254 022214 001201 .WORD $CRLF ;ADDRESS OF MESSAGE TO BE TYPED
4255 022216 010146 5$: MOV R1, -(SP) ;SAVE R1
4256 022220 012001 MOV (RO)+,R1 ;PICKUP "DATA TABLE" POINTER
4257 022222 001451 BEQ 9$ ;BR IF NO DATA TO BE TYPED
4258 022224 066701 156350 ADD RELOCF, R1 ;ADJUST POINTER
4259 022230 012000 MOV (RO)+,RO ;PICKUP "DATA FORMAT" POINTER
4260 022232 066700 156342 ADD RELOCF, RO ;ADJUST POINTER.
4261 022236 105720 6$: TSTB (RO)+ ;CHECK THE FORMAT
4262 022240 001006 BNE 7$ ;BR IF NOT 16-BIT OCTAL
4263 022242 013146 MOV 2(R1)+, -(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4264 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
4265 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4266 022244 013746 177776 MOV 2$PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4267 022250 004767 002434 JSR PC, $TYPOC ;GO TO THE SUBROUTINE
4268 022254 000426 BR 8$
4269 022256 100406 7$: BMI 17$ ;BRANCH IF NOT DECIMAL
4270 022260 013146 MOV 2(R1)+, -(SP) ;SAVE 2(R1)+ FOR TYPEOUT
4271 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
4272 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4273 022262 013746 177776 MOV 2$PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4274 022266 004767 002140 JSR PC, $TYPDS ;GO TO THE SUBROUTINE
4275 022272 000417 BR 8$ ;SKIP
4276 022274 122760 177777 17$: CMPB #-1, -1(RO) ;CHECK FOR 18-BIT ADDRESS FORMAT.
4277 022302 001004 BNE 18$ ;BR IF NOT 18-BIT ADDRESS FORMAT.
4278 022304 013146 MOV 2(R1)+, -(SP) ;PUT THE DATA ON THE STACK.
4279 022306 004767 002640 JSR PC, $TYPAD ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
4280 022312 000407 BR 8$ ;SKIP
4281 022314 18$:
4282 022314 005046 CLR -(SP) ;CLEAR THE WORD ON THE STACK.
4283 022316 113116 MOVB 2(R1)+, (SP) ;PUT THE DATA ON THE STACK.
4284 ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
4285 ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4286 022320 013746 177776 MOV 2$PSW, -(SP) ;PUT THE PROCESSOR STATUS ON THE STACK
4287 022324 004767 002334 JSR PC, $TYPOS ;GO TO THE SUBROUTINE
4288 022330 003 .BYTE 3 ;TYPE 3 DIGITS.
4289 022331 001 .BYTE 1 ;TYPE LEADING ZEROS.
4290 022332 005711 8$: TST (R1) ;IS THERE ANOTHER NUMBER?
4291 022334 001404 BEQ 9$ ;BR IF NO
4292 022336 004567 001150 JSR R5, $PRINT ;GO PRINT OUT THE FOLLOWING MESSAGE.
    
```

```

022342 022362      WORD 118      : ADDRESS OF MESSAGE TO BE TYPED
022344 000734      BR      68      : LOOP

022346 012601      98:  MOV  (SP),R1      : RESTORE R1
022348 012600      108: MOV  (SP),R0      : RESTORE R0
022350 004567      JSR  R5,SPRINT : GO PRINT OUT THE FOLLOWING MESSAGE.
022352 001201      .WORD SCALF   : ADDRESS OF MESSAGE TO BE TYPED
022354 000207      RTS  PC      : RETURN
022356 000011      118: .ASCIZ : TAB CHARACTER.
          EVEN
          .SBTTL ITY INPUT ROUTINE

```

\*\*\*\*\*  
: ENABL LSB

\*\*\*\*\*  
: SOFTWARE SWITCH REGISTER CHANGE ROUTINE.  
: ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL  
: SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL  
: WHEN OPERATING IN ITY FLAG MODE.

```

022364 022767 000176 156546 $CKSWR: CMP  @SWREG,SWR      : IS THE SOFT-SWR SELECTED?
022366 001104      BNE  156      : BRANCH IF NO
022368 105777 156544      TSTB @STKS      : CHAR THERE?
022370 100101      BPL  156      : IF NO, DON'T WAIT AROUND
022372 117746 156540      MOVB @STKB,-(SP)   : SAVE THE CHAR
022374 042716 177600      BIC  @C177,(SP)   : STRIP-OFF THE ASCII
022376 022726 000007      CMP  @7,(SP)+     : IS IT A CONTROL G?
022378 001072      BNE  156      : NO, RETURN TO USER
022380 126727 156510 000001      CMPB @AUTOB,@1   : ARE WE RUNNING IN AUTO-MODE?
022382 001466      BEQ  156      : BRANCH IF YES

```

```

022430 004567 001056      JSR  R5,SPRINT : GO PRINT OUT THE FOLLOWING MESSAGE.
022432 023311      .WORD SCNTLG  : ADDRESS OF MESSAGE TO BE TYPED

```

\$GTSWR: JSR R5,SPRINT : GO PRINT OUT THE FOLLOWING MESSAGE.  
: ADDRESS OF MESSAGE TO BE TYPED  
: SAVE SWREG FOR TYPEOUT

: \* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE \$TYPOC ROUTINE  
: \* WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY \*\*SYSMAC\*\*.

```

022450 013746 177776      MOV  @PSW,-(SP)   : PUT THE PROCESSOR STATUS ON THE STACK
022452 004767 002230      JSR  PC,STYPOC  : GO TO THE SUBROUTINE
022454 004567 001026      JSR  R5,SPRINT : GO PRINT OUT THE FOLLOWING MESSAGE.
022456 023327      .WORD SMNEW   : ADDRESS OF MESSAGE TO BE TYPED
022458 005046      198: CLR  -(SP)      : CLEAR COUNTER
022460 005046      78: CLR  -(SP)      : THE NEW SWR
022462 105777 156446      TSTB @STKS      : CHAR THERE?
022464 100375      SPL  78      : IF NOT TRY AGAIN

```

```

022500 117746 156442      MOVB @STKB,-(SP)   : PICK UP CHAR
022502 042716 177600      BIC  @C177,(SP)   : MAKE IT 7-BIT ASCII

```

```

022510 021627 000025      98:  CMP  (SP),@25   : IS IT A CONTROL-U?
022512 001006      BNE  108      : BRANCH IF NOT
022514 004567 000770      JSR  R5,SPRINT : GO PRINT OUT THE FOLLOWING MESSAGE.

```



```

4349 022522 023304          .WORD  SCNTLU          :ADDRESS OF MESSAGE TO BE TYPED
4350 022524 062706 000006 20$:  ADD      #6,SP          : : IGNORE PREVIOUS INPUT
4351 022530 000756          BR        19$          : : LET'S TRY IT AGAIN
4352
4353
4354 022532 021627 000015 10$:  CMP      (SP),#15        : : IS IT A <CR>?
4355 022536 001023          BNE      16$          : : BRANCH IF NO
4356 022540 005766 000004          TST      4(SP)         : : YES, IS IT THE FIRST CHAR?
4357 022544 001403          BEQ      11$          : : BRANCH IF YES
4358 022546 016677 000002 156364  MOV      2(SP),#SWR     : : SAVE NEW SWR
4359 022554 062706 000006 11$:  ADD      #6,SP          : : CLEAR UP STACK
4360 022560
4361 022560 004567 000726          JSR      RS,SPRINT     : : GO PRINT OUT THE FOLLOWING MESSAGE.
4362 022564 001201          .WORD  SCALF          : : ADDRESS OF MESSAGE TO BE TYPED
4363 022566 126727 156343 000001  CMPB     $INTAG,#1     : : RE-ENABLE TTY KBD INTERRUPTS?
4364 022574 001003          BNE      15$          : : BRANCH IF NOT
4365 022576 012777 000100 156340  MOV      #100,#STKS    : : RE-ENABLE TTY KBD INTERRUPTS
4366 022604 000002          RTI                     : : RETURN
4367 022606 004767 001142 15$:  JSR      PC,$TYPEC    : : ECHO CHAR
4368 022612 021627 000060 16$:  CMP      (SP),#60     : : CHAR < 0?
4369 022616 002420          BLT      18$          : : BRANCH IF YES
4370 022620 021627 000067          CMP      (SP),#67     : : CHAR > 7?
4371 022624 003015          BGT      18$          : : BRANCH IF YES
4372 022626 042726 000060          BIC      #60,(SP)+    : : STRIP-OFF ASCII
4373 022632 005766 000002          TST      2(SP)         : : IS THIS THE FIRST CHAR
4374 022636 001403          BEQ      17$          : : BRANCH IF YES
4375 022640 006316          ASL      (SP)         : : NO, SHIFT PRESENT
4376 022642 006316          ASL      (SP)         : : CHAR OVER TO MAKE
4377 022644 006316          ASL      (SP)         : : ROOM FOR NEW ONE.
4378 022646 005266 000002 17$:  INC      2(SP)         : : KEEP COUNT OF CHAR
4379 022652 056616 177776          BIS      -2(SP),(SP)  : : SET IN NEW CHAR
4380 022656 000705          BR        7$          : : GET THE NEXT ONE
4381 022660
4382 022660 004567 000626          JSR      RS,SPRINT     : : GO PRINT OUT THE FOLLOWING MESSAGE.
4383 022664 001200          .WORD  $QUES          : : ADDRESS OF MESSAGE TO BE TYPED
4384 022666 000716          BR        20$          : : SIMULATE CONTROL-U
4385 .DSABL  LSB
4386
4387
4388
4389
4390
4391
4392
4393
4394
4395
4396 022670 011646          $RDCHR: MOV      (SP),-(SP) : : PUSH DOWN THE PC
4397 022672 016666 000004 000002  MOV      4(SP),2(SP)   : : SAVE THE PS
4398 022700 105777 156240 1$:  TSTB     #STKS         : : WAIT FOR
4399 022704 100375          BPL      1$           : : A CHARACTER
4400 022706 117766 156234 000004  MOVB     #STKB,4(SP)   : : READ THE TTY
4401 022714 042766 177600 000004  BIC      #1<17>,4(SP)  : : GET RID OF JUNK IF ANY
4402 022722 026627 000004 000023  CMP      4(SP),#23     : : IS IT A CONTROL-S?
4403 022730 001013          BNE      3$           : : BRANCH IF NO
4404 022732 105777 156206 2$:  TSTB     #STKS         : : WAIT FOR A CHARACTER

```

```

: : *****
: : *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
: : *CALL:
: : * RDCHR          : : INPUT A SINGLE CHARACTER FROM THE TTY
: : * RETURN HERE   : : CHARACTER IS ON THE STACK
: : *              : : WITH PARITY BIT STRIPPED OFF
: :

```



```

4405 022736 100375          SPL      2$          ;; LOOP UNTIL ITS THERE
4406 022740 117746 156202  MOVE     3$STKB, -(SP) ;; GET CHARACTER
4407 022744 042716 177600  BIC     4$C177, (SP)  ;; MAKE IT 7-BIT ASCII
4408 022750 022627 000021  CMP     (SP)+, 4$21  ;; IS IT A CONTROL-Q?
4409 022754 001366          BNE     2$          ;; IF NOT DISCARD IT
4410 022756 000750          BR      1$          ;; YES, RESUME
4411 022760 026627 000004 000140 3$:  CMP     4$(SP), 4$140 ;; IS IT UPPER CASE?
4412 022766 002407          BLT     4$          ;; BRANCH IF YES
4413 022770 026627 000004 000175  CMP     4$(SP), 4$175 ;; IS IT A SPECIAL CHAR?
4414 022776 003003          BGT     4$          ;; BRANCH IF YES
4415 023000 042766 000040 000004  BIC     4$40, 4$(SP)  ;; MAKE IT UPPER CASE
4416 023006 000002          RTI     ;; GO BACK TO USER
4417
4418 *****
4419 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
4420 *CALL:
4421 *      RDLIN          ;; INPUT A STRING FROM THE TTY
4422 *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
4423 *      *              ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
4424
4425 023010 010346          $RDLIN: MOV     R3, -(SP)  ;; SAVE R3
4426 023012 005046          CLR     -(SP)        ;; CLEAR THE RUBOUT KEY
4427 023014 012703 023274 1$:  MOV     4$TTYIN, R3  ;; GET ADDRESS
4428 023020 022703 023304 2$:  CMP     4$TTYIN+8, R3 ;; BUFFER FULL?
4429 023024 101467          BLOS   4$          ;; BR IF YES
4430
4431 * THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
4432 * WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSTEMAC**.
4433 023026 013746 177776  MOV     3$PSW, -(SP)  ;; PUT THE PROCESSOR STATUS ON THE STACK
4434 023032 004767 177632  JSR     PC, $RDCHR    ;; GO TO THE SUBROUTINE
4435 023036 112613          MOVVB  (SP)+, (R3)    ;; GET CHARACTER
4436 023040 122713 000177 10$:  CMPB   4$177, (R3)   ;; IS IT A RUBOUT
4437 023044 001024          BNE     5$          ;; BR IF NO
4438 023046 005716          TST   (SP)          ;; IS THIS THE FIRST RUBOUT?
4439 023050 001010          BNE     6$          ;; BR IF NO
4440 023052 112767 000134 000212  MOVVB  4$'\, 9$      ;; TYPE A BACK SLASH
4441 023060 004567 000426  JSR     R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4442 023064 023272          .WORD  9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4443 023066 012716 177777  MOV     4$-1, (SP)   ;; SET THE RUBOUT KEY
4444 023072 005303 6$:  DEC     R3          ;; BACKUP BY ONE
4445 023074 020327 023274  CMP     R3, 4$TTYIN  ;; STACK EMPTY?
4446 023100 103441          BLO     4$          ;; BR IF YES
4447 023102 111367 000164  MOVVB  (R3), 9$      ;; SETUP TO TYPEOUT THE DELETED CHAR.
4448 023106 004567 000400  JSR     R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4449 023112 023272          .WORD  9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4450 023114 000741          BR      2$          ;; GO READ ANOTHER CHAR.
4451 023116 005716 5$:  TST   (SP)          ;; RUBOUT KEY SET?
4452 023120 001407          BEQ     7$          ;; BR IF NO
4453 023122 112767 000134 000142  MOVVB  4$'\, 9$      ;; TYPE A BACK SLASH
4454 023130 004567 000356  JSR     R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4455 023134 023272          .WORD  9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4456 023136 005016          CLR     (SP)        ;; CLEAR THE RUBOUT KEY
4457 023140 122713 000025 7$:  CMPB   4$25, (R3)   ;; IS CHARACTER A CTRL U?
4458 023144 001004          BNE     8$          ;; BR IF NO
4459 023146 004567 000340  JSR     R5, $SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4460 023152 023304          .WORD  9$          ;; ADDRESS OF MESSAGE TO BE TYPED
4461 023154 000717          BR      1$          ;; GO START OVER
4462 023156 122713 8$:  CMPB   4$22, (R3)   ;; IS CHARACTER A "R"?

```

```

4461 023162 001014      BNE      3$          ;; BRANCH IF NO
4462 023164 105013      CLRB     (R3)       ;; CLEAR THE CHARACTER
4463 023166 004567 000320   JSR      R5         SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4464 023172 001201      .WORD   $CALF      ;; ADDRESS OF MESSAGE TO BE TYPED
4465 023174 004567 000312   JSR      R5         SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4466 023200 023274      .WORD   $TTYIN     ;; ADDRESS OF MESSAGE TO BE TYPED
4467 023202 000706      BR       2$        ;; GO PICKUP ANOTHER CHAFTER
4468 023204
4469 023204 004567 000302   4$:      JSR      R5         SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4470 023210 001200      .WORD   $QUES     ;; ADDRESS OF MESSAGE TO BE TYPED
4471 023212 000700      BR       1$        ;; CLEAR THE BUFFER AND LOOP
4472 023214 111367 000052   3$:      MOVB    (R3),9$    ;; ECHO THE CHARACTER
4473 023220 004567 000266   JSR      R5         SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4474 023224 023272      .WORD   9$        ;; ADDRESS OF MESSAGE TO BE TYPED
4475 023226 122723 000015   CMPB    #15,(R3)+  ;; CHECK FOR RETURN
4476 023232 001272      BNE     2$        ;; LOOP IF NOT RETURN
4477 023234 105063 177777   CLRB    -1(R3)    ;; CLEAR RETURN (THE 15)
4478 023240 004567 000246   JSR      R5         SPRINT  ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4479 023244 001202      .WORD   $LF       ;; ADDRESS OF MESSAGE TO BE TYPED
4480 023246 005726      TST     (SP)+     ;; CLEAN RUBOUT KEY FROM THE STACK
4481 023250 012603      MOV     (SP)+,R3  ;; RESTORE R3
4482 023252 011646      MOV     (SP)-,(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
4483 023254 016666 000004 000002   MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
4484 023262 012766 023274 000004   MOV     # $TTYIN,4(SP)
4485 023270 000002      RTI
4486 023272 000          9$:      .BYTE   0         ;; RETURN
4487 023273 000          .BYTE   0         ;; STORAGE FOR ASCII CHAR. TO TYPE
4488 023274 000010      .BLKB  8         ;; TERMINATOR
4489 023304 052536 005015 000      $TTYIN: .ASCIZ  /?U/<15><12>  ;; RESERVE 8 BYTES FOR TTY INPUT
4490 023311 136 006507 000012 020122  $CNTLU: .ASCIZ  /?G/<15><12>  ;; CONTROL "U"
4491 023316 005015 053523 020122  $MSWR:  .ASCIZ  <15><12>/SWR = /  ;; CONTROL "G"
4492 023324 020075 000
4493 023327 040 047040 053505  $MNEW:  .ASCIZ  / NEW = /
4494 023334 036440 000040
4495      .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
4496
4497      ;; *****
4498      ;; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
4499      ;; *CHANGE IT TO BINARY.
4500      ;; *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
4501      ;; *OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
4502      ;; *FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
4503      ;; *THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
4504      ;; *CALL:
4505      ;; *      RDOCT          ;; READ AN OCTAL NUMBER
4506      ;; *      RETURN HERE   ;; LOW ORDER BITS ARE ON TOP OF THE STACK
4507      ;; *                  ;; HIGH ORDER BITS ARE IN $HIOCT
4508
4509 023340 011646 000004 000002  $RDOCT: MOV     (SP)-,(SP)  ;; PROVIDE SPACE FOR THE
4510 023342 016666      MOV     4(SP),2(SP)  ;; INPUT NUMBER
4511 023350 010046      MOV     R0,-(SP)    ;; PUSH R0 ON STACK
4512 023352 010146      MOV     R1,-(SP)    ;; PUSH R1 ON STACK
4513 023354 010246      MOV     R2,-(SP)    ;; PUSH R2 ON STACK
4514 023356
4515      1$:      ;; THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
4516      ;; * WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.

```



```

4517 023356 013746 177776      MOV      2#PSW, -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
4518 023362 004767 177422      JSR      PC, $RDLIN      ;GO TO THE SUBROUTINE
4519 023366 012600              MOV      (SP)+,R0        ;GET ADDRESS OF 1ST CHARACTER
4520 023370 010067 000102      MOV      R0,$$          ;AND SAVE IT
4521 023374 005001              CLR      R1              ;CLEAR DATA WORD
4522 023376 005002              CLR      R2
4523 023400 112046              2$:     MOVB     (R0)+,-(SP)  ;PICKUP THIS CHARACTER
4524 023402 001420              BEQ      3$              ;IF ZERO GET OUT
4525 023404 122716 000060      CMPB     #'0,(SP)        ;MAKE SURE THIS CHARACTER
4526 023410 003026              BGT      4$              ;IS AN OCTAL DIGIT
4527 023412 122716 000067      CMPB     #'7,(SP)
4528 023416 002423              4$:     BLT      R1          ;:*2
4529 023420 006301              ASL      R1
4530 023422 006102              ROL      R2
4531 023424 006301              ASL      R1          ;:*4
4532 023426 006102              ROL      R2
4533 023430 006301              ASL      R1          ;:*8
4534 023432 006102              ROL      R2
4535 023434 042716 177770      BIC      #'C7,(SP)      ;STRIP THE ASCII JUNK
4536 023440 062601              ADD      (SP)+,R1      ;ADD IN THIS DIGIT
4537 023442 000756              BR       2$              ;LOOP
4538 023444 005726              3$:     TST      (SP)+      ;CLEAN TERMINATOR FROM STACK
4539 023446 010166 000012      MOV      R1,12(SP)     ;SAVE THE RESULT
4540 023452 010267 000032      MOV      R2,$HIOCT
4541 023456 012602              MOV      (SP)+,R2      ;POP STACK INTO R2
4542 023460 012601              MOV      (SP)+,R1      ;POP STACK INTO R1
4543 023462 012600              MOV      (SP)+,R0      ;POP STACK INTO R0
4544 023464 000002              RTI
4545 023466 005726              4$:     TST      (SP)+      ;CLEAN PARTIAL FROM STACK
4546 023470 105010              CLRB     (R0)          ;SET A TERMINATOR
4547 023472 004567 000014      JSR      RS, $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4548 023476 000000              5$:     .WORD   0
4549 023500 004567 000006      JSR      RS, $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4550 023504 001200              .WORD   $QUES          ;ADDRESS OF MESSAGE TO BE TYPED
4551 023506 000723              BR       1$            ;TRY AGAIN
4552 023510 000000              $HIOCT: .WORD   0      ;HIGH ORDER BITS GO HERE
4553
4554
4555
4556
4557
4558
4559 023512 012567 000016      $PRINT: MOV      (RS)+, 1$ ;GET THE MESSAGE VIRTUAL ADDRESS.
4560 023516 066767 155056 000010  ADD      RELOC, 1$      ;MAKE IT PHYSICAL.
4561
4562
4563 023524 013746 177776      ;* THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
4564 023530 004767 000004      ;* WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
4565 023534 000000              MOV      2#PSW, -(SP)  ;PUT THE PROCESSOR STATUS ON THE STACK
4566 023536 000205              JSR      PC, $TYPE     ;GO TO THE SUBROUTINE
4567
4568
4569
4570
4571
4572
1$:     .WORD   0          ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
RTS     RS              ;RETURN.

.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.

```



```

4573      : *NOTE1:          $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
4574      : *NOTE2:          $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
4575      : *NOTE3:          $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
4576      : *
4577      : *CALL:
4578      : *1) USING A TRAP INSTRUCTION
4579      : *      TYPE      .MESADR      ;; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
4580      : *OR
4581      : *      TYPE
4582      : *      MESADR
4583      : *
4584
4585 023540 105767 155413 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
4586 023544 100002      BPL 1$          ;; BR IF YES
4587 023546 000000      HALT      ;; HALT HERE IF NO TERMINAL
4588 023550 000430      BR          ;; LEAVE
4589 023552 010046      1$: MOV RO, -(SP)      ;; SAVE RO
4590 023554 017600      MOV 02(SP), RO      ;; GET ADDRESS OF ASCIZ STRING
4591 023560 122767 000001 155436 CMPB #APTENV, $ENV      ;; RUNNING IN APT MODE
4592 023566 001011      BNE 62$          ;; NO, GO CHECK FOR APT CONSOLE
4593 023570 132767 000100 155427 BITB #APTPOOL, $ENVM      ;; SPOOL MESSAGE TO APT
4594 023576 001405      BEQ 62$          ;; NO, GO CHECK FOR CONSOLE
4595 023600 010067 000004      MOV RO, 61$      ;; SETUP MESSAGE ADDRESS FOR APT
4596 023604 004767 000222      JSR PC, $ATY3      ;; SPOOL MESSAGE TO APT
4597 023610 000000      61$: .WORD 0      ;; MESSAGE ADDRESS
4598 023612 132767 000040 155405 62$: BITB #APTCSUP, $ENVM      ;; APT CONSOLE SUPPRESSED
4599 023620 001003      BNE 60$          ;; YES, SKIP TYPE OUT
4600 023622 112046      2$: MOVB (RO)+, -(SP)      ;; PUSH CHARACTER TO BE TYPED ONTO STACK
4601 023624 001005      BNE 4$          ;; BR IF IT ISN'T THE TERMINATOR
4602 023626 005726      TST (SP)+      ;; IF TERMINATOR POP IT OFF THE STACK
4603 023630 012600      60$: MOV (SP)+, RO      ;; RESTORE RO
4604 023632 062716 000002      3$: ADD #2, (SP)      ;; ADJUST RETURN PC
4605 023636 000002      RTI      ;; RETURN
4606 023640 122716 000011      4$: CMPB #HT, (SP)      ;; BRANCH IF <HT>
4607 023644 001431      BEQ 8$          ;; BRANCH IF NOT <CRLF>
4608 023646 122716 000200      CMPB #CRLF, (SP)
4609 023652 001007      BNE 5$          ;; POP <CR><LF> EQUIV
4610 023654 005726      TST (SP)+      ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4611 023656 004567 177630      JSR R5, $PRINT
4612 023662 001201      $CRLF
4613 023664 105067 000130      CLRB $CHARCNT      ;; CLEAR CHARACTER COUNT
4614 023670 000754      BR 2$          ;; GET NEXT CHARACTER
4615 023672 004767 000056      5$: JSR PC, $TYPEC      ;; GO TYPE THIS CHARACTER
4616 023676 126726 155254      6$: CMPB $FILLC, (SP)+      ;; IS IT TIME FOR FILLER CHARS.?
4617 023702 001347      BNE 2$          ;; IF NO GO GET NEXT CHAR.
4618 023704 016746 155244      MOV $NULL, -(SP)      ;; GET # OF FILLER CHARS. NEEDED
4619      ;; AND THE NULL CHAR.
4620 023710 105366 000001      7$: DECIB 1(SP)      ;; DOES A NULL NEED TO BE TYPED?
4621 023714 002770      BLT 6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
4622 023716 004767 000032      JSR PC, $TYPEC      ;; GO TYPE A NULL
4623 023722 105367 000072      DECIB $CHARCNT      ;; DO NOT COUNT AS A COUNT
4624 023726 000770      BR 7$          ;; LOOP
4625
4626      : HORIZONTAL TAB PROCESSOR
4627
4628 023730 112716 000040      8$: MOVB #' , (SP)      ;; REPLACE TAB WITH SPACE

```

```

4629 023734 004767 000014          9$: JSR PC,$TYPEC      ;;TYPE A SPACE
4630 023740 132767 000007 000052 BITB #7,$CHARCNT    ;;BRANCH IF NOT AT
4631 023746 001372          BNE 9$              ;;TAB STOP
4632 023750 005726          TST (SP)+           ;;POP SPACE OFF STACK
4633 023752 000723          BR 2$              ;;GET NEXT CHARACTER
4634 023754 105777 155170 $TYPEC: TSTB @STPS  ;;WAIT UNTIL PRINTER IS READY
4635 023760 100375          BPL $TYPEC
4636 023762 116677 000002 155162 MOVB 2(SP),@STPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
4637 023770 122766 000015 000002 CMPB #CR,2(SP)     ;;IS CHARACTER A CARRIAGE RETURN?
4638 023776 001003          BNE 1$              ;;BRANCH IF NO
4639 024000 105067 000014          CLRB $CHARCNT     ;;YES--CLEAR CHARACTER COUNT
4640 024004 000406          BR $TYPEX          ;;EXIT
4641 024006 122766 000012 000002 1$: CMPB #LF,2(SP)   ;;IS CHARACTER A LINE FEED?
4642 024014 001402          BEQ $TYPEX         ;;BRANCH IF YES
4643 024016 105227          INCB (PC)+        ;;COUNT THE CHARACTER
4644 024020 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE
4645 024022 000207          $TYPEX: RTS PC
4646
4647          .SBTTL APT COMMUNICATIONS ROUTINE
4648
4649          ;*****
4650 024024 112767 000001 000376 $ATY1: MOVB #1,$FFLG  ;;TO REPORT FATAL ERROR
4651 024032 112767 000001 000366 $ATY3: MOVB #1,$MFLG  ;;TO TYPE A MESSAGE
4652 024040 000403          BR $ATYC
4653 024042 112767 000001 000360 $ATY4: MOVB #1,$FFLG  ;;TO ONLY REPORT FATAL ERROR
4654 024050          $ATYC:
4655 024050 010046          MOV RO,-(SP)      ;;PUSH RO ON STACK
4656 024052 010146          MOV RI,-(SP)      ;;PUSH RI ON STACK
4657 024054 105767 000346          TSTB $MFLG        ;;SHOULD TYPE A MESSAGE?
4658 024060 001450          BEQ 5$            ;;IF NOT: BR
4659 024062 122767 000001 155134 CMPB #APTENV,$ENV  ;;OPERATING UNDER APT?
4660 024070 001031          BNE 3$            ;;IF NOT: BR
4661 024072 132767 000100 155125 BITB #APTPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
4662 024100 001425          BEQ 3$            ;;IF NOT: BR
4663 024102 017600 000004          MOV @4(SP),RO     ;;GET MESSAGE ADDR.
4664 024106 062766 000002 000004 ADD #2,4(SP)       ;;BUMP RETURN ADDR.
4665 024114 005767 155064          1$: TST $MSGTYPE   ;;SEE IF DONE W/ LAST XMISSION?
4666 024120 001375          BNE 1$            ;;IF NOT: WAIT
4667 024122 010067 155072          MOV RO,$MSGAD     ;;PUT ADDR IN MAILBOX
4668 024126 105720          2$: TSTB (RO)+     ;;FIND END OF MESSAGE
4669 024130 001376          BNE 2$
4670 024132 166700 155062          SUB $MSGAD,RO     ;;SUB START OF MESSAGE
4671 024136 006200          ASR RO            ;;GET MESSAGE LNTH IN WORDS
4672 024140 010067 155056          MOV RO,$MSGGLT   ;;PUT LENGTH IN MAILBOX
4673 024144 012767 000004 155032 MOV #4,$MSGTYPE   ;;TELL APT TO TAKE MSG.
4674 024152 000413          BR 5$
4675 024154 017667 000004 000016 3$: MOV @4(SP),4$     ;;PUT MSG ADDR IN JSR LINKAGE
4676 024162 062766 000002 000004 ADD #2,4(SP)      ;;BUMP RETURN ADDRESS
4677 024170 016746 153602          MOV 177776,-(SP) ;;PUSH 177776 ON STACK
4678 024174 004767 177340          JSR PC,$TYPE     ;;CALL TYPE MACRO
4679 024200 000000          4$: .WORD 0
4680 024202          5$:
4681 024202 105767 000221          TSTB $LFLG        ;;SHOULD LOG AN ERROR?
4682 024206 001422          BEQ 10$          ;;IF NOT: BR
4683 024210 017600 000004          MOV @4(SP),RO     ;;GET ERROR #
4684 024214 062766 000002 000004 ADD #2,4(SP)      ;;BUMP RETURN ADDR.

```



```

4685 024222 012701 001344          MOV    #SASTAT,R1      ;; POINT TO TABLE START
4686 024226 005711          TST    (R1)           ;; END OF TABLE?
4687 024230 100404          BMI    8$            ;; IF SO: BR
4688 024232 020021          CMP    RO,(R1)+      ;; PROPER ENTRY?
4689 024234 001406          BEQ    9$            ;; IF SO: BR
4690 024236 005721          TST    (R1)+         ;; MOVE PAST COUNTER WORD
4691 024240 000772          BR     6$            ;; KEEP LOOKING
4692 024242 026701 155244          CMP    $APTR,R1     ;; TABLE FULL?
4693 024246 001402          BEQ    10$           ;; IF SO: BR -- NO MORE ROOM
4694 024250 010021          MOV    RO,(R1)+     ;; SET UP NEW ENTRY
4695 024252 005211          INC    (R1)         ;; BUMP ERROR COUNT
4696 024254 105767 000150          TSTB  $FFLG         ;; SHOULD REPORT FATAL ERROR?
4697 024260 001416          BEQ    12$           ;; IF NOT: BR
4698 024262 005767 154736          TST    $ENV         ;; RUNNING UNDER APT?
4699 024266 001413          BEQ    12$           ;; IF NOT: BR
4700 024270 005767 154710          TST    $MSGTYPE     ;; FINISHED LAST MESSAGE?
4701 024274 001375          BNE    11$           ;; IF NOT: WAIT
4702 024276 017667 000004 154702          MOV    24(SP), $FATAL ;; GET ERROR #
4703 024304 062766 000002 000004          ADD    #2,4(SP)     ;; BUMP RETURN ADDR.
4704 024312 005267 154666          INC    $MSGTYPE     ;; TELL APT TO TAKE ERROR
4705 024316 105067 000106          CLRB  $FFLG         ;; CLEAR FATAL FLAG
4706 024322 105067 000101          CLRB  $LFLG         ;; CLEAR LOG FLAG
4707 024326 105067 000074          CLRB  $MFLG         ;; CLEAR MESSAGE FLAG
4708 024332 012601          MOV    (SP)+,R1     ;; POP STACK INTO R1
4709 024334 012600          MOV    (SP)+,RO     ;; POP STACK INTO RO
4710 024336 000207          RTS    PC           ;; RETURN
4711 024340          $ATY6:
4712 024340 010046          MOV    RO,-(SP)     ;; PUSH RO ON STACK
4713 024342 016700 155144          MOV    $APTR,RO
4714 024346 162700 001344          SUB    #SASTAT,RO
4715 024352 005767 154626          TST    $MSGTY
4716 024356 001375          BNE    1$           ;; GET SIZE OF STAT TABLE
4717 024360 010067 154636          MOV    RO,$MSGLG   ;; SEE IF DONE LAST COMMUNICATION
4718 024364 012767 001344 154626          MOV    #SASTAT,$MSGAD ;; IF NOT: WAIT
4719 024372 012767 000002 154604          MOV    #2,$MSGTY   ;; SET MESSAGE LENGTH
4720 024400 012600          MOV    (SP)+,RO    ;; SET MESSAGE ADDR.
4721 024402 000207          RTS    PC           ;; TELL APT TO TAKE STATS.
4722 024404          $ATY7:
4723 024404 010046          MOV    RO,-(SP)     ;; POP STACK INTO RO
4724 024406 012701 001344          MOV    #SASTAT,R1  ;; RETURN
4725 024412 005721          TST    (R1)+       ;; GET START OF TABLE
4726 024414 100402          BMI    2$           ;; END OF TABLE?
4727 024416 005021          CLR    (R1)+       ;; IF SO: BR
4728 024420 000774          BR     1$           ;; CLEAR ERROR COUNT
4729 024422          2$:
4730 024422 012600          MOV    (SP)+,RO    ;; KEEP CLEARING
4731 024424 000207          RTS    PC           ;; POP STACK INTO RO
4732 024426 000          $MFLG: .BYTE 0     ;; RETURN
4733 024427 000          $LFLG: .BYTE 0     ;; MESSG. FLAG
4734 024430 000          $FFLG: .BYTE 0     ;; LOG FLAG
4735 024432          .EVEN            ;; FATAL FLAG
4736 000200          APTSIZE=200
4737 000001          APTENV=001
4738 000100          APTSPool=100
4739 000040          APTCSUP=040
4740          ;;*****

```





```

4797 024616 012602      MOV      (SP)+,R2      ;; POP STACK INTO R2
4798 024620 012601      MOV      (SP)+,R1      ;; POP STACK INTO R1
4799 024622 012600      MOV      (SP)+,R0      ;; POP STACK INTO R0
4800 024624 004567      JSR      R5,SPRINT     ;; GO PRINT OUT THE FOLLOWING MESSAGE.
4801 024630 024654      .WORD   $DBLK         ;; ADDRESS OF MESSAGE TO BE TYPED
4802 024632 016666      MOV      2(SP),4(SP)   ;; ADJUST THE STACK
4803 024640 012616      MOV      (SP)+,(SP)
4804 024642 000002      RTI
4805 024644 023420      $DTBL: 10000.
4806 024646 001750      1000.
4807 024650 000144      100.
4808 024652 000012      10.
4809 024654 000004      $DBLK: .BLKW 4

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

4810
4811
4812
4813
4814
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835 024664 017646 000000
4836 024670 116667 000001 000213
4837 024676 112667 000211
4838 024702 062716 000002
4839 024706 000406
4840 024710 112767 000001 000173
4841 024716 112767 000006 000167
4842 024724 112767 000005 000156
4843 024732 010346
4844 024734 010446
4845 024736 010546
4846 024740 116704 000147
4847 024744 005404
4848 024746 062704 000006
4849 024752 110467 000134
4850 024756 116704 000127
4851 024762 016605 000012
4852 024766 005003

;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;CALL:
;*      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
;*      TYPOS    ;; CALL FOR TYPEOUT
;*      .BYTE   N              ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;*      .BYTE   M              ;; M=1 OR 0
;*                               ;; 1=TYPE LEADING ZEROS
;*                               ;; 0=SUPPRESS LEADING ZEROS
;$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;$TYPOS OR $TYPOC
;CALL:
;*      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
;*      STYPON  ;; CALL FOR TYPEOUT
;$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;CALL:
;*      MOV      NUM,-(SP)      ;; NUMBER TO BE TYPED
;*      STYPOC  ;; CALL FOR TYPEOUT
$TYPOS: MOV      2(SP),-(SP)      ;; PICKUP THE MODE
        MOV      1(SP),$OFILL    ;; LOAD ZERO FILL SWITCH
        MOV      (SP)+,$SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
        ADD      #2,(SP)        ;; ADJUST RETURN ADDRESS
        BR       $STYPON
$TYPOC: MOV      #1,$OFILL      ;; SET THE ZERO FILL SWITCH
        MOV      #6,$SOMODE+1   ;; SET FOR SIX(6) DIGITS
$STYPON: MOV      #5,$OCNT      ;; SET THE ITERATION COUNT
        MOV      R3,-(SP)      ;; SAVE R3
        MOV      R4,-(SP)      ;; SAVE R4
        MOV      R5,-(SP)      ;; SAVE R5
        MOVE     $SOMODE+1,R4   ;; GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4         ;; SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,$SOMODE    ;; 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

```



```

4853 024770 006105 1$: ROL R5 ::ROTATE MSB INTO "C"
4854 024772 000404 BR R3 ::GO DO MSB
4855 024774 006105 2$: ROL R5 ::FORM THIS DIGIT
4856 024776 006105 ROL R5
4857 025000 006105 ROL R5
4858 025002 010503 MOV R5,R3
4859 025004 006103 3$: ROL R3 ::GET LSB OF THIS DIGIT
4860 025006 105367 000100 DECB $OMODE ::TYPE THIS DIGIT?
4861 025012 100017 BPL R3 ::BR IF NO
4862 025014 042703 177770 BIC #177770,R3 ::GET RID OF JUNK
4863 025020 001002 BNE R4 ::TEST FOR 0
4864 025022 005704 TST R4 ::SUPPRESS THIS 0?
4865 025024 001403 BEQ R5 ::BR IF YES
4866 025026 005204 4$: INC R4 ::DON'T SUPPRESS ANYMORE 0'S
4867 025030 052703 000060 BIS #'0,R3 ::MAKE THIS DIGIT ASCII
4868 025034 052703 000040 5$: BIS #' ,R3 ::MAKE ASCII IF NOT ALREADY
4869 025040 110367 000042 MOV R3,$S ::SAVE FOR TYPING
4870 025044 004567 176442 JSR R5, $SPRINT ::GO PRINT OUT THE FOLLOWING MESSAGE.
4871 025050 025106 .WORD R5 ::ADDRESS OF MESSAGE TO BE TYPED
4872 025052 105367 000032 7$: DECB $OCNT ::COUNT BY 1
4873 025056 003346 BGT R5 ::BR IF MORE TO DO
4874 025060 002402 BLT R5 ::BR IF DONE
4875 025062 005204 INC R4 ::INSURE LAST DIGIT ISN'T A BLANK
4876 025064 000743 BR R5 ::GO DO THE LAST DIGIT
4877 025066 012605 6$: MOV (SP)+,R5 ::RESTORE R5
4878 025070 012604 MOV (SP)+,R4 ::RESTORE R4
4879 025072 012603 MOV (SP)+,R3 ::RESTORE R3
4880 025074 016666 000002 000004 MOV 2(SP),4(SP) ::SET THE STACK FOR RETURNING
4881 025102 012616 MOV (SP)+,(SP)
4882 025104 000002 RTI ::RETURN
4883 025106 000 .BYTE 0 ::STORAGE FOR ASCII DIGIT
4884 025107 000 .BYTE 0 ::TERMINATOR FOR TYPE ROUTINE
4885 025110 000 $OCNT: .BYTE 0 ::OCTAL DIGIT COUNTER
4886 025111 000 $OFILL: .BYTE 0 ::ZERO FILL SWITCH
4887 025112 000000 $OMODE: .WORD 0 ::NUMBER OF DIGITS TO TYPE
4888 .ERROR TRAP SERVICE ROUTINE
4889 025114 005727 ERRTRP: TST (PC)+ ::CHECK IF PREV TRAP TO 4 REPORTED
4890 025116 000000 1$: .WORD 0 ::CONTAINS ERROR REPORTED FLAG
4891 025120 001010 BNE R5 ::BRANCH IF NOT REPORTED
4892 025122 005267 177770 INC R5 ::SET DOUBLE TRAP FLAG.
4893 025126 011667 154034 MOV (SP), $TMP3 ::SAVE THE BAD PC FOR TYPING.
4894 025132 004767 174502 JSR PC, $ERROR ::*** ERROR *** (GO TYPE A MESSAGE)
4895 025136 000031 .WORD 31 ::ERROR TYPE CODE.
4896 025140 000401 BR R5 ::SKIP HALT
4897 025142 000000 2$: HALT ::ERROR! SECOND TRAP TO 4 OCCURRED
4898 BEFORE FIRST WAS PRINTED
4899 025144 005067 177746 3$: CLR R5
4900 025150 000002 RTI ::RETURN TO PROGRAM AND TRY TO RECOVER
4901
4902 .SBTTL PHYSICAL ADDRESS TYPE ROUTINE
4903 :* ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
4904 $TYPAD:
4905 025152 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
4906 025154 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
4907 025156 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
4908 025160 010346 MOV R3,-(SP) ::PUSH R3 ON STACK

```



```

4909 025162 016602 000012      MOV      12(SP), R2      ;GET BASE ADDRESS
4910 025166 005003              CLR      R3              ;WORKING & INDEX REGISTER
4911 025170 005767 153412      TST     MMAVA            ;CHECK FOR MEM MGMT AVAILABLE
4912 025174 001430              BEQ     1$               ;BRANCH IF NO MEM MGMT
4913 025176 032737 000001 177572 BIT     #1, 2#SRO        ;CHECK IF MEM MGMT ENABLED
4914 025204 001424              BEQ     1$               ;BRANCH IF MEM MGMT NOT ENABLED
4915 025206 010201              MOV     R2, R1          ;COPY VIRTUAL ADR
4916 025210 006101              ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
4917 025212 006101              ROL     R1
4918 025214 006101              ROL     R1
4919 025216 006101              ROL     R1
4920 025220 006101              ROL     R1
4921 025222 042701 177761      BIC     #177761, R1      ;CLR ALL EXCEPT BITS 1,2,3
4922 025226 062701 172340      ADD     #KIPARO, R1     ;SET TO APPROPRIATE PAR
4923 025232 011101              MOV     (R1), R1        ;GET CONTENTS OF PAR
4924 025234 012700 000006      MOV     #6, R0          ;SET UP COUNTER
4925 025240 006301 4$:      ASL     R1              ;SHIFT PAR
4926 025242 006103              ROL     R3              ;SAVE OVERFLOW BITS
4927 025244 077003              SOB     R0, 4$         ;COUNT SIX SHIFTS
4928 025246 042702 160000      BIC     #160000, R2     ;SAVE BANK BITS
4929 025252 060102              ADD     R1, R2          ;COMPUTE PHYSICAL ADDRESS
4930 025254 005503              ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
4931 025256 006302 1$:      ASL     R2              ;FIRST DIGIT TO R3
4932 025260 006103              ROL     R3
4933 025262 012700 000006      MOV     #6, R0          ;DIGIT COUNT
4934 025266 000404              BR     3$              ;PRINT FIRST DIGIT
4935 025270 006302 2$:      ASL     R2              ;
4936 025272 006103              ROL     R3              ;
4937 025274 005301              DEC     R1              ;
4938 025276 001374              BNE     2$             ;
4939 025300 012701 000003 3$:      MOV     #3, R1          ;DIGIT SHIFT COUNT
4940 025304 062703 000060      ADD     #60, R3         ;MAKE IT AN ASCII DIGIT
4941 025310 110367 000036      MOV     R3, R5         ;LOAD DIGIT INTO MESSAGE
4942 025314 004567 176172      JSR     R5, $SPRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4943 025320 025352              .WORD   8$             ;ADDRESS OF MESSAGE TO BE TYPED
4944 025322 005003              CLR     R3              ;CLEAR INDEX
4945 025324 005300              DEC     R0              ;DEC DIGIT COUNT
4946 025326 001360              BNE     2$             ;
4947 025330 012603              MOV     (SP)+, R3       ;: POP STACK INTO R3
4948 025332 012602              MOV     (SP)+, R2       ;: POP STACK INTO R2
4949 025334 012601              MOV     (SP)+, R1       ;: POP STACK INTO R1
4950 025336 012600              MOV     (SP)+, R0       ;: POP STACK INTO R0
4951 025340 012616              MOV     (SP)+, (SP)     ;ADJUST THE STACK TO CLEAR DATA
4952 025342 004567 176144      JSR     R5, $SPRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
4953 025346 027005              .WORD   8$             ;ADDRESS OF MESSAGE TO BE TYPED
4954 025350 000207              RTS     PC              ;RETURN
4955 025352 000              8$:      .BYTE   0              ;ONE DIGIT MESSAGE BUFFER
4956 025353 000              .BYTE   0              ;MESSAGE TERMINATOR

```

```

4957
4958      .SBTTL  STANDARD PROGRAM MESSAGES
4959      ;*****
4960      ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
4961      ;THE PROGRAM
4962      ;*****
4963 025354 005015 052113 030461 MMAMES: .ASCIZ <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
4964 025362 024040 042515 047515

```

4965	025370	054522	046440	047101	
4966	025376	043501	046505	047105	
4967	025404	024524	040440	040526	
4968	025412	046111	041101	042514	
4969	025420	000			
4970	025421	015	046412	046505	MEMMES: .ASCIZ <15><12>'MEMORY MAP:'
4971	025426	051117	020131	040515	
4972	025434	035120	000		
4973	025437	015	041012	052131	BYTMES: .ASCIZ <15><12>'BYTE MEMORY MAP:'
4974	025444	020105	042515	047515	
4975	025452	054522	046440	050101	
4976	025460	000072			
4977	025462	005015	040520	044522	MTMAP: .ASCIZ <15><12>'PARITY MEMORY MAP:'
4978	025470	054524	046440	046505	
4979	025476	051117	020131	040515	
4980	025504	035120	000		
4981	025507	015	043012	047522	FROM: .ASCIZ <15><12>'FROM '
4982	025514	020115	000		
4983	025517	040	047524	000040	TO: .ASCIZ ' TO '
4984	025524	005015	047111	052523	INSUFF: .ASCIZ <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
4985	025532	043106	041511	042511	
4986	025540	052116	046440	046505	
4987	025546	051117	027131	027056	
4988	025554	044506	051522	020124	
4989	025562	033061	020113	047516	
4990	025570	020124	046101	020114	
4991	025576	044124	051105	020505	
4992	025604	000			
4993	025605	015	047012	020117	MTR: .ASCIZ <15><12>'NO PARITY REGISTERS FOUND'
4994	025612	040520	044522	054524	
4995	025620	051040	043505	051511	
4996	025626	042524	051522	043040	
4997	025634	052517	042116	000	
4998	025641	015	051012	051505	PWRMSG: .ASCIZ <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
4999	025646	040524	052122	047111	
5000	025654	020107	043101	042524	
5001	025662	020122	020101	047520	
5002	025670	042527	020122	040506	
5003	025676	046111	051125	006505	
5004	025704	000012			
5005	025706	005015	047516	050040	NOPE: .ASCIZ <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
5006	025714	051101	052111	020131	
5007	025722	051105	047522	051522	
5008	025730	043040	052517	042116	
5009	025736	047440	020116	042515	
5010	025744	047515	054522	051440	
5011	025752	040503	006516	000012	
5012	025760	005015	051120	043517	PROREL: .ASCII <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
5013	025766	040522	020115	047516	
5014	025774	020127	042522	044523	
5015	026002	042504	020123	040502	
5016	026010	045503	040440	020124	
5017	026016	020060	047524	034040	
5018	026024	113			
5019	026025	015	044012	052111	.ASCIZ <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
5020	026032	041440	047117	044524	



5021	026040	052516	020105	047110	
5022	026046	020122	047516	047110	
5023	026054	046101	051040	047110	
5024	026062	044516	043516	020015	
5025	026070	000			
5026	026071	015	051012	043505	MX1: .ASCIZ <15><12>'REGISTER AT '
5027	026076	051511	042524	020122	
5028	026104	052101	000040		
5029	026110	041440	047117	051124	MX2: .ASCIZ 'CONTROLS '
5030	026116	046117	020123	000	
5031	026123	015	041412	051117	MX3: .ASCIZ <15><12>'CORE PARITY '
5032	026130	020105	040520	044522	
5033	026136	054524	000040		
5034	026142	005015	047515	020123	MX4: .ASCIZ <15><12>'MOS PARITY '
5035	026150	040520	044522	054524	
5036	026156	000040			
5037	026160	005015	051515	030461	MX5: .ASCIZ <15><12>'MS11-K CSR '
5038	026166	045455	041440	051123	
5039	026174	000040			
5040	026176	051515	030461	045455	MX6: .ASCIZ 'MS11-K MEMORY PRESENT!! TO COMPLETELY TEST RUN DZMML...'
5041	026204	046440	046505	051117	
5042	026212	020131	051120	051505	
5043	026220	047105	020524	020041	
5044	026226	047524	041440	046517	
5045	026234	046120	052105	046105	
5046	026242	020131	042524	052123	
5047	026250	051040	047125	042040	
5048	026256	046532	046115	027056	
5049	026264	000056			
5050	026266	005015	047516	046440	NOMEM: .ASCIZ <15><12>'NO MEMORY FOUND.'
5051	026274	046505	051117	020131	
5052	026302	047506	047125	027104	
5053	026310	000			
5054	026311	015	005012	044412	FADMES: .ASCII <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
5055	026316	050116	052125	040440	
5056	026324	046114	050040	051101	
5057	026332	046501	052105	051105	
5058	026340	020123	047111	047440	
5059	026346	052103	046101	056	
5060	026353	015	043012	051111	.ASCIZ <15><12>'FIRST ADDRESS: '
5061	026360	052123	040440	042104	
5062	026366	042522	051523	020072	
5063	026374	000040			
5064	026376	005015	040514	052123	LADMES: .ASCIZ <15><12>'LAST ADDRESS: '
5065	026404	040440	042104	042522	
5066	026412	051523	020072	020040	
5067	026420	000			
5068	026421	015	037412	042101	BADADR: .ASCIZ <15><12>'?ADDRESS IN UNMAPPED BANK?'
5069	026426	051104	051505	020123	
5070	026434	047111	052440	046516	
5071	026442	050101	042520	020104	
5072	026450	040502	045516	000077	
5073	026456	005015	042523	042514	CONST: .ASCIZ <15><12>'SELECT CONSTANT: '
5074	026464	052103	041440	047117	
5075	026472	052123	047101	035124	
5076	026500	000			



5077	026501	015	052412	042516	UNEXPT: .ASCIZ <15><12>'UNEXPECTED MEMORY PARITY ERROR'
5078	026506	050130	041505	042524	
5079	026514	020104	042515	047515	
5080	026522	054522	050040	051101	
5081	026530	052111	020131	051105	
5082	026536	047522	000122		
5083	026542	005015	051120	043517	PRELOC: .ASCIZ <15><12>'PROGRAM RELOCATED TO '
5084	026550	040522	020115	042522	
5085	026556	047514	040503	042524	
5086	026564	020104	047524	000040	
5087	026572	005015	047515	042522	MTOE: .ASCIZ <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
5088	026600	052040	040510	020116	
5089	026606	047117	020105	040520	
5090	026614	044522	054524	042440	
5091	026622	051122	051117	043040	
5092	026630	052517	042116	000056	
5093	026636	005015	041523	047101	SCANM: .ASCIZ <15><12>'SCANNING MEMORY FOR BAD PARITY.'
5094	026644	044516	043516	046440	
5095	026652	046505	051117	020131	
5096	026660	047506	020122	040502	
5097	026666	020104	040520	044522	
5098	026674	054524	000056		
5099	026700	005015	040520	044522	PEWNC: .ASCIZ <15><12>'PARITY ERROR WILL NOT CLEAR.'
5100	026706	054524	042440	051122	
5101	026714	051117	053440	046111	
5102	026722	020114	047516	020124	
5103	026730	046103	040505	027122	
5104	026736	000			
5105	026737	015	047012	020117	NOMTST: .ASCIZ <15><12>'NO MEMORY TESTED.'
5106	026744	042515	047515	054522	
5107	026752	052040	051505	042524	
5108	026760	027104	000		
5109	026763	015	051412	044513	SKPMES: .ASCIZ <15><12>'SKIPPING TEST #'
5110	026770	050120	047111	020107	
5111	026776	042524	052123	021440	
5112	027004	000			
5113	027005	377	000377		FILL2: .ASCIZ <377><377>
5114					
5115					.SBTTL ERROR REPORTING MESSAGES AND TABLES.
5116					::*****
5117					::* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
5118					::*****
5119	027010	040520	044522	054524	DM1: .ASCIZ 'PARITY REGISTER DATA ERROR.'
5120	027016	051040	043505	051511	
5121	027024	042524	020122	040504	
5122	027032	040524	042440	051122	
5123	027040	051117	000056		
5124	027044	042101	051104	051505	DM2: .ASCIZ 'ADDRESS TEST ERROR(TST1-5).'
5125	027052	020123	042524	052123	
5126	027060	042440	051122	051117	
5127	027066	052050	052123	026461	
5128	027074	024465	000056		
5129	027100	047503	051516	040524	DM4: .ASCIZ 'CONSTANT DATA ERROR(TST6-10).'
5130	027106	052116	042040	052101	
5131	027114	020101	051105	047522	
5132	027122	024122	051524	033124	

S133	027130	030455	024460	000056		
S134	027136	047523	040524	044524	DM5:	.ASCIZ 'ROTATING BIT ERROR(TST11-12).'
S135	027144	043516	041040	052111		
S136	027152	042440	051122	051117		
S137	027160	052050	052123	030461		
S138	027166	030455	024461	000056		
S139	027174	047515	020123	042522	DM6:	.ASCIZ 'MOS REFRESH TEST ERROR (TST 30-31).'
S140	027202	051106	051505	020110		
S141	027210	042524	052123	042440		
S142	027216	051122	051117	024040		
S143	027224	051524	020124	030063		
S144	027232	031455	024461	000056		
S145	027240	020063	047530	020122	DM7:	.ASCIZ '3 XOR 9 PATTERN ERROR(TST13-16).'
S146	027246	020071	040520	052124		
S147	027254	051105	020116	051105		
S148	027262	047522	024122	051524		
S149	027270	030524	026463	033061		
S150	027276	027051	000			
S151	027301	115	051101	044103	DM10:	.ASCIZ "MARCHING 1'S AND 0'S ERROR(TST 27)."
S152	027306	047111	020107	023461		
S153	027314	020123	047101	020104		
S154	027322	023460	020123	051105		
S155	027330	047522	024122	051524		
S156	027336	020124	033462	027051		
S157	027344	000				
S158	027345	120	051101	052111	DM11:	.ASCIZ 'PARITY MEMORY ADDRESS ERROR(TST17).'
S159	027352	020131	042515	047515		
S160	027360	054522	040440	042104		
S161	027366	042522	051523	042440		
S162	027374	051122	051117	052050		
S163	027402	052123	033461	027051		
S164	027410	000				
S165	027411	104	052101	050111	DM12:	.ASCIZ "DATIP WITH WRONG PARITY DIDN'T TRAP(TST17)."
S166	027416	053440	052111	020110		
S167	027424	051127	047117	020107		
S168	027432	040520	044522	054524		
S169	027440	042040	042111	023516		
S170	027446	020124	051124	050101		
S171	027454	052050	052123	033461		
S172	027462	027051	000			
S173	027465	127	047522	043516	DM13:	.ASCIZ 'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
S174	027472	050040	051101	052111		
S175	027500	020131	051124	050101		
S176	027506	042520	026104	041040		
S177	027514	052125	047040	020117		
S178	027522	042522	044507	052123		
S179	027530	051105	051440	047510		
S180	027536	051527	042440	051122		
S181	027544	051117	043040	040514		
S182	027552	027107	000			
S183	027555	120	051101	052111	DM14:	.ASCIZ 'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
S184	027562	020131	042522	044507		
S185	027570	052123	051105	047040		
S186	027576	052117	046440	050101		
S187	027604	042520	020104	051501		
S188	027612	041440	047117	051124		



## E15

CZQMCFD 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 104  
 CZQMCF.P11 14-FEB-78 08:19 ERROR REPORTING MESSAGES AND TABLES.

SEQ 0186

5189	027620	046117	044514	043516		
5190	027626	052040	044510	020123		
5191	027634	042101	051104	051505		
5192	027642	024123	051524	030524		
5193	027650	024467	000056			
5194	027654	047515	042522	052040	DM16:	.ASCIZ 'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
5195	027662	040510	020116	047117		
5196	027670	020105	042522	044507		
5197	027676	052123	051105	044440		
5198	027704	042116	041511	052101		
5199	027712	042105	050040	051101		
5200	027720	052111	020131	051105		
5201	027726	047522	027122	000		
5202	027733	104	052101	020101	DM17:	.ASCIZ "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17)."
5203	027740	044123	052517	042114		
5204	027746	023516	020124	040510		
5205	027754	042526	041440	040510		
5206	027762	043516	042105	053440		
5207	027770	042510	020116	040520		
5208	027776	044522	054524	042440		
5209	030004	051122	051117	052040		
5210	030012	040522	050120	042105		
5211	030020	052050	052123	033461		
5212	030026	027051	000			
5213	030031	122	047101	047504	DM20:	.ASCIZ 'RANDOM DATA ERROR(TST20).'
5214	030036	020115	040504	040524		
5215	030044	042440	051122	051117		
5216	030052	052050	052123	030062		
5217	030060	027051	000			
5218	030063	111	051516	051124	DM21:	.ASCIZ 'INSTRUCTION EXECUTION ERROR(TST21-26).'
5219	030070	041525	044524	047117		
5220	030076	042440	042530	052503		
5221	030104	044524	047117	042440		
5222	030112	051122	051117	052050		
5223	030120	052123	030462	031055		
5224	030126	024466	000056			
5225	030132	051120	043517	040522	DM23:	.ASCIZ 'PROGRAM CODE CHANGED WHEN RELOCATED.'
5226	030140	020115	047503	042504		
5227	030146	041440	040510	043516		
5228	030154	042105	053440	042510		
5229	030162	020116	042522	047514		
5230	030170	040503	042524	027104		
5231	030176	000				
5232	030177	124	040522	050120	DM24:	.ASCIZ 'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
5233	030204	042105	020054	052502		
5234	030212	020124	047516	051040		
5235	030220	043505	051511	042524		
5236	030226	020122	040510	020104		
5237	030234	051105	047522	020122		
5238	030242	044502	020124	042523		
5239	030250	027124	000			
5240	030253	124	040522	050120	DM25:	.ASCIZ 'TRAPPED TO 114.'
5241	030260	042105	052040	020117		
5242	030266	030461	027064	000		
5243	030273	106	044501	042514	DM26:	.ASCIZ 'FAILED TO TRAP.'
5244	030300	020104	047524	052040		



F15

5245	030306	040522	027120	000
5246	030313	050	041501	044524
5247	030320	047117	042440	040516
5248	030326	046102	020105	040527
5249	030334	047123	052047	051440
5250	030342	052105	027051	000
5251	030347	015	052012	040522
5252	030354	050120	042105	052040
5253	030362	020117	020064	000
5254				
5255				
5256				
5257				
5258				
5259	030367	120	004503	042522
5260	030374	004507	027523	004502
5261	030402	040527	000123	
5262	030406	027526	041520	050011
5263	030414	050057	004503	040515
5264	030422	051411	041057	053411
5265	030430	051501	000	
5266	030433	126	050057	004503
5267	030440	027520	041520	046411
5268	030446	004501	027523	000102
5269	030454	027526	041520	050011
5270	030462	050057	004503	042522
5271	030470	004507	040515	000
5272	030475	126	050057	004503
5273	030502	027520	041520	046411
5274	030510	052501	004524	042522
5275	030516	004507	027523	004502
5276	030524	040527	000123	
5277	030530	027526	041520	050011
5278	030536	050057	004503	052511
5279	030544	004524	040515	051411
5280	030552	041057	053411	051501
5281	030560	000		
5282	030561	126	050057	004503
5283	030566	027520	041520	051411
5284	030574	041522	046440	004501
5285	030602	051504	020124	040515
5286	030610	051411	041057	053411
5287	030616	051501	000	
5288	030621	126	050057	004503
5289	030626	027520	041520	052011
5290	030634	050122	050057	000103
5291	030642	027526	041520	050011
5292	030650	050057	004503	051124
5293	030656	027520	041520	051011
5294	030664	043505	053411	051501
5295	030672	000		
5296	030673	126	050057	004503
5297	030700	027520	041520	051011
5298	030706	043505	053411	051501
5299	030714	000		
5300	030715	122	043505	053411

```

:*****
:DATA COLUMN HEADINGS
:*****

```

DM1:	.ASCIZ	'PC	REG	S/B	WAS'		
DM2:	.ASCIZ	'V/PC	P/PC	MA	S/B	WAS'	
DM12:	.ASCIZ	'V/PC	P/PC	MA	S/B'		
DM14:	.ASCIZ	'V/PC	P/PC	REG	MA'		
DM15:	.ASCIZ	'V/PC	P/PC	MAUT	REG	S/B	WAS'
DM21:	.ASCIZ	'V/PC	P/PC	IUT	MA	S/B	WAS'
DM23:	.ASCIZ	'V/PC	P/PC	SRC MA	DST MA	S/B	WAS'
DM24:	.ASCIZ	'V/PC	P/PC	TRP/PC'			
DM25:	.ASCIZ	'V/PC	P/PC	TRP/PC	REG	WAS'	
DM26:	.ASCIZ	'V/PC	P/PC	REG	WAS'		
DM30:	.ASCIZ	'REG	WAS	MA	WAS'		





ABASE = 000000	384	425																		
ACDW1 = 000000	384	426																		
ACDW2 = 000000	384	427																		
ACPUOP = 000000	384	428																		
ADDW0 = 000000	384	429																		
ADDW1 = 000000	384	430																		
ADDW10 = 000000	384	439																		
ADDW11 = 000000	384	440																		
ADDW12 = 000000	384	441																		
ADDW13 = 000000	384	442																		
ADDW14 = 000000	384	443																		
ADDW15 = 000000	384	444																		
ADDW2 = 000000	384	431																		
ADDW3 = 000000	384	432																		
ADDW4 = 000000	384	433																		
ADDW5 = 000000	384	434																		
ADDW6 = 000000	384	435																		
ADDW7 = 000000	384	436																		
ADDW8 = 000000	384	437																		
ADDW9 = 000000	384	438																		
ADEVCT = 000000	384	430																		
ADEVM = 000000	384	426																		
AE = 000001	179	233	2519	3750																
AENV = 000000	384	435																		
AENVM = 000000	384	436																		
AFATAL = 000000	384	437																		
AMADR1 = 000000	384	412																		
AMADR2 = 000000	384	416																		
AMADR3 = 000000	384	419																		
AMADR4 = 000000	384	422																		
AMAMS1 = 000000	384	406																		
AMAMS2 = 000000	384	414																		
AMAMS3 = 000000	384	417																		
AMAMS4 = 000000	384	420																		
AMSGAD = 000000	384	432																		
AMSGLG = 000000	384	433																		
AMSGTY = 000000	384	436																		
AMTYP1 = 000000	384	407																		
AMTYP2 = 000000	384	415																		
AMTYP3 = 000000	384	418																		
AMTYP4 = 000000	384	421																		
APASS = 000000	384	389																		
APRIOR = 000000	384																			
APTC SU = 000040	4598	4739																		
APTENV = 000001	4187	4591	4659	4737																
APTSIZ = 000200	4876	4736																		
APTSP0 = 000100	4593	4661	4738																	
ASWREG = 000000	384	397																		
ATESTN = 000000	384	388																		
AUNIT = 000000	384	391																		
AUSWR = 000000	384	398																		
AVECT1 = 000000	384	423																		
AVECT2 = 000000	384	424																		
BADADR = 026421	1446	5068																		
BANKNO = 016130	1632	1641	1667	1677	3397															
BITPT = 001544	522	1134	1135	1171	1172	1173	1174	1182	1183	1192	1194	1197	1200							















# M15

PMEMAP	001540	519#	1148*	1149*	1173*	1174*	1255*	1256*	1259*	1260*	2409	2411		
PRELOC	026542	3506	5083#											
PRGMAP	000602	253	254	274	289#	911*	912*	3043	3045	3049	3517	3549	3557*	3575
PROREL	025760	3601*	3610	3650*	3651*	4084	4085							
PRO	= 000000	5012#												
PR1	= 000040	60#												
PR2	= 000100	61#												
PR3	= 000140	62#												
PR4	= 000200	63#												
PR5	= 000240	64#												
PR6	= 000300	65#												
PR7	= 000340	66#												
PS	= 177776	67#												
PSCAN	017744	40#	41											
PSW	= 177776	3725	3773	3778	3789#									
		41#	899	1239	1378	1405	1462	3091	3962	3970	4004	4127	4148	4167
		4199	4233	4266	4273	4286	4332	4431	4517	4563				
PWRMSG	025641	326	4998#											
PWRVEC	= 000024	132#	295*	296*	305*	311*	323*	324*	856*	857*	3589*	3642*		
RADTAB	001622	559#	3595	3634										
RANTST	012470	2553#												
RELOC	016336	3481#	3551	3580	3617	3627								
RELOC	000600	269	288#	913*	1487	3233	3583*	3587*	3596	3604	3625	3649*	3740	4066
		4237	4258	4260	4560	4764								
RELTOP	016460	3056	3517#											
RELO	017062	276	3059	3610#										
RESCHK	005252	1289	1328#											
RESLDR	017270	282	3062	3659#										
RESRVD	001516	505#	561	1292*	1296	1301	1306	1314	2486*	2487	2488	2517		
RESTAR	000300	201	226#	328	910									
RESTOR	000304	202	228#											
REST1	000306	227	229#											
REST2	000324	231	233#											
RESVEC	= 000010	127#												
ROTATE	016210	1781	1804	3426#										
RW	= 000006	176#	3125	3126	3127	3128	3132							
SAVLDR	017350	921	3108	3677#										
SAVTST	001534	515#	1008*	1009*	1381*	1382*	1423*	1424*	3043	3045	4082	4083		
SCANM	026636	3798	5093#											
SELECT	002646	199	846#											
SELFLG	001556	531#	844*	846*	1361									
SETAE	017602	2416	3741	3743	3746#									
SETCON	016170	1778	1801	2407	3416#									
SKPMES	026763	4121	5109#											
SPRNT	020312	1298	1316	1349	2424	3891#								
SPRNTA	020400	3901	3905	3910#										
SPRNTB	020404	3893	3911#											
SPRNTP	020336	2471	2492	2505	2522	3899#								
SPRNTQ	020324	3722	3769	3775	3827	3895#								
SPRNTD	020342	1309	1534	1570	1682	2447	2463	2531	3897	3900#				
SPRNT1	020350	1645	3903#											
SPRNT2	020366	1517	1609	1727	1763	1786	1809	1843	1850	1857	1864	1887	1896	1905
		1946	1953	1960	1967	1990	1999	2008	2049	2056	2063	2070	2097	2106
		2115	2122	2131	2140	2147	2156	2165	2172	2181	2190	2234	2241	2248
		2255	2282	2291	2300	2307	2316	2325	2332	2341	2350	2357	2366	2375
		2570	2916	2925	2947	2996	3029	3908#						



# N15

SPRNT3	020362	2623	2672	2722	2771	2820	2870	3907*	3664*	3673*	4913							
SR0	= 177572	148#	235	251*	928*	1028*	1491	3141*										
SR1	= 177574	149#																
SR2	= 177576	150#																
SR3	= 172516	151#																
STACK	= 001100	31#	229	270	272	560	854	936	952	1023								
START	002640	198	844#															
STARTA	002654	232	284	845	847#													
START1	006104	281	1472	1482#	1486	3057	3110											
STKLMT	= 177774	42#																
SWR	001140	303	316*	360#	852	863*	865	871*	878*	895	925	1025	1034	1737				
		1739	2403	3047	3591	3593*	3644	3646*	3738	3742	3761	4008	4022	4024				
		4032	4039	4068	4173	4181	4194	4201	4313	4358*								
SWREG	000176	196#	871	895	4313	4329												
SW0	= 000001	95#																
SW00	= 000001	85#	95															
SW01	= 000002	84#	94															
SW02	= 000004	83#	93															
SW03	= 000010	82#	92															
SW04	= 000020	81#	91															
SW05	= 000040	80#	90	4068														
SW06	= 000100	79#	89	1094	2403													
SW07	= 000200	78#	88	3047														
SW08	= 000400	77#	87	1737														
SW09	= 001000	76#	86															
SW1	= 000002	94#																
SW10	= 002000	75#																
SW11	= 004000	74#																
SW12	= 010000	73#	925	1025														
SW13	= 020000	72#																
SW14	= 040000	71#																
SW15	= 100000	70#																
SW2	= 000004	93#																
SW3	= 000010	92#																
SW4	= 000020	91#																
SW5	= 000040	90#	3742															
SW6	= 000100	89#	3738	3761														
SW7	= 000200	88#																
SW8	= 000400	87#																
SW9	= 001000	86#																
TBITVE	= 000014	128#																
TEMP	001614	550#	2907*	2956														
TIMOUT	003622	203	1023#	1085														
TKVEC	= 000060	135#																
TMAP	004562	1184	1208#															
TMPFAD	001564	537#	3219	3344	4065*	4070	4072*	4094	4096*	4097*	4098	4113	4135					
TMPLAD	001576	543#	3170*	3189*	3271*	3290*	3293	4081*	4102	4104*	4105	4113	4140					
TMPPT	001550	525#	3172*	3173*	3177*	3178*	3180	3182	3184	3186	3198*	3203*	3205	3208				
		3213*	3214*	3250	3252	3272*	3273*	3278*	3279*	3281	3283	3285	3287	3310*				
		3313*	3315	3324*	3325*													
		1056	1078	3944	4983#													
TO	025517	136#																
TPVEC	= 000064	134#																
TRAPVE	= 000034	129#																
TRTVEC	= 000014	512#																
TSTMAP	001530	3353	3156	3158	3180	3182	3192	3205	3262	3264	3281	3283	3305	3315				
			3355	3362	4082*	4083*	4084*	4085*	4088	4090*	4091*	4100*	4101*	4107*				







SETEND	001330	445	468																		
SFATAL	001206	387	4702*																		
SFFLG	024430	4650*	4653*	4696	4705*	4734															
SFILLC	001156	368	4616	4647																	
SFILLS	001155	367	4617																		
SGADR	001120	351	573	575	577	579	581	584	587	596	3496*	3891*	3900*	3903*							
		3904*	3908*	3909*																	
SGDAT	001124	353	573	575	577	581	584	587	3494*	3892*	3910*										
SGET42	014212	3095																			
SGTSWR	022436	900	4326																		
SHD	000000	12																			
SHIBTS	001330	463																			
SHIOCT	023510	1385	1398	1409	1412*	1414	1425	4540*	4552												
SICNT	001104	344	4043*	4044	4046*	4145															
SILLUP	000756	295	311	330																	
SINTAG	001135	358	4363	4495																	
SITEMB	001114	348	4180*	4189	4213	4226															
SLF	001202	379	4213	4479	4489	4553	4647														
SLFLG	024427	4681	4706*	4733*																	
SLPADR	001106	345	4486*	4487*	1744*	4034*	4050*	4055	4133*	4145											
SLPERR	001110	346	4034	4051*	4145	4203															
SMADR1	001236	412																			
SMADR2	001242	416																			
SMADR3	001246	419																			
SMADR4	001252	422																			
SMAIL	001204	385	464	468	875	893	4049	4187	4591												
SMAMS1	001234	406																			
SMAMS2	001240	414																			
SMAMS3	001244	417																			
SMAMS4	001250	420																			
SMBADR	001332	464																			
SMFLG	024426	4651*	4657	4707*	4732																
SMNEW	023327	4335	4493																		
SMSGAD	001220	392	4667*	4670	4718*																
SMSGLG	001222	393	4672*	4717*																	
SMSGTY	001204	386	4665	4673*	4700	4704*	4715	4719*													
SMSWR	023316	4328	4491																		
SMTYP1	001235	407																			
SMTYP2	001241	415																			
SMTYP3	001245	418																			
SMTYP4	001251	421																			
SMXCNT	021626	4047	4145																		
SNULL	001154	366	4618	4647																	
SNWTST=	000001	1495	1497	1541	1543	1578	1580	1618	1620	1653	1655	1694	1696	1697							
		1714	1716	1747	1749	1771	1794	1816	1918	2021	2206	2390	2392	2393							
		2582	2584	2631	2633	2680	2682	2730	2732	2779	2781	2828	2830	2831							
		2881	2961	2963	3004																
SOCNT	025110	4842*	4872*	4885																	
SOMODE	025112	4837*	4841*	4846	4849*	4860*	4887														
SOVER	021112	4009	4027	4035	4045	4054															
SPASS	001212	389	875*	3078*	3079*	3088	3111	4041	4060	4086	4146										
SPASTM	001336	466																			
SPRINT	023512	325	888	931	961	1006	1014	1032	1045	1055	1077	1115	1209	1217							
		1223	1229	1233	1241	1250	1270	1373	1400	1445	1457	1064	3086	3093							
		3505	3700	3718	3797	3835	3860	3922	3943	3950	4076	4120	4175	4181							
		4222	4245	4247	4251	4253	4292	4298	4324	4327	4334	4348	4361	4382							



		4439	4446	4452	4457	4463	4465	4469	4473	4478	4547	4549	4559#	4611
		4800	4870	4942	4952									
\$PWAD	000752	328#												
\$PWADN	000610	295#	323	856										
\$PWARMG	000746	326#												
\$PWUP	000662	305#	311#											
\$QJES	001200	377#	4213	4383	4470	4469	4550	4553	4647					
\$RDCHR	022670	396#	432#											
\$RDLIN	023010	424#	451#											
\$RDOCT	023340	1379	1406	1463	4509#									
\$RDSZ =	000010	417#												
\$SAVR6	000762	304*	312	313*	314*	332#								
\$SCOPE	020630	1505	1551	1588	1628	1663	1705	1719	1752	1775	1798	1820	1922	2025
		2210	2399	2551	2608	2657	2706	2756	2805	2854	2904	2976	3008	3039
		4001#												
\$SETUP =	000130	188#	855	856	858	859	866	890	3077	4002	165	4197	4208	4308
		4495#												
\$STUP =	177777	198#												
\$SVLAD	021056	4017	4048#											
\$SVPC =	000010	213#	218#											
\$SWR =	167400	1#	12	17	18	19	20	21	22	23	24	29	374	395
		376	859	1510	1553	1590	1630	1665	1707	1721	1784	1777	1900	1987
		1927	2030	2215	2401	2553	2613	2662	2711	2761	2810	2993	3006	3077
		3010	3072	3077	3096	3110	3111	3993	3994	3995	4036	4037	4006	4030
		4022	4023	4028	4029	4030	4037	4038	4039	4051	4054	4145	4156	4157
		4158	4159	4160	4173	4181	4194	4201	4213					
\$SWREG	001226	397#	878											
\$SWRMK =	000340	1#	24	25	3997	3998	4024	4025						
\$TESTN	001210	388#	4049#											
\$TIMES	001170	279#	374#	3041*	3077*	4037*	4044	4047*	4145					
\$TKB	001146	363#	4306	4317	4341	4400	4406							
\$TKS	001144	362#	4306	4315	4338	4365*	4398	4404						
\$TMPD	001160	370#	579	581	584	592	594	596	3377*	3387*	3389*	3895*	3899*	3907*
\$TMP1	001162	371#	592	594	596	3896*								
\$TMP2	001164	372#	970*	971*	974*	985*	1363	1411						
\$TMP3	001166	373#	598	960*	976*	982*	984*	986*	1412	4893*				
\$TN =	000032	1#	12	1495	1510#	1541	1553#	1578	1590#	1618	1630#	1653	1665#	1694
		1707#	1714	1716	1721#	1734	1747	1754#	1757	1771	1777#	1794	1800#	1816
		1819	1825#	1918	1921	1927#	2021	2024	2030#	2206	2209#	2215#	2230	2401#
		2547	2552#	2582	2607	2613#	2631	2656	2662#	2680	2705	2711#	2730	2755#
		2761#	2779	2804	2810#	2828	2853	2859#	2879	2906#	2961	2977#	3004	3010#
\$TPB	001152	365#	4636*	4647										
\$TPFLG	001157	369#	4585	4647										
\$TPS	001150	364#	4634	4647										
\$TSTM	001334	465#												
\$TSTNM	001102	280*	342#	1743*	3042*	3992	4026	4048*	4049	4054	4062	4124	4146	4172
		4213												
\$TTYIN	023274	4426	4427	4443	4466	4484	4488#							
\$TYPAD	025152	1049	1060	1082	3509	4279	4904#							
\$TYPDS	024432	3092	4274	4753#										
\$TYPE	023540	4564	4585#	4678										
\$TYPEC	023754	4367	4615	4622	4629	4634#	4635							
\$TYPEX	024022	4640	4642	4645#										
\$TYPOC	024710	1240	4234	4267	4333	4840#								
\$TYPON	024724	4839	4842#											
\$TYPOS	024664	3963	3971	4128	4287	4835#								





ABORT	118	4260	3164	3197	3235	3407	3500	3519	3577	3612	3661				
CKSWR	118	4000	4146	4165	4197										
CKWD	118	1300	1515	1532	1607	1643	1680	1725	1761	1784	1807	1841	1848	1855	1868
	118	1300	1515	1532	1607	1643	1680	1725	1761	1784	1807	1841	1848	1855	1868
	118	1300	1515	1532	1607	1643	1680	1725	1761	1784	1807	1841	1848	1855	1868
	118	1300	1515	1532	1607	1643	1680	1725	1761	1784	1807	1841	1848	1855	1868
CKWD2	118	1567	1605	1720	1760	1839	1847	1854	1861	1883	1893	1902	1942	1950	1957
	118	1567	1605	1720	1760	1839	1847	1854	1861	1883	1893	1902	1942	1950	1957
	118	1567	1605	1720	1760	1839	1847	1854	1861	1883	1893	1902	1942	1950	1957
COMMEN	118	138	838	1476	3363	3772	5678	9932	3025	2278	2288	2297	2324	2313	2322
ENDCOM	118	12	138	842	1480										
ERROR	118														
GETPR	118	138													
GETSWR	118	138	890												
GYSWR	118	897													
LDPDR	118	3124	3126	3127	3128	3132									
MORETA	333	447													
MULT	118	138													
NEWTST	118	138	1495	1541	1578	1618	1653	1694	1714	1747	1771	1794	1816	1918	2021
	2206	2390	2547	2582	2631	2680	2730	2779	2828	2879	2961	3004			
POP	118	138	316	317	2515	2528	3391	3408	3510	3602	3640	3727	3753	3781	3844
	3854	3862	3882	3974	3981	4541	4708	4709	4720	4729	4795	4947			
PRINT	118	931	961	1006	1014	1031	1045	1055	1077	1115	1209	1217	1223	1229	1232
	124	1373	1399	1445	1456	3063	3505	3700	3718	3797	3835	3860	3922	4076	4119
	494	495													
PUSH	118	138	297	303	2474	2497	3380	3398	3481	3520	3537	3633	3703	3748	3763
	3789	3809	3816	3875	3926	3952	4511	4654	4656	4677	4711	4722	4753	4904	
RDCHR	118	429													
RDDEC	118	429													
RDLIN	118	4514													
RDOCT	118	1376	1403	1460											
REPORT	118	138													
RESREG	118														
SAVREG	118														
SCOPE	333														
SCOPEX	333	4056													
SCOPEX	333	4006													
SETPRI	118	138													
SETUP	118	138	847												
SIMTRP	118	4897	1237	1376	1403	1460	3089	3960	3968	4002	4125	4146	4165	4197	4231
	4264	4271	4284	4330	4429	4515	4561								
SKIP	118	138	1757												
SLASH	118	138	621	631											
SPACE	118	138													
STOP	118	138	211	220	225	293	309	335	380	383	449	451	458	471	501
	503	554	558	570	572	602	604	942	951	1019	1022	1088	1091	1122	1131
	114	1146	1203	1206	1277	1281	1323	1326	1366	1368	1495	1503	1541	1549	1578
	158	1618	1626	1653	1661	1690	1693	1694	1703	1714	1747	1747	1750	1771	1778
	179	1796	1816	1818	1833	1835	1876	1878	1918	1920	1936	1938	1979	1981	2021
	200	2038	2040	2086	2088	2201	2208	2223	2225	2271	2273	2390	2397	2547	2549
	207	2606	2631	2655	2680	2704	2730	2754	2779	2803	2828	2852	2879	2902	2961
	209	3004	3006	3066	3117	3123	3145	3147	3244	3249	3334	3339	3372	3375	3494
	309	3413	3415	3423	3425	3446	3448	3478	3480	3514	3516	3607	3609	3654	3658









J16

CZQMCFO 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 20-FEB-78 07:56 PAGE 124  
CZQMCF.P11 14-FEB-78 08:19 CROSS REFERENCE TABLE -- MACRO NAME3

SEQ 0204

ERRORS DETECTED: 0

CZQMCF.BIN,CZQMCF.LST/CRF/SOL/NL:TOC=CZQMCF.SML,CZQMCF.P11  
RUN-TIME: 21 28 1 SECONDS  
RUN-TIME RATIO: 130/S1=2.5  
CORE USED: 39K (77 PAGES)



K16