IDENTIFICATION

PRODUCT CODE:	AC-9045F-MC
PRODUCT NAME:	CZQMCF0 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:

DIGTAL PDP UNIBUS MASSBUS

DEC DECUS DECTAPE

REVISION HISTORY

Revision A:	May 1975
Revision B:	October 1975
Revision C:	October 1976
Revision D:	June 1977
Revision E:	December 1977
Revision F:	February 1978

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION.

1.1 Program Purpose (Abstract)

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

A. Unique addressing techniques
B. Worse case noise patterns, and
C. Instruction execution thruout memory.

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

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

1.2 System Requirements

A. Hardware Requirements

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

B. Software Requirements

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

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

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

1.3 Related Documents and Standards

A. Programming Practices - Document No. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC Package - MAINDEC-11-DZQAC-C2-D
C. 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 SW05=1, then:

Load Address 214,
Press START.

B. If running without memory management, then:

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

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

Load Address 777707 (PC)
Deposit 214
Press CONTinue

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

Load Address	772340 (KIPAR0)
Deposit	<(relocation factor)/100>
(Example:	Relocation factor=540000, then deposit 005400)
Load Address	777572 (SR0)
Deposit	000001
Load Address	777707 (PC)
Deposit	214
Press	Continue

2.1.6 Starting address 220:

Byte address memory map typeout routine. This routine performs DATI, DATIP, DATO, and DATOB on all possible addresses, and types the ranges of addresses which do not cause a timeout trap.

2.2 Special Environments

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

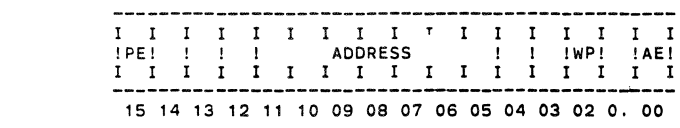
4.0 PERFORMANCE AND PROGRESS REPORTS

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:

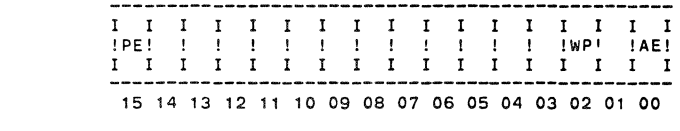
5.1 CORE PARITY REGISTER



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

```
---------------------------------------------------
I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
!DE! !SI! !      ADDRESS       !SE!IP!DC!EC!EE!
I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I  I
---------------------------------------------------
 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
```

BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

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

		DIAGNOSTIC CHECK MODE. THE INHIBIT MODE POINTER INDICATES WHICH 16K IS BEING INHIBITED,,,BIT 3 =1 THE SECOND 16K OF MEMORY IS INHIBITTED. WHEN BIT 13 IS SET TO A 0, BIT 3 BECOMES INOPERATIVE.
BIT02	DIAGNOSTIC CHECK A	WHEN SET ENABLES READ-WRITE OF CHECK BITS(SEE BITS 11-5)
BIT01	DISABLE ERROR CORRECTION	WHEN SET NO ERROR CORRECTION TAKES PLACE
BIT00	DOUBLE ERROR ENABLE	WHEN SET ENABLES TRAP TO VECTOR 114 ON DOUBLE ERROR.

6.0 SUB-TEST SUMMARIES

6.1 Section 1: Address Tests.

These tests verify the uniqueness of every memory address.

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

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

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

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

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

6.2 Section 2: Worst Case Noise Tests.

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

TEST 6 and TEST 7 Are supplied to allow the operator to select a single word data pattern (SA=204) and SCOPE on

either the writing (DATO) in TEST 6 or the reading (DATI) in TEST 7 of that data.

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

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

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

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

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

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

6.3 Section 3: Instruction Execution Tests.

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

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

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

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

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

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

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

6.4 Section 4: Mos Tests

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

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

6.5 Special Toggle In Tests

6.5.1 Toggle-in-program #1

The following is a toggle in memory address test. This test is useful when an address selection failure is suspected involving the first 8K of memory. This program writes the value of each address into itself starting with the lower limit and continuing to the upper limit. After all addresses have been written each address is checked for the correct contents starting with the upper limit and continuing to the lower limit.

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

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

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

6.5.2 Toggle-in-Program #2

The following is also a toggle in program to be used with toggle-i -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 ISTING

Attached

FLOW CHART
****

CZQMCF0 0-124K MEM EXER 16K

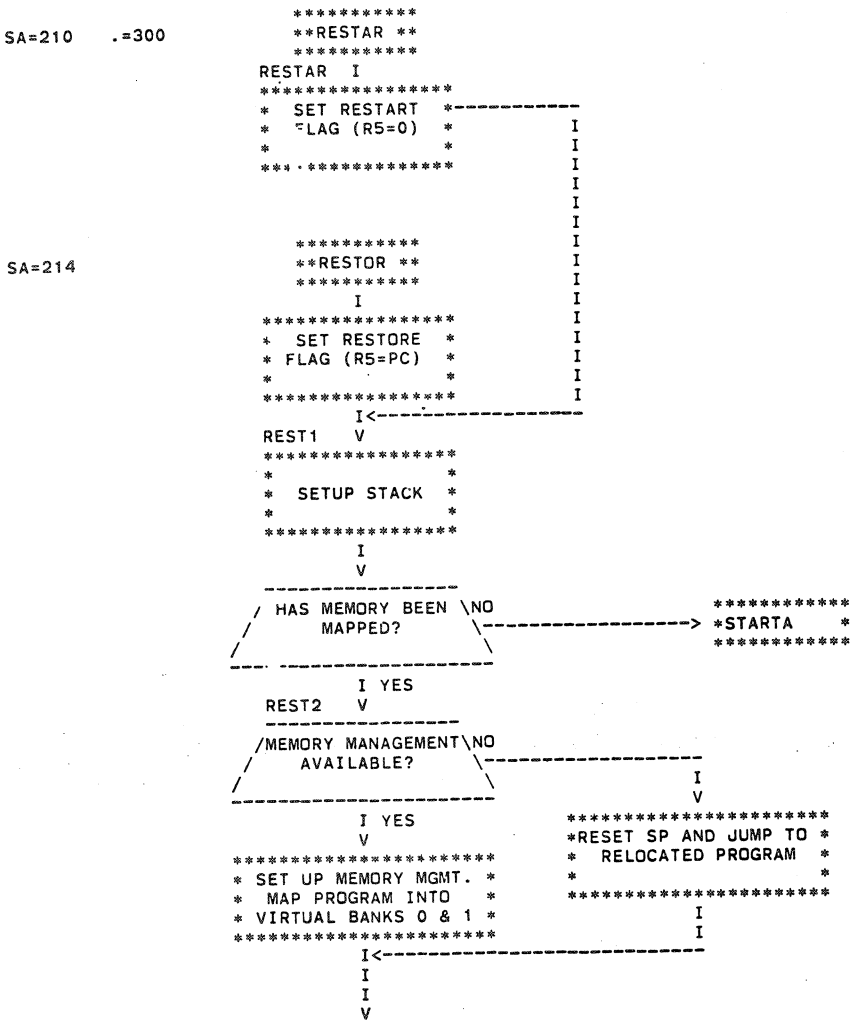
******* ********

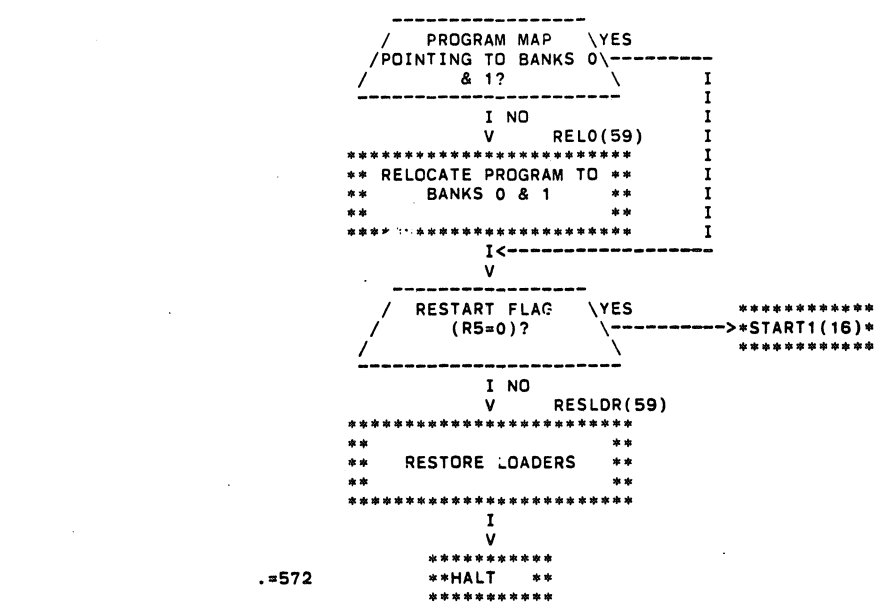
COPYRIGHT 1978
DIGITAL EQUIPMENT CORPORATION
MAYNARD, MASS. 01754

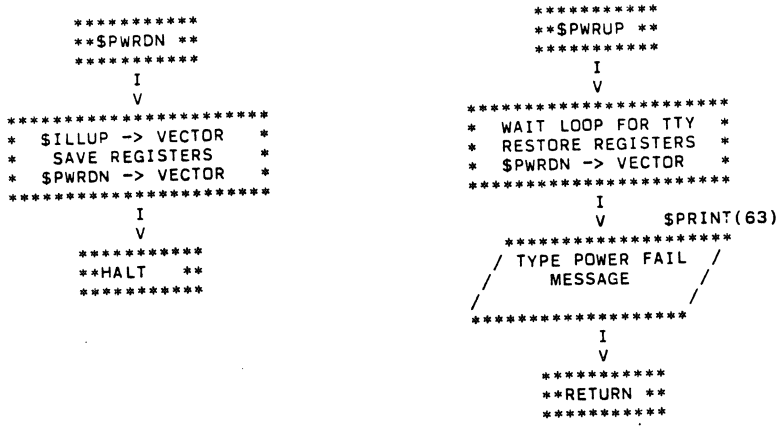
TABLE OF CONTENTS
****

TABLE OF CONTENTS
****

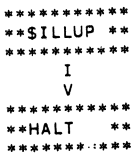
```
                        **********************
                        * SWITCH SETTINGS AND *
                        *  BASIC DEFINITIONS  *
                        *                     *
                        *  *******************

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


SA=210 .=300
**RESTAR **
RESTAR
SET RESTART FLAG (R5=0)
SA=214
**RESTOR **
SET RESTORE FLAG (R5=PC)
REST1
SETUP STACK
HAS MEMORY BEEN MAPPED?
NO
*STARTA
YES
REST2
MEMORY MANAGEMENT AVAILABLE?
NO
RESET SP AND JUMP TO RELOCATED PROGRAM
YES
SET UP MEMORY MGMT. MAP PROGRAM INTO VIRTUAL BANKS 0 & 1

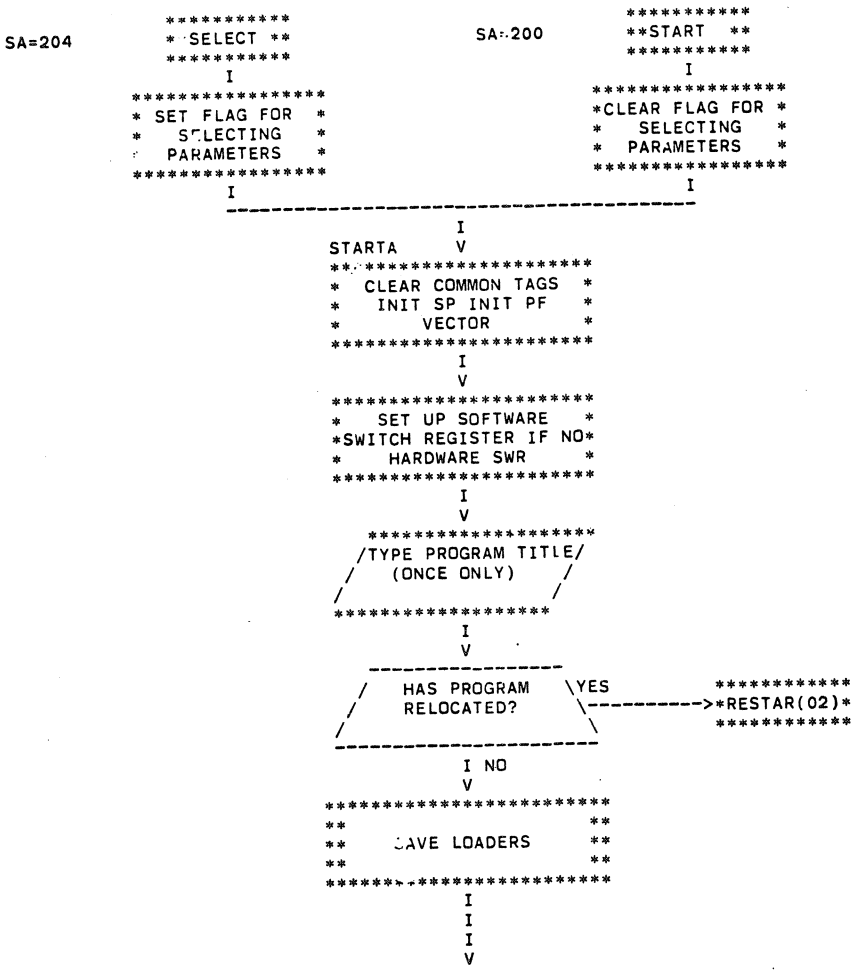

PROGRAM MAP POINTING TO BANKS 0 & 1?
YES
I NO
REL0(59)
** RELOCATE PROGRAM TO BANKS 0 & 1 **
RESTART FLAG (R5=0)?
YES
START1(16)
I NO
RESLDR(59)
** RESTORE LOADERS **
.=572
**HALT **

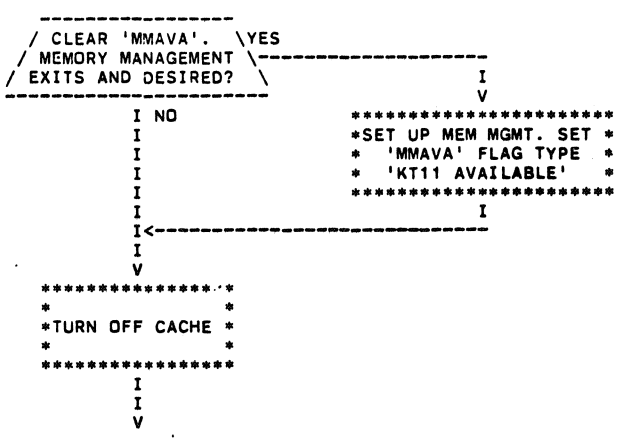

**$PWRDN **
$ILLUP -> VECTOR
SAVE REGISTERS
$PWRDN -> VECTOR
**HALT **

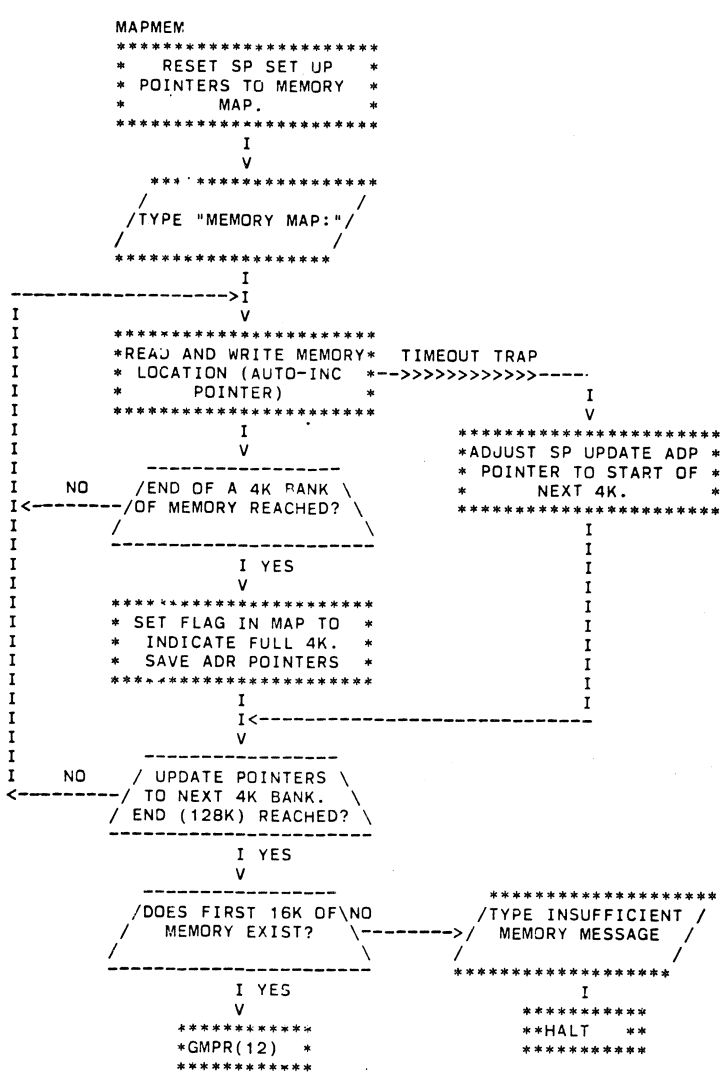
**$PWRUP **
WAIT LOOP FOR TTY
RESTORE REGISTERS
$PWRDN -> VECTOR
$PRINT(63)
TYPE POWER FAIL
MESSAGE
**RETURN **

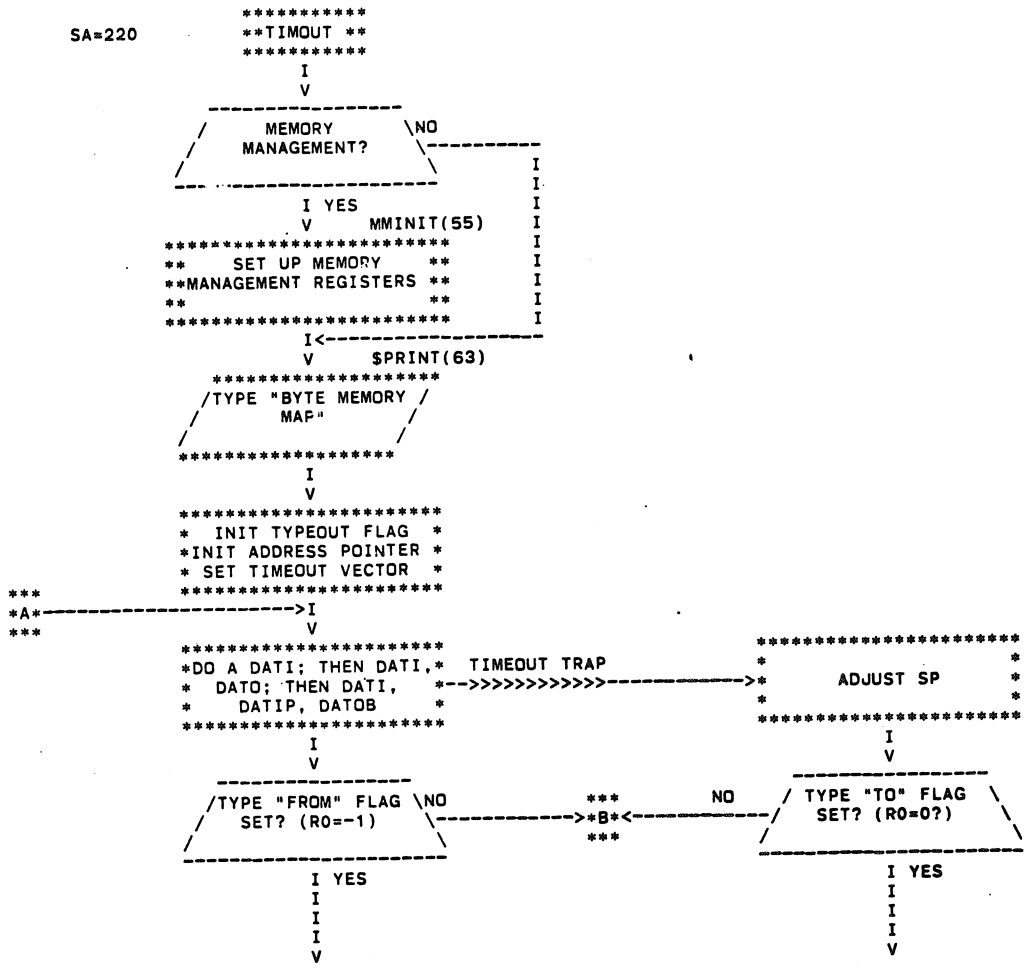

**$ILLUP **
**HALT **

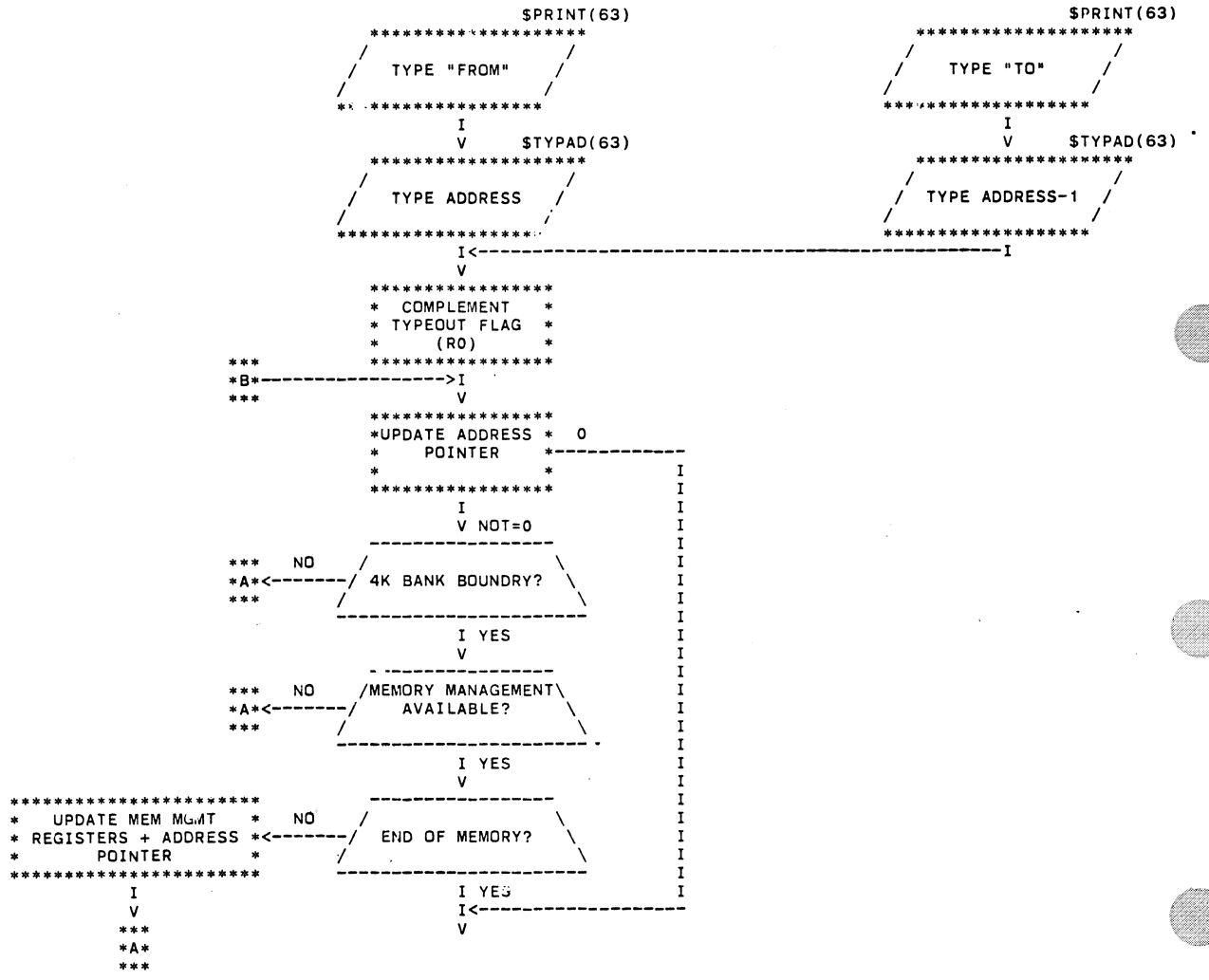
```
                        **********************
                        *  STANDARD 'SYSMAC'  *
                .=1100  *     COMMON TAGS     *
                        *                     *
                        **********************


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


                        **********************
                        *COMMON TAGS FOR THIS *
                        *       PROGRAM       *
                        *                     *
                        **********************


                        **********************
                        * RELATIVE ADDRESSING *
                        *  TABLE. ERROR DATA  *
                        *       POINTER       *
                        **********************


                        **********************
                        * MEMORY PARITY WORSE *
                        * CASE PATTERNS TABLE *
                        *                     *
                        **********************


                        **********************
                        *    MEMORY PARITY    *
                        *REGISTER ADDRESS AND *
                        *      MAP TABLE      *
                        **********************


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

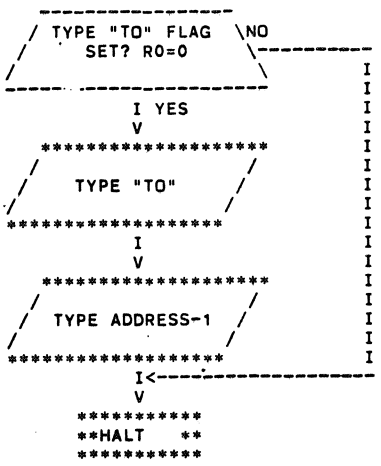

SA=204
SELECT
SET FLAG FOR SELECTING PARAMETERS
SA=200
START
CLEAR FLAG FOR SELECTING PARAMETERS
STARTA
CLEAR COMMON TAGS
INIT SP INIT PF VECTOR
SET UP SOFTWARE
SWITCH REGISTER IF NO HARDWARE SWR
TYPE PROGRAM TITLE
(ONCE ONLY)
HAS PROGRAM RELOCATED?
YES
RESTAR(02)
NO
SAVE LOADERS

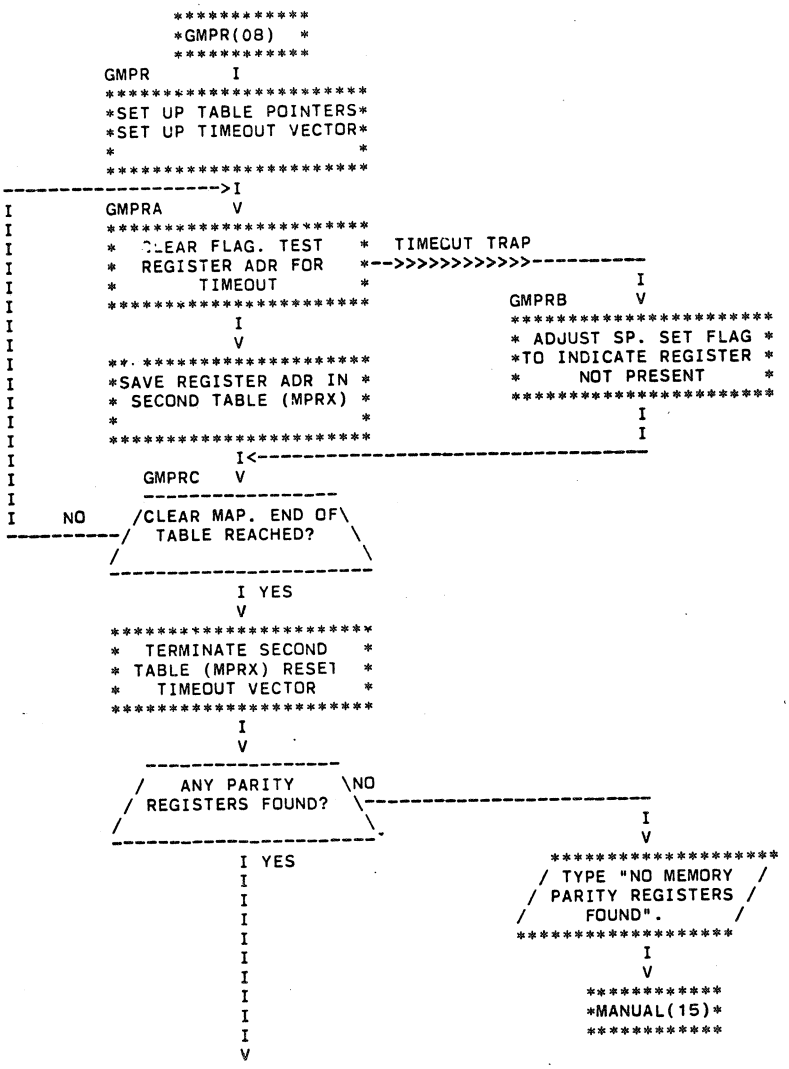

CLEAR 'MMAVA'. MEMORY MANAGEMENT EXITS AND DESIRED?
YES
NO
SET UP MEM MGMT. SET 'MMAVA' FLAG TYPE 'KT11 AVAILABLE'
TURN OFF CACHE

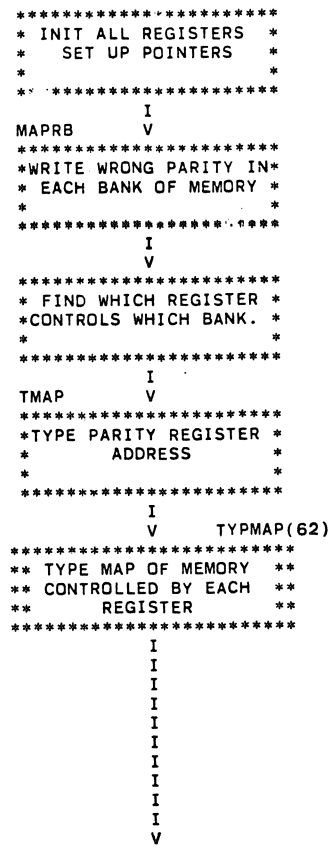

MAPMEM
RESET SP SET UP POINTERS TO MEMORY MAP.
/TYPE "MEMORY MAP:"/
READ AND WRITE MEMORY* LOCATION (AUTO-INC POINTER)
TIMEOUT TRAP
ADJUST SP UPDATE ADP * POINTER TO START OF * NEXT 4K.
NO
/END OF A 4K BANK \ OF MEMORY REACHED? \
YES
SET FLAG IN MAP TO INDICATE FULL 4K. SAVE ADR POINTERS
NO
/ UPDATE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

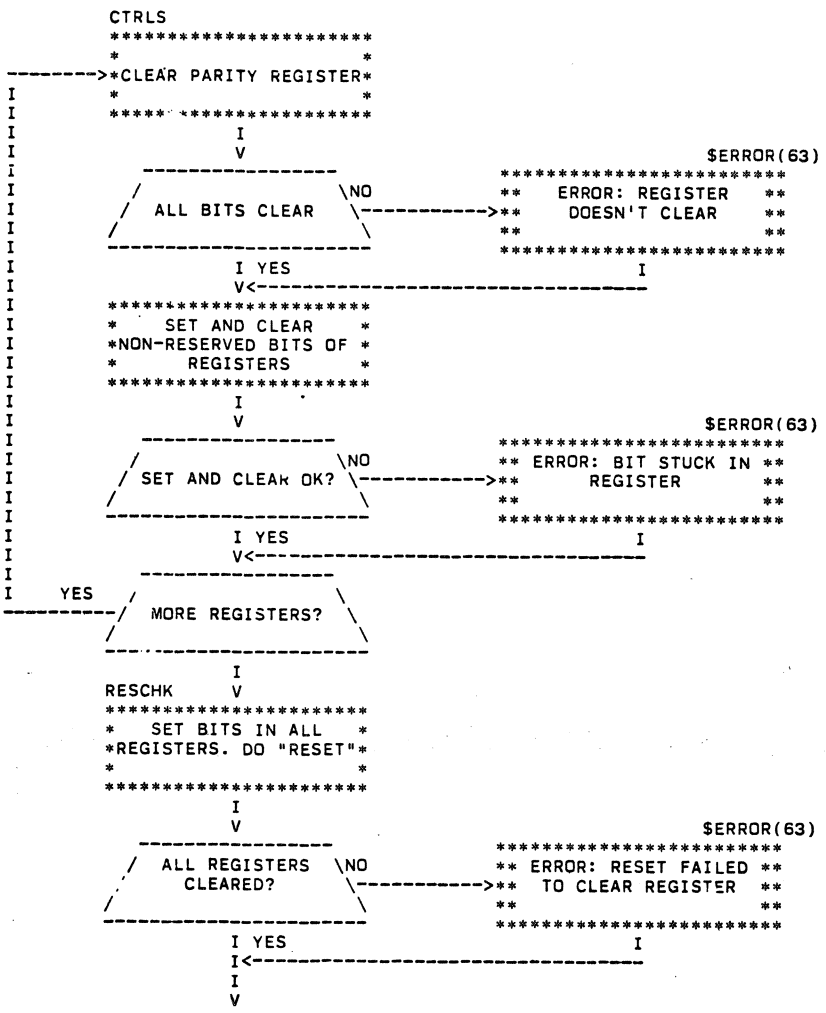

SA=220
**TIMOUT **
MEMORY MANAGEMENT?
NO
I YES
MMINIT(55)
SET UP MEMORY MANAGEMENT REGISTERS
$PRINT(63)
TYPE "BYTE MEMORY MAP"
INIT TYPEOUT FLAG
INIT ADDRESS POINTER
SET TIMEOUT VECTOR
A
DO A DATI; THEN DATI, DATO; THEN DATI, DATIP, DATOB
TIMEOUT TRAP
ADJUST SP
TYPE "FROM" FLAG SET? (R0=-1)
NO
B
NO
TYPE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

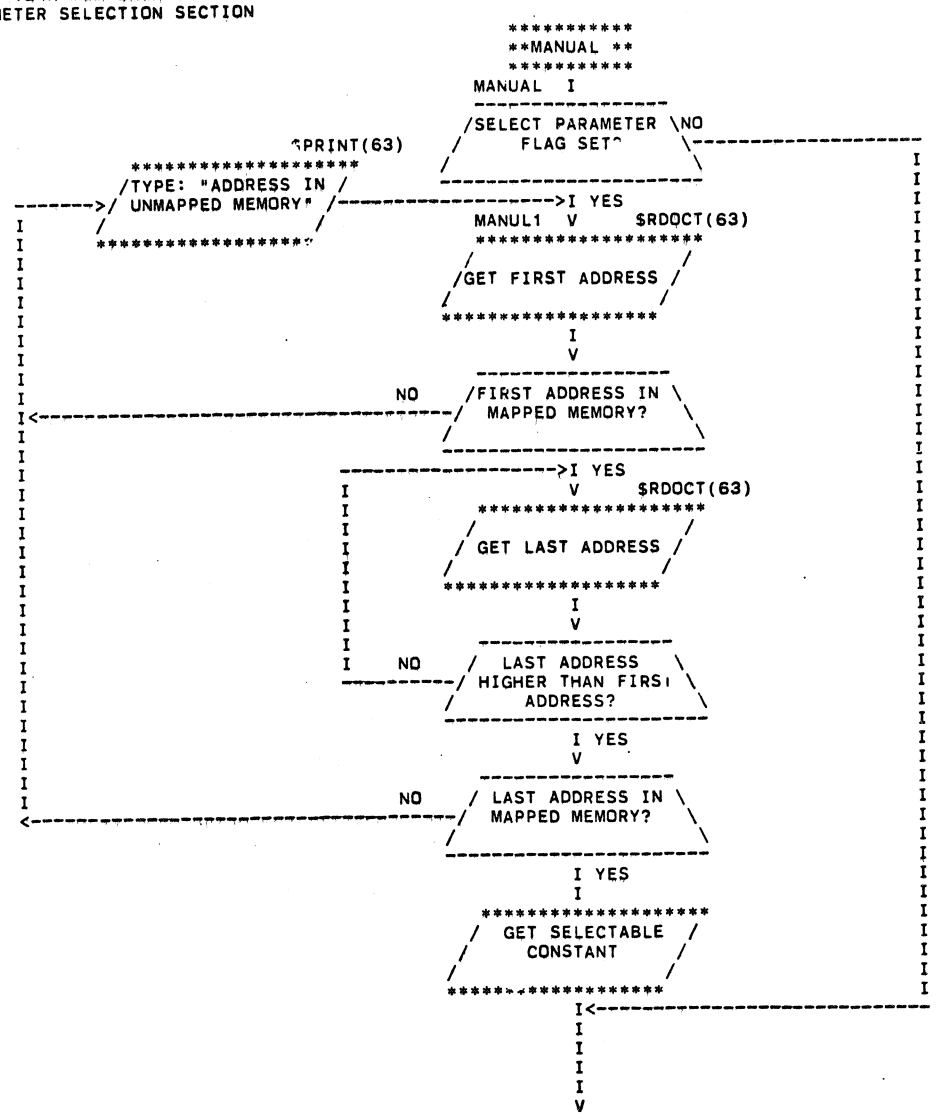

$PRINT(63)
TYPE "FROM"
$TYPAD(63)
TYPE ADDRESS
$PRINT(63)
TYPE "TO"
$TYPAD(63)
TYPE ADDRESS-1
COMPLEMENT TYPEOUT FLAG (R0)
UPDATE ADDRESS POINTER
0
NOT=0
NO
4K BANK BOUNDRY?
YES
NO
MEMORY MANAGEMENT AVAILABLE?
YES
NO
END OF MEMORY?
YES
UPDATE MEM MGMT REGISTERS + ADDRESS POINTER
A
B

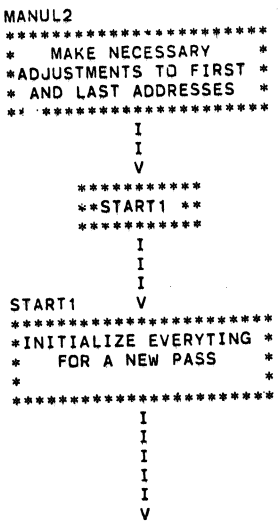

TYPE "TO" FLAG SET? R0=0
NO
I YES
TYPE "TO"
TYPE ADDRESS-1
**HALT **

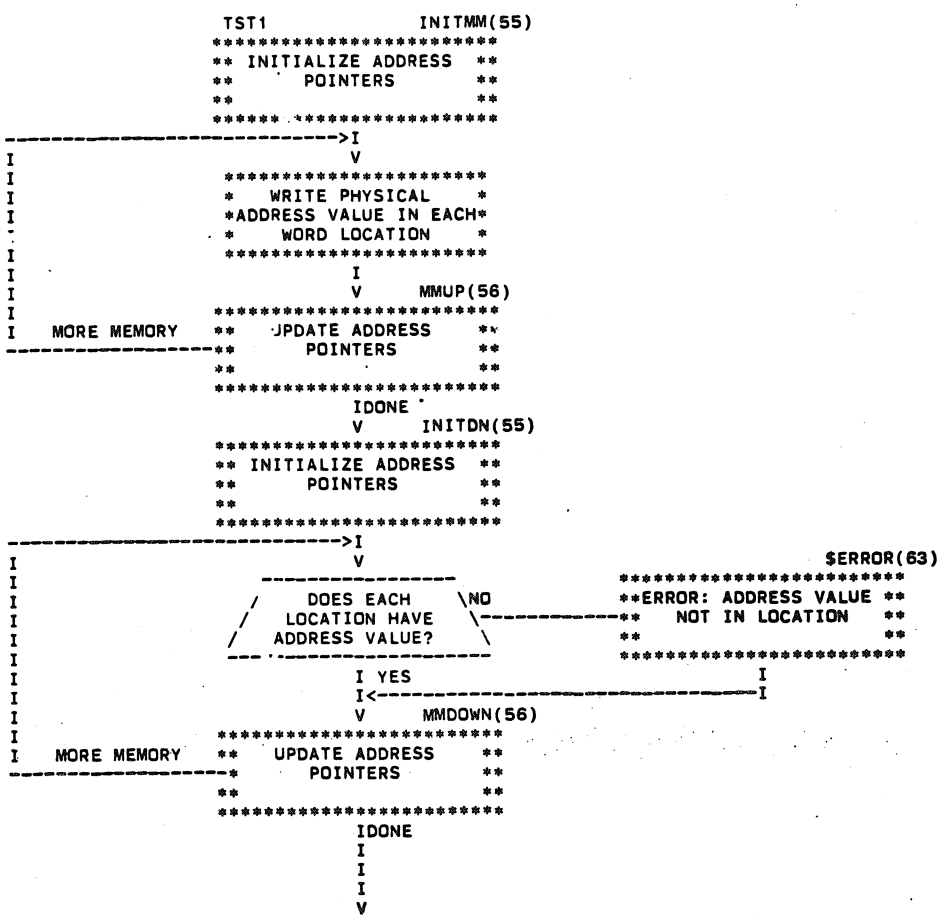

*GMPR(08) *
GMPR
SET UP TABLE POINTERS
SET UP TIMEOUT VECTOR
GMPRA
* CLEAR FLAG. TEST *
* REGISTER ADR FOR *
* TIMEOUT *
TIMEOUT TRAP
GMPRB
* ADJUST SP. SET FLAG *
*TO INDICATE REGISTER *
* NOT PRESENT *
*SAVE REGISTER ADR IN *
* SECOND TABLE (MPRX) *
GMPRC
NO
CLEAR MAP. END OF TABLE REACHED?
YES
* TERMINATE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

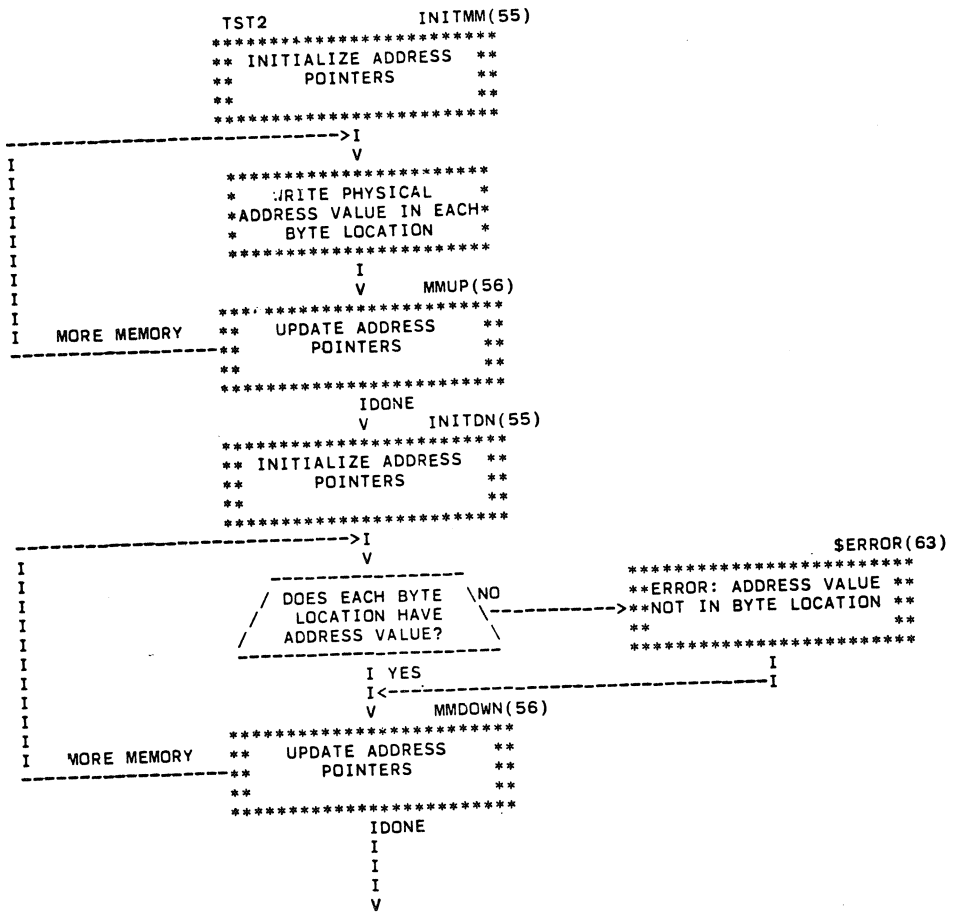

INIT ALL REGISTERS
SET UP POINTERS
MAPRB
WRITE WRONG PARITY IN
EACH BANK OF MEMORY
FIND WHICH REGISTER
CONTROLS WHICH BANK.
TMAP
TYPE PARITY REGISTER
ADDRESS
TYPMAP(62)
TYPE MAP OF MEMORY
CONTROLLED BY EACH
REGISTER

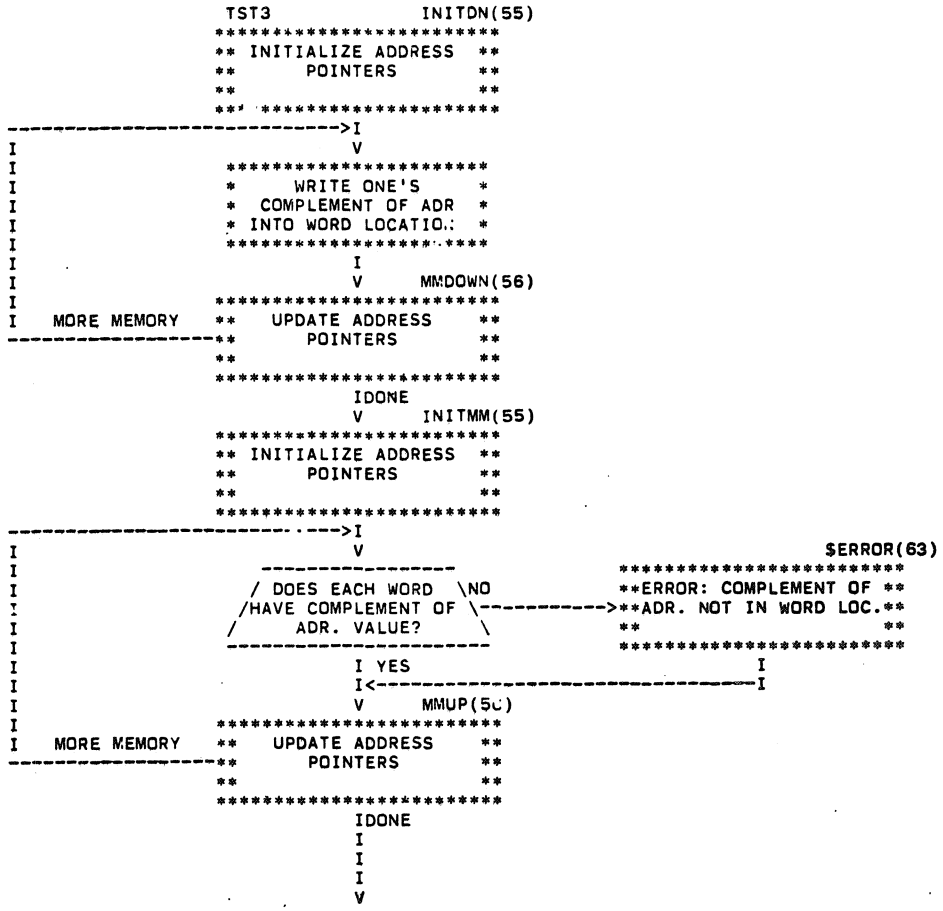

CTRLS
CLEAR PARITY REGISTER
ALL BITS CLEAR
NO
YES
$ERROR(63)
ERROR: REGISTER DOESN'T CLEAR
SET AND CLEAR NON-RESERVED BITS OF REGISTERS
SET AND CLEAR OK?
ERROR: BIT STUCK IN REGISTER
MORE REGISTERS?
RESCHK
SET BITS IN ALL REGISTERS. DO "RESET"
ALL REGISTERS CLEARED?
ERROR: RESET FAILED TO CLEAR REGISTER

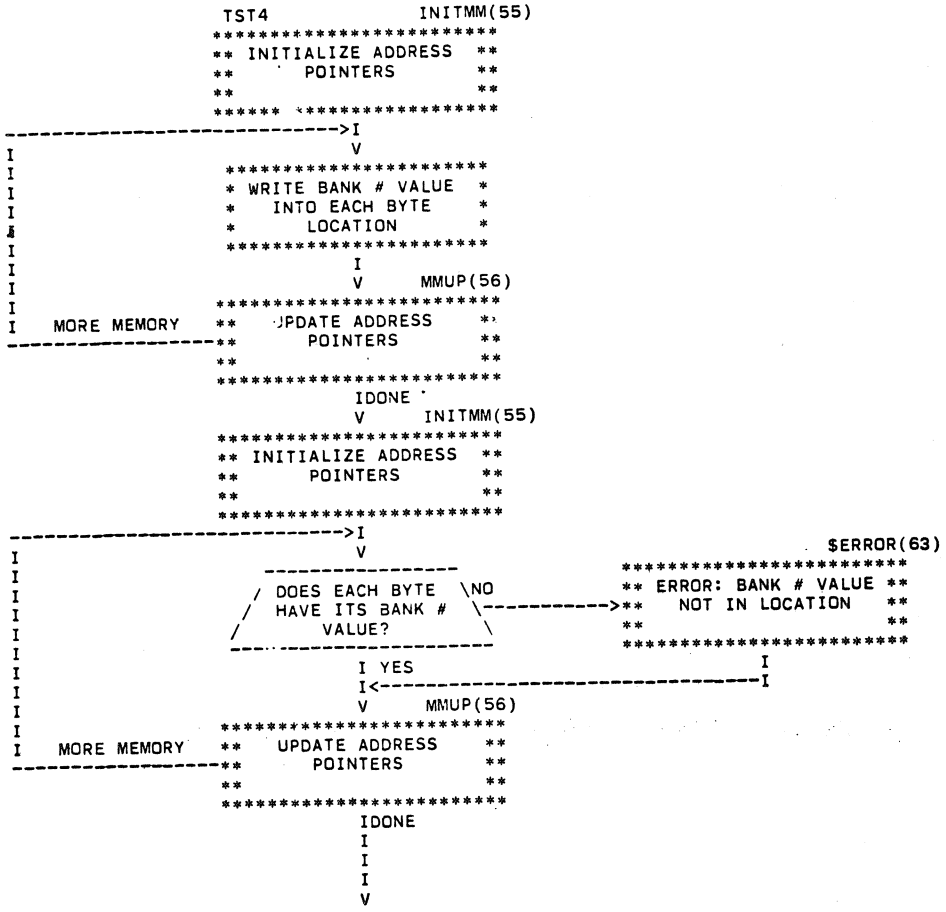

**MANUAL **
MANUAL I
SELECT PARAMETER FLAG SET
NO
YES
$PRINT(63)
TYPE: "ADDRESS IN UNMAPPED MEMORY"
MANUL1 V $RDOCT(63)
GET FIRST ADDRESS
FIRST ADDRESS IN MAPPED MEMORY?
NO
YES
$RDOCT(63)
GET LAST ADDRESS
LAST ADDRESS HIGHER THAN FIRST ADDRESS?
NO
YES
LAST ADDRESS IN MAPPED MEMORY?
NO
YES
GET SELECTABLE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

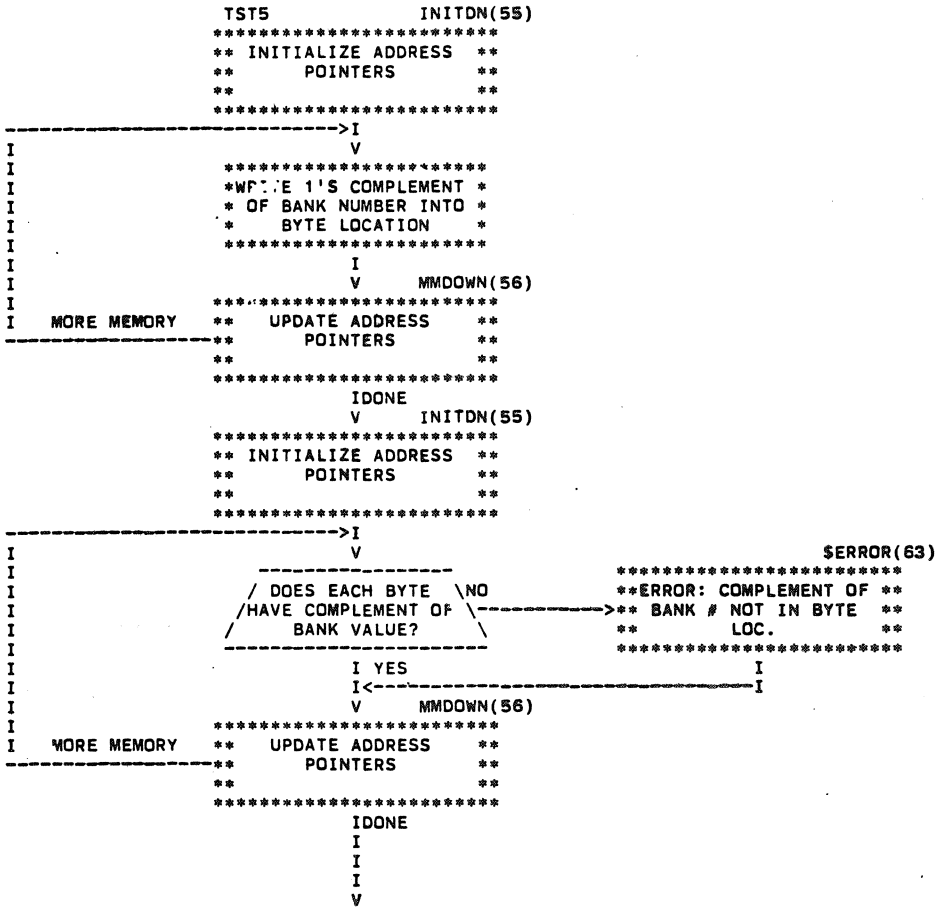

MANUL2
MAKE NECESSARY
ADJUSTMENTS TO FIRST
AND LAST ADDRESSES
START1
START1
INITIALIZE EVERYTING
FOR A NEW PASS

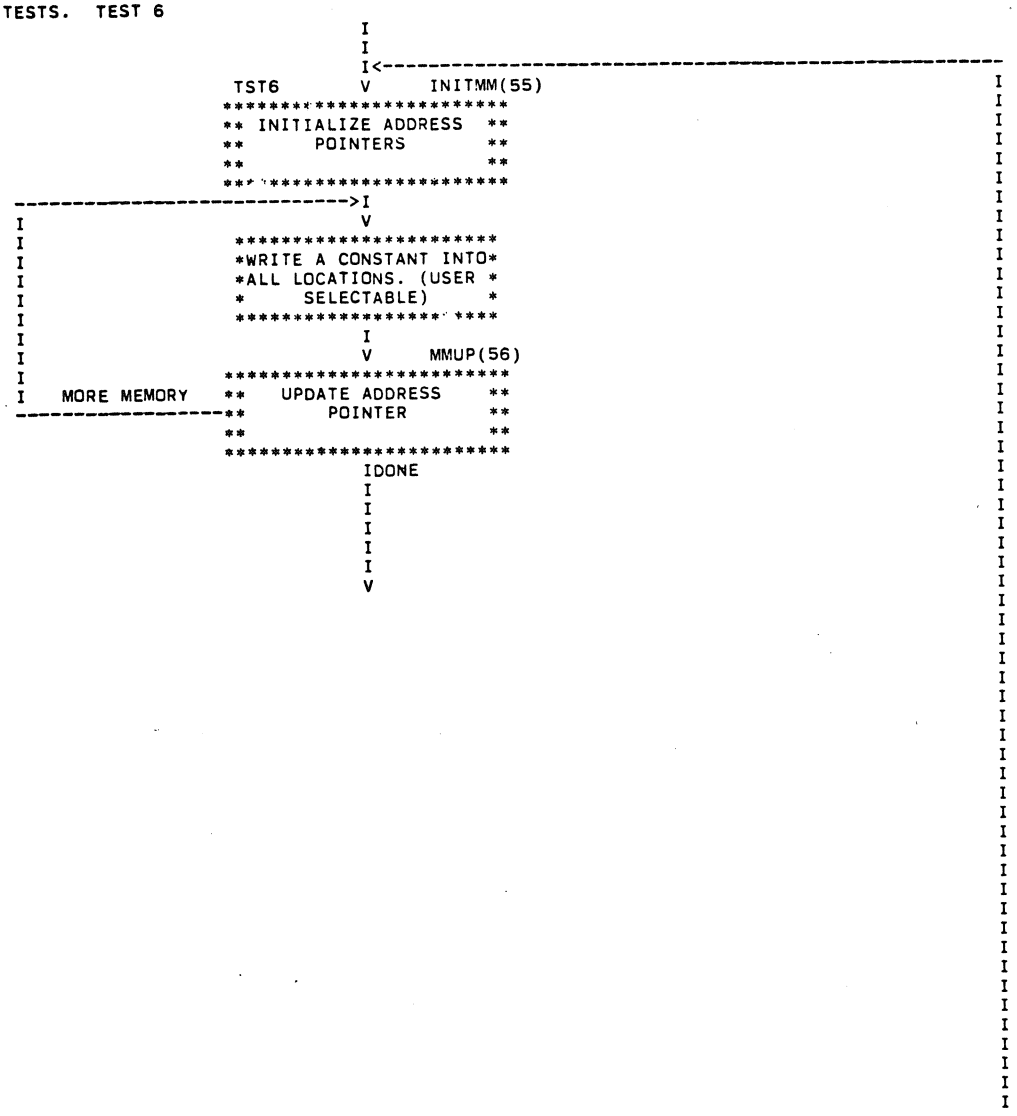

TST1
INITMM(55)
INITIALIZE ADDRESS POINTERS
WRITE PHYSICAL ADDRESS VALUE IN EACH WORD LOCATION
MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITDN(55)
INITIALIZE ADDRESS POINTERS
DOES EACH LOCATION HAVE ADDRESS VALUE?
NO
$ERROR(63)
ERROR: ADDRESS VALUE NOT IN LOCATION
I YES
MMDOWN(56)
UPDATE ADDRESS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

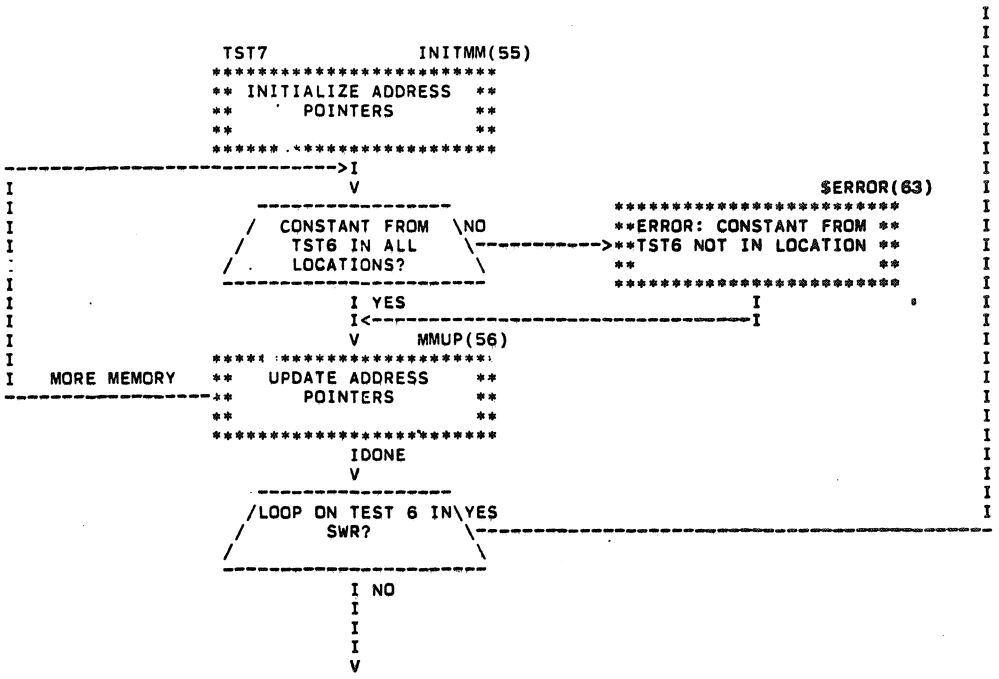

TST2
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE PHYSICAL
ADDRESS VALUE IN EACH
BYTE LOCATION
MMUP(56)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE
INITDN(55)
INITIALIZE ADDRESS
POINTERS
DOES EACH BYTE
LOCATION HAVE
ADDRESS VALUE?
NO
$ERROR(63)
**ERROR: ADDRESS VALUE **
**NOT IN BYTE LOCATION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

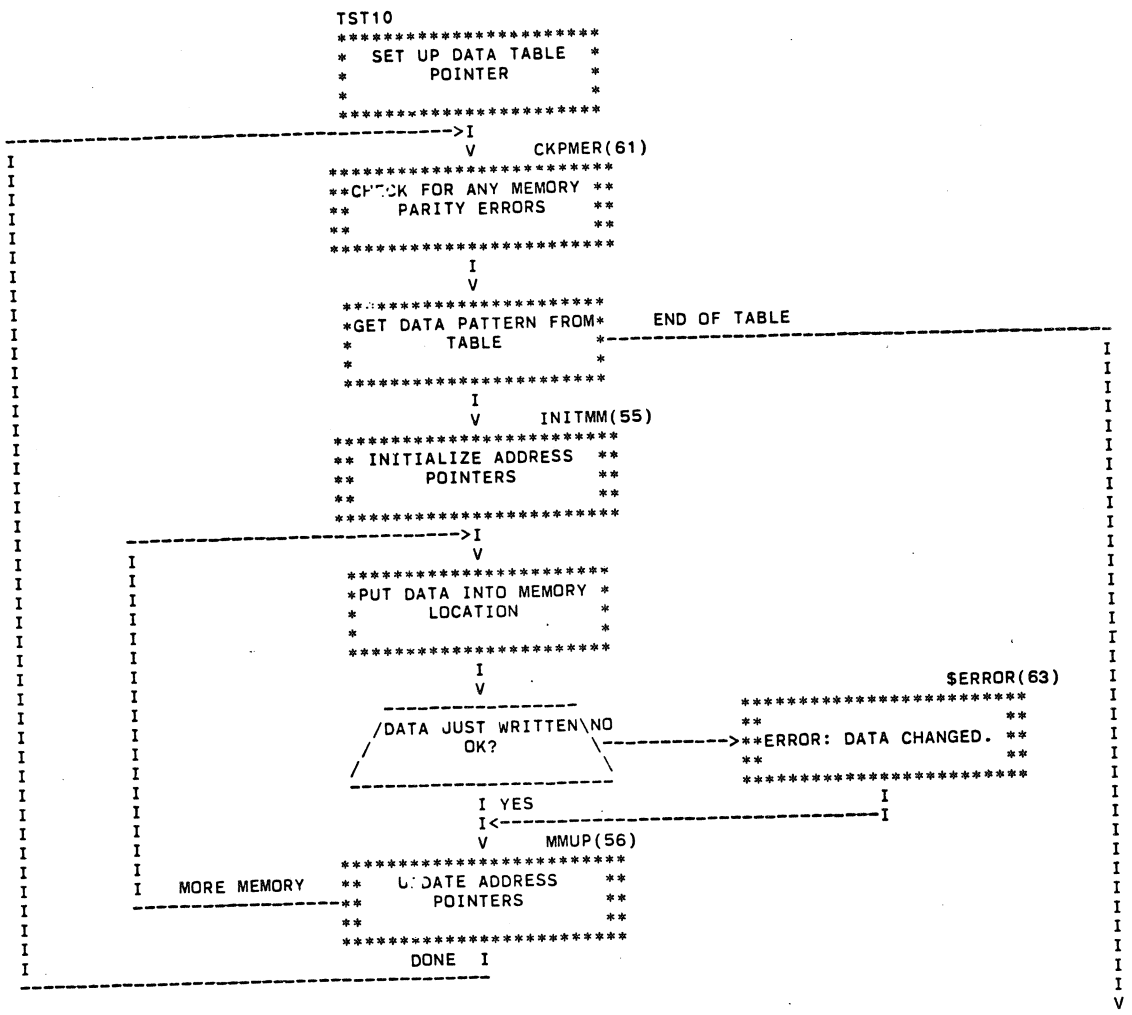

TST3
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE ONE'S
COMPLEMENT OF ADR
INTO WORD LOCATIO.:
MMDOWN(56)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
DOES EACH WORD
HAVE COMPLEMENT OF
ADR. VALUE?
NO
**ERROR: COMPLEMENT OF **
ADR. NOT IN WORD LOC.
I YES
MMUP(5C)
UPDATE ADDRESS
POINTERS
MORE MEMORY
IDONE

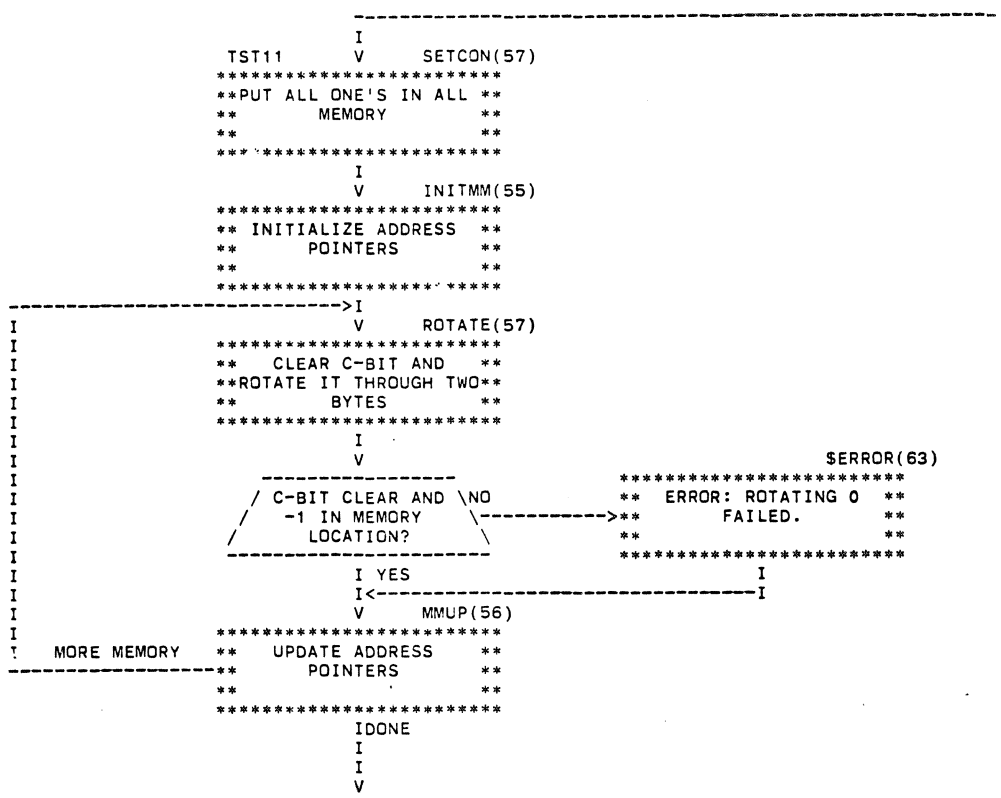

TST4
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE BANK # VALUE
INTO EACH BYTE
LOCATION
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DOES EACH BYTE HAVE ITS BANK # VALUE?
NO
$ERROR(63)
** ERROR: BANK # VALUE **
** NOT IN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

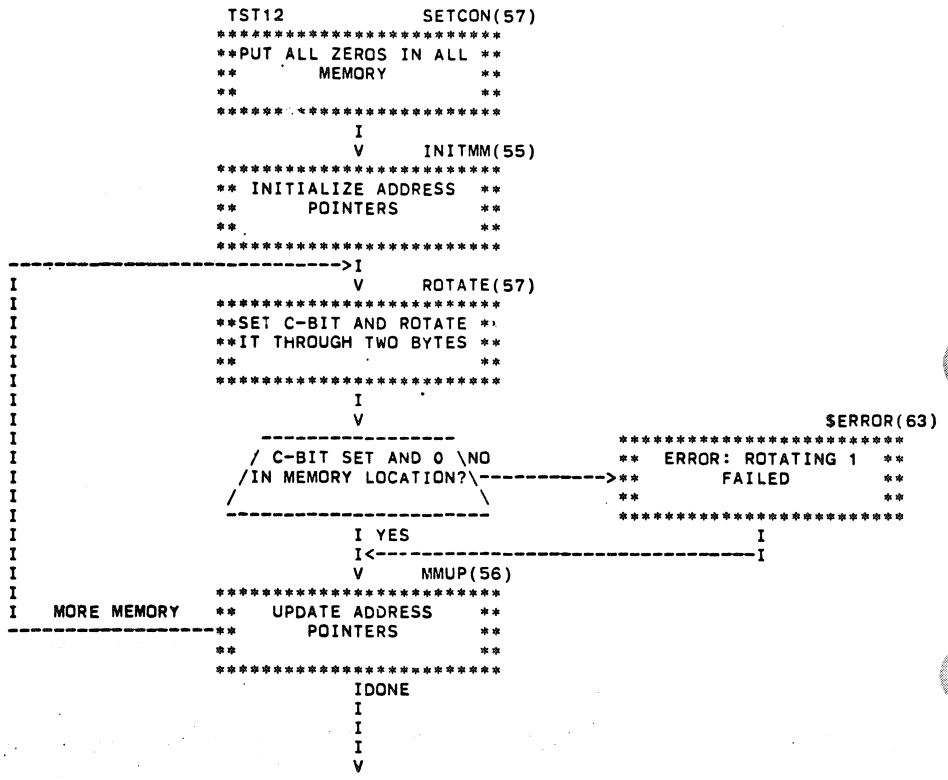

TST5
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
*WRITE 1'S COMPLEMENT *
* OF BANK NUMBER INTO *
* BYTE LOCATION *
MMDOWN(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITDN(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
/ DOES EACH BYTE \NO
/HAVE COMPLEMENT OF \
/ BANK VALUE?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

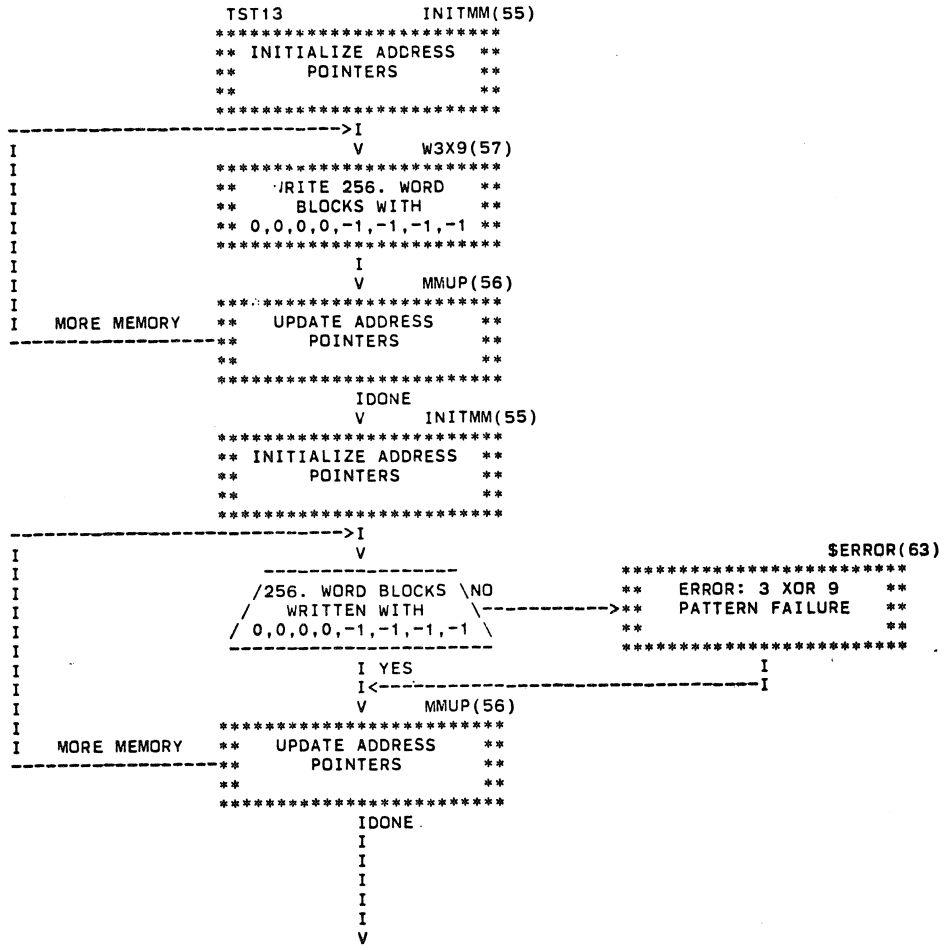

TST6
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
WRITE A CONSTANT INTO
*ALL LOCATIONS. (USER *
* SELECTABLE) *
MMUP(56)
** UPDATE ADDRESS **
** POINTER **
MORE MEMORY
IDONE

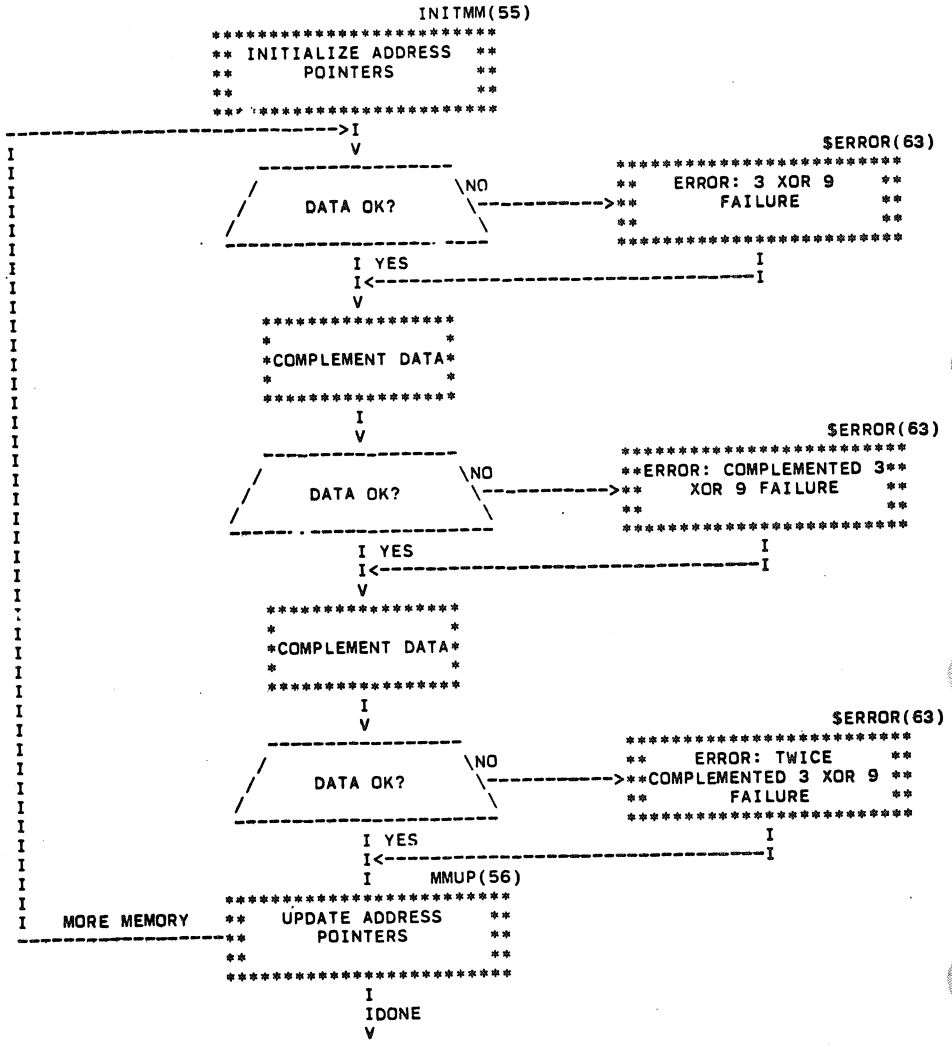

TST7
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
CONSTANT FROM TST6 IN ALL LOCATIONS?
NO
$ERROR(63)
**ERROR: CONSTANT FROM **
**TST6 NOT IN LOCATION **
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS **
** POINTERS **
IDONE
LOOP ON TEST 6 IN SWR?
YES
NO

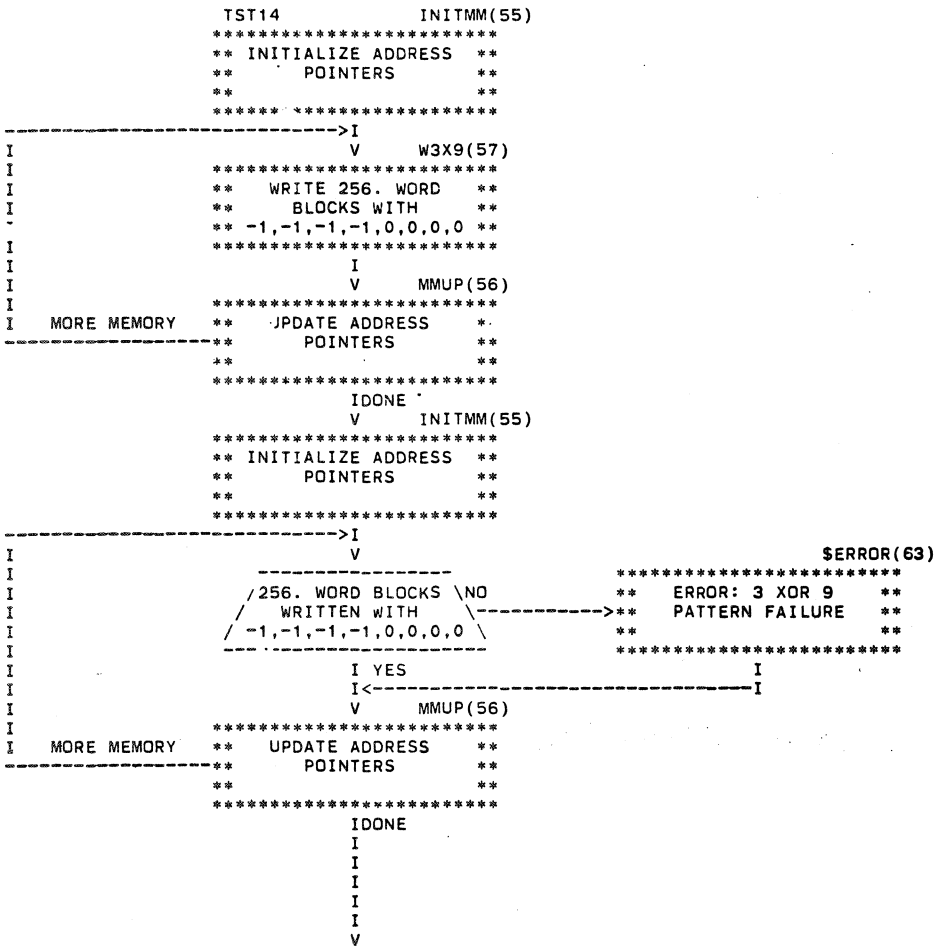

TST10
SET UP DATA TABLE POINTER
CKPMER(61)
CHECK FOR ANY MEMORY PARITY ERRORS
GET DATA PATTERN FROM TABLE
END OF TABLE
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT DATA INTO MEMORY LOCATION
$ERROR(63)
DATA JUST WRITTEN OK?
NO
ERROR: DATA CHANGED.
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
DONE

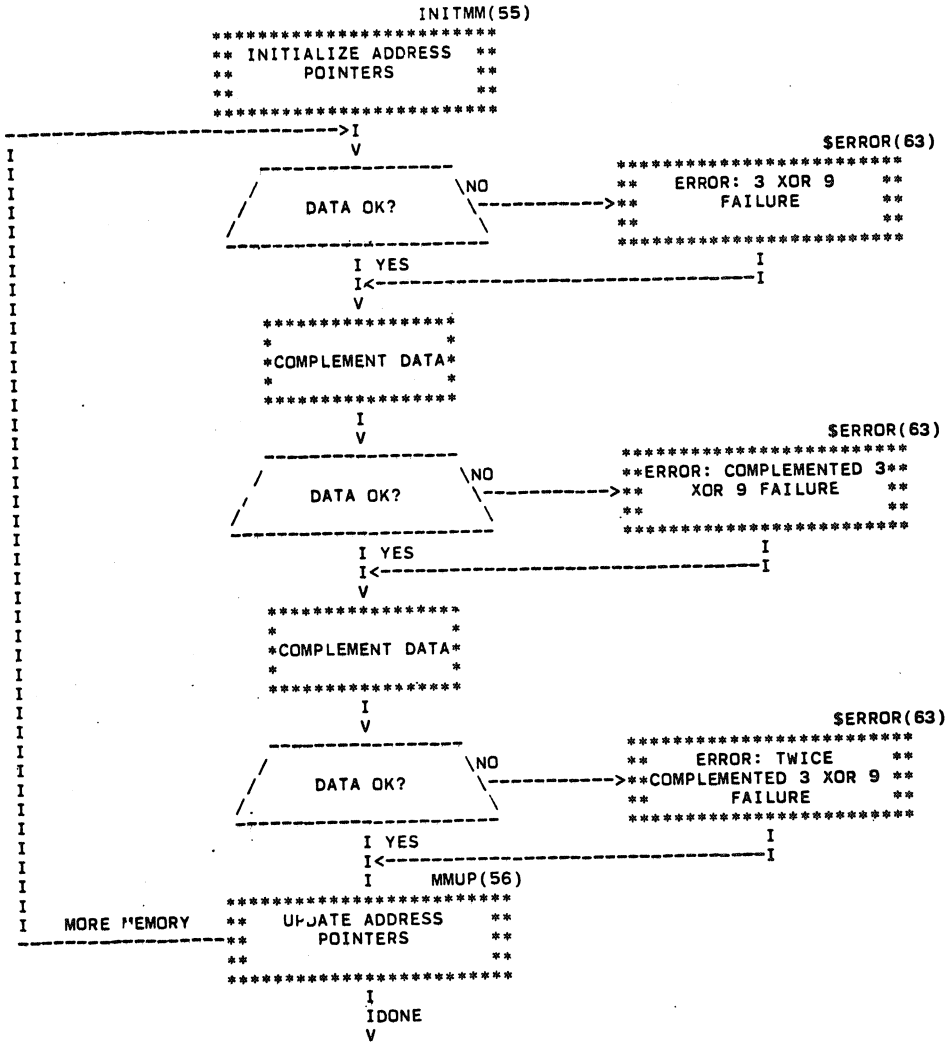

TST11
SETCON(57)
PUT ALL ONE'S IN ALL MEMORY
INITMM(55)
INITIALIZE ADDRESS POINTERS
ROTATE(57)
CLEAR C-BIT AND ROTATE IT THROUGH TWO BYTES
C-BIT CLEAR AND -1 IN MEMORY LOCATION?
NO
$ERROR(63)
ERROR: ROTATING 0 FAILED.
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

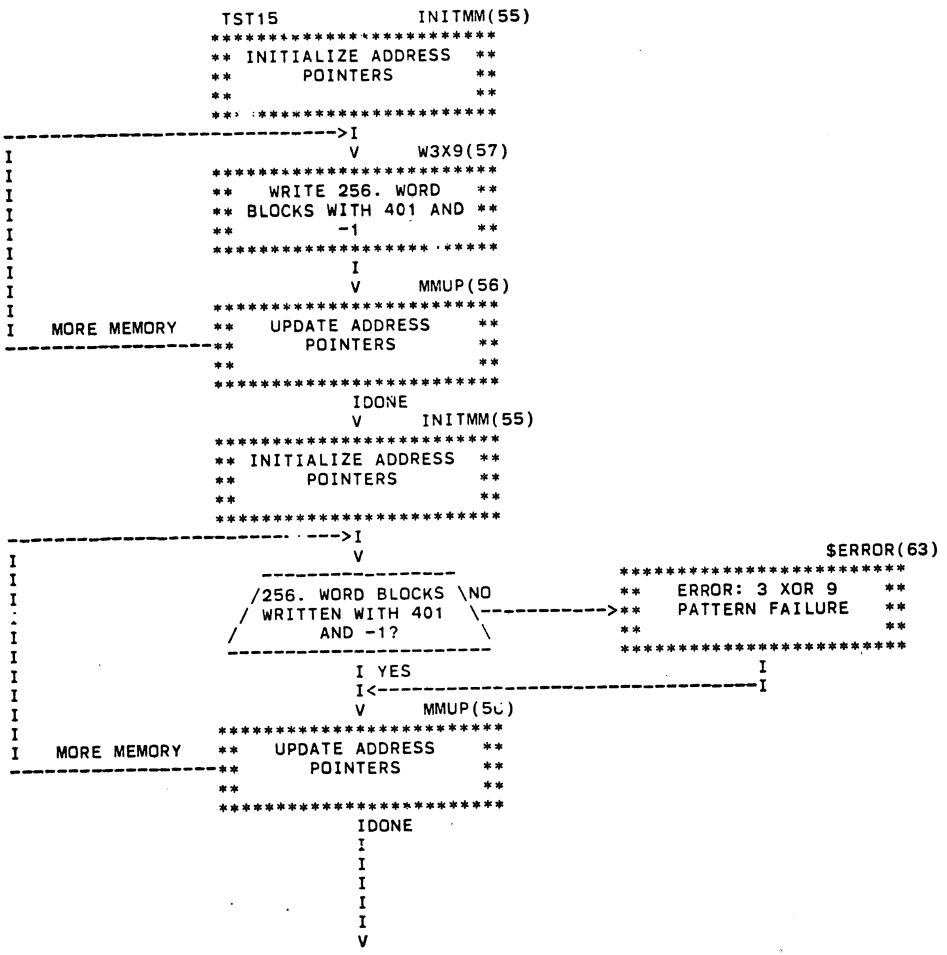

TST12
SETCON(57)
PUT ALL ZEROS IN ALL MEMORY
INITMM(55)
** INITIALIZE ADDRESS POINTERS **
ROTATE(57)
**SET C-BIT AND ROTATE IT THROUGH TWO BYTES **
C-BIT SET AND 0 IN MEMORY LOCATION?
YES
NO
$ERROR(63)
** ERROR: ROTATING 1 FAILED **
MMUP(56)
** UPDATE ADDRESS POINTERS **
MORE MEMORY
IDONE

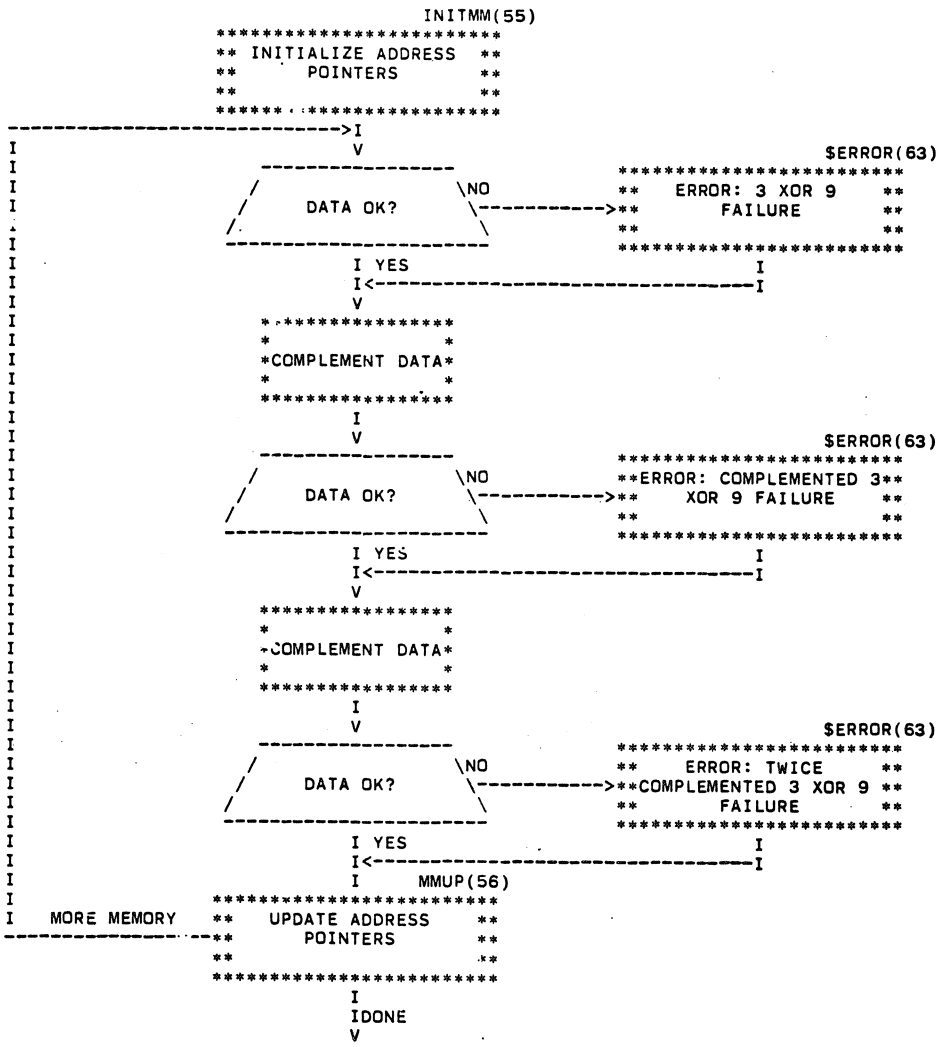

TST13
INITMM(55)
INITIALIZE ADDRESS POINTERS
W3X9(57)
WRITE 256. WORD BLOCKS WITH 0,0,0,0,-1,-1,-1,-1
MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
IDONE
INITMM(55)
INITIALIZE ADDRESS POINTERS
$ERROR(63)
/256. WORD BLOCKS WRITTEN WITH 0,0,0,0,-1,-1,-1,-1 \
NO
ERROR: 3 XOR 9 PATTERN FAILURE
YES
MMUP(56)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

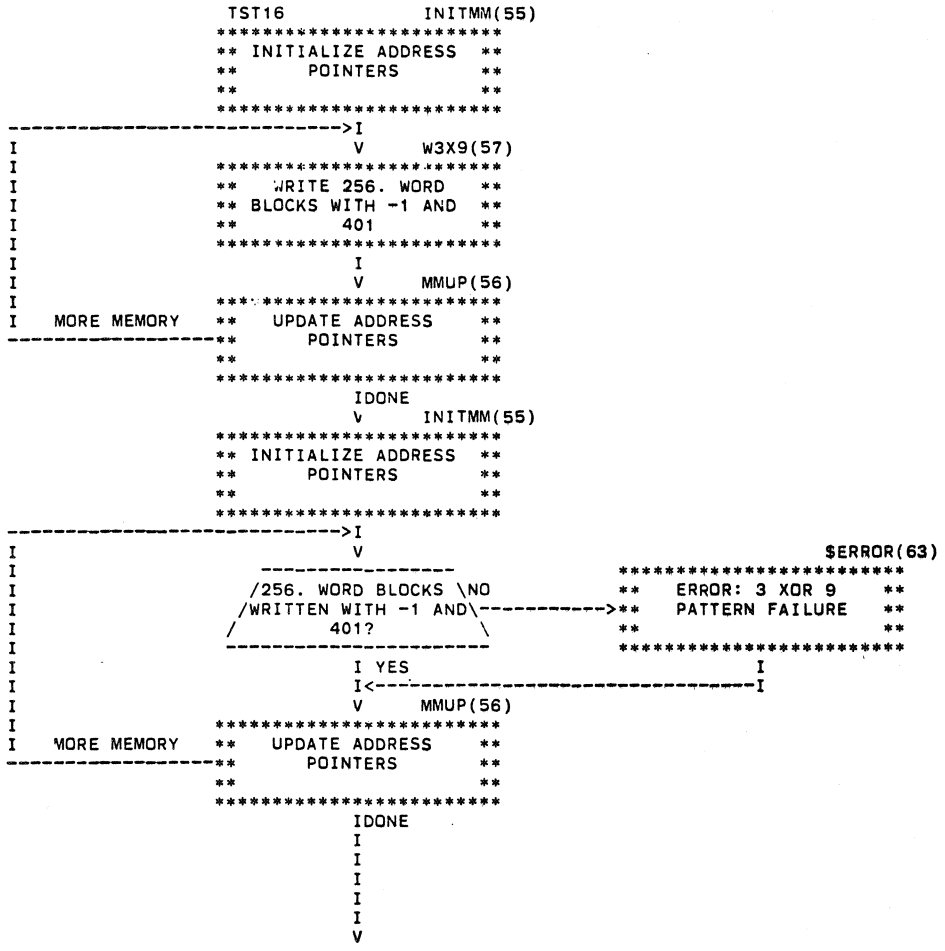

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
ERROR: 3 XOR 9
FAILURE
I YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: COMPLEMENTED 3
XOR 9 FAILURE
I YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: TWICE
COMPLEMENTED 3 XOR 9
FAILURE
I YES
I MMUP(56)
** UPDATE ADDRESS **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

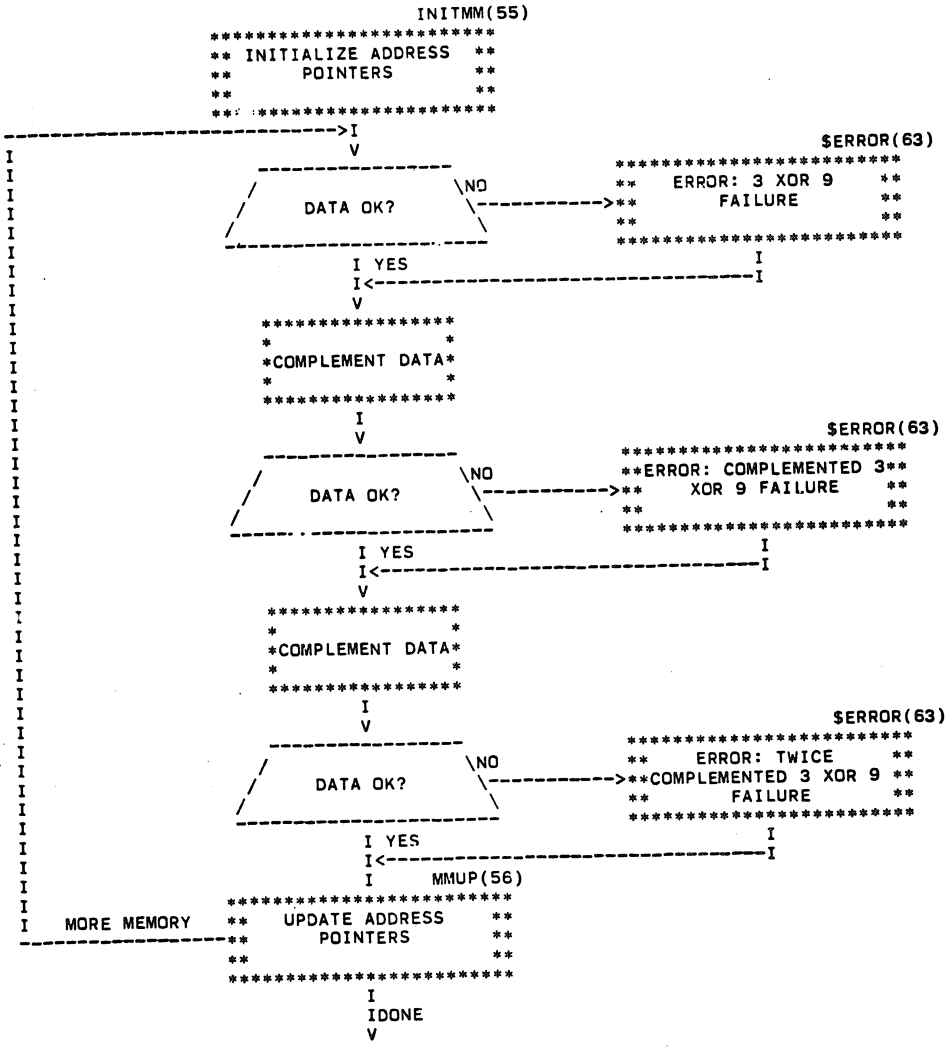

TST14
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH **
** -1,-1,-1,-1,0,0,0,0 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
/256. WORD BLOCKS \NO
/ WRITTEN WITH \
\ -1,-1,-1,-1,0,0,0,0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

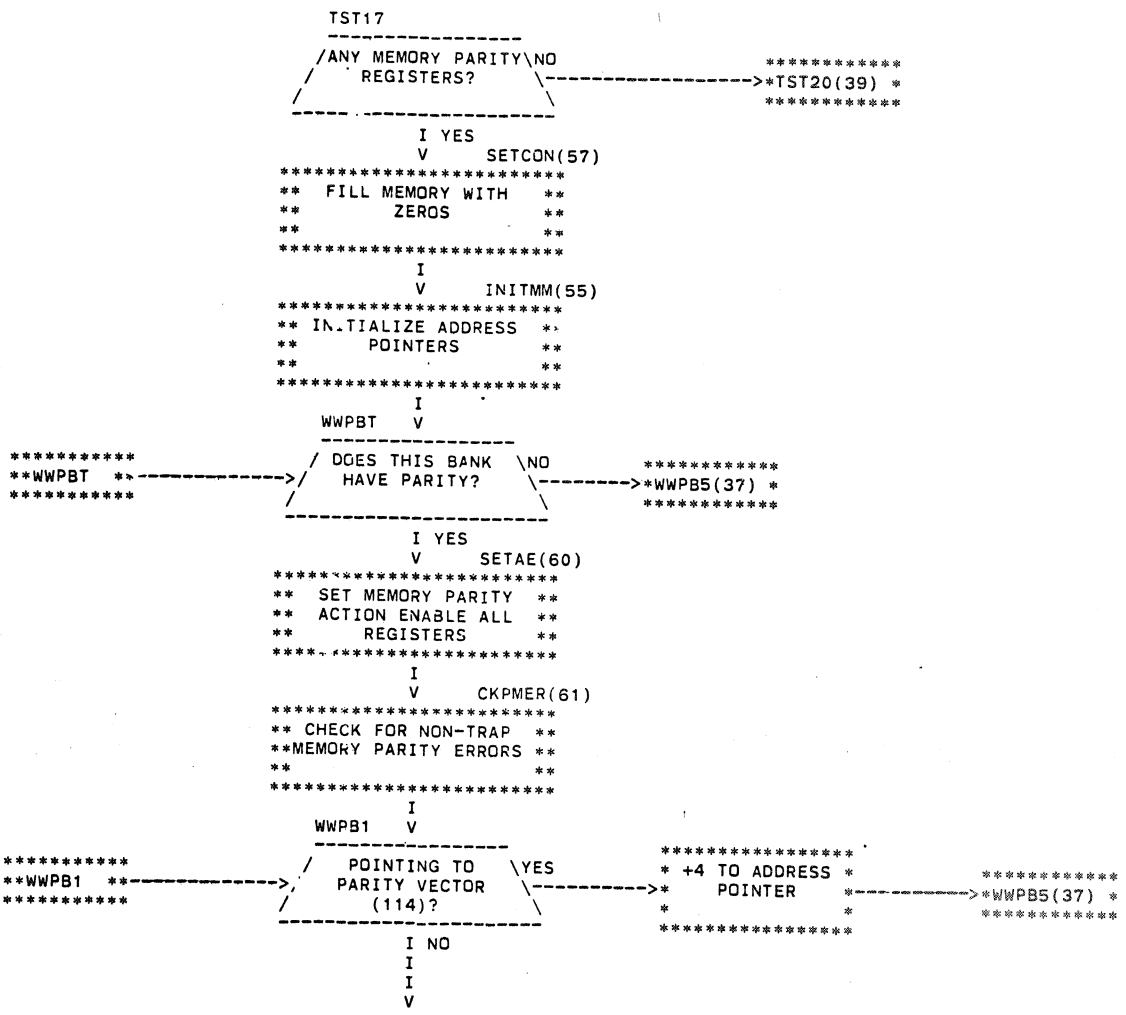

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
ERROR: 3 XOR 9
FAILURE
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: COMPLEMENTED 3
XOR 9 FAILURE
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: TWICE
COMPLEMENTED 3 XOR 9
FAILURE
YES
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS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

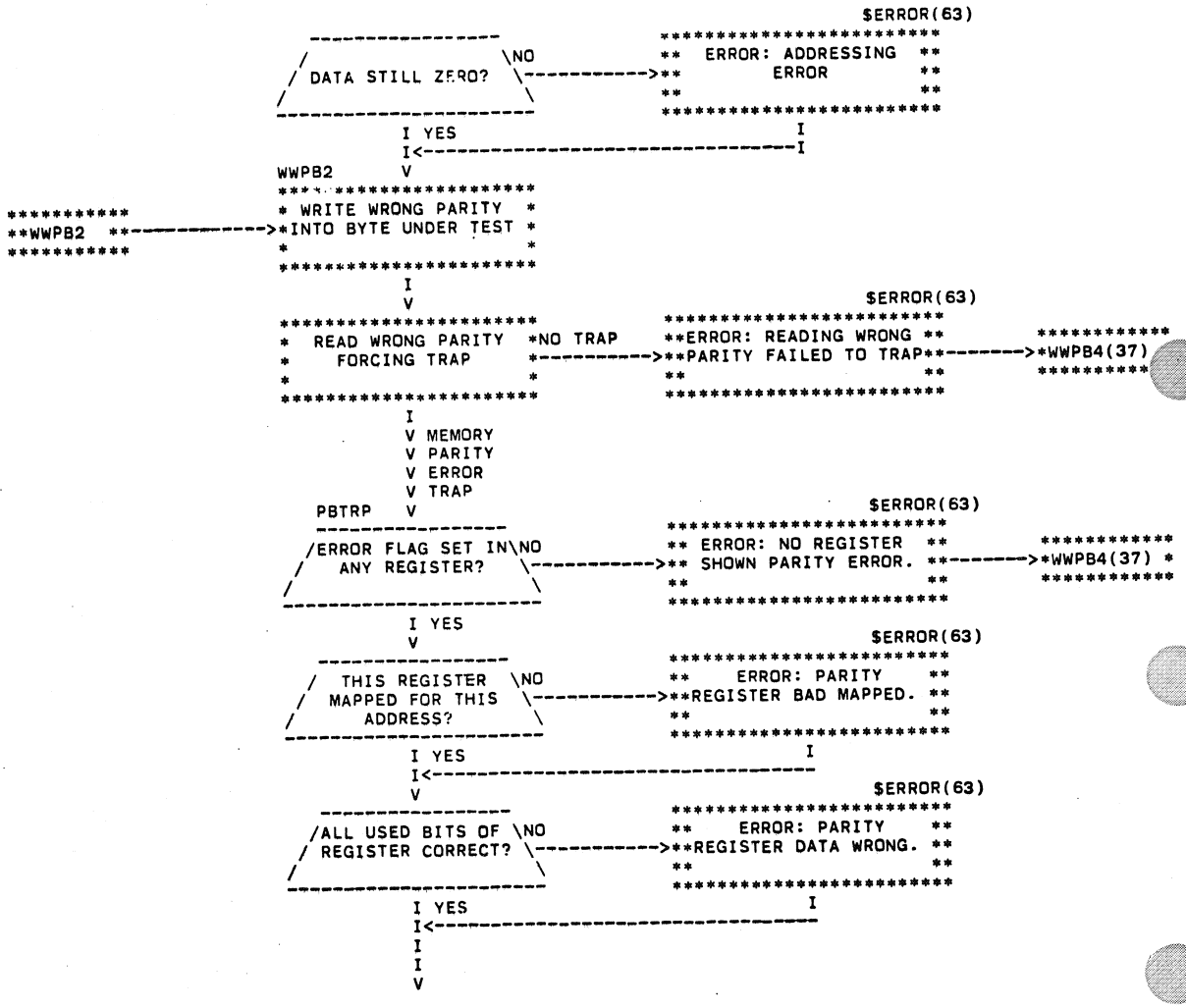

TST15
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH 401 AND **
** -1 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
/256. WORD BLOCKS \NO
/ WRITTEN WITH 401 \
/ AND -1? \
$ERROR(63)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

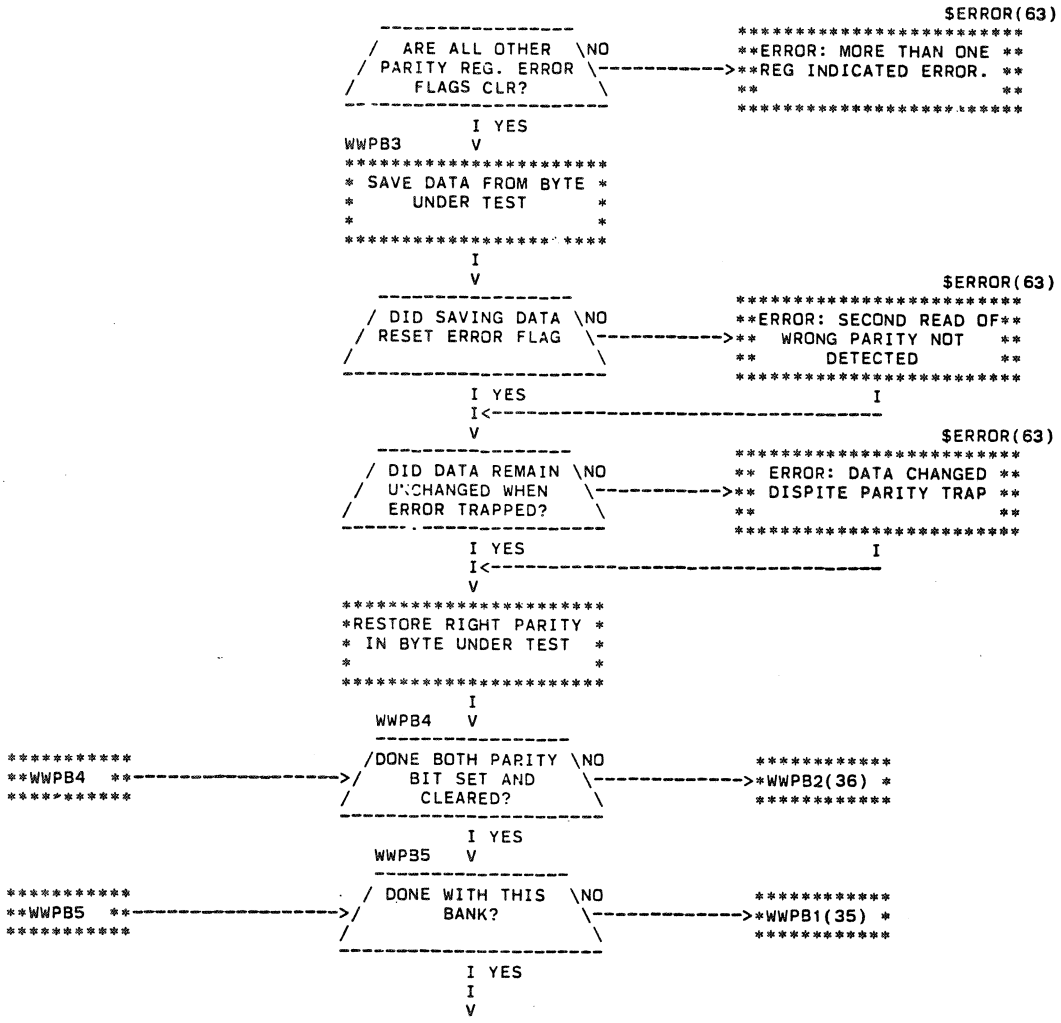

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
** ERROR: 3 XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
YES
MMUP(56)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

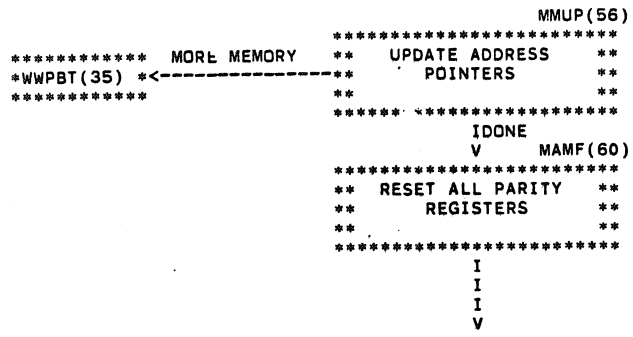

TST16
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
W3X9(57)
** WRITE 256. WORD **
** BLOCKS WITH -1 AND **
** 401 **
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
$ERROR(63)
/256. WORD BLOCKS \NO
/WRITTEN WITH -1 AND\
/ 401? \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

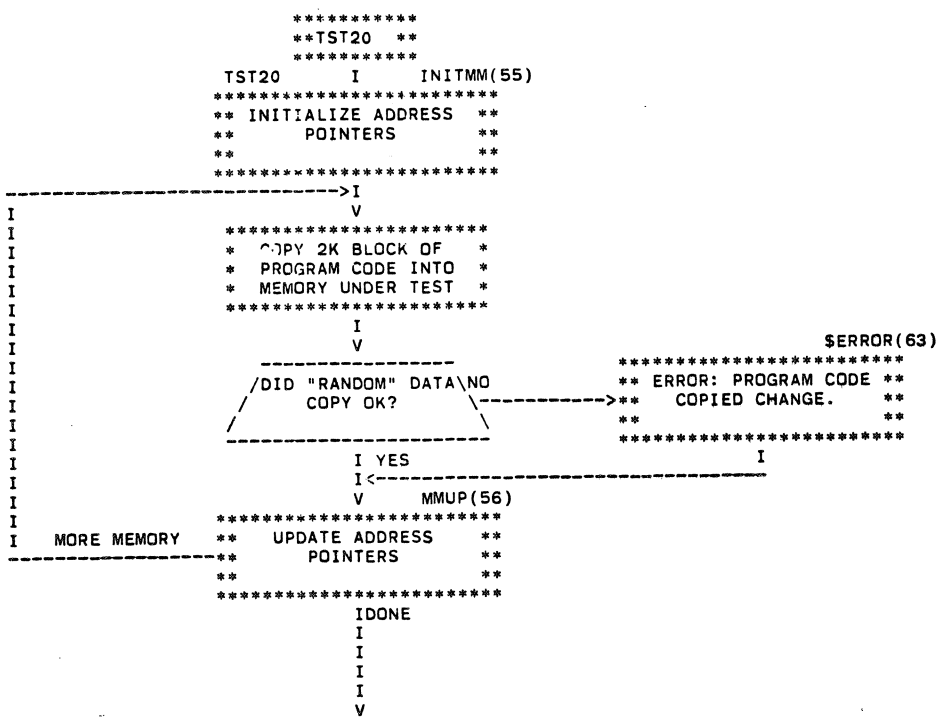

INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
DATA OK?
NO
$ERROR(63)
** ERROR: 3 XOR 9 **
** FAILURE **
I YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
ERROR: COMPLEMENTED 3
** XOR 9 FAILURE **
I YES
COMPLEMENT DATA
DATA OK?
NO
$ERROR(63)
** ERROR: TWICE **
**COMPLEMENTED 3 XOR 9 **
** FAILURE **
I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

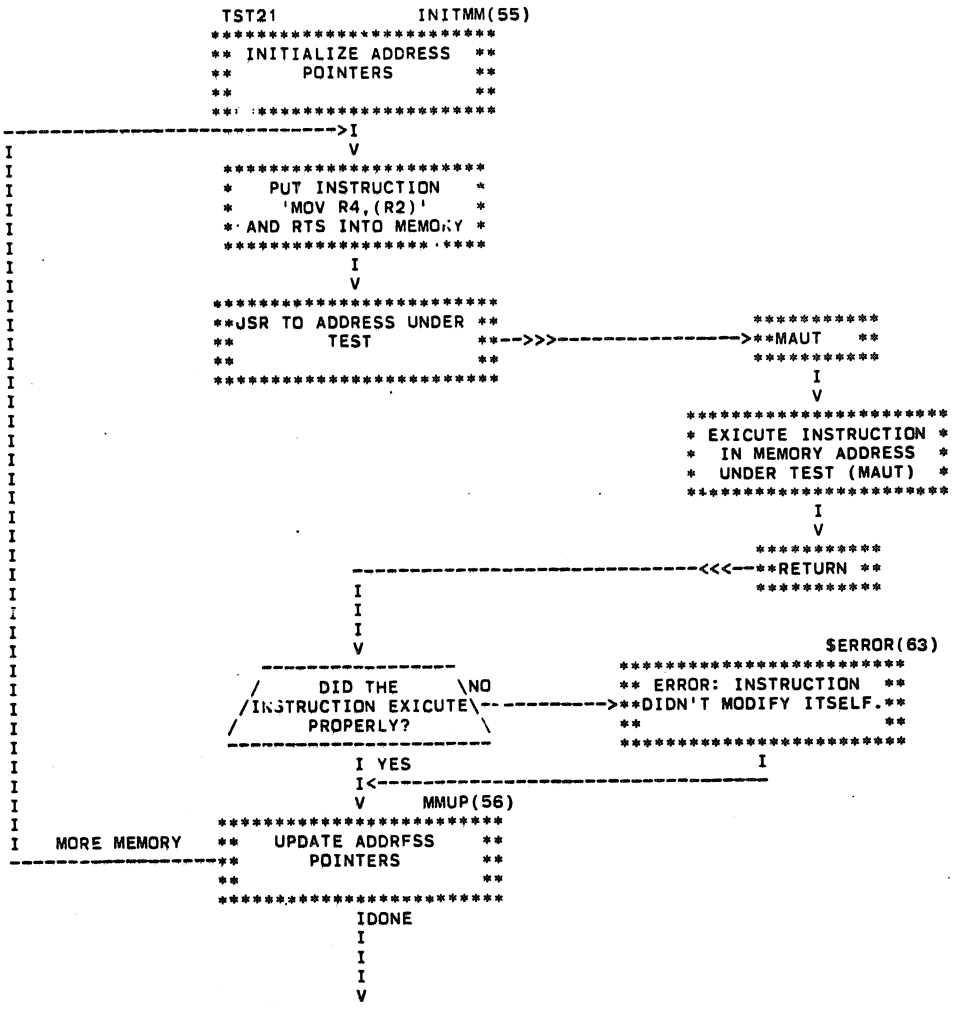

TST17
ANY MEMORY PARITY REGISTERS?
NO
TST20(39)
I YES
SETCON(57)
FILL MEMORY WITH ZEROS
INITMM(55)
INITIALIZE ADDRESS POINTERS
WWPBT
WWPBT
DOES THIS BANK HAVE PARITY?
NO
WWPB5(37)
I YES
SETAE(60)
SET MEMORY PARITY ACTION ENABLE ALL REGISTERS
CKPMER(61)
CHECK FOR NON-TRAP MEMORY PARITY ERRORS
WWPB1
WWPB1
POINTING TO PARITY VECTOR (114)?
YES
+4 TO ADDRESS POINTER
WWPB5(37)
I NO

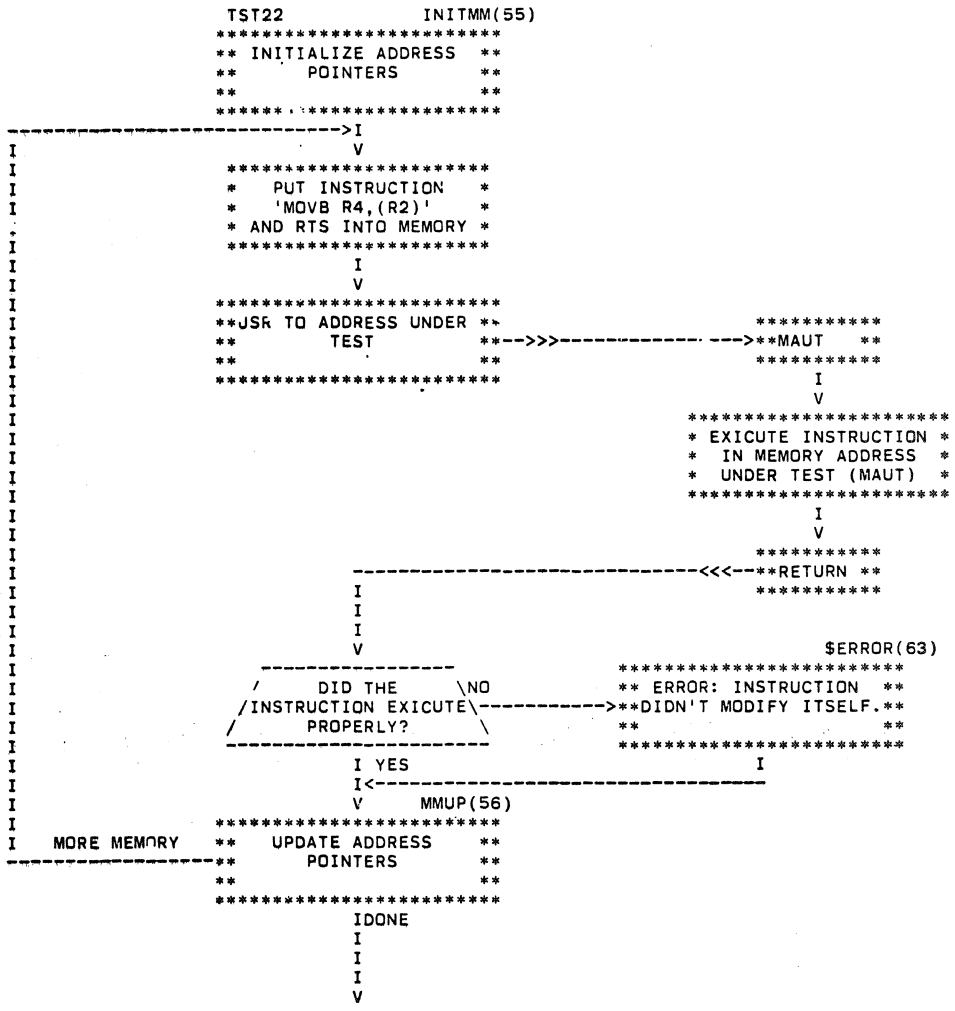

$ERROR(63)
DATA STILL ZERO?
NO
ERROR: ADDRESSING ERROR
YES
WWPB2
WRITE WRONG PARITY INTO BYTE UNDER TEST
WWPB2
READ WRONG PARITY FORCING TRAP
NO TRAP
ERROR: READING WRONG PARITY FAILED TO TRAP
WWPB4(37)
MEMORY PARITY ERROR TRAP
PBTRP
ERROR FLAG SET IN ANY REGISTER?
NO
ERROR: NO REGISTER SHOWN PARITY ERROR.
WWPB4(37)
YES
THIS REGISTER MAPPED FOR THIS ADDRESS?
NO
ERROR: PARITY REGISTER BAD MAPPED.
YES
ALL USED BITS OF REGISTER CORRECT?
NO
ERROR: PARITY REGISTER DATA WRONG.
YES

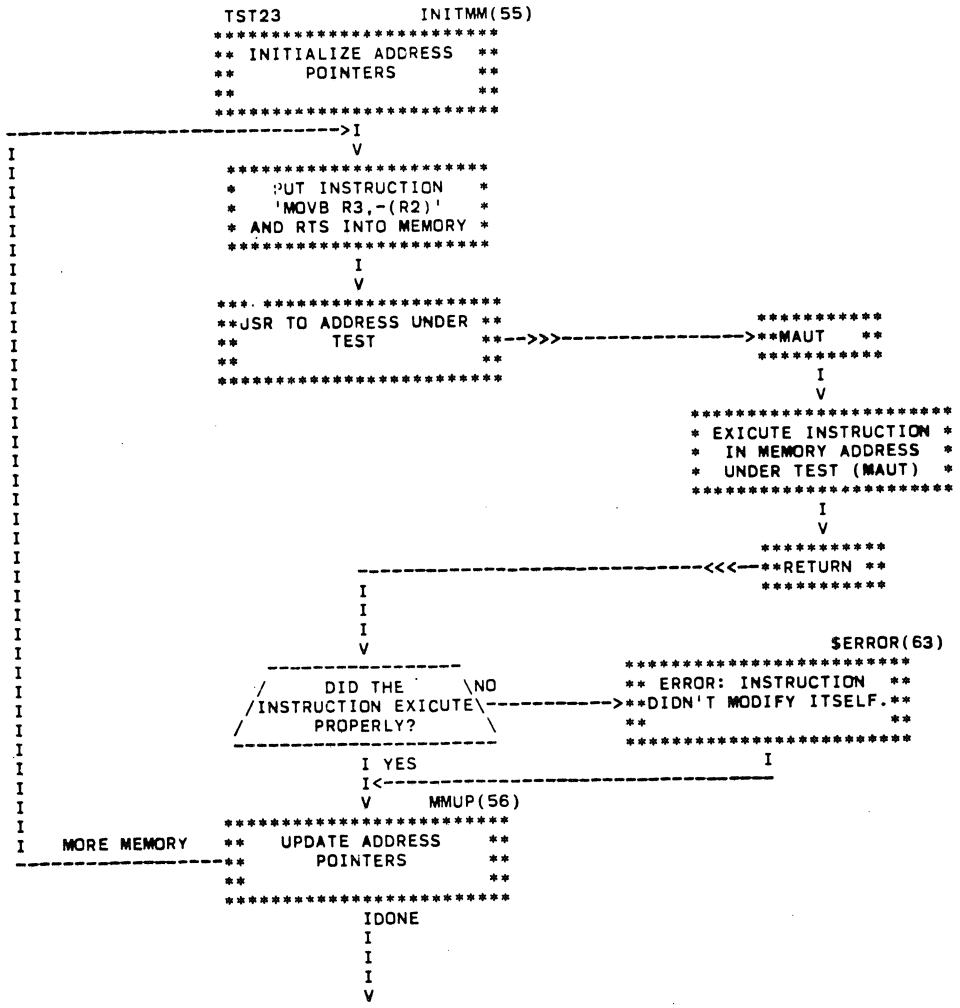

$ERROR(63)
ARE ALL OTHER PARITY REG. ERROR FLAGS CLR?
NO
**ERROR: MORE THAN ONE REG INDICATED ERROR.
YES
WWPB3
SAVE DATA FROM BYTE UNDER TEST
$ERROR(63)
DID SAVING DATA RESET ERROR FLAG
NO
**ERROR: SECOND READ OF WRONG PARITY NOT DETECTED
YES
$ERROR(63)
DID DATA REMAIN UNCHANGED WHEN ERROR TRAPPED?
NO
** ERROR: DATA CHANGED DISPITE PARITY TRAP
YES
RESTORE RIGHT PARITY IN BYTE UNDER TEST
WWPB4
**WWPB4
DONE BOTH PARITY BIT SET AND CLEARED?
NO
*WWPB2(36)
YES
WWPB5
**WWPB5
DONE WITH THIS BANK?
NO
*WWPB1(35)
YES

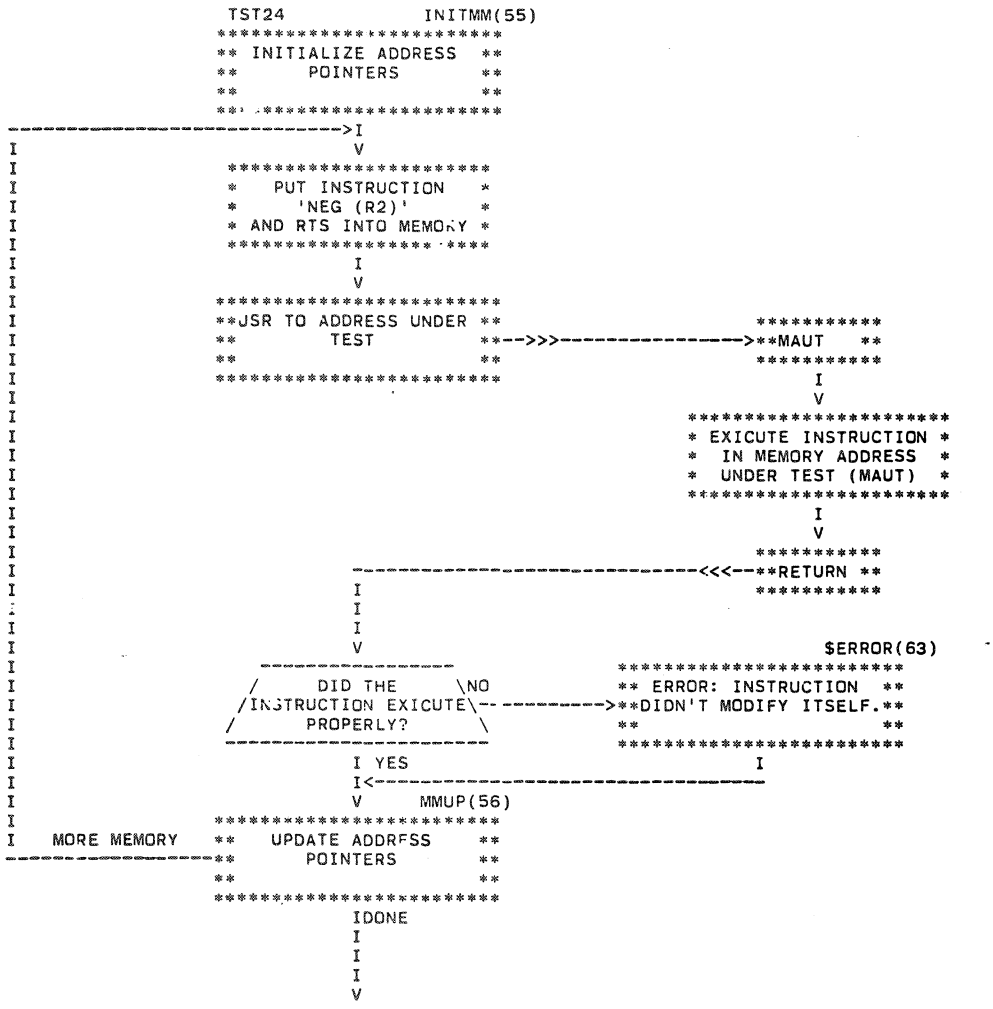

MMUP(56)
UPDATE ADDRESS POINTERS
MORE MEMORY
*WWPBT(35)
IDONE
MAMF(60)
RESET ALL PARITY REGISTERS

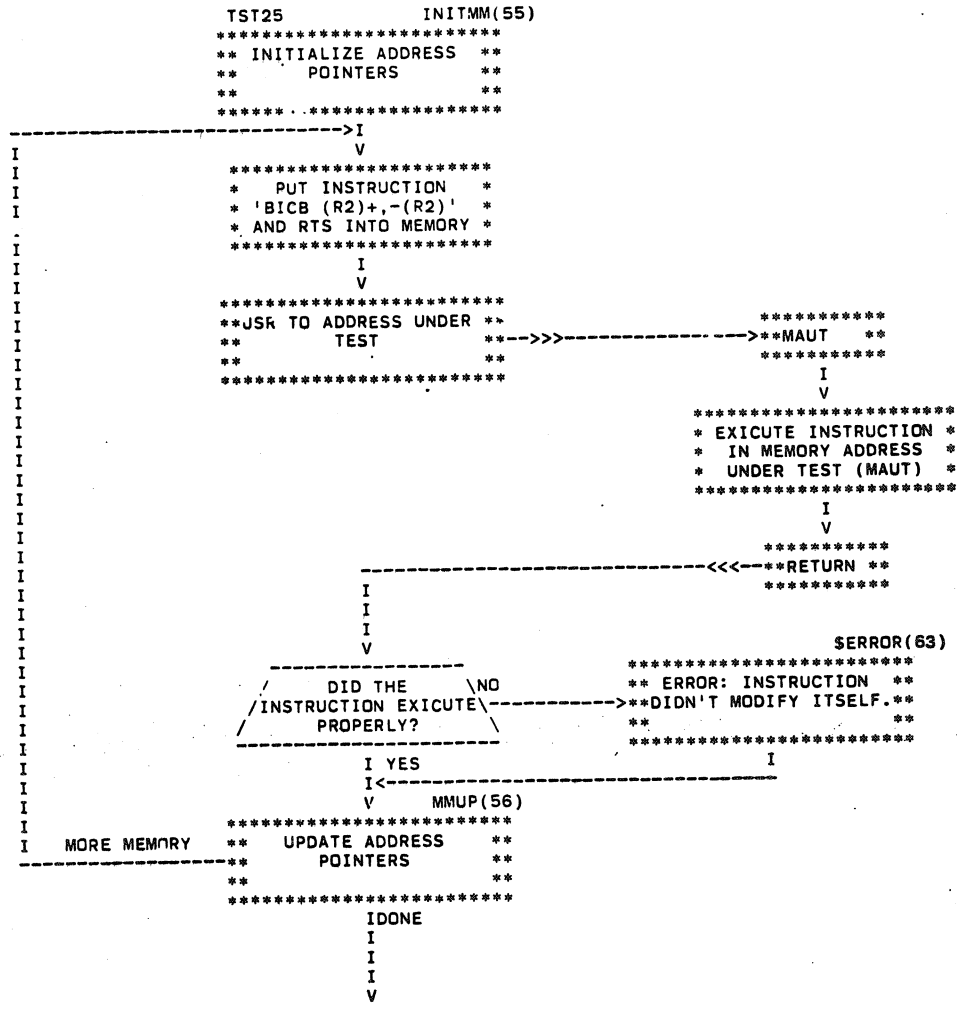

**TST20 **
TST20
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
* COPY 2K BLOCK OF *
* PROGRAM CODE INTO *
* MEMORY UNDER TEST *
/DID "RANDOM" DATA\NO
/ COPY OK? \
\
I YES
MMUP(56)
** UPDATE ADDRESS **
** POINTERS **
MORE MEMORY
IDONE
$ERROR(63)
** ERROR: PROGRAM CODE **
** COPIED CHANGE. **

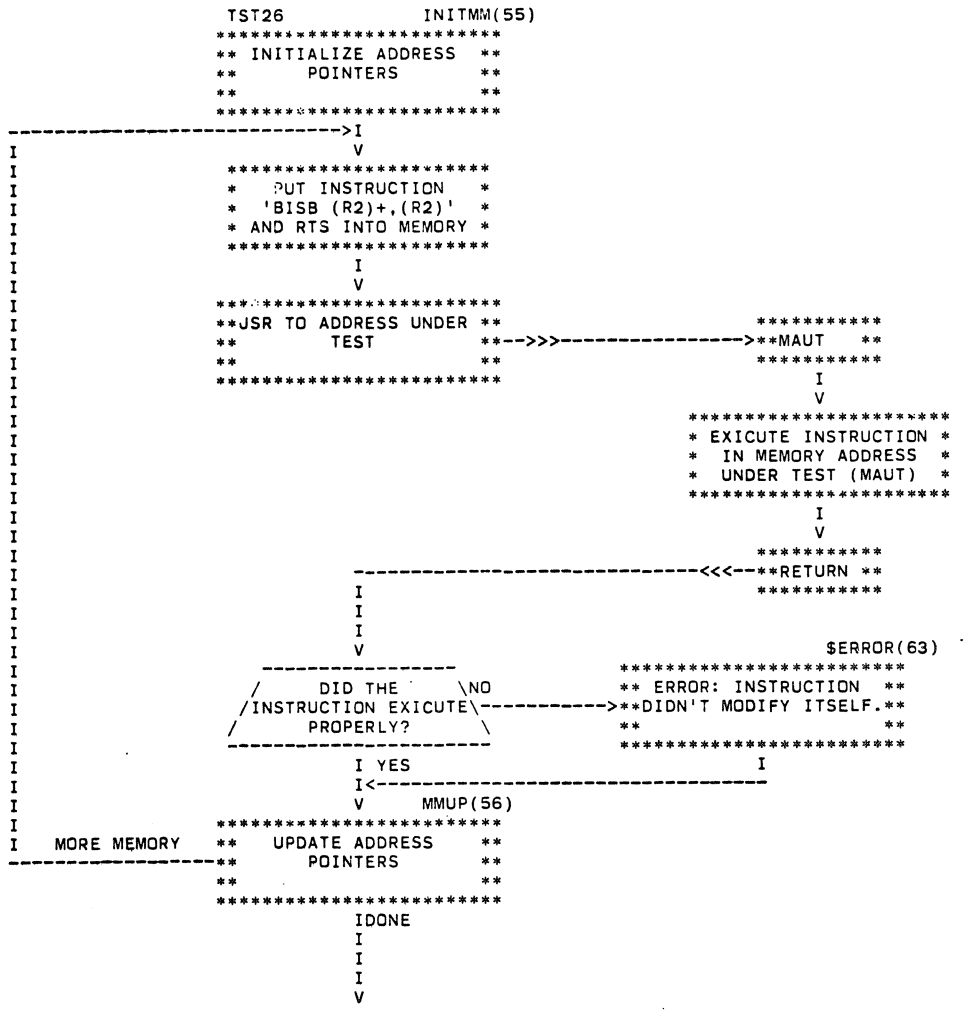

TST21
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'MOV R4,(R2)'
AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
RETURN
$ERROR(63)
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
ERROR: INSTRUCTION
DIDN'T MODIFY ITSELF.
YES
MMUP(56)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

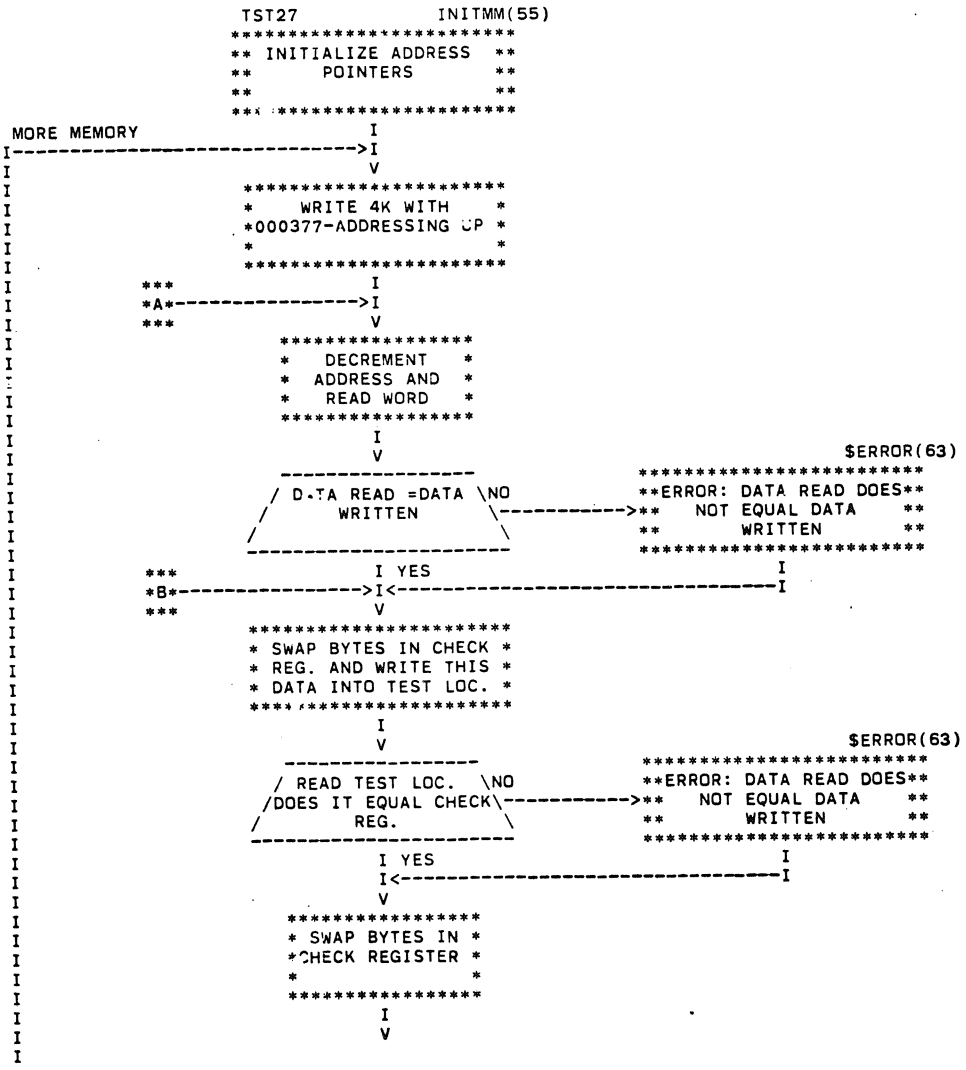

TST22
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'MOVB R4,(R2)'
AND RTS INTO MEMORY
**JSR TO ADDRESS UNDER **
** TEST **
**MAUT **
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
**RETURN **
$ERROR(63)
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
** ERROR: INSTRUCTION **
DIDN'T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

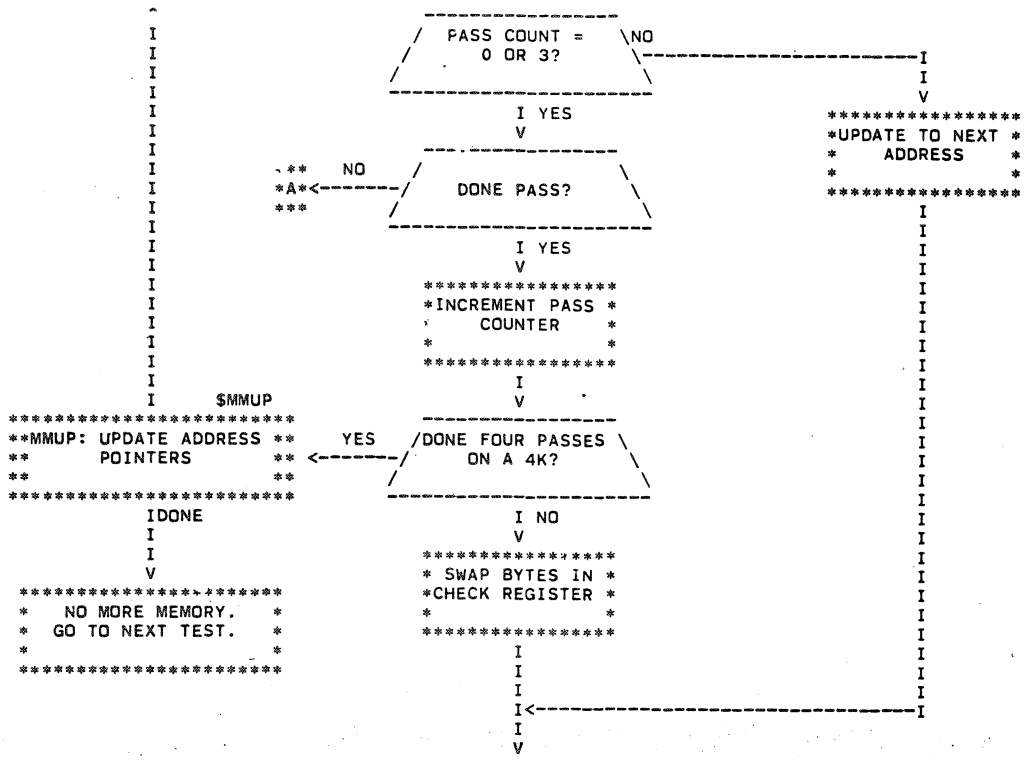

TST23
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'MOVB R3,-(R2)'
AND RTS INTO MEMORY
USR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
YES
MMUP(56)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

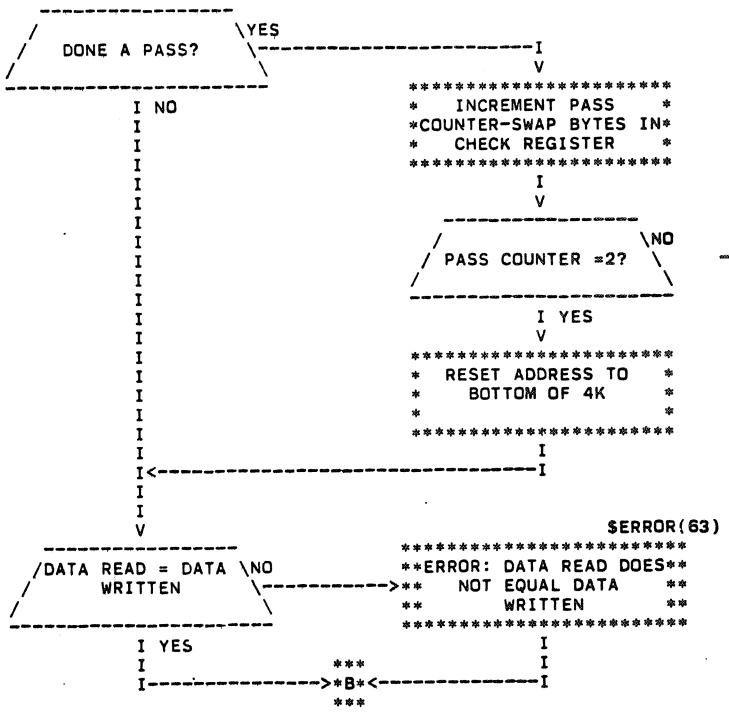

TST24
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'NEG (R2)'
AND RTS INTO MEMORY
**JSR TO ADDRESS UNDER **
** TEST **
**MAUT **
EXICUTE INSTRUCTION
IN MEMORY ADDRESS
UNDER TEST (MAUT)
**RETURN **
$ERROR(63)
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
** ERROR: INSTRUCTION **
DIDN'T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

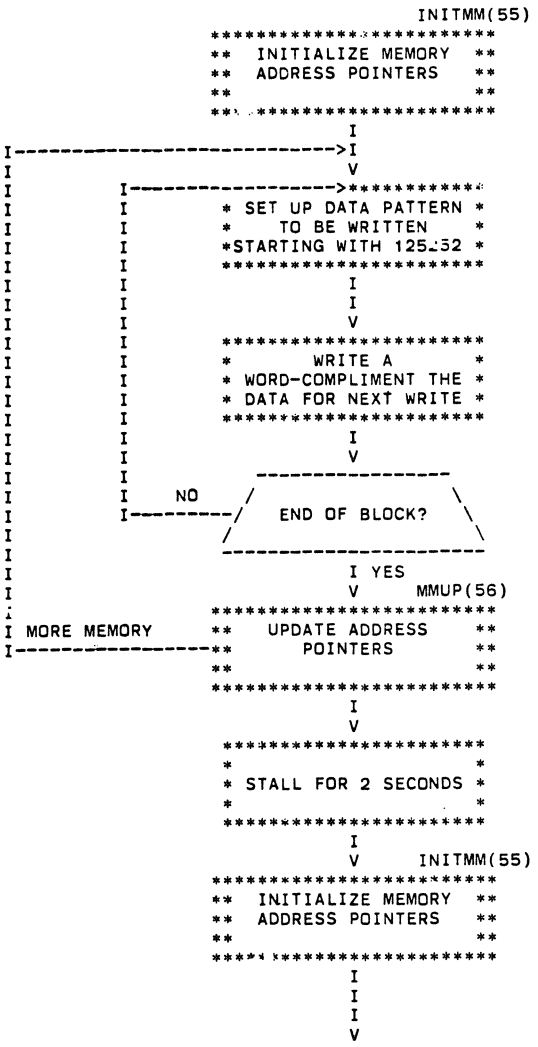

TST25
INITMM(55)
INITIALIZE ADDRESS POINTERS
PUT INSTRUCTION 'BICB (R2)+,-(R2)' AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
MMUP(56)
UPDATE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

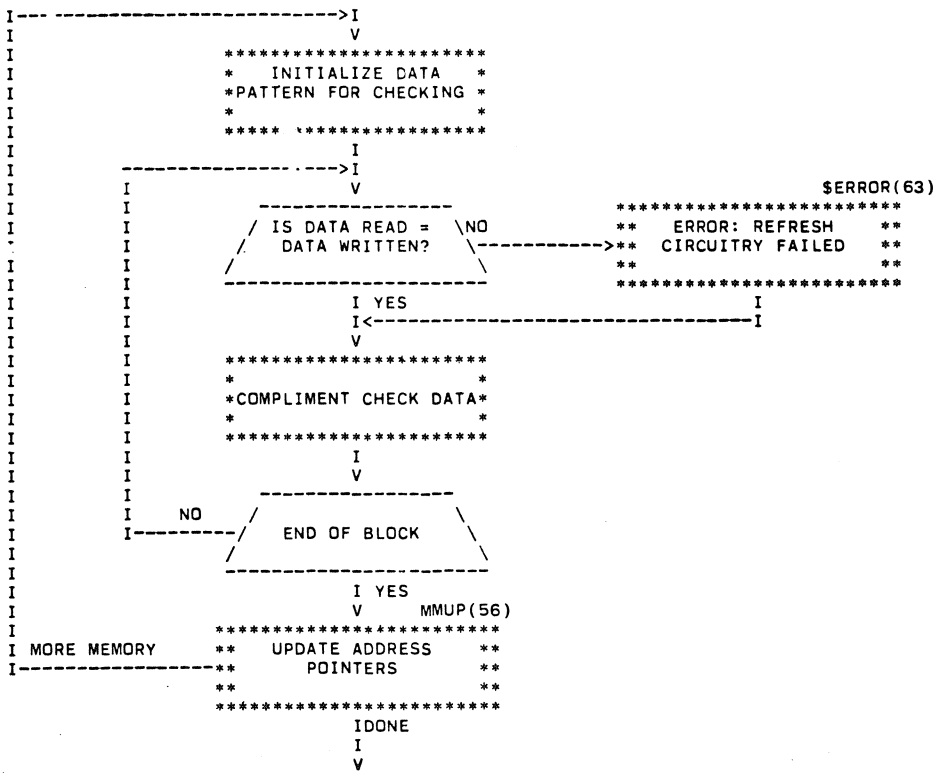

TST26
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
PUT INSTRUCTION
'BISB (R2)+,(R2)'
AND RTS INTO MEMORY
JSR TO ADDRESS UNDER TEST
MAUT
EXICUTE INSTRUCTION IN MEMORY ADDRESS UNDER TEST (MAUT)
RETURN
DID THE INSTRUCTION EXICUTE PROPERLY?
NO
YES
$ERROR(63)
ERROR: INSTRUCTION DIDN'T MODIFY ITSELF.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

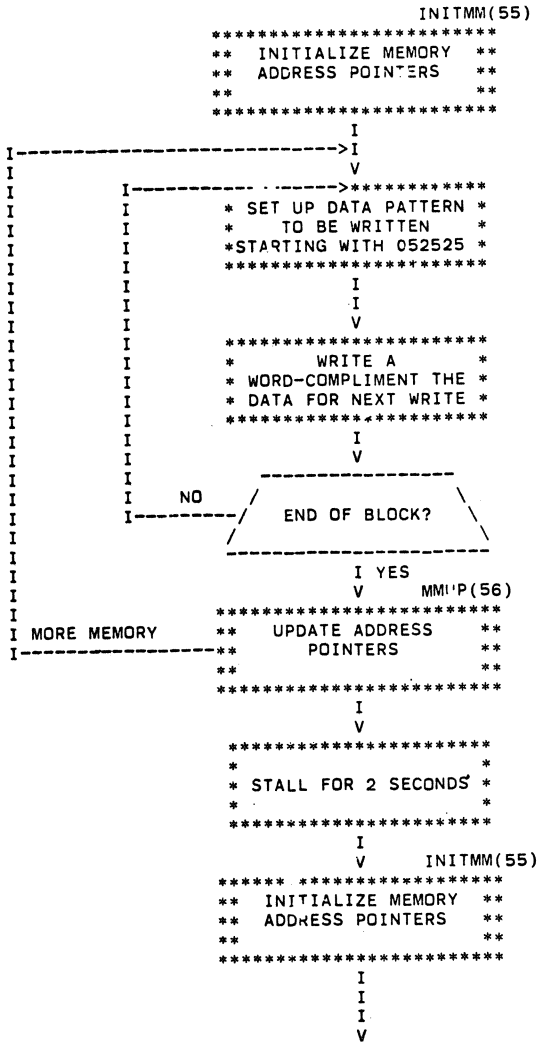

TST27
INITMM(55)
** INITIALIZE ADDRESS **
** POINTERS **
MORE MEMORY
WRITE 4K WITH
000377-ADDRESSING UP
DECREMENT
ADDRESS AND
READ WORD
DATA READ =DATA WRITTEN
NO
YES
$ERROR(63)
**ERROR: DATA READ DOES NOT EQUAL DATA WRITTEN
SWAP BYTES IN CHECK REG. AND WRITE THIS DATA INTO TEST LOC.
READ TEST LOC. DOES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

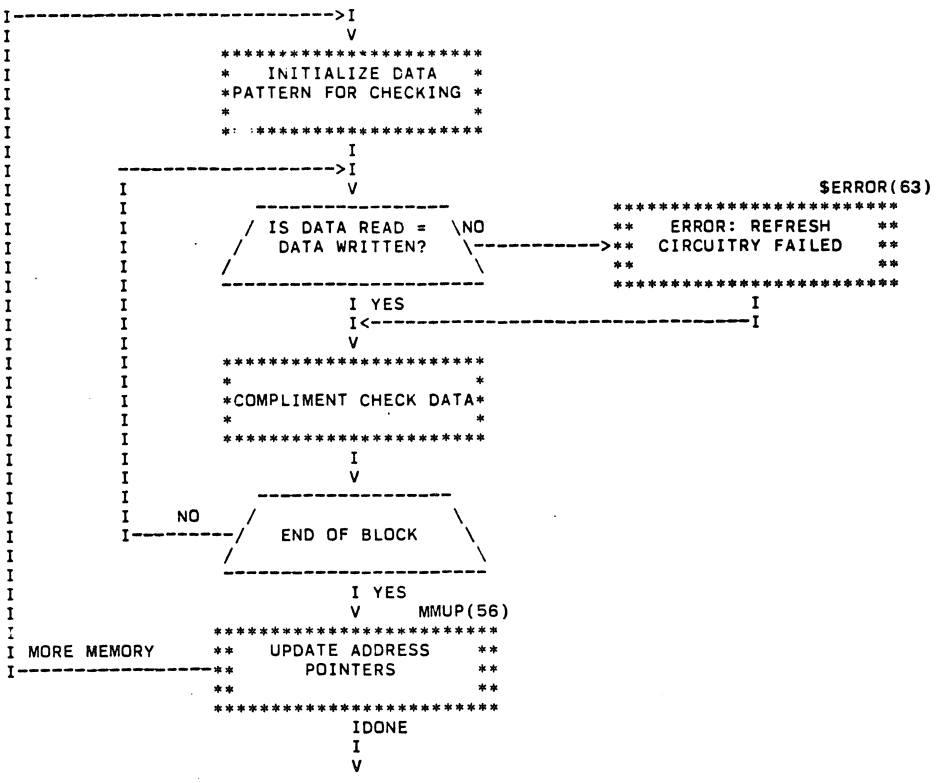

PASS COUNT = 0 OR 3?
NO
YES
DONE PASS?
NO
A
YES
DONE FOUR PASSES ON A 4K?
NO
UPDATE TO NEXT ADDRESS
INCREMENT PASS COUNTER
* SWAP BYTES IN *CHECK REGISTER *
$MMUP
MMUP: UPDATE ADDRESS POINTERS
IDONE
* NO MORE MEMORY. GO TO NEXT TEST. *


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

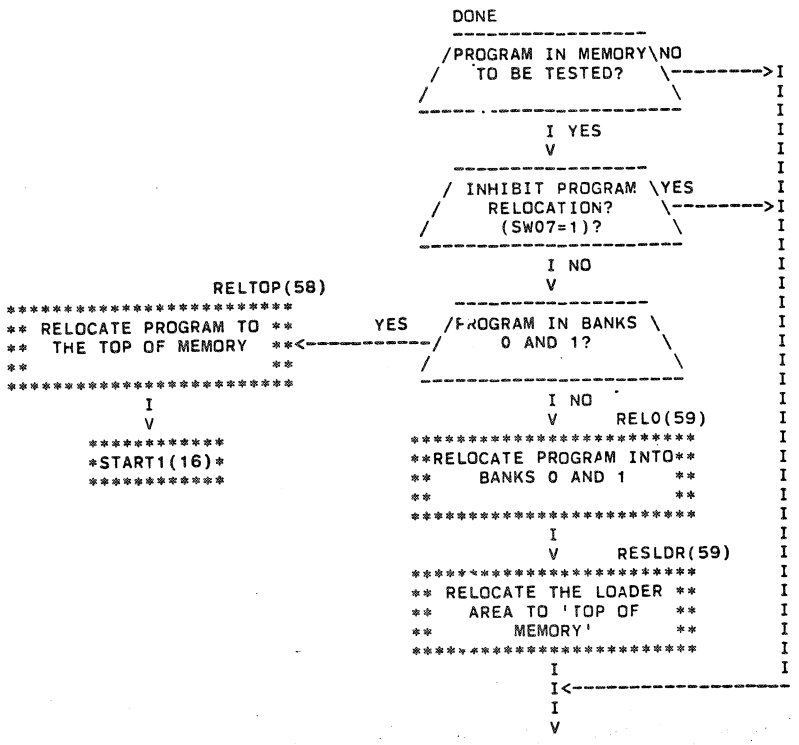

INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS
SET UP DATA PATTERN TO BE WRITTEN STARTING WITH 125252
WRITE A WORD-COMPLIMENT THE DATA FOR NEXT WRITE
NO
END OF BLOCK?
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
STALL FOR 2 SECONDS
INITMM(55)
INITIALIZE MEMORY ADDRESS POINTERS


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


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

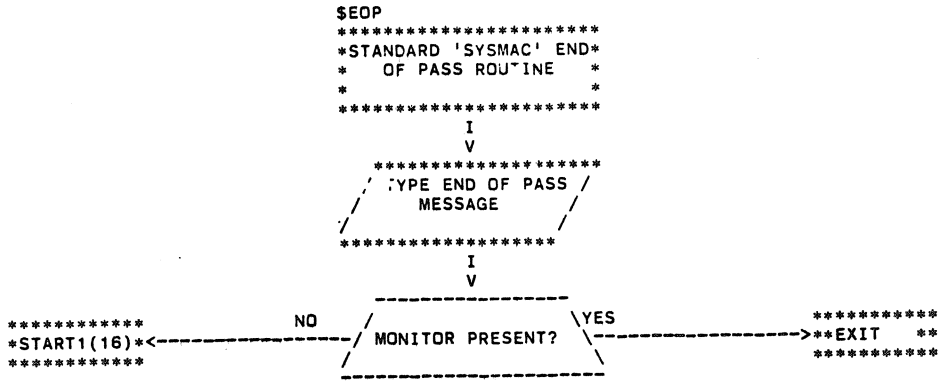

INITIALIZE DATA
PATTERN FOR CHECKING
IS DATA READ = DATA WRITTEN?
NO
YES
$ERROR(63)
ERROR: REFRESH CIRCUITRY FAILED
COMPLIMENT CHECK DATA
NO
END OF BLOCK
YES
MMUP(56)
MORE MEMORY
UPDATE ADDRESS POINTERS
IDONE

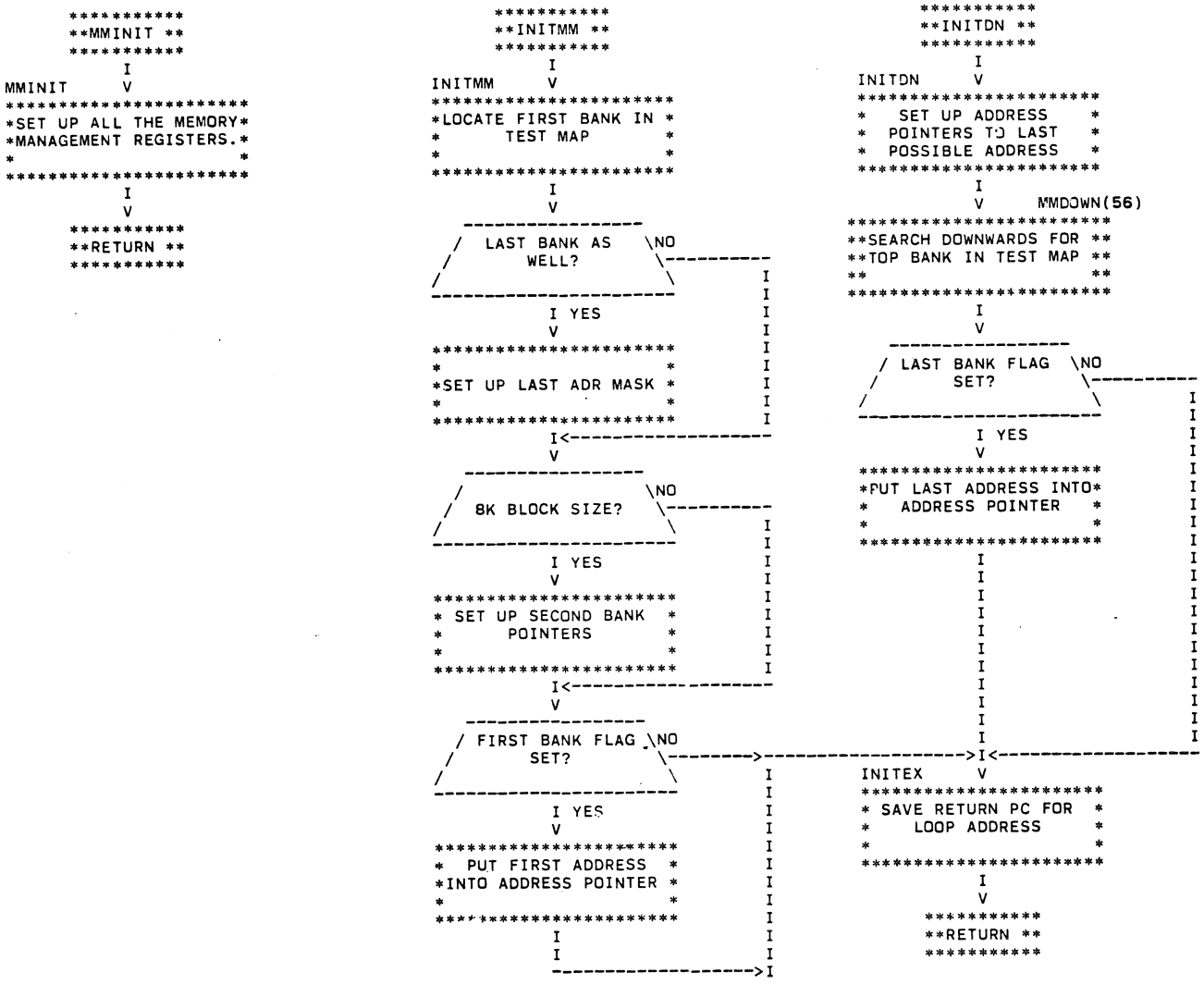
DONE

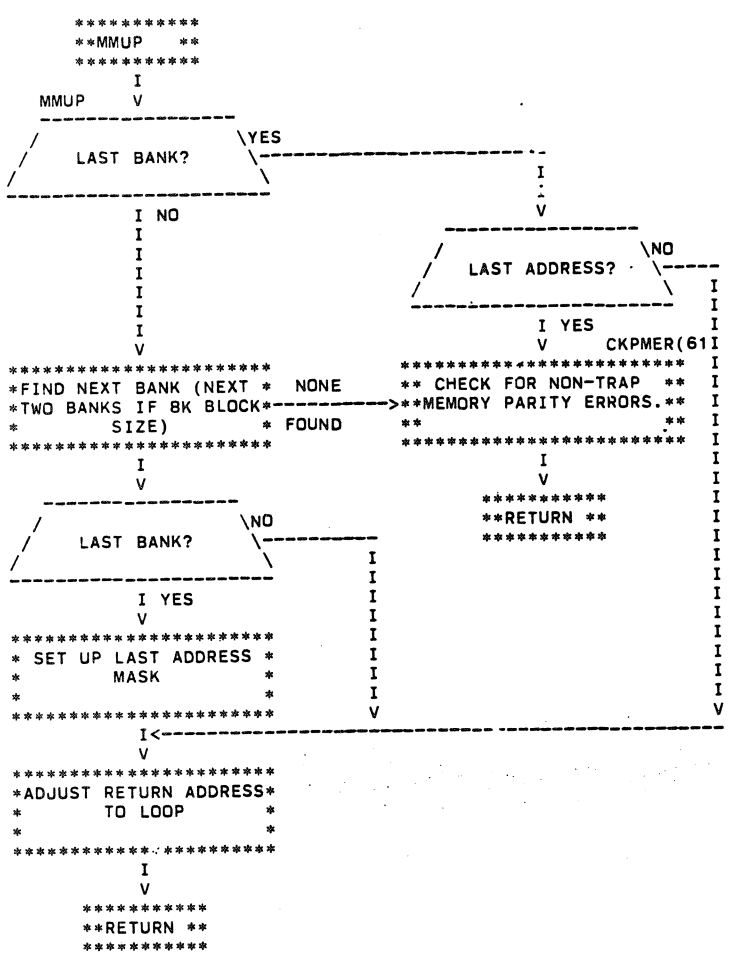

DONE
PROGRAM IN MEMORY TO BE TESTED?
NO
YES
INHIBIT PROGRAM RELOCATION? (SW07=1)?
YES
NO
PROGRAM IN BANKS 0 AND 1?
YES
NO
RELTOP(58)
** RELOCATE PROGRAM TO THE TOP OF MEMORY **
START1(16)
RELO(59)
RELOCATE PROGRAM INTO BANKS 0 AND 1
RESLDR(59)
** RELOCATE THE LOADER AREA TO 'TOP OF MEMORY' **

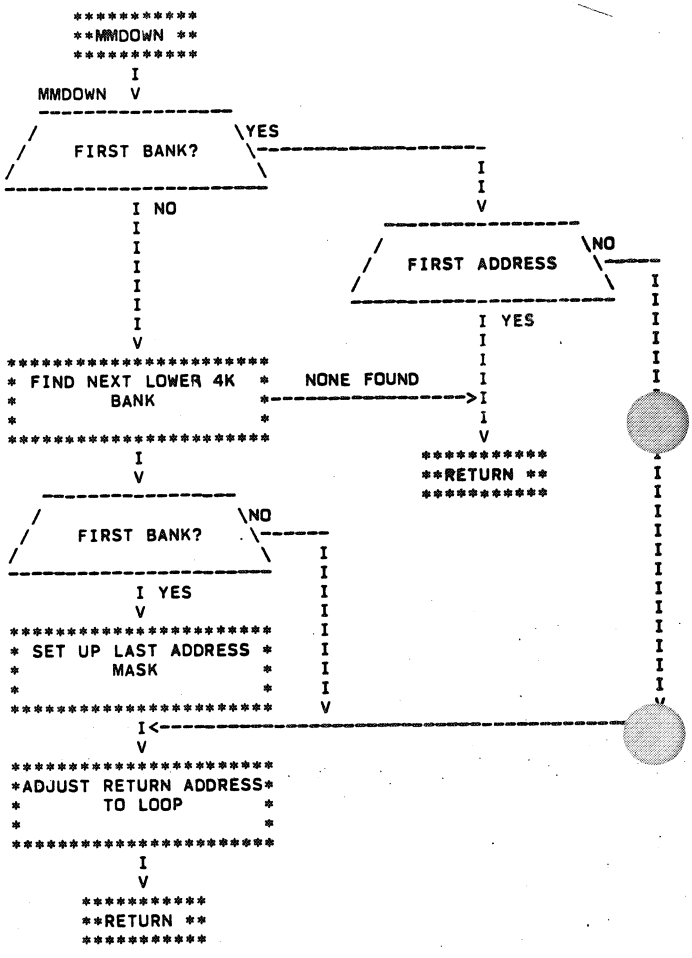
CZQMCF0 0-124K MEM EXER 16K
END OF PASS

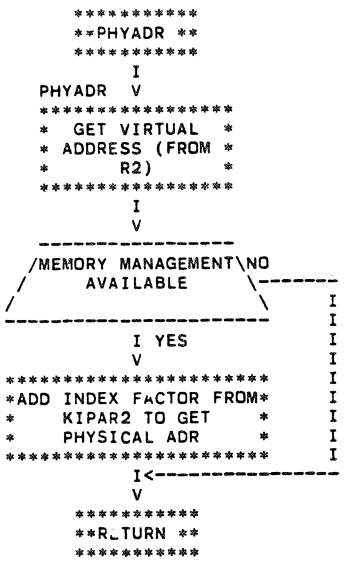
DECFLO VER 00.07 20-FEB-78 07:58

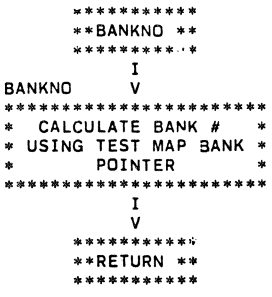


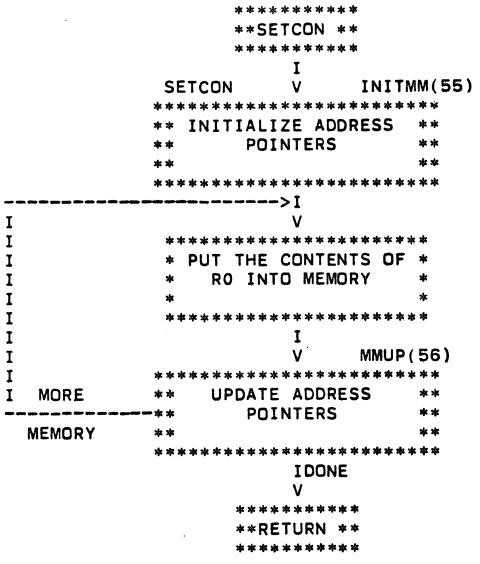

**MMINIT **
MMINIT
SET UP ALL THE MEMORY
MANAGEMENT REGISTERS.
**RETURN **
**INITMM **
INITMM
*LOCATE FIRST BANK IN *
* TEST MAP *
LAST BANK AS WELL?
I YES
*SET UP LAST ADR MASK *
8K BLOCK SIZE?
* SET UP SECOND BANK *
* POINTERS *
FIRST BANK FLAG SET?
*PUT FIRST ADDRESS *
*INTO ADDRESS POINTER *
**INITDN **
INITDN
* SET UP ADDRESS *
* POINTERS TO LAST *
* POSSIBLE ADDRESS *
MMDOWN(56)
**SEARCH DOWNWARDS FOR **
**TOP BANK IN TEST MAP **
LAST BANK FLAG SET?
PUT LAST ADDRESS INTO
* ADDRESS POINTER *
INITEX
* SAVE RETURN PC FOR *
* LOOP ADDRESS *
**RETURN **

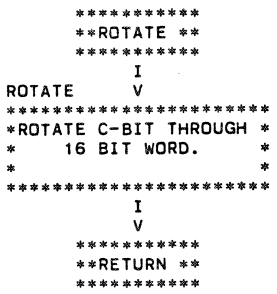

**MMUP **
MMUP
LAST BANK?
YES
NO
LAST ADDRESS?
NO
YES
CKPMER(61
*FIND NEXT BANK (NEXT *
TWO BANKS IF 8K BLOCK
* SIZE) *
NONE
FOUND
** CHECK FOR NON-TRAP **
MEMORY PARITY ERRORS.
**RETURN **
LAST BANK?
NO
YES
* SET UP LAST ADDRESS *
* MASK *
ADJUST RETURN ADDRESS
* TO LOOP *
**RETURN **

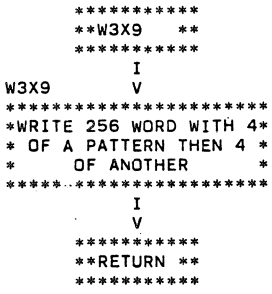

**MMDOWN **
MMDOWN
FIRST BANK?
YES
NO
FIRST ADDRESS
NO
YES
* FIND NEXT LOWER 4K *
* BANK *
NONE FOUND
**RETURN **
FIRST BANK?
NO
YES
* SET UP LAST ADDRESS *
* MASK *
ADJUST RETURN ADDRESS
* TO LOOP *
**RETURN **

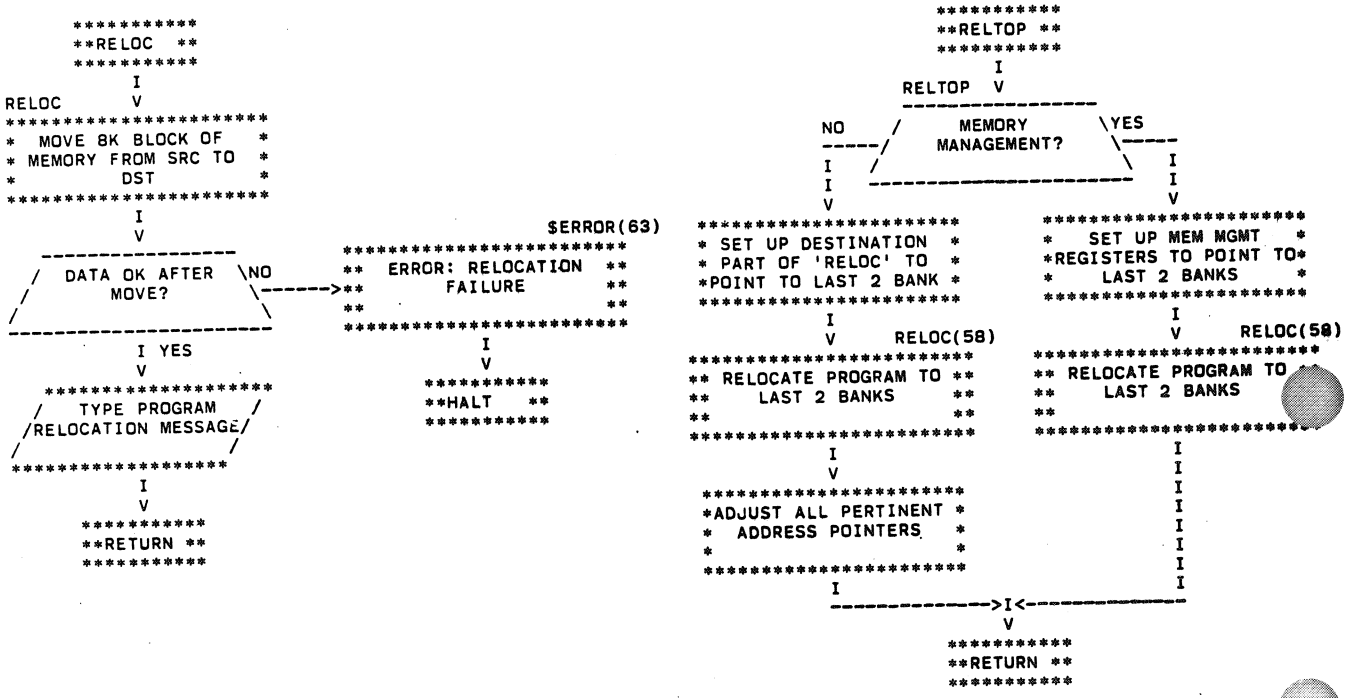

**PHYADR **
PHYADR
GET VIRTUAL ADDRESS (FROM R2)
MEMORY MANAGEMENT AVAILABLE
NO
YES
ADD INDEX FACTOR FROM KIPAR2 TO GET PHYSICAL ADR
**RETURN **

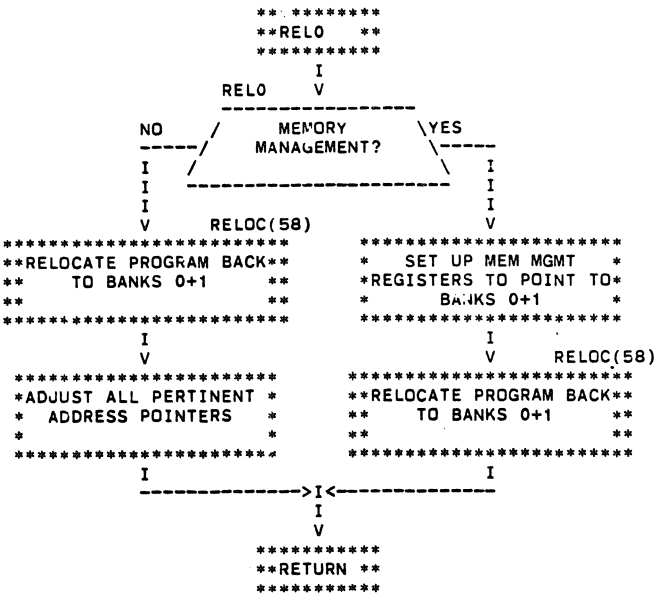

**BANKNO **
BANKNO
CALCULATE BANK # USING TEST MAP BANK POINTER
**RETURN **

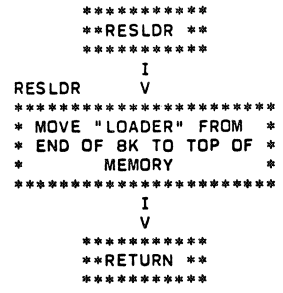

**SETCON **
SETCON
INITMM(55)
** INITIALIZE ADDRESS POINTERS **
PUT THE CONTENTS OF R0 INTO MEMORY
MMUP(56)
MORE MEMORY
** UPDATE ADDRESS POINTERS **
IDONE
**RETURN **

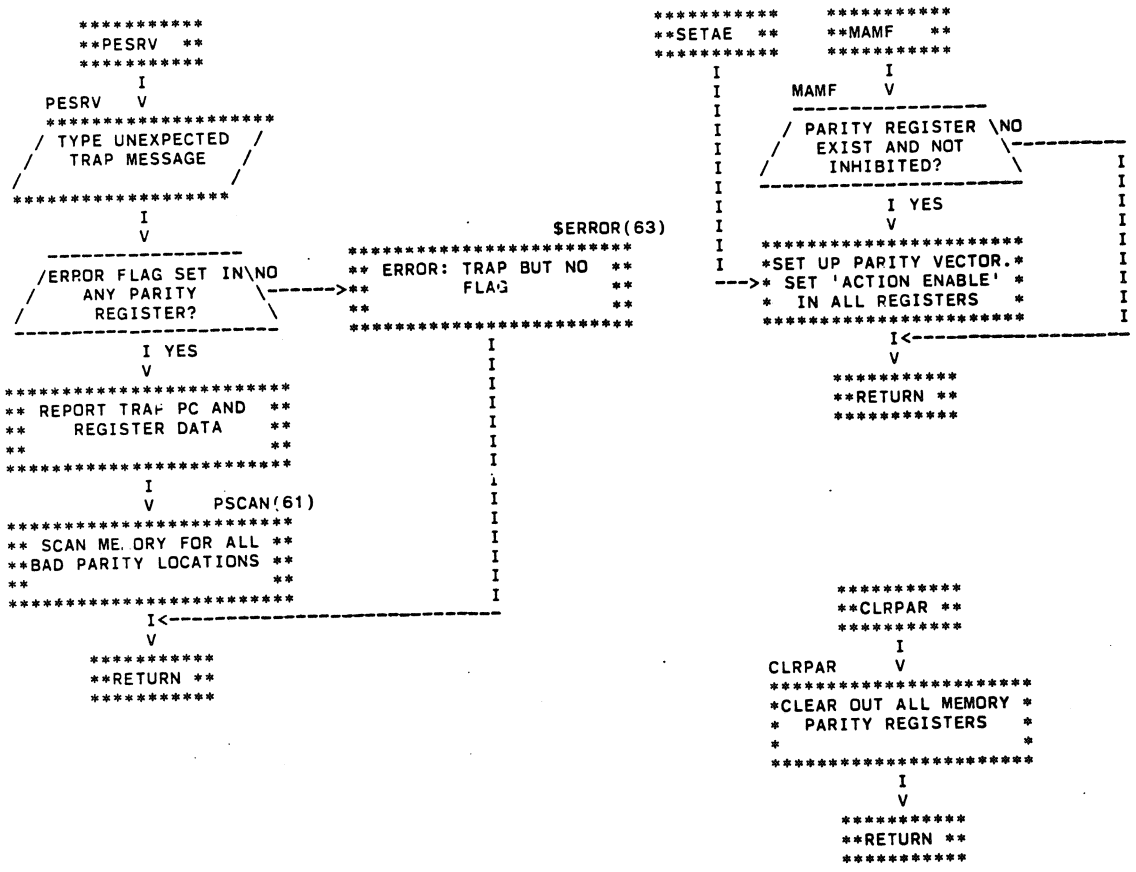

**ROTATE **
ROTATE
*ROTATE C-BIT THROUGH 16 BIT WORD.
**RETURN **

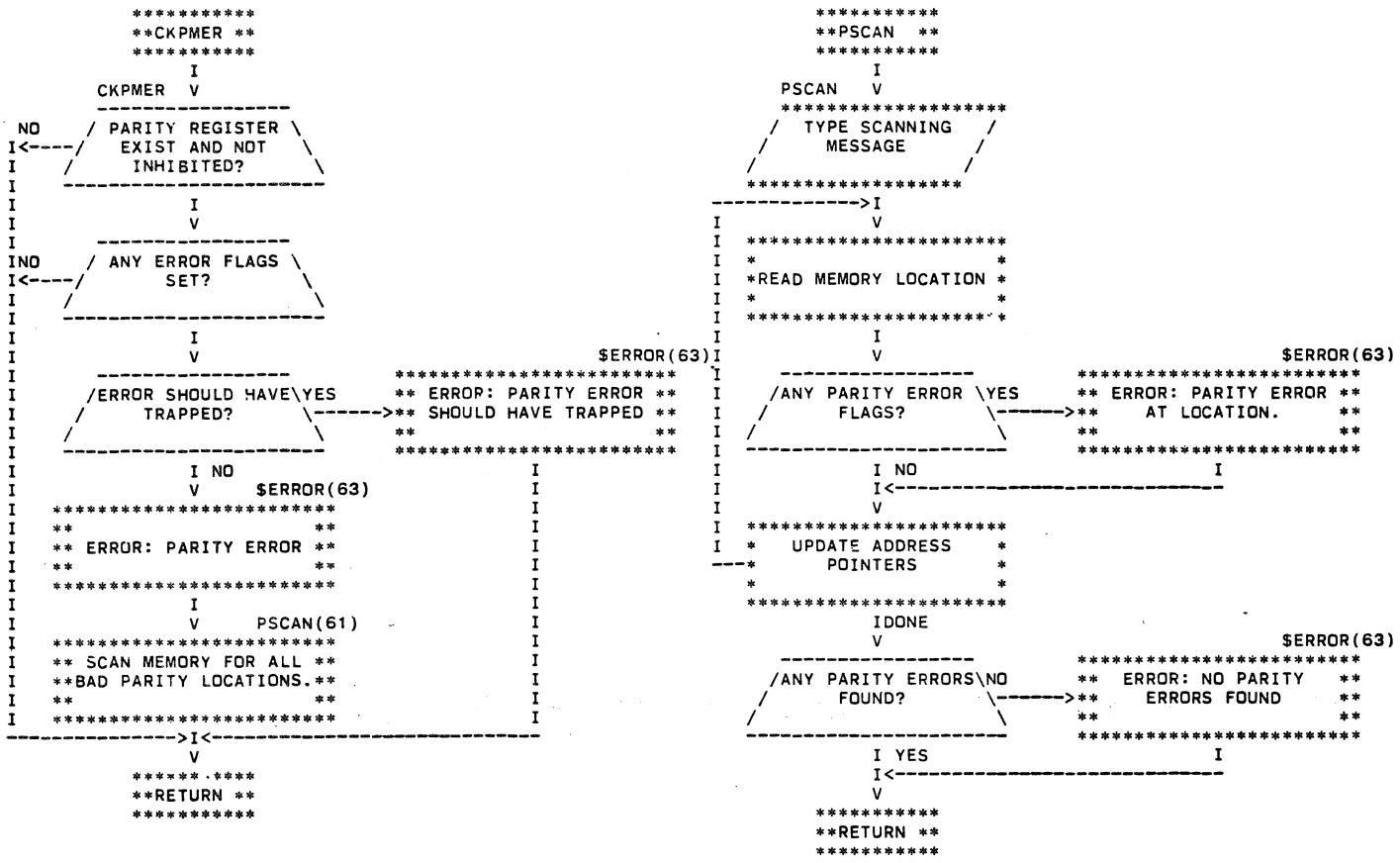

**W3X9 **
W3X9
WRITE 256 WORD WITH 4 OF A PATTERN THEN 4 * OF ANOTHER
**RETURN **

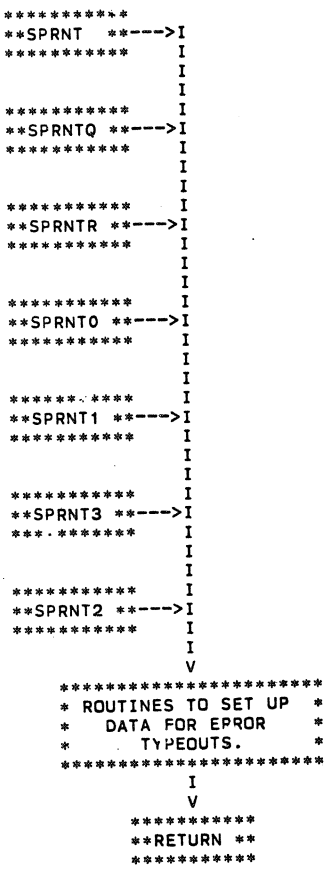

**RELOC **
RELOC
MOVE 8K BLOCK OF MEMORY FROM SRC TO DST
DATA OK AFTER MOVE?
NO
$ERROR(63)
ERROR: RELOCATION FAILURE
**HALT **
I YES
TYPE PROGRAM /RELOCATION MESSAGE/
**RETURN **
**RELTOP **
RELTOP
MEMORY MANAGEMENT?
NO
YES
SET UP DESTINATION PART OF 'RELOC' TO POINT TO LAST 2 BANK
RELOC(58)
RELOCATE PROGRAM TO LAST 2 BANKS
ADJUST ALL PERTINENT ADDRESS POINTERS
SET UP MEM MGMT REGISTERS TO POINT TO LAST 2 BANKS
RELOC(58)
RELOCATE PROGRAM TO LAST 2 BANKS
**RETURN **

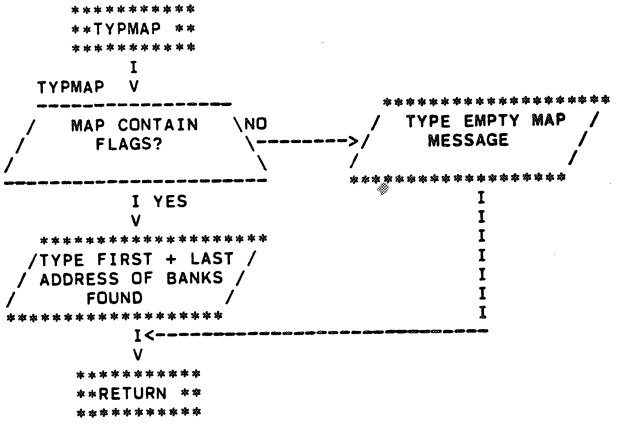

**RELO **
RELO
MEMORY MANAGEMENT?
NO
YES
RELOC(58)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
*ADJUST ALL PERTINENT *
* ADDRESS POINTERS *
* SET UP MEM MGMT *
REGISTERS TO POINT TO
* BANKS 0+1 *
RELOC(58)
RELOCATE PROGRAM BACK
** TO BANKS 0+1 **
**RETURN **

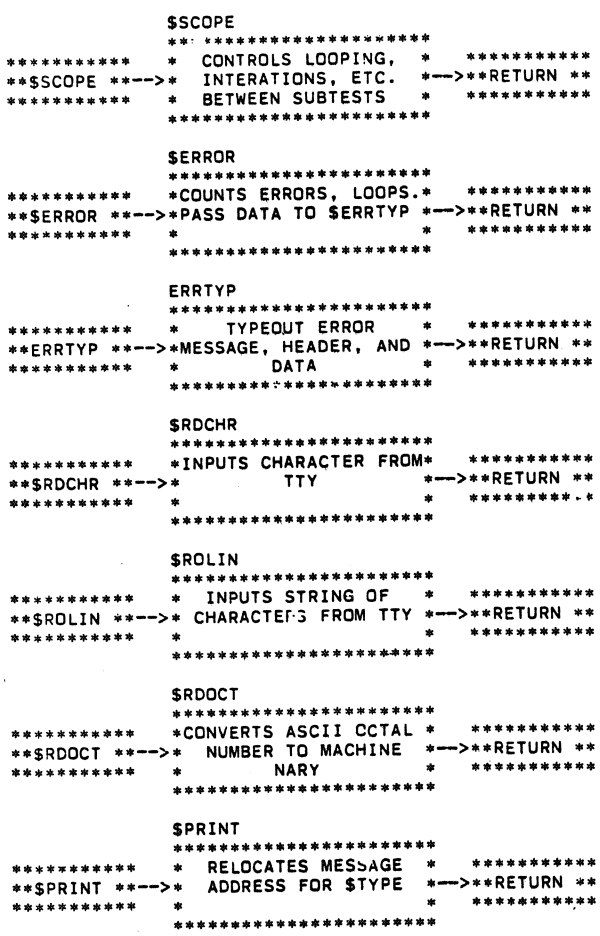

**RESLDR **
RESLDR
* MOVE "LOADER" FROM *
* END OF 8K TO TOP OF *
* MEMORY *
**RETURN **

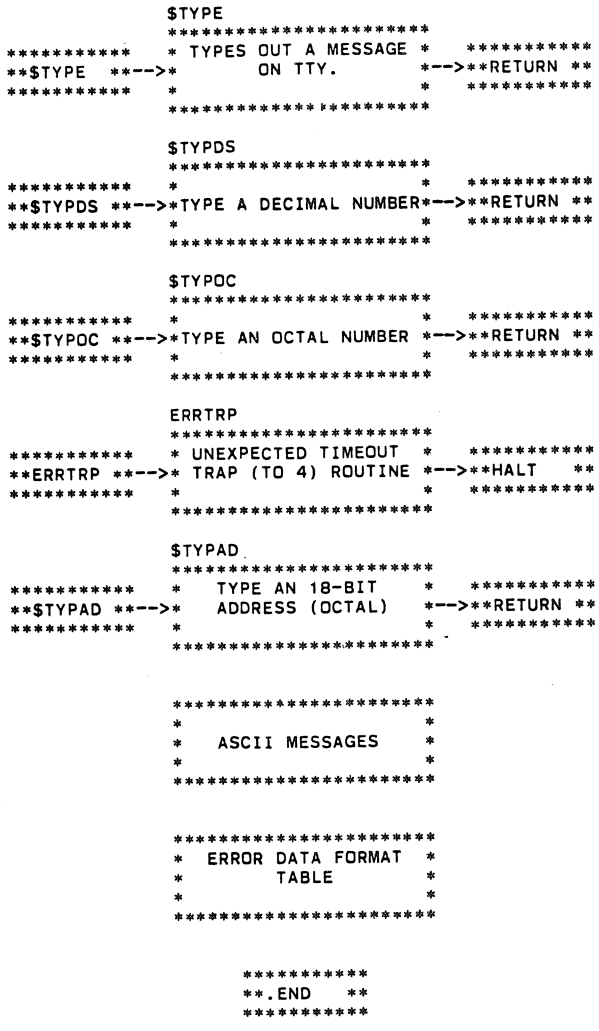

**PESRV **
PESRV
TYPE UNEXPECTED TRAP MESSAGE
ERROR FLAG SET IN ANY PARITY REGISTER?
NO
YES
$ERROR(63)
** ERROR: TRAP BUT NO FLAG **
** REPORT TRAP PC AND REGISTER DATA **
PSCAN(61)
** SCAN MEMORY FOR ALL BAD PARITY LOCATIONS **
**RETURN **
**SETAE **
**MAMF **
MAMF
PARITY REGISTER EXIST AND NOT INHIBITED?
NO
YES
SET UP PARITY VECTOR.
* SET 'ACTION ENABLE' *
* IN ALL REGISTERS *
**RETURN **
**CLRPAR **
CLRPAR
*CLEAR OUT ALL MEMORY *
* PARITY REGISTERS *
**RETURN **


**CKPMER **
CKPMER
PARITY REGISTER EXIST AND NOT INHIBITED?
NO
ANY ERROR FLAGS SET?
NO
ERROR SHOULD HAVE TRAPPED?
YES
$ERROR(63)
** ERROR: PARITY ERROR **
$ERROR(63)
** ERROR: PARITY ERROR **
** SHOULD HAVE TRAPPED **
NO
PSCAN(61)
** SCAN MEMORY FOR ALL **
BAD PARITY LOCATIONS.
**RETURN **
**PSCAN **
PSCAN
TYPE SCANNING MESSAGE
*READ MEMORY LOCATION *
ANY PARITY ERROR FLAGS?
YES
$ERROR(63)
** ERROR: PARITY ERROR **
AT LOCATION.
NO
UPDATE ADDRESS POINTERS
IDONE
ANY PARITY ERRORS FOUND?
NO
$ERROR(63)
** ERROR: NO PARITY ERRORS FOUND
YES
**RETURN **


**SPRNT
**SPRNTQ
**SPRNTR
**SPRNT0
**SPRNT1
**SPRNT3
**SPRNT2
* ROUTINES TO SET UP *
* DATA FOR EPROR *
* TYPEOUTS. *
**RETURN **


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


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


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

BANKNO	57#	57															
CKPMER	24	35	56	61#	61												
CLRPAR	60#	60															
CTRLS	14																
DONE	53																
ERRTRP	63	63#															
ERRTYP	63	63#															
EXIT	54																
GMPR	08	12	12														
GMPRA	12																
GMPRB	12																
GMPRC	12																
HALT	03	04	04	08	11	58	63										
INITDN	17	18	19	21	21	55#	55										
INITEX	55																
INITMM	17	18	19	20	20	22	23	24	25	26	27	27	28	29	29	30	31	31
	32	33	33	34	35	39	40	41	42	43	44	45	46	49	49	51	51	55#
	55	57																
MAMF	38	60#	60															
MANUAL	12	15#	15															
MANUL1	15																	
MANUL2	16																	
MAPRB	13																	
MAUT	40	41	42	43	44	45												
MMDOWN	17	18	19	21	21	55	56#	56										
MMINIT	09	55#	55															
MMUP	17	18	19	20	20	22	23	24	25	26	27	27	28	29	29	30	31	31
	32	33	33	34	38	39	40	41	42	43	44	45	49	50	51	52	56#	56
	57																	
PBTRP	36																	
PESRV	60#	60																
PHYADR	57#	57																
PSCAN	60	61	61#	61														
REL0	03	53	59#	59														
RELOC	58#	58	58	58	59	59												
RELTOP	53	58#	58															
RESCHK	14																	
RESLDR	03	53	59#	59														
REST1	02																	
REST2	02																	
RESTAR	02#	02	06															
RESTOR	02																	
RETURN	04	40	41	42	43	44	45	55	55	56	56	56	56	57	57	57	57	57
	58	58	59	59	60	60	60	61	61	62	62	63	63	63	63	63	63	63
	63	63	63	63														
ROTATE	25	26	57#	57														
SELECT	06#																	
SETAE	35	60#																
SETCON	25	26	35	57#	57													
SPRNT	62#																	
SPRNT0	62#																	
SPRNT1	62#																	

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

```
                63#
$SCOPE          63      63#
$TYPAD          10      10      63      63#
$TYPDS          63      63#
$TYPE           63      63#
$TYPOC          63      63#
.END            63
```

```
                .TITLE  CZQMCF0  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-DZQAC-C3), JAN 19, 1977.
                ;*

                                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                                        ;*
                                        ;*      SWITCH                  USE
                                        ;*      ------  --------------------
                                        ;*        15            HALT ON ERROR
                                        ;*        14            LOOP ON TEST
                                        ;*        13            INHIBIT ERROR TYPEOUTS
                                        ;*        12            INHIBIT KT11 (AT START TIME ONLY)
                                        ;*        11            INHIBIT ITERATIONS
                                        ;*        10            BELL ON ERROR
                                        ;*         9            LOOP ON ERROR
                                        ;*         8            LOOP ON TEST IN SWR<4:0>
                                        ;*         7            INHIBIT PROGRAM RELOCATION
                                        ;*         6            INHIBIT PARITY ERROR DETECTION
                                        ;*         5            INHIBIT EXRERCISING VECTOR AREA.
                                        .SBTTL  BASIC DEFINITIONS

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

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

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



```
  000007                                PC=     %7              ;;PROGRAM COUNTER

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

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

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

```
                        000001          BIT00=  1
                                        .EQUIV  BIT09,BIT9
                                        .EQUIV  BIT08,BIT8
                                        .EQUIV  BIT07,BIT7
                                        .EQUIV  BIT06,BIT6
                                        .EQUIV  BIT05,BIT5
                                        .EQUIV  BIT04,BIT4
                                        .EQUIV  BIT03,BIT3
                                        .EQUIV  BIT02,BIT2
                                        .EQUIV  BIT01,BIT1
                                        .EQUIV  BIT00,BIT0

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


                                        .SBTTL  MEMORY MANAGEMENT DEFINITIONS

                                        ;*KT11 VECTOR ADDRESS

                        000250          MMVEC=  250

                                        ;*KT11 STATUS REGISTER ADDRESSES

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

                                        ;*KERNEL "I" PAGE DESCRIPTOR REGISTERS

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

                                        ;*KERNEL "I" PAGE ADDRESS REGISTERS

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



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

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

                                        ;*  PARITY MEMORY DEFINITIONS.
                        000001          AE=1                            ;PARITY ACTION ENABLE
                        000114          PARVEC=114                      ;PARITY TRAP VECTOR

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

                                        ;* CACHE REGISTER DEFINITIONS.
                        177746          IMPCHE= 177746

                                        .SBTTL  TRAP CATCHER

                        000000                  .=0
                                        ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
                                        ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                        ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                        000174                  .=174
  000174  000000        DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
  000176  000000        SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
                                        .SBTTL  STARTING ADDRESS(ES)
  000200  000137  002640                JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
  000204  000167  002436                JMP     SELECT          ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
                                                                ;SELECT VARIOUS PARAMETERS.
  000210  000167  000064                JMP     RESTAR          ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
  000214  000167  000064                JMP     RESTOR          ;RESTORE LOADERS TO END OF MEMORY AND HALT.
  000220  000167  003376                JMP     TIMOUT          ;TYPE OUT MEMORY MAP, BYTE BY BYTE.

                        000004                  .=ERRVEC
  000004  025114                        .WORD   ERRTRP
  000006  000000                        .WORD   0

                                        .SBTTL  ACT11 HOOKS

                                        ;;*************************************************************
                                        ;HOOKS REQUIRED BY ACT11
                        000010                  $SVPC=.                 ;SAVE PC
                        000046                  .=46
  000046  014222                        $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
                        000052                  .=52
  000052  040000                        .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
                        000010                  .=$SVPC         ;; RESTORE PC
```

```
                000300                          .=300
                                        ;;*****************************************************************
                                        ;*  THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
                                        ;*   THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
                                        ;*   THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
                                        ;*   MANAGEMENT IS DISABLED BY "CONSOLE START".
                                        ;;*****************************************************************
000300  005005                  RESTAR: CLR     R5              ;CLEAR FLAG TO INDICATE RESTART.
000302  000401                          BR      REST1           ;GO RESTORE PROGRAM BEFORE RESTARTING.
000304  010705                  RESTOR: MOV     PC,     R5      ;PUT DATA INTO FLAG FOR RESTORE.
000306  012706  001100          REST1:  MOV     #STACK, SP      ;SET UP THE STACK POINTER.
000312  005767  001206                  TST     MEMMAP          ;CHECK IF THE MEMORY HAS BEEN MAPPED.
000316  001002                          BNE     REST2           ;BR IF MEMORY MAPPED.
000320  000167  002330                  JMP     START4          ;GO START
000324  005767  000256          REST2:  TST     MMAVA           ;CHECK IF MEM MGMT AVAILABLE.
000330  001470                          BEQ     10$             ;BR IF NO MEM MGMT.
000332  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ACTIVE.
000340  001034                          BNE     2$              ;BR IF MEM MGMT ALREADY SET UP.
000342  012700  172300                  MOV     #KIPDR0,R0      ;POINT TO FIRST MEM MGMT DDATA REG.
000346  012701  000010                  MOV     #8.,    R1      ;SET UP COUNTER.
000352  012720  077406          1$:     MOV     #077406,(R0)+   ;MAP FIRST 28K 1-FOR-1.
000356  005301                          DEC     R1              ;COUNT REGESTERS.
000360  001374                          BNE     1$              ;BR IF MORE REG.
000362  012700  172340                  MOV     #KIPAR0,R0      ;POINT TO FIRST MEM MGMT ADDRESS REG.
000366  005020                          CLR     (R0)+           ;PAR0 MAPPED INTO BANK0.
000370  012720  000200                  MOV     #200,   (R0)+   ;PAR1 MAPPED INTO BANK1.
000374  012720  000400                  MOV     #400,   (R0)+   ;PAR2 MAPPED INTO BANK2.
000400  012720  000600                  MOV     #600,   (R0)+   ;PAR3 MAPPED INTO BANK3.
000404  012720  001000                  MOV     #1000,  (R0)+   ;PAR4 MAPPED INTO BANK4.
000410  012720  001200                  MOV     #1200,  (R0)+   ;PAR5 MAPPED INTO BANK5.
000414  012720  001400                  MOV     #1400,  (R0)+   ;PAR6 MAPPED INTO BANK6.
000420  012720  007600                  MOV     #7600,  (R0)+   ;PAR7 MAPPED INTO BANK37.
000424  012737  000001  177572          MOV     #BIT0,  @#SR0   ;ENABLE MEM MGMT.
000432  005000                  2$:     CLR     R0              ;INIT TEMP PAR REG.
000434  016701  000142                  MOV     PRGMAP, R1      ;GET THE PROGRAM MAP...LO 64K.
000440  016702  000140                  MOV     PRGMAP+2,R2     ;...HI 64K.
000444  006202                  3$:     ASR     R2              ;SHIFT THE MAP POINTER...HI
000446  006001                          ROR     R1              ;...LO.
000450  103404                          BCS     4$              ;BR WHEN FIRST BANK FOUND.
000452  062700  000200                  ADD     #200,   R0      ;UPDATE TMP PAR TO NEXT BANK.
000456  100372                          BPL     3$              ;BR IF MORE.
000460  000000                          HALT            ;FATAL ERROR!!!  MAP EMPTY?
000462  010037  172340          4$:     MOV     R0,     @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
000466  000137  000472                  JMP     @#5$            ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
000472  062700  000200          5$:     ADD     #200,   R0      ;KEEP UPDATING TEMP PAR REG.
000476  006202                          ASR     R2              ;SHIFT POINTER...HI
000500  006001                          ROR     R1              ;...LO
000502  103373                          BCC     5$              ;BR IF TOP BANK NOT YET FOUND.
000504  010037  172342                  MOV     R0,     @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
000510  000410                          BR      20$             ;BR TO RELOCATE SECTION.
000512  016700  000062          10$:    MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
000516  062700  001100                  ADD     #STACK, R0      ;SET UP STACK POINTER.
000522  010006                          MOV     R0,     SP      ;SET STACK TO RELOCATE PROGRAM.
000524  062700  177432                  ADD     #20$-STACK,R0   ;ADJUST R0 TO RELOCATED "20$" ADDRESS.
000530  000110                          JMP     (R0)            ;GO TO "20$" (RELOCATED).
000532  022767  000003  000042  20$:    CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.
```



```
000540  001402                          BEQ     21$             ;BR IF IN BANKS 0 AND 1.
000542  004767  016714                  JSR     PC,     RELO    ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
000546  005705                  21$:    TST     R5              ;CHECK RESTART/RESTORE FLAG.
000550  001006                          BNE     22$             ;BR IF RESTORE.
000552  005067  000412                  CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
000556  105067  000320                  CLRB    $TSTNM
000562  000167  005316                  JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
000566  004767  016476          22$:    JSR     PC,     RESLDR  ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
000572  000000                          HALT                    ;HALT AFTER RESTORING THE LOADERS.
000574  000167  002054                  JMP     START4          ;CONTINUE WILL RESTART THE PROGRAM.
                                ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
                                ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
                                ;*  CIRCUMSTANCES.
000600  000000                  RELOCF: .WORD   0       ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
000602  000000  000000          PRGMAP: .WORD   0,0     ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
000606  000000                  MMAVA:  .WORD   0       ;MEMORY MANAGEMENT AVAILABLE FLAG.
```

```
                                        .SBTTL  POWER DOWN AND UP ROUTINES

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

                                        ;;**************************************************************
                                        ;POWER UP ROUTINE
000662  012737  000756  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
000670  016706  000066                  MOV     $SAVR6,SP       ;;GET SP
000674  005067  000062                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
000700  005267  000056          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
000704  001375                          BNE     1$              ;;OF  WORD
000706  012677  000226                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
000712  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
000714  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
000716  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
000720  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
000722  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
000724  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
000726  012737  000610  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
000734  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
000742  004567  022544                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000746  025641                  $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
000750  012716                          MOV     (PC)+,(SP)      ;;RESTART AT RESTART
000752  000300                  $PWRAD: .WORD   RESTART         ;;RESTART ADDRESS
000754  000002                          RTI
000756  000000                  $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
000760  000776                          BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
000762  000000                  $SAVR6: 0                       ;;PUT THE SP HERE
```



```
                                        .SBTTL  COMMON TAGS

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

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

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

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



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

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

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

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

```
                                ;;*************************************************************
                                ;*THE FOLLOWING TAGS ARE USER DEFINED
                                ;;*************************************************************
001514  000000                  $VERPC: .WORD   0       ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
001516  070032                  RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
                                                        ;NOTE:  FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
001520  000000                  LWAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
001522  000000                  LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE
001524                          MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
001524  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001526  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001530                          TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
001530  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001532  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001534                          SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
                                                        ; TEST EACH BANK ONCE.
001534  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001536  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001540                          PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
001540  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001542  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001544                          BITPT:                  ;POINTER TO CURRENT 4K BANK OF MEMORY
001544  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001546  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001550                          TMPPT:                  ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
001550  000000                          .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
001552  000000                          .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
001554  000000                  MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
                                                        ;SET UP BY "INITMM" AND "INITDN" ROUTINEES.
                                                        ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
001556  000                     SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
001557  000                     FLAGBK: .BYTE   0       ;BK BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP".
001560  000                     OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
        001562                          .EVEN
001562  000000                  FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
                                                        ;FIRST ADDRESS IS USER SELECTABLE.
001564  000000                  TMPFAD: .WORD   0       ;ADJUSTED FIRST ADDRESS.
001566  000000                  FADMSK: .WORD   0       ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
                                                        ; TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
001570  000000  000000          FADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
001574  000000                  LSTADR: .WORD   0       ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
                                                        ;LAST ADDRESS IS USER SELECTABLE.
001576  000000                  TMPLAD: .WORD   0       ;ADJUSTED LAST ADDRESS.
001600  000000                  LADMSK: .WORD   0       ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
                                                        ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
001602  000000  000000          LADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
001606  000000                  BLKMSK: .WORD   0       ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
001610  000000                  .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
001612  000004                  WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND
001614  000000                  TEMP:   .WORD   0       ;TEMPORARY STORAGE
001616  000000                  CASFLG: .WORD   0       ;CACHE PRESENT FLAG
001620  177746                  CASREG: .WORD   177746  ;CACHE CONTROL REGISTER

                                ;;*************************************************************
                                ;*  RELATIVE ADDRESSING TABLE.
                                ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
```



```
                                ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMEMT TAGS.
                                ;;*************************************************************
001622                          RADTAB:
001622  001100                  .STACK: STACK           ;STACK POINTER INITIAL ADDRESS.
001624  001516                  .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
001626  002076                  .MPR0:  MPR0            ;MEMORY PARITY REGISTER TABLE ADDRESS.
001630  002276                  .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
001632  012052                  .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
001634  002050                  .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
001636  017430                  .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
001640  002340                  .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
001642  000010                  .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
001644  014004                  .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
                                ;;*************************************************************
                                ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
                                ;;*************************************************************
001646  001116  001120  001124  DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0
001654  001126  000000
001660  001514  001116  001120  DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
001666  001124  001126  000000
001674  001514  001116  001120  DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0
001702  001124  000000
001706  001514  001116  001160  DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0
001714  001120  000000
001720  001514  001116  001120  DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
001726  001160  001124  001126
001734  000000
001736  001514  001116  001160  DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
001744  001120  001124  001126
001752  000000
001754  001514  001116  001120  DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
001762  001122  001124  001126
001770  000000
001772  001514  001116  001122  DT24:   $VERPC,$ERRPC,$BDADR,0
002000  000000
002002  001514  001116  001122  DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
002010  001160  001162  000000
002016  001514  001116  001160  DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
002024  001162  000000
002030  001160  001162  001120  DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
002036  001126  000000
002042  001166  000000          DT31:   $TMP3,0
002046  177777                          .WORD   -1      ;TABLE TERMINATOR.

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

002050  125325                  MPPATS: 125325                  ;EVEN,ODD
002052  152652                          152652                  ;ODD,EVEN
002054  052452                          052452                  ;EVEN,ODD
002056  025125                          025125                  ;ODD,EVEN
002060  102070                          102070                  ;EVEN,EVEN
002062  072527                          072527                  ;ODD,ODD
002064  177777                          177777                  ;EVEN,EVEN
```

```
                                        107030                  ;ODD,ODD
                                        152525                  ;ODD,EVEN
                                        0                       ;EXTRA PATTERN HOLDER FOR
                                                                ;FUTURE USE
                                MPEND:  0                       ;TABLE TERMINATOR


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



```
002210  000000                  0                       ;CONTROL MAP (LOW 64K)
002212  000000                  0                       ;CONTROL MAP (HIGH 64K)
002214  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002216  172125         MPR10:   172124 +1               ;PARITY STATUS REGISTER
002220  000000                  0                       ;CONTROL MAP (LOW 64K)
002222  000000                  0                       ;CONTROL MAP (HIGH 64K)
002224  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002226  172127         MPR11:   172126 +1               ;PARITY STATUS REGISTER
002230  000000                  0                       ;CONTROL MAP (LOW 64K)
002232  000000                  0                       ;CONTROL MAP (HIGH 64K)
002234  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002236  172131         MPR12:   172130 +1               ;PARITY STATUS REGISTER
002240  000000                  0                       ;CONTROL MAP (LOW 64K)
002242  000000                  0                       ;CONTROL MAP (HIGH 64K)
002244  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002246  172133         MPR13:   172132 +1               ;PARITY STATUS REGISTER
002250  000000                  0                       ;CONTROL MAP (LOW 64K)
002252  000000                  0                       ;CONTROL MAP (HIGH 64K)
002254  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002256  172135         MPR14:   172134 +1               ;PARITY STATUS REGISTER
002260  000000                  0                       ;CONTROL MAP (LOW 64K)
002262  000000                  0                       ;CONTROL MAP (HIGH 64K)
002264  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
002266  172137         MPR15:   172136 +1               ;PARITY STATUS REGISTER
002270  000000                  0                       ;CONTROL MAP (LOW 64K)
002272  000000                  0                       ;CONTROL MAP (HIGH 64K)
002274  000000                  0                       ;MASK FOR MOS,CORE,MS11-K
                                ;THIS IS THE END OF THE TABLE !
002276  000021         MPRX:    .BLKW   17,             ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
                                                        ; (THE EXTRA WORD IS FOR A TERMINATOR.)
```

```
                                .SBTTL  ERROR POINTER TABLE

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

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


002340                          $ERRTB:
                                ;*  ITEM 1
002340  027010                      DM1             ;PARITY REGISTER DATA ERROR.
002342  030367                      DH1             ;PC,REG,S/B,WAS
002344  001646                      DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
002346  030734                      DF1             ;16,18,16,16
                                ;*  ITEM 2
002350  027044                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002352  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002354  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002356  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 3
002360  027044                      DM2             ;ADDRESS TEST ERROR(TST1-5).
002362  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002364  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002366  030745                      DF3             ;16,18,18,8,8
                                ;*  ITEM 4
002370  027100                      DM4             ;CONSTANT DATA ERROR(TST6-10).
002372  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002374  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002376  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 5
002400  027136                      DM5             ;ROTATING BIT ERROR(TST11-12).
002402  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002404  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002406  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 6
002410  027174                      DM6             ;MOS REFRESH TEST ERROR (TST30-31).
002412  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002414  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002416  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 7
002420  027240                      DM7             ;3 XOR 9 PATTERN ERROR(TST13-16).
002422  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002424  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002426  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 10
002430  027301                      DM10            ;MARCHING 1'S AND 0'S ERROR(TST27).
002432  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002434  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002436  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 11
```



```
002440  027345                      DM11            ;PARITY MEMORY ADDRESS ERROR(TST17).
002442  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002444  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002446  030745                      DF3             ;16,18,18,8,8
                                ;*  ITEM 12
002450  027411                      DM12            ;DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
002452  030433                      DH12            ;V/PC,P/PC,MA,S/B
002454  001674                      DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
002456  030745                      DF3             ;16,18,18,8
                                ;*  ITEM 13
002460  027465                      DM13            ;WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
002462  030433                      DH12            ;V/PC,P/PC,MA,S/B
002464  001674                      DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
002466  030745                      DF3             ;16,18,18,8
                                ;*  ITEM 14
002470  027555                      DM14            ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
002472  030454                      DH14            ;V/PC,P/PC,REG,MA
002474  001706                      DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
002476  030752                      DF14            ;16,18,18,18
                                ;*  ITEM 15
002500  027010                      DM1             ;PARITY REGISTER DATA ERROR.
002502  030475                      DH15            ;V/PC,P/PC,MAUT,REG,S/B,WAS
002504  001720                      DT15            ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
002506  030752                      DF14            ;16,18,18,18,16,16
                                ;*  ITEM 16
002510  027654                      DM16            ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
002512  030454                      DH14            ;V/PC,P/PC,REG,MA
002514  001706                      DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
002516  030752                      DF14            ;16,18,18,18
                                ;*  ITEM 17
002520  027733                      DM17            ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
                                                    ; TRAPPED(TST21).
002522  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002524  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002526  030745                      DF3             ;16,18,18,8,8
                                ;*  ITEM 20
002530  030031                      DM20            ;RANDOM DATA ERROR(TST20).
002532  030406                      DH2             ;V/PC,P/PC,MA,S/B,WAS
002534  001660                      DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
002536  030740                      DF2             ;16,18,18,16,16
                                ;*  ITEM 21
002540  030063                      DM21            ;INSTRUCTION EXICUTION ERROR(TST21-26).
002542  030530                      DH21            ;V/PC,P/PC,IUT,MA,S/B,WAS
002544  001736                      DT21            ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
002546  030760                      DF21            ;16,18,16,18,16,16
                                ;*  ITEM 22         ;NOT USED
                                ;*  ITEM 23
002550  030132                      DM23            ;PROGRAM CODE CHANGED WHEN RELOCATED.
002552  030561                      DH23            ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
002554  001754                      DT23            ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
002556  030752                      DF14            ;16,18,18,18,16,16
                                ;*  ITEM 24
002560  030177                      DM24            ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
002562  030621                      DH24            ;V/PC,P/PC,TRP/PC
002564  001772                      DT24            ;$VERPC,$ERRPC,$BDADR
002566  030752                      DF14            ;16,18,18
```

```
                                ;*  ITEM 25
002570  030253                        DM25            ;TRAPPED TO 114.
002572  030642                        DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
002574  002002                        DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
002576  030752                        DF14            ;16,18,18,18,16
                                ;*  ITEM 26
002600  030273                        DM26            ;FAILED TO TRAP.
002602  030673                        DH26            ;V/PC,P/PC,REG,WAS
002604  002016                        DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
002606  030740                        DF2             ;16,18,18,16
                                ;*  ITEM 27
002610  030313                        DM27            ;(ACTION ENABLE WASN'T SET).
002612  030673                        DH26            ;V/PC,P/PC,REG,WAS
002614  002016                        DT26            ;$VERPC,$ERRPC,$TMP0,$BDDAT
002616  030740                        DF2             ;16,18,18,16
                                ;*  ITEM 30
002620  000000                        0               ;NO MESSAGE.
002622  030715                        DH30            ;REG,WAS,MA,WAS
002624  002030                        DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
002626  030766                        DF30            ;18,16,18,8
                                ;*  ITEM 31
002630  030347                        DM31            ;TRAPPED TO 4
002632  000000                        0               ;NO HEADER
002634  002042                        DT31            ;$TMP3
002636  030766                        DF30            ;18
```



```
                                .SBTTL START:   SETUP AND MAP MEMORY

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

002640  105067  176712       START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
002644  000403                       BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
002646  112767  177777  176702 SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
002654                       STARTA:
                                .SBTTL  INITIALIZE THE COMMON TAGS
                                ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
002654  012706  001100               MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
002660  005026                       CLR     (R6)+           ;;CLEAR MEMORY LOCATION
002662  022706  001140               CMP     #$WR,R6 ;;DONE?
002666  001374                       BNE     .-6             ;;LOOP BACK IF NO
002670  012706  001100               MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                ;;INITIALIZE A FEW VECTORS
002674  012737  000610  000024       MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
002702  012737  000340  000026       MOV     #340,@#PWRVEC+2 ;;LEVEL 7
002710  016767  011242  011232       MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
002716  013746  000004               MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
002722  012737  002756  000004       MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
002730  012767  177570  176202       MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
002736  012767  177570  176176       MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
002744  022777  177777  176166       CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
002752  001012                       BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                             ;;AND  THE HARDWARE SWR IS NOT = -1
002754  000403                       BR      65$             ;;BRANCH IF NO TIMEOUT
002756  012716  002764       64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
002762  000002                       RTI
002764  012767  000176  176146 65$:  MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
002772  012767  000174  176142       MOV     #DISPREG,DISPLAY
003000  012637  000004       66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

003004  005067  176202               CLR     $PASS           ;;CLEAR PASS COUNT
003010  132767  000200  176207       BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
003016  001403                       BEQ     67$             ;;YES,USE NON-APT SWITCH
003020  012767  001226  176112       MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
003026                       67$:
003026  005067  176470               CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
003032  005077  176104               CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
003036  005227  177777               INC     #-1             ;;FIRST TIME?
003042  001040                       BNE     68$             ;;BRANCH IF NO
003044  022737  014222  000042       CMP     #$ENDAD,@#42    ;;ACT-11?
003052  001434                       BEQ     68$             ;;BRANCH IF YES
003054  004567  020432               JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
003060  003132                       .WORD   69$             ;ADDRESS OF MESSAGE TO BE TYPED
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
003062  005737  000042               TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
003066  001015                       BNE     70$             ;;BRANCH IF YES
```

```
 893  003070  126727  176130  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
 894  003076  001411                          BEQ     70$             ;;BRANCH IF YES
 895  003100  026727  176034  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
 896  003106  001010                          BNE     71$             ;;BRANCH IF NO
 897                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
 898                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 899  003110  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 900  003114  004767  017316                  JSR     PC,     $GTSWR  ;GO TO THE SUBROUTINE
 901  003120  000403                          BR      71$
 902  003122  112767  000001  176004  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
 903  003130                          71$:
 904  003130  000405                          BR      68$             ;;GET OVER THE ASCIZ
 905                                  ;;69$:  .ASCIZ  <CRLF>'CZQMCF0'<CRLF>
 906  003144                          68$:
 907  003144  010700                          MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
 908  003146  022700  003146                  CMP     #.,     R0      ;CHECK IF THE PROGRAM IS RELOCATED.
 909  003152  001402                          BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
 910  003154  000167  175120                  JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
 911  003160  012767  000003  175414  10$:    MOV     #3,     PRGMAP  ;INITIALIZE PROGRAM MAP....LO 64K.
 912  003166  005067  175412                  CLR     PRGMAP+2        ;...HI 64K.
 913  003172  005067  175402                  CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
 914  003176  105737  001224                  TSTB    @#$ENV          ;CHECK FOR APT11
 915  003202  001011                          BNE     13$             ;BR IF APT11
 916  003204  005727  000042                  TST     @#42            ;CHECK FOR STANDALONE
 917  003210  001406                          BEQ     13$             ;BR IF STANDALONE
 918  003212  023737  000042  000046          CMP     @#42,@#46       ;CHECK FOR ACT11
 919  003220  001402                          BEQ     13$             ;BR IF ACT11
 920                                                                  ;MUST BE XXDP
 921  003222  004767  014122                  JSR     PC,SAVLDR       ;GO SAVE LOADERS
 922
 923                                  ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
 924  003226  005067  175354          13$:    CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG
 925  003232  032777  010000  175700          BIT     #SW12,  @SWR    ;CHECK FOR INHIBIT KT11 SWITCH
 926  003240  001014                          BNE     IMPCK           ;BRANCH IF SET
 927  003242  012737  003272  000004          MOV     #IMPCK,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
 928  003250  005037  177572                  CLR     @#SR0           ;CLEAR MEM MGMT STATUS REG
 929  003254  004767  011020                  JSR     PC,     MMINIT  ;MEM MGMT INITIALIZATION ROUTINE.
 930  003260  005267  175322                  INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
 931  003264  004567  020222                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 932  003270  025354                          .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
 933                                                                  ;"KT11 AVAILABLE"
 934
 935                                  ;*      CHECK IF CACHE PRESENT, IF SO TURN IT OFF!!!
 936  003272  012706  001100          IMPCK:  MOV     #STACK, SP
 937  003276  005067  176314                  CLR     CASFLG          ;CLEAR CACHE PRESENT FLAG
 938  003302  012737  003324  000004          MOV     #MAPMEM,@#ERRVEC
 939  003310  052767  000014  174430          BIS     #14,    IMPCHE
 940  003316  012767  000001  176272          MOV     #1,     CASFLG  ;SET CACHE PRESENT FLAG
 941
 942                                  ;;**********-**********-**********-**********-**********-**********
 943                                  ;*  ROUTINE TO MAP ALL OF MEMORY.
 944                                  ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
 945                                  ;*      R0 = MEMMAP POINTER...LO 64K.
 946                                  ;*      R1 = MEMMAP POINTER...HI 64K.
 947                                  ;*      R2 = ADDRESS POINTER
 948                                  ;*      R3 = BANK POINTER...LO 64K.
```



```
 949                                  ;*      R4 = BANK POINTER...HI 64K.
 950                                  ;*      R5 = SCRATCH REGISTER.
 951                                  ;;**********-**********-**********-**********-**********-**********
 952  003324  012706  001100          MAPMEM: MOV     #STACK, SP      ;RESET THE STACK
 953  003330  012700  001524                  MOV     #MEMMAP,R0      ;SET UP MEMORY MAP POINTER...LO 64K.
 954  003334  012701  001526                  MOV     #MEMMAP+2,R1    ;...HI 64K.
 955  003340  005010                          CLR     (R0)            ;CLR MEMORY MAP...LO 64K.
 956  003342  005011                          CLR     (R1)            ;...HI 64K.
 957  003344  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
 958  003346  012703  000001                  MOV     #1,     R3      ;SETUP 4K BANK POINTER...LO 64K.
 959  003352  005004                          CLR     R4              ;...HI 64K.
 960  003354  005067  175606                  CLR     $TMP3           ;INIT TEMPORARY HIGH ADDRESS BITS.
 961  003360  004567  020126                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 962  003364  025421                          .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
 963                                                                  ;"MEMORY MAP:"
 964  003366  012737  003502  000004          MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
 965  003374  011222          1$:     MOV     (R2),   (R2)+   ;READ+WRITE ALL MEMORY
 966  003376  032702  017777                  BIT     #MASK4K,R2      ;CHECK FOR 4K BOUNDRY
 967  003402  001374                          BNE     1$              ;BRANCH IF MORE IN BANK
 968  003404  050310                          BIS     R3,     (R0)    ;SET FLAG FOR BANK...LO 64K.
 969  003406  050411                          BIS     R4,     (R1)    ;...HI 64K.
 970  003410  010267  175550                  MOV     R2,     $TMP2   ;SAVE ADDRESS POINTER.
 971  003414  005367  175544                  DEC     $TMP2           ;ADJUST TO LAST ADR, LAST BANK.
 972  003420  005767  175162                  TST     MMAVA           ;CHECK FOR MEM MGMT.
 973  003424  001432                          BEQ     3$              ;BR IF NO MEM MGMT.
 974  003426  042767  160000  175530          BIC     #160000,$TMP2   ;CLEAR BANK BITS ON RELATIVE ADDRESS.
 975  003434  013705  172344                  MOV     @#KIPAR2,R5     ;SAVE KIPAR2.
 976  003440  005067  175522                  CLR     $TMP3           ;MAKE SURE HI BITS ARE INIT.
 977  003444  006305                          ASL     R5              ;SHIFT IT 6 PLACES.
 978  003446  006305                          ASL     R5
 979  003450  006305                          ASL     R5
 980  003452  006305                          ASL     R5
 981  003454  006305                          ASL     R5
 982  003456  006167  175504                  ROL     $TMP3
 983  003462  006305                          ASL     R5
 984  003464  006167  175476                  ROL     $TMP3
 985  003470  060567  175470                  ADD     R5,     $TMP2   ;MAKE LAST ADR PHYSICAL.
 986  003474  005567  175466                  ADC     $TMP3
 987  003500  000404                          BR      3$              ;GO TO UPDATE POINTERS.
 988
 989                                  ;*  TIMEOUT TRAPS TO HERE
 990  003502  022626          2$:     CMP     (SP)+,  (SP)+   ;RESTORE THE STACK POINTER
 991  003504  052702  017777                  BIS     #MASK4K,R2      ;LAST ADDRESS OF 4K BANK
 992  003510  005202                          INC     R2              ;FIRST ADDRESS OF NEXT BANK.
 993  003512  005767  175070          3$:     TST     MMAVA           ;CHECK FOR MEM MGMT
 994  003516  001411                          BEQ     4$              ;BRANCH IF NO MEM MGMT
 995  003520  062737  000200  172344          ADD     #200, @#KIPAR2  ;UPDATE THIRD PAR
 996  003526  012702  040000                  MOV     #40000, R2      ;POINT TO START OF THIRD PAR
 997  003532  006303                          ASL     R3              ;UPDATE LO BANK POINTER.
 998  003534  006104                          ROL     R4              ;UPDATE HI BANK POINTER
 999  003536  100316                          BPL     1$              ;BRANCH IF MORE MEMORY TO MAP.
1000  003540  000402                          BR      5$              ;EXIT WHEN DONE.
1001
1002  003542  106303          4$:     ASLB    R3              ;UPDATE MAP POINTER
1003  003544  100313                  BPL     1$              ;BRANCH IF NOT YET DONE
1004  003546  012737  025114  000004  5$:     MOV     #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
```

```
 1005  003554  004767  014632                  JSR     PC,     TYPMAP  ;GO TYPE THE MAP.
 1006  003560  004567  017726                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1007  003564  001201                          .WORD   $CRLF           ;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),   R0      ;GET LOW MEM MAP
 1011  003600  042700  177760                  BIC     #177760, R0     ;MASK ALL BUT BOTTOM 4 BANKS
 1012  003604  020027  000017                  CMP     R0,     #17     ;CHECK THAT BOTTOM 16K IS ALL THERE!
 1013  003610  001530                          BEQ     GMPR            ;BRANCH IF BOTTOM 16K EXISTS
 1014  003612  004567  017674                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1015  003616  025524                          .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
 1016                                                                  ;"FIRST 16K OF MEMORY NOT ALL THERE!"
 1017  003620  000000                  6$:     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          TIMOUT: 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,  @SWR    ;CHECK IF MEM MGMT TO BE INHIBITED.
 1026  003640  001011                          BNE     1$              ;BR IF NO MEM MGMT.
 1027  003642  012737  003664  000004          MOV     #1$,    @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
 1028  003650  005037  177572                  CLR     @#SR0           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
 1029  003654  004767  010420                  JSR     PC,     MMINIT  ;INIT ALL MEM MGMT REGISTERS.
 1030  003660  005267  174722                  INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
 1031  003664                          1$:
 1032  003664  004567  017622                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1033  003670  025437                          .WORD   BYTMES          ;ADDRESS OF MESSAGE TO BE TYPED
 1034                                                                  ;"BYTE MEMORY MAP:"
 1035  003672  005000                          CLR     R0              ;SET UP TYPE OUT FLAG.
 1036  003674  005002                          CLR     R2              ;SET ADDRESS POINTER TO ZERO.
 1037  003676  012737  003742  000004          MOV     #20$,   @#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
 1038  003704  105712                  10$:    TSTB    (R2)            ;DO DATI ONLY.
 1039  003706  032702  000001                  BIT     #BIT0,  R2      ;CHECK FOR WORD ADDRESS.
 1040  003712  001001                          BNE     11$             ;BR IF ODD BYTE ADDRESS.
 1041  003714  011212                          MOV     (R2),   (R2)    ;DO DATI, DATO...NOP FOR READ ONLY MAP.
 1042  003716  151212                  11$:    BISB    (R2),   (R2)    ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
 1043  003720  005700                          TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 1044  003722  001023                          BNE     30$             ;BR IF ALREADY TYPED "FROM".
 1045  003724  004567  017562                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1046  003730  025507                          .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
 1047                                                                  ;"FROM"
 1048  003732  010246                          MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
 1049  003734  004767  021212                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 1050  003740  000413                          BR      29$             ;GO TO ADDRESS POINTER UPDATE.
 1051                                  ;*  TIME OUTS COME HERE.
 1052  003742  022626                  20$:    CMP     (SP)+,  (SP)+   ;POP TWO OFF STACK.
 1053  003744  005700                          TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 1054  003746  001411                          BEQ     30$             ;BR IF ALREADY TYPED "TO".
 1055  003750  004567  017536                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1056  003754  025517                          .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
 1057                                                                  ;"TO"
 1058  003756  005302                          DEC     R2              ;BACK UP ONE BYTE.
 1059  003760  010246                          MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
 1060  003762  004767  021164                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
```



```
 1061  003766  005202                          INC     R2              ;RESET ADDRESS POINTER.
 1062  003770  005100                  29$:    COM     R0              ;RESET PREVIOUS TYPOUT FLAG.
 1063  003772  005202                  30$:    INC     R2              ;UPDATE ADDPESS POINTER TO NEXT BYTE.
 1064  003774  001403                          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 TYPEOUT.
 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                  31$:    TST     R0              ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
 1076  004046  001407                          BEQ     32$             ;BR TO EXIT IF TYPING ALL DONE.
 1077  004050  004567  017436                  JSR     R5,     $PRINT  ;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,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 1083  004066  000000                  32$:    HALT                    ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
 1084                                                                  ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
 1085  004070  000654                          BR      TIMOUT          ;LOOP BACK AND DO AGAIN UPON CONTINUE.
 1086
 1087                                  .SBTTL  MAP PARITY REGISTERS
 1088                                  ;;*********** ***************************************************
 1089                                  ;*  SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
 1090                                  ;*  THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
 1091                                  ;;*************************************************************
 1092
 1093  004072  012704  002276          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
 1094  004076  032777  000100  175034          BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
 1095  004104  001036                          BNE     GMPRD           ;BR IF INHIBIT PARITY.
 1096  004106  012703  002076                  MOV     #MPR0,  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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 1116  004200  025605                          .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
```

```
 1117                                                    ;"NO MEMORY PARITY REGISTERS FOUND"
 1118  004202  005014                  GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
 1119  004204  000167  001156                  JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING
 1120
```



```
 1121                                   .SBTTL  MAP PARITY MEMORY
 1122                                   ;;*****************************************************************
 1123                                   ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
 1124                                   ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
 1125                                   ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
 1126                                   ;       - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
 1127                                   ;       - PARITY GENERATE OR DETECT LOGIC FAILED
 1128                                   ;       - PARITY ERROR BIT FAILED TO SET
 1129                                   ;       - PARITY BITS IN MEMORY LOCATION FAILED
 1130                                   ;       - I.E. BIT STUCK AT GOOD PARITY VALUE
 1131                                   ;;*****************************************************************
 1132
 1133  004210  004767  014054          MPAMEM: JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
 1134  004214  012767  000001  175322          MOV     #1,     BITPT   ;INITIALIZE 4K POINTER
 1135  004222  005067  175320                  CLR     BITPT+2         ;CLEAR HI 64K POINTER
 1136  004226  012702  014000                  MOV     #14000, R2      ;SET ADR POINTER TO 14000.
 1137  004232  005767  174350                  TST     MMAVA           ;CHECK FOR MEM MGMT
 1138  004236  001404                          BEQ     MAPRB           ;BRANCH IF NO MEM MGMT
 1139  004240  012702  054000                  MOV     #54000, R2      ;SET ADR POINTER TO PAR2
 1140  004244  004767  010030                  JSR     PC,     MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.
 1141
 1142                                   ;;*****************************************************************
 1143                                   ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
 1144                                   ;*  THEN WRITE TEST LOCATION VIA DATO & READ TEST LOCATION VIA DATI
 1145                                   ;*  THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
 1146                                   ;;*****************************************************************
 1147
 1148  004250  005067  175264          MAPRB:  CLR     PMEMAP          ;CLEAR THE PARITY MEMORY MAP
 1149  004254  005067  175262                  CLR     PMEMAP+2
 1150  004260  012703  002076          1$:     MOV     #MPR0,  R3      ;INITIALIZE TABLE ADDRESS
 1151  004264  012713  000001          2$:     BIT     #1,     (R3)    ;IS THIS REGISTER PRESENT?
 1152  004270  001052                          BNE     3$              ;NO - GET THE NEXT ONE
 1153  004272  013773  001612  000000          MOV     @#WWP,  @(R3)   ;YES - SET WRITE WRONG PARITY
 1154                                                                  ;AND CLEAR REST OF REGISTER
 1155  004300  011212                          MOV     (R2),   (R2)    ;WRITE WRONG PARITY
 1156  004302  005712                          TST     (R2)            ;READ WRONG PARITY
 1157  004304  043773  001612  000000          BIC     @#WWP,  @(R3)   ;CLEAR WRITE WRONG PARITY
 1158  004312  005773  000000                  TST     @(R3)           ;OTHERWISE, CHECK TO SEE IF THIS
 1159                                                                  ;CONTROL REGISTER GOT A PARITY
 1160                                                                  ;ERROR
 1161  004316  100014                          BPL     6$              ;BRANCH IF IT DIDN'T AND CHECK
 1162  004320  032773  007740  000000          BIT     #7740,  @(R3)   ;IS IT A CORE PAR. REG.
 1163  004326  001404                          BEQ     5$              ;BRANCH IF NOT.
 1164  004330  012763  070032  000006          MOV     #70032, 6(R3)   ;IF IT IS SET UP MASK
 1165  004336  000413                          BR      7$              ;AND BRANCH TO SET BITS.
 1166  004340  012763  077772  000006  5$:     MOV     #77772, 6(R3)   ;IF MOS SET UP MASK
 1167  004346  000407                          BR      7$              ;AND BRANCH TO SET BIT.
 1168  004350  032773  007740  000000  6S:     BIT     #7740,  @(R3)   ;IF ANY BITS ARE SET
 1169  004356  001417                          BEQ     3$              ;THEN CSR IS MS11-K.
 1170  004360  012763  070000  000006          MOV     #70000, 6(R3)   ;IF MS11-K SET MASK.
 1171  004366  056763  175152  000002  7S:     BIS     BITPT,  2(R3)   ;SET FLAG IN MAP FOR THIS PARITY REGISTER
 1172  004374  056763  175146  000004          BIS     BITPT+2,4(R3)
 1173  004402  056767  175136  175130          BIS     BITPT,  PMEMAP  ;SET FLAG IN PARITY MAP
 1174  004410  056767  175132  175124          BIS     BITPT+2,PMEMAP+2
 1175  004416  062703  000010          3S:     ADD     #10,    R3      ;STEP UP TO NEXT REGISTER
 1176  004422  020327  002276                  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  001401                          BEQ     10$             ;BR IF NO MEM MGMT
1181  004440  062737  000200  172344  4$:     ADD     #200,   @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
1182  004446  006367  175072                  ASL     BITPT           ;UPDATE BANK POINTER...LO 64K.
1183  004452  006167  175070                  ROL     BITPT+2         ;...HI 64K.
1184  004456  103441                          BMI     TMAP            ;BR IF ALL DONE.
1185  004460  023727  172344  001000          CMP     @#KIPAR2,#1000  ;THIS CODE TESTS IF MS11-K IS
1186  004466  001013                          BNE     12$             ;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$:     MOV     #20004,@#WRP
1192  004516  036767  175022  175000  12$:     BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS...LO 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  062762  020000          10$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
1200  004554  106367  174764                  ASLB    BITPT           ;MOVE POINTER TO NEXT BANK.
1201  004560  100367                          BPL     11$             ;BR IF MORE TO LOOK FOR.
1202
1203                                  ;;*****************************************************************
1204                                  ;*  ROUTINE T  TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
1205                                  ;*  AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
1206                                  ;;*****************************************************************
1207
1208  004562  004767  013502          TMAP:   JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS PRESENT
1209  004566  004567  016720                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1210  004572  025462                          .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
1211                                                                  ;"PARITY MEMORY MAP:"
1212  004574  012703  002076                  MOV     #MPR0,  R3      ;INITIALIZE TABLE POINTER
1213  004600  032713  000001          1$:     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,     $PRINT  ;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
1242  004710  026110                          .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
1243                                                                  ;"CONTROLS"
1244  004712  010300                          MOV     R3,     R0      ;SET UP R0 FOR TYPMAP ROUTINE.
1245  004714  005720                          TST     (R0)+           ;UPDATE POINTER TO MAP.
1246  004716  004767  013413                  JSR     PC,     TYPMAP  ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
1247  004722  062703  000010          2$:     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,     $PRINT  ;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          8$:     CMP     (R5)+,  @#MPR15 ;SEARCH FOR MS11-K CSR IN
1267  005040  001366                          BNE     6$              ;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,     $PRINT  ;OUTPUT MESSAGE TO RUN MS11-K TEST.
1271  005056  026176                          .WORD   MX6
1272  005060  005737  002276          9$:     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     #MPR0,  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          64$:  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          65$:  JSR     PC,     SPRNT0  ;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          67$:  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.
 1322
 1323                                  ;;**********-**********-**********-**********-**********
 1324                                  ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
 1325                                  ;*  THIS IS A ONCE ONLY TEST.
 1326                                  ;;**********-**********-**********-**********-**********
 1327
 1328  005252  012704  002076          RESCHK: MOV     #MPR0,  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     #MPR0,  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  0022/6                  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  001403                          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.
```

```
                                        .SBTTL  USER PARAMETER SELECTION SECTION
                                        ;;*************************************************************
                                        ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                        ;;*************************************************************
005424  012700  000001          MANUL1: MOV     #BIT0,  R0          ;SET UP BANK POINTER.
005430  005001                          CLR     R1                  ;...HI 64K.
005432  005002                          CLR     R2                  ;CLEAR ADDRESS POINTER.
005434  005003                          CLR     R3                  ;...HI ADDRESS BITS.
005436  004567  016050                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
005442  026311                          .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**.
005444  013746  177776                  MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
005450  004767  015664                  JSR     PC,     $RDOCT      ;GO TO THE SUBROUTINE
005454  042716  000001                  BIC     #BIT0,  (SP)        ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
005460  005067  174030                  CLR     SAVTST              ;INIT TEST MAP...LO 64K.
005464  005067  174046                  CLR     SAVTST+2            ;...HI 64K.
005470  062702  020000          1$:     ADD     #20000, R2          ;UPDATE ADDRESS POINTER TO NEXT BANK.
005474  005503                          ADC     R3
005476  020367  016006                  CMP     R3,     $HIOCT      ;CHECK HI ADDRESS BITS.
005502  103403                          BLO     2$                  ;BR IF NOT HI ENOUGH YET.
005504  101006                          BHI     3$                  ;BR IF PAST SELECTED ADDRESS.
005506  020216                          CMP     R2,     (SP)        ;CHECK THE LO ADDRESS BITS.
005510  101004                          BHI     3$                  ;BR IF PAST SELECTED ADDRESS.
005512  006300                  2$:     ASL     R0                  ;UPDATE POINTER...LO 64K.
005514  006101                          ROL     R1                  ;...HI 64K.
005516  100364                          BPL     1$                  ;BR BACK TO CHECK NEXT BANK.
005520  000507                          BR      17$                 ;BR IF OVERFLOW.
005522  030067  173776          3$:     BIT     R0,     MEMMAP      ;CHECK IF BANK EXISTS.
005526  001003                          BNE     4$                  ;BR IF BANK EXISTS.
005530  030167  173772                  BIT     R1,     MEMMAP+2    ;CHECK HI 64K.
005534  001501                          BEQ     17$                 ;BR IF ADDRESS IN UN-MAPPED BANK.
005536  016704  015746          4$:     MOV     $HIOCT, R4          ;SAVE FIRST ADR HI BITS.
005542                          10$:
005542  004567  015744                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
005546  026376                          .WORD   LADMES              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"LAST ADDRESS:"
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
005550  013746  177776                  MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
005554  004767  015560                  JSR     PC,     $RDOCT      ;GO TO THE SUBROUTINE
005560  005716                          TST     (SP)                ;CHECK IF ADR 0 SELECTED (DEFAULT).
005562  001003                          BNE     11$                 ;BR IF NOT 0 (DEFAULT)
005564  005767  015720                  TST     $HIOCT              ;CHECK HI BITS.
005570  001005                          BNE     11$                 ;BR IF NOT 0 (DEFAULT).
005572  016716  173366                  MOV     $TMP2,  (SP)        ;SET UP DEFAULT LAST ADR.
005576  016767  173364  015704          MOV     $TMP3,  $HIOCT
005604  012667  173764          11$:    MOV     (SP)+,  LSTADR      ;GET THE DATA.
005610  020467  015674                  CMP     R4,     $HIOCT      ;CHECK FOR LAST ADR BELOW FIRST ADR.
005614  101352                          BHI     10$                 ;BR IF LAST BELOW FIRST.
005616  103403                          BLO     12$                 ;BR IF LAST ABOVE FIRST.
005620  021667  173750                  CMP     (SP),   LSTADR      ;CHECK FOR LAST BELOW FIRST.
005624  101346                          BHI     10$                 ;BR IF LAST BELOW FIRST.
005626  032716  017777          12$:    BIT     #MASK4K,(SP)        ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005632  001404                          BEQ     13$                 ;BR IF ON BOUNDRY.
```



```
005634  010067  173730                  MOV     R0,     FADMAP      ;SET UP FIRST ADDRESS MAP.
005640  010167  173726                  MOV     R1,     FADMAP+2
005644  050067  173664          13$:    BIS     R0,     SAVTST      ;SET FLAG IN TEST MAP...LO 64K.
005650  050167  173662                  BIS     R1,     SAVTST+2    ;...HI 64K.
005654  020367  015630          14$:    CMP     R3,     $HIOCT      ;CHECK FOR PAST LAST ADR.
005660  103404                          BLO     15$                 ;BR IF BELOW LAST ADR.
005662  101020                          BHI     16$                 ;BR IF GONE PAST LAST ADR.
005664  020267  173704                  CMP     R2,     LSTADR      ;CHECK FOR PAST LAST ADR.
005670  101015                          BHI     16$                 ;BR IF GONE PAST LAST ADR.
005672  062702  020000          15$:    ADD     #20000, R2          ;UPDATE ADDRESS POINTER.
005676  005503                          ADC     R3                  ;...HI BITS.
005700  006300                          ASL     R0                  ;UPDATE BANK POINTER...LO 64K.
005702  006101                          ROL     R1                  ;...HI 64K.
005704  100415                          BMI     17$                 ;BR IF OVERFLOW.
005706  030067  173612                  BIT     R0,     MEMMAP      ;CHECK IF THIS BANK EXISTS.
005712  001354                          BNE     13$                 ;BR IF BANK EXISTS.
005714  030167  173606                  BIT     R1,     MEMMAP+2    ;CHECK IF THIS BANK EXISTS.
005720  001351                          BNE     13$                 ;BR IF BANK EXISTS.
005722  000754                          BR      14$                 ;BR IF BANK DOESN'T EXIST.
005724  030067  173574          16$:    BIT     R0,     MEMMAP      ;CHECK IF THIS BANK EXISTS.
005730  001010                          BNE     20$                 ;BR IF IT EXISTS.
005732  030167  173570                  BIT     R1,     MEMMAP+2    ;CHECK IF THIS BANK EXISTS.
005736  001005                          BNE     20$                 ;BR IF IT EXISTS.
005740  005726                  17$:    TST     (SP)+               ;ADJUST THE STACK.
005742  004567  015544                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
005746  026421                          .WORD   BADADR              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"?ADDRESS IN UNMAPPED BANK?"
005750  000606                          BR      MANUAL              ;LOOP BACK TO THE BEGINNING.
005752  010067  173624          20$:    MOV     R0,     LADMAP      ;SET UP MAP FOR LAST ADDRESS.
005756  010167  173622                  MOV     R1,     LADMAP+2
005762  005767  172620          21$:    TST     MMAVA               ;CHECK FOR MEMORY MANAGEMENT.
005766  001404                          BEQ     22$                 ;BR IF NO MEM MGMT.
005770  042716  160000                  BIC     #160000,(SP)        ;ADJUST FSTADR TO VITRUAL BANK 0.
005774  062716  040000                  ADD     #40000, (SP)        ;...TO VIRTUAL BANK 2.
006000  012667  173556          22$:    MOV     (SP)+,  FSTADR      ;SAVE FISRT ADDRESS OFF THE STACK.
006004                          30$:
006004  004567  015502                  JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
006010  026456                          .WORD   CONST               ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"SELECT CONSTANT: "
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
006012  013746  177776                  MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
006016  004767  015316                  JSR     PC,     $RDOCT      ;GO TO THE SUBROUTINE
006022  012667  173562                  MOV     (SP)+,  .CONST      ;SAVE THE CONSTANT
006026  005767  172554          MANUL2: TST     MMAVA               ;CHECK IF MEM MGMT IS AVAILABLE.
006032  001436                          BEQ     31$                 ;BR IF NO MEM MGMT.
006034  042767  160000  173532          BIC     #160000,LSTADR      ;ADJUST LSTADR TO VIRTUAL BANK 0.
006042  062767  040000  173524          ADD     #40000, LSTADR      ;...VIRTUAL BANK 2.
006050  062767  000002  173516  31$:    ADD     #2,     LSTADR      ;ADJUST LAST ADDRESS UP ONE WORD.
006056  042767  000001  173510          BIC     #BIT0,  LSTADR      ;MAKE SURE IT IS A WORD ADDRESS.
006064  032767  017777  173502          BIT     #MASK4K,LSTADR      ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
006072  001004                          BNE     START1              ;BR IF NOT ON BOUNDRY.
006074  005067  173502                  CLR     LADMAP              ;CLEAR OUT THE LAST ADDRESS MAP.
006100  005067  173500                  CLR     LADMAP+2
```

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

  1482  006104  016706  173512          START1: MOV     .STACK, SP      ;SET STACK POINTER
  1483  006110  005767  173502                  TST     CASFLG          ;CHECK CACHE PRESENT FLAG
  1484  006114  001403                          BEQ     1$              ;BRANCH IF NO CACHE
  1485  006116  052777  000014  173474          BIS     #14,    @CASREG ;TURN OFF CACHE
  1486  006124  012767  006104  172754  1$:     MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
  1487  006132  066767  172442  172746          ADD     RELOCF, $LPADR
  1488  006140  004767  011372                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
  1489  006144  005767  172436                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
  1490  006150  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
  1491  006152  032737  000001  177572          BIT     #BIT0,  @#MSR   ;CHECK IF MEM MGMT ENABLED.
  1492  006160  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
  1493  006162  004767  006112                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
```



```
  1494                                  .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
  1495                                  ;;*************************************************************
  1496                                  ;*TEST 1         WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
  1497                                  ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
  1498                                  ;*      R1 = DATA READ FROM MEMORY (WAS)
  1499                                  ;*      R2 = VIRTUAL ADDRESS
  1500                                  ;*      R3 = NOT USED
  1501                                  ;*      R4 = NOT USED
  1502                                  ;*      R5 = BLOCK BOUNDRY BIT MASK.
  1503                                  ;;*************************************************************
  1504  006166                          TST1:
  1505  006166  004567  012436                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  1506  006172  000001                          .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
  1507                                                                  ; REQUIRED FOR THIS TEST.
  1508  006174  000167  005604                  JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
  1509                                                                  ; AVAILABLE FOR TEST.
  1510                                  ;*  UPWARDS WORD ADDRESSING.
  1511  006200  004467  006222                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1512  006204  004767  007644          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
  1513  006210  010012                  2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
  1514  006212  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  1515  006214  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  1516  006216  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  1517  006220  004767  012142          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  1518  006224  004767  013410                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  1519  006230  000002                          .WORD   2               ;ERROR TYPE CODE.
  1520  006232                          65$:
  1521  006232  062700  000002                  ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
  1522  006236  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  1523  006240  001363                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  1524  006242  004767  006736                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  1525
  1526                                  ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
  1527                                  ;*  DOWNWARDS WORD ADDRESSING.
  1528  006246  004467  006612                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1529  006252  004767  007576          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
  1530  006256  162700  000002          4$:     SUB     #2,     R0      ;DEC DATA BY 2
  1531  006262  014201                          MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
  1532  006264  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  1533  006266  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  1534  006270  004767  012046          66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
  1535  006274  004767  013340                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  1536  006300  000002                          .WORD   2               ;ERROR TYPE CODE.
  1537  006302                          67$:
  1538  006302  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  1539  006304  001364                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  1540  006306  004767  007362                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
                                        ;;************************************************************
                                        ;*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.
                                        ;;************************************************************
006312                                  TST2:
006312  004567  012312                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006316  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                        ;*  UPWARDS BYTE ADDRESSING.
006320  004467  006102                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006324  004767  007524          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006330  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
006332  005200                          INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
006334  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006336  001374                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006340  004767  006640                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                        ;*  DOWNWARDS BYTE ADDRESSING.
006344  004467  006514                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006350  004767  007500          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006354  005300                  4$:     DEC     R0              ;DEC DATA BY 1
006356  114201                          MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
006360  120001                          CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
006362  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006364  004767  011752          64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
006370  004767  013244                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006374  000003                          .WORD   3               ;ERROR TYPE CODE.
006376                                  65$:
006376  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006400  001365                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006402  004767  007266                  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.
                                        ;;************************************************************
006406                                  TST3:
006406  004567  012216                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006412  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                        ;*  DOWNWARDS WORD ADDRESSING.
006414  004467  006444                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006420  004767  007430          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006424  005100                          COM     R0              ;COMPLEMENT THE ADR
006426  062700  000002          2$:     ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
006432  010042                          MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
006434  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```



```
006436  001373                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006440  004767  007230                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
                                        ;*  UPWARDS WORD ADDRESSING.
006444  004467  005756                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006450  004767  007400          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006454  005100                          COM     R0              ;COMPLEMENT IT
006456                          4$:
006456  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006460  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006462  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006464  004767  011676          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006470  004767  013144                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006474  000002                          .WORD   2               ;ERROR TYPE CODE.
006476                          65$:
006476  162700  000002                  SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
006502  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006504  001364                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006506  004767  006472                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.

                                        ;;************************************************************
                                        ;*TEST 4         WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
                                        ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                        ;*      R1 = DATA READ FROM MEMORY (WAS)
                                        ;*      R2 = VIRTUAL ADDRESS
                                        ;*      R3 = NOT USED
                                        ;*      R4 = NOT USED
                                        ;*      R5 = BLOCK BOUNDRY BIT MASK.
                                        ;;************************************************************
006512                                  TST4:
006512  004567  012112                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006516  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                        ;*  UPWARDS BYTE ADDRESSING.
006520  004467  005702                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006524  004767  007400          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006530  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
006532  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006534  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006536  004767  006442                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
                                        ;*  UPWARDS BYTE ADDRESSING.
006542  004467  005660                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006546  004767  007356          3$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006552  112201                  4$:     MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
006554  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006556  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006560  004767  011564          64$.    JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
006564  004767  013050                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006570  000003                          .WORD   3               ;ERROR TYPE CODE.
006572                          65$:
006572  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006574  001366                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006576  004767  006402                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
```

```
                                ;;*****************************************************************
                                ;*TEST 5         WRITE 1'S COMPLEMENT OF BANK #.
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDRY BIT MASK.
                                ;;*****************************************************************
006602                          TST5:
006602  004567  012022                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006606  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                ;*  DOWNWARDS BYTE ADDRESSING.
006610  004467  006250                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006614  004767  007310          1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
006620  005100                          COM     R0              ;1'S COMPLEMENT OF BANK #
006622  110042          2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
006624  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006626  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006630  004767  007040                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN CAN BE READ.
                                ;*  DOWNWARDS BYTE ADDRESSING.
006634  004467  006224                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006640  004767  007264          3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
006644  005100                          COM     R0              ;SET 1'S COMPLEMENT OF BANK #
006646  114201          4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
006650  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006652  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006654  004767  011462          64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
006660  004767  012754                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006664  000003                          .WORD   3               ;ERROR TYPE CODE.
006666                          65$:
006666  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006670  001366                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006672  004767  006776                  JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```



```
                                .SBTTL SECTION 2:       WORST CASE NOISE TESTS
                                ;;*****************************************************************
                                ;*  THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
                                ;*  MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
                                ;;*****************************************************************
                                ;;*****************************************************************
                                ;*TEST 6         WRITE A CONSTANT INTO MEMORY.
                                        ;*  THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
                                ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                ;*      R1 = DATA READ FROM MEMORY (WAS)
                                ;*      R2 = VIRTUAL ADDRESS
                                ;*      R3 = NOT USED
                                ;*      R4 = NOT USED
                                ;*      R5 = BLOCK BOUNDRY BIT MASK.
                                ;;*****************************************************************
006676                          TST6:
006676  004567  011726                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006702  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006704  016700  172700          TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
006710  004467  005512                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006714  010022          1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
006716  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006720  001375                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006722  004767  006256                  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.
                                ;;*****************************************************************
006726                          TST7:
006726  004567  011676                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006732  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
006734  016700  172650                  MOV     .CONST, R0      ;GET USER CONSTANT
006740  004467  005462                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006744                          1$:
006744  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006746  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006750  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006752  004767  011410          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006756  004767  012656                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006762  000004                          .WORD   4               ;ERROR TYPE CODE.
006764                          65$:
006764  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006766  001366                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
006770  004767  006210                  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.
006774  032777  000400  172136          BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
007002  001416                          BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
007004  017746  172130                  MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
007010  042716  177740                  BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
007014  022726  000006                  CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
007020  001007                          BNE     TST10           ;BRANCH IF NOT TEST 6
007022  162767  000001  172052          SUB     #1,     $TSTNM  ;RESET TEST NUM
007030  162767  000030  172050          SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
```

```
  1745  007036  000722                    BR      TST6A           ;GO TO TEST 6
  1746
  1747                                ;;**************************************************************
  1748                                ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
  1749                                ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
  1750                                ;**************************************************************
  1751  007040                        TST10:
  1752  007040  004567  011564               JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  1753  007044  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
  1754  007046  016704  172562               MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
  1755  007052  004767  010560       1$:     JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
  1756  007056  012400                        MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
  1757  007060  001420                        BEQ     TST11           ;;BR IF END OF TABLE.
  1758  007062  004467  005340               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1759  007066  010012       2$:     MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
  1760  007070  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  1761  007072  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  1762  007074  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  1763  007076  004767  011264       64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  1764  007102  004767  012532               JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  1765  007106  000004                        .WORD   4               ;ERROR TYPE CODE.
  1766  007110                        65$:
  1767  007110  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  1768  007112  001365                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  1769  007114  004767  006064               JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
  1770  007120  000754                        BR      1$              ;BR BACK TO DO NEXT PATTERN
```



```
  1771
  1772                                ;;**************************************************************
  1773                                ;*TEST 11       ROTATE A "0" BIT THROUGH A FIELD OF ONES.
  1774  007122                        ;;**************************************************************
  1775  007122  004567  011502       TST11:  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  1776  007126  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
  1777  007130  012700  177777               MOV     #-1,    R0      ;SET CHECK WORD
  1778  007134  004767  007030               JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
  1779  007140  004467  005262               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1780  007144  000241       1$:     CLC                     ;CLEAR CARRY BIT IN PSW
  1781  007146  004767  007036               JSR     PC,     ROTATE
  1782  007152  016201  177776               MOV     -2(R2), R1      ;GET RESULT
  1783  007156  103402                        BCS     63$             ;BRANCH IF 'C' BIT WAS SET
  1784  007160  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  1785  007162  001405                        BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  1786  007164  004767  011176       63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  1787  007170  004767  012444               JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  1788  007174  000005                        .WORD   5               ;ERROR TYPE CODE.
  1789  007176                        64$:
  1790  007176  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  1791  007200  001361                        BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
  1792  007202  004767  005776               JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  1793
  1794                                ;;**************************************************************
  1795                                ;*TEST 12       ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
  1796                                ;;**************************************************************
  1797  007206                        TST12:
  1798  007206  004567  011416               JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  1799  007212  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
  1800  007214  005000                        CLR     R0              ;SET CHECK WORD
  1801  007216  004767  006746               JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
  1802  007222  004467  005200               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  1803  007226  000261       1$:     SEC                     ;SET 'C' BIT IN PSW
  1804  007230  004767  006754               JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
  1805  007234  016201  177776               MOV     -2(R2), R1      ;GET RESULT
  1806  007240  103002                        BCC     63$             ;BRANCH IF 'C' IS CLEAR
  1807  007242  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  1808  007244  001405                        BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  1809  007246  004767  011114       63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  1810  007252  004767  012362               JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  1811  007256  000005                        .WORD   5               ;ERROR TYPE CODE.
  1812  007260                        64$:
  1813  007260  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  1814  007262  001361                        BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
  1815  007264  004767  005714               JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;************************************************************
                                ;*TEST 13        3 XOR 9 TEST PATTERN.
                                ;;************************************************************
007270                          TST13:
007270  004567  011334                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007274  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007276  000167  000312                  JMP     TST14           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007302  005000                  .3X9:   CLR     R0              ;SET UP TEST DATA
007304  012703  177777                  MOV     #-1,    R3      ;SET COM DATA REG
007310  004467  005112                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007314  004767  006736          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007320  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007322  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007324  004767  005654                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;************************************************************
                                ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
                                ;;************************************************************
007330  005000                          CLR     R0              ;SET CHECK WORD
007332  004467  005070                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007336  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007342                          12$:
007342  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007344  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007346  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007350  004767  011012          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007354  004767  012260                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007360  000007                          .WORD   7               ;ERROR TYPE CODE.
007362                          65$:
007362  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007364  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007366  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007370  004767  010772          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007374  004767  012240                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007400  000007                          .WORD   7               ;ERROR TYPE CODE.
007402                          67$:
007402  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007404  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007406  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007410  004767  010752          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007414  004767  012220                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007420  000007                          .WORD   7               ;ERROR TYPE CODE.
007422                          69$:
007422  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007424  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007426  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007430  004767  010732          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007434  004767  012200                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007440  000007                          .WORD   7               ;ERROR TYPE CODE.
007442                          71$:
007442  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007444  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007446  001335                          BNE     12$
007450  005100                          COM     R0              ;COMPLEMENT CHECK WORD
```



```
007452  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007454  001330                          BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
007456  004767  005522                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                                ;;************************************************************
                                ;*  CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
                                ;;************************************************************
007462  005000                          CLR     R0
007464  004467  004736                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007470  012704  000100          21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007474  012703  000004          22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
007500                          23$:
007500  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007502  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007504  001405                          BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007506  004767  010654          72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007512  004767  012122                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007516  000007                          .WORD   7               ;ERROR TYPE CODE.
007520                          73$:
007520  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007522  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
007524  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007526  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007530  001405                          BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007532  004767  010630          74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007536  004767  012076                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007542  000007                          .WORD   7               ;ERROR TYPE CODE.
007544                          75$:
007544  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007546  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
007550  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007552  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007554  001405                          BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007556  004767  010[ 1          76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007562  004767  012052                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007566  000007                          .WORD   7               ;ERROR TYPE CODE.
007570                          77$:
007570  005303                          DEC     R3              ;DECREMENT 4 WORD COUNTER
007572  001342                          BNE     23$             ;BR IF NOT DONE.
007574  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007576  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007600  001335                          BNE     22$             ;BR IF NOT DONE.
007602  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007604  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007606  001330                          BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
007610  004767  005370                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 14       COMPLEMENT 3 XOR 9 TEST PATTERN
                                ;;*****************************************************************
007614                          TST14:
007614  004567  011010                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007620  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007622  000167  000316                  JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007626  012700  177777                  MOV     #-1,    R0      ;SET UP TEST DATA
007632  005003                          CLR     R3              ;SET COM DATA REG
007634  004467  004566                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007640  004767  006412          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007644  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007646  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007650  004767  005330                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.


                                ;;*****************************************************************
                                ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
007654  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007660  004467  004542                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007664  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007670                          12$:
007670  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007672  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007674  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007676  004767  010464          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007702  004767  011732                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007706  000007                          .WORD   7               ;ERROR TYPE CODE.
007710                          65$:
007710  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007712  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007714  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007716  004767  010444          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007722  004767  011712                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007726  000007                          .WORD   7               ;ERROR TYPE CODE.
007730                          67$:
007730  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007732  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007734  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007736  004767  010424          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007742  004767  011672                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007746  000007                          .WORD   7               ;ERROR TYPE CODE.
007750                          69$:
007750  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007752  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007754  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007756  004767  010404          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007762  004767  011652                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007766  000007                          .WORD   7               ;ERROR TYPE CODE.
007770                          71$:
007770  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007772  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007774  001335                          BNE     12$
```



```
007776  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010000  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010002  001330                          BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
010004  004767  005174                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.


                                ;;*****************************************************************
                                ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
                                ;;*****************************************************************
010010  012700  177777                  MOV     #-1,    R0      ;SET UP CHECK WORD.
010014  004467  004406                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010020  012704  000100          21$:    MOV     #64.,   P4      ;SET 256. WORD COUNTER
010024  012703  000004          22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
010030                          23$:
010030  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010032  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010034  001405                          BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010036  004767  010324          72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010042  004767  011572                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010046  000007                          .WORD   7               ;ERROR TYPE CODE.
010050                          73$:
010050  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010052  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
010054  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010056  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010060  001405                          BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010062  004767  010300          74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010066  004767  011546                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010072  000007                          .WORD   7               ;ERROR TYPE CODE.
010074                          75$:
010074  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010076  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
010100  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010102  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010104  001405                          BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010106  004767  010254          76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010112  004767  011522                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010116  000007                          .WORD   7               ;ERROR TYPE CODE.
010120                          77$:
010120  005303                          DEC     R3              ;DECREMENT 4 WORD COUNTER
010122  001342                          BNE     23$             ;BR IF NOT DONE.
010124  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010126  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
010130  001335                          BNE     22$             ;BR IF NOT DONE.
010132  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010134  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010136  001330                          BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
010140  004767  005040                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                        ;;*************************************************************
                        ;*TEST 15       MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
                        ;;*************************************************************
010144                  TST15:
010144  004567  010460          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010150  000777                  .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                        ; REQUIRED FOR THIS TEST.
010152  000167  000610          JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
010156  012700  000401          MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN
010162  012703  177777          MOV     #-1,    R3      ;SET COM DATA REG
010166  004467  004234          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010172  004767  006060  1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
010176  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010200  001374                  BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
010202  004767  004776          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                        ;;*************************************************************
                        ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                        ;;*************************************************************
010206  012700  000401          MOV     #401,   R0      ;RESET PARITY "ALL ZEROS" PATTERN.
010212  012703  177777          MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
010216  004467  004204          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010222  012704  000100  11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
010226                  12$:
010226  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010230  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010232  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010234  004767  010126  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010240  004767  011374          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010244  000007                  .WORD   7               ;ERROR TYPE CODE.
010246                  65$:
010246  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010250  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010252  001405                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010254  004767  010106  66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010260  004767  011354          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010264  000007                  .WORD   7               ;ERROR TYPE CODE.
010266                  67$:
010266  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010270  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010272  001405                  BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010274  004767  010066  68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010300  004767  011334          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010304  000007                  .WORD   7               ;ERROR TYPE CODE.
010306                  69$:
010306  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010310  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010312  001405                  BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010314  004767  010046  70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010320  004767  011314          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010324  000007                  .WORD   7               ;ERROR TYPE CODE.
010326                  71$:
010326  010046                  MOV     R0,     -(SP)   ;SAVE R0
010330  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
010332  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
```



```
010334  005304                  DEC     R4              ;COUNT 256. WORDS
010336  001333                  BNE     12$             ;BRANCH IF MORE
010340  010046                  MOV     R0,     -(SP)   ;SAVE R0
010342  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
010344  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
010346  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010350  001324                  BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
010352  004767  004626          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                        ;;*************************************************************
                        ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
                        ;;*************************************************************
010356  012700  000401          MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN.
010362  012703  177777          MOV     #-1,    R3      ;SET UP ALL ONES PATTERN.
010366  004467  004034          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010372  012704  000100  21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
010376                  22$:
010376  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010400  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010402  001405                  BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010404  004767  007756  72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010410  004767  011224          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010414  000007                  .WORD   7               ;ERROR TYPE CODE.
010416                  73$:
010416  005100                  COM     R0              ;COMPLEMENT CHECK WORD
010420  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
010422  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010424  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010426  001405                  BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010430  004767  007732  74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010434  004767  011200          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010440  000007                  .WORD   7               ;ERROR TYPE CODE.
010442                  75$:
010442  005100                  COM     R0              ;COMPLEMENT CHECK WORD
010444  005142                  COM     -(R2)           ;RESTORE DATA
010446  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010450  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010452  001405                  BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010454  004767  007706  76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010460  004767  011154          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010464  000007                  .WORD   7               ;ERROR TYPE CODE.
010466                  77$:
010466  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010470  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010472  001405                  BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010474  004767  007666  78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010500  004767  011134          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010504  000007                  .WORD   7               ;ERROR TYPE CODE.
010506                  79$:
010506  005100                  COM     R0              ;COMPLEMENT CHECK WORD
010510  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
010512  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010514  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010516  001405                  BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010520  004767  007642  80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010524  004767  011110          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
 2133  010530  000007                        .WORD   7               ;ERROR TYPE CODE.
 2134  010532                          81$:
 2135  010532  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2136  010534  005142                        COM     -(R2)           ;RESTORE DATA
 2137  010536  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2138  010540  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2139  010542  001405                        BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2140  010544  004767  007616          82$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2141  010550  004767  011064                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2142  010554  000007                        .WORD   7               ;ERROR TYPE CODE.
 2143  010556                          83$:
 2144  010556  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2145  010560  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2146  010562  001405                        BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2147  010564  004767  007576          84$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2148  010570  004767  011044                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2149  010574  000007                        .WORD   7               ;ERROR TYPE CODE.
 2150  010576                          85$:
 2151  010576  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2152  010600  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 2153  010602  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2154  010604  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2155  010606  001405                        BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2156  010610  004767  007552          86$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2157  010614  004767  011020                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2158  010620  000007                        .WORD   7               ;ERROR TYPE CODE.
 2159  010622                          87$:
 2160  010622  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2161  010624  005142                        COM     -(R2)           ;RESTORE DATA
 2162  010626  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2163  010630  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2164  010632  001405                        BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2165  010634  004767  007526          88$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2166  010640  004767  010774                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2167  010644  000007                        .WORD   7               ;ERROR TYPE CODE.
 2168  010646                          89$:
 2169  010646  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2170  010650  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2171  010652  001405                        BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2172  010654  004767  007506          90$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2173  010660  004767  010754                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2174  010664  000007                        .WORD   7               ;ERROR TYPE CODE.
 2175  010666                          91$:
 2176  010666  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2177  010670  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 2178  010672  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2179  010674  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2180  010676  001405                        BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2181  010700  004767  007462          92$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2182  010704  004767  010730                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2183  010710  000007                        .WORD   7               ;ERROR TYPE CODE.
 2184  010712                          93$:
 2185  010712  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2186  010714  005142                        COM     -(R2)           ;RESTORE DATA
 2187  010716  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2188  010720  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
```



```
 2189  010722  001405                        BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2190  010724  004767  007436          94$:  JSR     PC,     SPRNT2  ;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$.
 2222
 2223                                  ;;*****************************************************************
 2224                                  ;*  CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
 2225                                  ;;*****************************************************************
 2226  011030  012700  177777                MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
 2227  011034  012703  000401                MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
 2228  011040  004467  003362                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2229  011044  012704  000100          11$.  MOV     #64.,   R4      ;SET 256. WORD COUNTER
 2230  011050                          12$:
 2231  011050  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2232  011052  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2233  011054  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2234  011056  004767  007304          64$:  JSR     PC,     SPRNT2  ;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,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2242  011102  004767  010532                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2243  011106  000007                        .WORD   7               ;ERROR TYPE CODE.
 2244  011110                          67$:
```

```
 2245  011110  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2246  011112  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2247  011114  001405                        BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2248  011116  004767  007244        68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2249  011122  004767  010512                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2250  011126  000007                        .WORD   7               ;ERROR TYPE CODE.
 2251  011130                          69$:
 2252  011130  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2253  011132  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2254  011134  001405                        BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2255  011136  004767  007224        70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2256  011142  004767  010472                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2257  011146  000007                        .WORD   7               ;ERROR TYPE CODE.
 2258  011150                          71$:
 2259  011150  010046                        MOV     R0,     -(SP)   ;SAVE R0
 2260  011152  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
 2261  011154  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 2262  011156  005304                        DEC     R4              ;COUNT 256. WORDS
 2263  011160  001333                        BNE     12$             ;BRANCH IF MORE
 2264  011162  010046                        MOV     R0,     -(SP)   ;SAVE R0
 2265  011164  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
 2266  011166  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 2267  011170  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 2268  011172  001324                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
 2269  011174  004767  004004                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
 2270
 2271                                  ;;*****************************************************************
 2272                                  ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
 2273                                  ;;*****************************************************************
 2274  011200  012700  177777                MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
 2275  011204  012703  000401                MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
 2276  011210  004467  003212                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2277  011214  012704  000100        21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
 2278  011220                          22$:
 2279  011220  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2280  011222  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2281  011224  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2282  011226  004767  007134        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2283  011232  004767  010402                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2284  011236  000007                        .WORD   7               ;ERROR TYPE CODE.
 2285  011240                          73$:
 2286  011240  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2287  011242  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 2288  011244  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2289  011246  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2290  011250  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2291  011252  004767  007110        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2292  011256  004767  010356                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2293  011262  000007                        .WORD   7               ;ERROR TYPE CODE.
 2294  011264                          75$:
 2295  011264  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 2296  011266  005142                        COM     -(R2)           ;RESTORE DATA
 2297  011270  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2298  011272  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2299  011274  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2300  011276  004767  007064        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```



```
 2301  011302  004767  010332                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2302  011306  000007                        .WORD   7               ;ERROR TYPE CODE.
 2303  011310                          77$:
 2304  011310  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2305  011312  020001                        CMP     R0,     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        78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2308  011322  004767  010312                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2309  011326  000007                        .WORD   7               ;ERROR TYPE CODE.
 2310  011330                          79$:
 2311  011330  005100                        COM     R0              ;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     R0,     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        80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2317  011346  004767  010266                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2318  011352  000007                        .WORD   7               ;ERROR TYPE CODE.
 2319  011354                          81$:
 2320  011354  005100                        COM     R0              ;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     R0,     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        82$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2326  011372  004767  010242                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2327  011376  000007                        .WORD   7               ;ERROR TYPE CODE.
 2328  011400                          83$:
 2329  011400  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2330  011402  020001                        CMP     R0,     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        84$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2333  011412  004767  010222                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2334  011416  000007                        .WORD   7               ;ERROR TYPE CODE.
 2335  011420                          85$:
 2336  011420  005100                        COM     R0              ;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     R0,     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        86$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2342  011436  004767  010176                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2343  011442  000007                        .WORD   7               ;ERROR TYPE CODE.
 2344  011444                          87$:
 2345  011444  005100                        COM     R0              ;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     R0,     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        88$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 2351  011462  004767  010152                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2352  011466  000007                        .WORD   7               ;ERROR TYPE CODE.
 2353  011470                          89$:
 2354  011470  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 2355  011472  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2356  011474  001405                        BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

```
2357  011476  004767  006664          90$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
2358  011502  004767  010132          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2359  011506  000007                  .WORD   7               ;ERROR TYPE CODE.
2360  011510                          91$:
2361  011510  005100                  COM     R0              ;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     R0,     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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2368  011532  000007                  .WORD   7               ;ERROR TYPE CODE.
2369  011534                          93$:
2370  011534  005100                  COM     R0              ;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     R0,     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,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2377  011556  000007                  .WORD   7               ;ERROR TYPE CODE.
2378  011560                          95$:
2379  011560  010046                  MOV     R0,     -(SP)   ;SAVE R0
2380  011562  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
2381  011564  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
2382  011566  005304                  DEC     R4              ;DECREMENT 256. WORD COUNTER
2383  011570  001213                  BNE     22$             ;BRANCH IF MORE.
2384  011572  010046                  MOV     R0,     -(SP)   ;SAVE R0
2385  011574  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
2386  011576  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 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                                    ;*TEST 17       WORSE CASE NOISE PARITY BYTE TESTING
2392                                    ;*  CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
2393                                    ;*      1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
2394                                    ;*      2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
2395                                    ;*      3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
2396                                    ;*      4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
2397                                    ;;*****************************************************************
2398  011610                            TST17:
2399  011610  004567  007014                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
2400  011614  000000                    .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
2401  011616  005767  170454           WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
2402  011622  001404                    BEQ     1$              ;BR IF NO PARITY MEMORY.
2403  011624  032777  000100  167306    BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
2404  011632  001402                    BEQ     2$              ;BR IF NOT SET.
2405  011634  000167  000622           1$:     JMP     TST20           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
2406  011640  005000                    2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
2407  011642  004767  004322                   JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
2408  011646  004467  002554                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
2409  011652  036767  167666  167660    WWPBYT: BIT     BITPT,  PMEMAP  ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
2410  011660  001010                    BNE     2$              ;BR IF PARITY MEM.
2411  011662  036767  167660  167652           BIT     BITPT+2,PMEMAP+2 ;...HI 64K.
2412  011670  001004                    BNE     2$              ;BR IF PARITY MEM.
2413  011672  050502                    BIS     R5,     R2      ;POINT TO END OF BLOCK.
2414  011674  005202                    INC     R2              ;FIRST ADR OF NEXT BLOCK.
2415  011676  000167  000540           JMP     WWPB5           ;BR TO FIND NEXT BLOCK.
2416  011702  004767  005674           2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
2417  011706  004767  005724                   JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
2418  011712  020227  000114           WWPB1:  CMP     R2,     #114    ;CKECK IF POINTING TO PARITY ERROR VECTOR.
2419  011716  001004                    BNE     3$              ;BR IF NOT AT VECTOR.
2420  011720  062702  000004                   ADD     #4,     R2      ;SKIP PARITY VECTOR.
2421  011724  000167  000512           JMP     WWPB5           ;CHECK FOR BLOCK END.
2422  011730  111201                    3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
2423  011732  001405                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
2424  011734  004767  006352           64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
2425  011740  004767  007674                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
2426  011744  000011                    .WORD   11              ;ERROR TYPE CODE.
2427  011746                            65$:
2428  011746  105067  167606                   CLRB    DEFLG           ;CLEAR ODD/EVEN FLAG.
2429  011752  112700  000252                   MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
2430  011756  110012           WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
2431  011760  016703  167644                   MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
2432  011764  056773  167622  000000   10$:    BIS     WWP,@(R3)       ;SET WRITE WRONG PARITY.
2433  011772  052733  000001                   BIS     #AE,@(R3)+
2434  011776  005713                    TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
2435  012000  001371                    BNE     10$             ;BR IF MORE REGS IN TABLE.
2436                                    ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
2437  012002  110012                    MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
2438  012004  016703  167620                   MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
2439  012010  056733  167576           11$:    BIC     WWP,    @(R3)+  ;CLEAR WRITE WRONG PARITY.
2440  012014  005713                    TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
2441  012016  001374                    BNE     11$             ;BR IF MORE PARITY REGISTERS.
2442  012020  016737  167606  000114           MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
2443                                    ;*  DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
2444  012026  105412                    NEGB    (R2)            ;DATIP (DATOB AND COM PARITY BIT.)
2445                                    ;*  SHOULD HAVE TRAPPED TO PBTRP.
```

```
 2446  012030  016737  167602  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
 2447  012036  004767  006700          64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
 2448  012042  004767  007572                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2449  012046  000012                          .WORD   12              ;ERROR TYPE CODE.
 2450  012050  000562                          BR      WWPB4           ;SKIP TRAP SERVICE.
 2451
 2452                                  ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
 2453  012052  016737  167560  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
 2454  012060  022626                          CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
 2455  012062  016703  167540                  MOV     .MPR0,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
 2456  012066  032713  000001          21$:    BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER EXISTS.
 2457  012072  001003                          BNE     22$             ;BR IF IT DOESN'T EXIST.
 2458  012074  017301  000000                  MOV     @(R3),  R1      ;GET THE CONTENTS.
 2459  012100  100413                          BMI     23$             ;BR IF ERROR FLAG SET.
 2460  012102  062703  000010          22$:    ADD     #10,    R3      ;MOVE POINTER TO NEXT REG.
 2461  012106  020367  167516                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
 2462  012112  103765                          BLO     21$             ;BR IF MORE REGISTERS.
 2463  012114  004767  006222          64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
 2464  012120  004767  007514                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2465  012124  000013                          .WORD   13              ;ERROR TYPE CODE.
 2466  012126  000533                          BR      WWPB4           ;EXIT AFTER ERROR.
 2467  012130  036763  167410  000002  23$:    BIT     BITPT,  2(R3)   ;CHECK THE MAP FOR THIS REGISTER.
 2468  012136  001011                          BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
 2469  012140  036763  167402  000004          BIT     BITPT+2,4(R3)   ;CHECK THE HI 64K.
 2470  012146  001005                          BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
 2471  012150  004767  006162          65$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 2472  012154  004767  007460                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2473  012160  000014                          .WORD   14              ;ERROR TYPE CODE.
 2474  012162                          24$:
 2475  012162  010046                          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     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
 2485  012216  052700  100001          25$:    BIS     #BIT15+BIT0,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
 2486  012222  016367  000006  167256          MOV     6(R3),  RESRVD  ;GET APPROPIATE MASK.
 2487  012230  046700  167262                  BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2488  012234  046701  167256                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2489                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
 2490  012240  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2491  012242  001405                          BEQ     67$             ;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     @(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.
```



```
 2502  012302  001370                          BNE     26$             ;BR IF THIS PARITY REG DOESN'T EXIST.
 2503  012304  017301  000000                  MOV     @(R3),  R1      ;SAVE AND CHECK FOR ERROR FLAG.
 2504  012310  100365                          BPL     26$             ;BR IF NO ERROR FLAG.
 2505  012312  004767  006020          68$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 2506  012316  004767  007316                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2507  012322  000016                          .WORD   16              ;ERROR TYPE CODE.
 2508  012324  000757                          BR      26$             ;BR AFTER ERROR.
 2509  012326  111204                  WWPB3:  MOVB    (R2),   R4      ;GET THE DATA FOR CHECKING.
 2510                                  ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
 2511                                  ;*  ACTION ENABLE IS NOT SET IN CONTROLING REG, SO NO TRAP SHOULD OCCURE.
 2512  012330  111212                          MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
 2513                                  ;NOTE:  THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
 2514                                  ;       WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
 2515  012332  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 2516  012334  017301  000000                  MOV     @(R3),  R1      ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
 2517  012340  046701  167152                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 2518                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
 2519  012344  042700  000001                  BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
 2520  012350  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2521  012352  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2522  012354  004767  005756          64$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 2523  012360  004767  007254                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2524  012364  000015                          .WORD   15              ;ERROR TYPE CODE.
 2525  012366                          65$:
 2526  012366  012773  000001  000000          MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
 2527  012374  010401                          MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
 2528  012376  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 2529  012400  120001                          CMPB    R0,     R1      ;CHECK THE DATA.
 2530  012402  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2531  012404  004767  005732          66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
 2532  012410  004767  007224                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 2533  012414  000017                          .WORD   17              ;ERROR TYPE CODE.
 2534  012416                          67$:
 2535  012416  110012                  WWPB4:  MOVB    R0,     (R2)    ;RESTORE DATA.
 2536  012420  105712                          TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
 2537  012422  012700  000253                  MOV     #253,   R0      ;SET ODD PARITY DATA.
 2538  012426  105167  167126                  COMB    DEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
 2539  012432  100002                          BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
 2540  012434  000167  177316                  JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
 2541  012440  005202                  27$:    INC     R2              ;MOVE POINTER TO NEXT MEMORY BYTE.
 2542  012442  030502                  WWPB5:  BIT     R5,     R2      ;CHECK FOR END OF BLOCK.
 2543  012444  001402                          BEQ     30$             ;BR IF END OF BLOCK FOUND.
 2544  012446  000167  177240                  JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
 2545  012452  004767  002526          30$:    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
 2546  012456  004767  005054                  JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
                                        ;;*****************************************************************
                                        ;*TEST 20       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
                                        ;;*****************************************************************
012462                                  TST20:
012462  004567  006142                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012466  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
012470  010703                          RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
012472  042703  007777                          BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
012476  004467  001724                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012502  010246                          1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
012504  010346                                  MOV     R3,     -(SP)   ;SAVE "DATA" POINTER.
012506  012322                          2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
012510  032703  007777                          BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
012514  001002                                  BNE     3$              ;BRANCH IF MORE
012516  162703  010000                          SUB     #10000, R3      ;RESET POINTER TO START OF "RANDOM DATA"
012522  030502                          3$:     BIT     R5,     R2      ;CHECK FOR END OF BLOCK
012524  001370                                  BNE     2$              ;BRANCH IF MORE.
012526  012603                                  MOV     (SP)+,  R3      ;RESET "DATA" POINTER.
012530  012602                                  MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
012532  012300                          4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
012534  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
012536  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012540  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012542  004767  005620                  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
012546  004767  007066                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012552  000020                                  .WORD   20              ;ERROR TYPE CODE.
012554                                  65$:
012554  032703  007777                          BIT     #7777,  R3      ;CHECK FOR END OF "DATA TABLE"
012560  001002                                  BNE     5$              ;BR IF MORE.
012562  162703  010000                          SUB     #10000, R3      ;RESET POINTER TO TOP OF "DATA TABLE".
012566                                  5$:
012566  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012570  001360                                  BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
012572  004767  002405                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```



```
                                        .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
                                        ;;*****************************************************************
                                        ;*TEST 21       EXECUTE DATI, DATO THRU MEMORY.
                                        ;*  EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
                                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
                                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                        ;*
                                        ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                        ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                        ;*
                                        ;*   1ST PASS /  40000           010412          000205
                                        ;*  THRU TEST /  40002           000205          000205
                                        ;*
                                        ;*   2ND PASS /  40002           010412          000205
                                        ;*  THRU TEST /  40004           000205          000205
                                        ;*
                                        ;*      ETC., ETC., ETC.
                                        ;*
                                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                                        ;*  R2 = ADDRESS OF IUT/DATA.
                                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                        ;*  R4 = RTS R5 (CODE 205).
                                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                        ;;*****************************************************************
012576                                  TST21:
012576  004567  006026                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012602  000003                                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                        ; REQUIRED FOR THIS TEST.
012604  000167  000056                          JMP     TST22           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
012610  012703  010412                  DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
012614  012704  000205                          MOV     #205,   R4      ;GET 'RTS R5'
012620  010400                                  MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
012622  004467  001600                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012626  010322                          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012630  010412                          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012632  004542                                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
012634  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012636  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012640  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012642  004767  005514                  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012646  004767  006766                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012652  000021                                  .WORD   21              ;ERROR TYPE CODE.
012654                                  65$:
012654  010322                                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012656  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012660  001363                                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
012662  004767  002316                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                        ;;*****************************************************************
                        ;*TEST 22       EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
                        ;*  EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                        ;*
                        ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                        ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                        ;*
                        ;*   1ST PASS /  40000          110412          110605
                        ;*  THRU TEST /  40002          000205          000205
                        ;*
                        ;*   2ND PASS /  40002          110412          110605
                        ;*  THRU TEST /  40004          000205          000205
                        ;*
                        ;*       ETC., ETC., ETC.
                        ;*
                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                        ;*  R2 = ADDRESS OF IUT/DATA.
                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                        ;*  R4 = RTS R5 (CODE 205).
                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
                        ;;*****************************************************************
012666                  TST22:
012666  004567  005736          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012672  000003          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                ; REQUIRED FOR THIS TEST.
012674  000167  000060          JMP     TST23           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                ; AVAILABLE FOR TEST.
012700  012703  110412  DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
012704  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
012710  012700  110605          MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
012714  004467  001506          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012720  010322  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012722  010412  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012724  004562          JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
012726  012201          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012730  020001          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012732  001405          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012734  004767  005422  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012740  004767  006674          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012744  000021          .WORD   21              ;ERROR TYPE CODE.
012746                  65$:
012746  010322          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012750  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012752  001363          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
012754  004767  002224          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```



```
                        ;;*****************************************************************
                        ;*TEST 23       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
                        ;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(R2)' THROUGHOUT MEMORY.
                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                        ;*
                        ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                        ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                        ;*
                        ;*   1ST PASS /  40000          110342          161342
                        ;*  THRU TEST /  40002          000205          000205
                        ;*
                        ;*   2ND PASS /  40002          110342          161342
                        ;*  THRU TEST /  40004          000205          000205
                        ;*
                        ;*       ETC., ETC., ETC.
                        ;*
                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                        ;*  R1 = DATA READ FROM MEMORY (WAS).
                        ;*  R2 = ADDRESS OF IUT/DATA.
                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                        ;*  R4 = RTS R5 (CODE 205).
                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
                        ;;*****************************************************************
012760                  TST23:
012760  004567  005644          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012764  000003          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                ; REQUIRED FOR THIS TEST.
012766  000167  000064          JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                ; AVAILABLE FOR TEST.
012772  012703  110342  DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
012776  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
013002  012700  161342          MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
013006  004467  001414          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013012  010322  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013014  010412  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013016  004562  177776          JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
013022  005302          DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
013024  012201          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013026  020001          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013030  001405          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013032  004767  005324  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013036  004767  006576          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013042  000021          .WORD   21              ;ERROR TYPE CODE.
013044                  65$:
013044  010322          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013046  030502          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013050  001361          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013052  004767  002126          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************************
                                ;*TEST 24       EXECUTE DATI, DATIP, DATO THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          005412          172366
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          005412          172366
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;;*****************************************************************
013056                          TST24:
013056  004567  005546                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013062  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
013064  000167  000060          DIPDO:  JMP     TST25           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
013070  012703  005412          DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
013074  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
013100  012700  172366                  MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
013104  004467  001316                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013110  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013112  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013114  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013116  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013120  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013122  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013124  004767  005232          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013130  004767  006504                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013134  000021                  .WORD   21              ;ERROR TYPE CODE.
013136                          65$:
013136  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013140  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013142  001363                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013144  004767  002034                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```



```
                                ;;*****************************************************************
                                ;*TEST 25       EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION BICB (R2)+,-(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BICB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000          142242          142000
                                ;*  THRU TEST /  40002          000205          000205
                                ;*
                                ;*   2ND PASS /  40002          142242          142000
                                ;*  THRU TEST /  40004          000205          000205
                                ;*
                                ;*      ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;;*****************************************************************
013150                          TST25:
013150  004567  005454                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013154  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                        ; REQUIRED FOR THIS TEST.
013156  000167  000060                  JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                        ; AVAILABLE FOR TEST.
013162  012703  142242          DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
013166  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
013172  012700  142000                  MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
013176  004467  001224                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013202  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013204  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013206  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013210  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013212  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013214  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013216  004767  005140          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013222  004767  006412                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013226  000021                  .WORD   21              ;ERROR TYPE CODE.
013230                          65$:
013230  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013232  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013234  001363                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013236  004767  001742                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;*****************************************************
                                ;*TEST 26       EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
                                ;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
                                ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
                                ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
                                ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
                                ;*
                                ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
                                ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
                                ;*
                                ;*   1ST PASS /  40000           152212          157212
                                ;*  THRU TEST /  40002           000205          000205
                                ;*
                                ;*   2ND PASS /  40002           152212          157212
                                ;*  THRU TEST /  40004           000205          000205
                                ;*
                                ;*       ETC., ETC., ETC.
                                ;*
                                ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
                                ;*  R1 = DATA READ FROM MEMORY (WAS).
                                ;*  R2 = ADDRESS OF IUT/DATA.
                                ;*  R3 = INSTRUCTION UNDER TEST (IUT).
                                ;*  R4 = RTS R5 (CODE 205).
                                ;*  R5 = BLOCK BOUNDRY BIT MASK.
                                ;;*****************************************************
013242                          TST26:
013242  004567  005362                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013246  000003                       .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                              ; REQUIRED FOR THIS TEST.
013250  000167  000062               JMP     TST27           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                              ; AVAILABLE FOR TEST.
013254  012703  152212       DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
013260  012704  000205               MOV     #205,   R4      ;GET 'RTS R5'
013264  012700  157212               MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
013270  004467  001132               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013274  010322              1$:      MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013276  010412              2$:      MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013300  004542                       JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013302  005302                       DEC     R2              ;RESET R2 TO POINT TO IUT.
013304  012201                       MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013306  020001                       CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ
013310  001405                       BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013312  004767  005044       64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013316  004767  006316               JSR     PC,     SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013322  000021                       .WORD   21              ;ERROR TYPE CODE.
013324                          65$:
013324  010322                       MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013326  030502                       BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013330  001362                       BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013332  004767  001646               JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```



```
                                .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.
                                ;;*****************************************************
013336                          TST27:
013336  004567  005266               JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013342  000000                       .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013344  004467  001056               JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013350  010267  166240               MOV     R2,TEMP         ;SAVE BANK STARTING ADDRESS
013354  005003              1$:      CLR     R3              ;CLEAR PASS COUNTER
013356  012700  000377               MOV     #000377,R0      ;SETUP TO WRITE PATTERN
013362  010022              2$:      MOV     R0,(R2)+        ;WRITE PATTERN
013364  030502                       BIT     R5,R2           ;END OF 4K?
013366  001375                       BNE     2$              ;CONTINUE WRITING IF NO.
013370  014201              3$:      MOV     -(R2),R1        ;GET DATA WRITEN
013372  020001                       CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013374  001405                       BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013376  004767  004764       64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013402  004767  006232               JSR     PC,     SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013406  000010                       .WORD   10              ;ERROR TYPE CODE.
013410                          65$:
013410  000300              4$:      SWAB    R0              ;SWAP BYTES OF DATA
013412  010012                       MOV     R0,(R2) ;WRITE SWAPPED WORD
013414  011201                       MOV     (R2),R1         ;GET DATA WRITEN
013416  020001                       CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013420  001405                       BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013422  004767  004740       66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013426  004767  006206               JSR     PC,     SERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013432  000010                       .WORD   10              ;ERROR TYPE CODE.
013434                          67$:
013434  000300                       SWAP    R0              ;PUT DATA BACK TO ORINGINAL
013436  005703                       TST     R3              ;IF ON PASS 0 OR PASS 3
013440  001403                       BEQ     5$              ;WE ARE ADDRESSING DOWN
013442  020327  000003               CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
013446  001010                       BNE     6$              ;UPWARD
```

```
 2934  013450  030502                  5$:   BIT     R5,R2           ;DONE A PASS?
 2935  013452  001346                        BNE     3$              ;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     9$              ;IF YES BRANCH
 2939  013464  000300                        SWAB    R0              ;ELSE SET UP NEW READ WORD
 2940  013466  000404                        BR      7$              ;GO TO START OF ADDRESS UP
 2941  013470  062702  000002              6$:   ADD     #2,R2           ;UPDATE TO NEXT ADDRESS
 2942  013474  030502                        BIT     R5,R2           ;DONE A PASS
 2943  013476  001411                        BEQ     8$              ;IF YES BRANCH
 2944  013500  011201                  7$:   MOV     (R2),R1         ;GET DATA WRITTEN
 2945  013502  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 2946  013504  001405                        BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 2947  013506  004767  0046F1          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      4$
 2952  013522  005203                  8$:   INC     R3              ;INCREMENT      PASS COUNTER
 2953  013524  000300                        SWAB    R0              ;SET UP NEW READ WORD
 2954  013526  020327  000002              CMP     R3,#2           ;ADDRESSING UP?
 2955  013532  001316                        BNE     3$              ;IF NO GO TO DOWN SEQUENCE
 2956  013534  016702  166054                MOV     TEMP,R2         ;IF YES RESET ADDRESS TO START
 2957  013540  000757                        BR      7$              ;GO TO UP SEQUENCE
 2958  013542  004467  000660          9$:   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                                  ;*      R0=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                          TST30:
 2975  013552  004567  005052                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 2976  013556  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 2977  013560  004467  000642                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 2978  013564  012700  125252                MOV     #125252 ,R0     ;SETUP DATA PATTERN
 2979  013570  010022                  1$:   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     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 2983  013600  004767  001400                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 2984  013604  005003                        CLR     R3              ;SET UP COUNTER FOR STALL
 2985  013606  012704  000046                MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
 2986  013612  005303                  2$:   DEC     R3
 2987  013614  001376                        BNE     2$
 2988  013616  005304                        DEC     R4
 2989  013620  001374                        BNE     2$
```



```
 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          64$:  JSR     PC,     SPRNT2  ;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                                  ;*TEST 31       WRITE CHECKERBOARD STARTING WITH 052525 DATA
 3006                                  ;;*************************************************************
 3007  013664                          TST31:
 3008  013664  004567  004740                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 3009  013670  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 3010  013672  004467  000530                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 3011  013676  012700  052525                MOV     #052525,R0      ;SETUP DATA PATTERN
 3012  013702  010022                  1$:   MOV     R0,     (R2)+   ;WRITE A WORD
 3013  013704  005100                        COM     R0
 3014  013706  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 3015  013710  001374                        BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 3016  013712  004767  001266                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 3017  013716  005003                        CLR     R3              ;SET COUNTER FOR LOOP
 3018  013720  012704  000046                MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
 3019  013724  005303                  2$:   DEC     R3
 3020  013726  001376                        BNE     2$
 3021  013730  005304                        DEC     R4
 3022  013732  001374                        BNE     2$
 3023  013734  004467  000466                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 3024  013740  012700  052525                MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
 3025  013744                          3$:
 3026  013744  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 3027  013746  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 3028  013750  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 3029  013752  004767  004410          64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 3030  013756  004767  005656                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 3031  013762  000006                        .WORD   6               ;ERROR TYPE CODE.
 3032  013764                          65$:
 3033  013764  005100                        COM     R0
 3034  013766  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 3035  013770  001365                        BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
 3036  013772  004767  001206                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                 CLRB    $TSTNM          ;RESET TEST NUMBER,
 3043  014014  036767  164562  165512  1$:    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     $EOP            ;SKIP RELOCATION IF SWITCH SET.
 3049  014044  022767  000003  164530         CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IN FIRST 8K.
 3050  014052  001013                         BNE     4$              ;BR IF NOT IN FIRST 8K.
 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 8K.
 3060  014106  005737  000042                 TST     @#42            ;TEST FOR
 3061  014112  001402                         BEQ     6$              ;IF NOT R  .NG UNDER MON. DONT
 3062  014114  004767  003150          5$:    JSR     PC,     RESLDR  ;RESTORE L ADERS.
 3063  014120                           6$:
 3064  014120  004567  007366                 JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3065  014124  001201                         .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```



```
 3066                                   ;;*****************************************************************
 3067
 3068                                   .SBTTL  END OF PASS ROUTINE
 3069
 3070                                   ;*INCREMENT THE PASS NUMBER ($PASS)
 3071                                   ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
 3072                                   ;*IF THERES A MONITOR GO TO IT
 3073                                   ;*IF THERE ISN'T JUMP TO START1
 3074
 3075  014126                           $EOP:
 3076  014126  000240                          NOP
 3077  014130  005067  165034                  CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
 3078  014134  005267  165052                  INC     $PASS           ;;INCREMENT THE PASS NUMBER
 3079  014140  042767  100000  165044          BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
 3080  014146  005327                          DEC     (PC)+           ;;LOOP?
 3081  014150  000001                  $EOPCT: .WORD   1
 3082  014152  003040                          BGT     $DOAGN          ;;YES
 3083  014154  012737                          MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
 3084  014156  000001                  $ENDCT: .WORD   1
 3085  014160  014150                          $EOPCT
 3086  014162  004567  007324                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3087  014166  014260                          .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
 3088  014170  016746  165016                  MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
 3089                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
 3090                                   ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **$YSMAC**.
 3091  014174  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 3092  014200  004767  010226                  JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
 3093  014204  004567  007302                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3094  014210  014275                          .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
 3095  014212                           $GET42:
 3096
 3097  014212  016700  163624                  MOV     42,     R0              ;;GET MONITOR ADDRESS
 3098  014216  001416                          BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
 3099  014220  000005                          RESET                   ;;CLEAR THE WORLD
 3100  014222  004710                  $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
 3101  014224  000240                          NOP                     ;;SAVE ROOM
 3102  014226  000240                          NOP                     ;;FOR
 3103  014230  000240                          NOP                     ;;ACT11
 3104  014232  023737  000042  000046          CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
 3105  014240  001405                          BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
 3106  014242  105737  001224                  TSTB    @#$ENV          ;;CHECK FOR APT11
 3107  014246  001002                          BNE     $DOAGN          ;;IF APT11 THEN RESTART
 3108  014250  004767  003074                  JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
 3109  014254                           $DOAGN:
 3110  014254  000167  171624                  JMP     START1          ;;RETURN*****
 3111  014260  005015  047105  020104  $ENDMG: .ASCIZ  <15><12>/END PASS #/
 3112  014266  040520  051523  02.440
 3113  014274     000
 3114  014275     377     377     000  $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
 3115                                   .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
 3116                                   .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
 3117                                   ;;*****************************************************************
 3118                                   ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
 3119                                   ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
 3120                                   ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
 3121                                   ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
```

```
                                        ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                        ;;*************************************************************
014300                                  MMINIT:
014300  012737  077406  172300                  MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
014306  012737  077406  172302                  MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
014314  012737  077406  172304                  MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
014322  012737  077406  172306                  MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
014330  005037  172310                          CLR     @#KIPDR4
014334  005037  172312                          CLR     @#KIPDR5
014340  005037  172314                          CLR     @#KIPDR6
014344  012737  077406  172316                  MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
014352  005037  172340                          CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
014356  012737  000200  172342                  MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
014364  005037  172341                          CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
014370  005037  172346                          CLR     @#KIPAR3
014374  005037  172350                          CLR     @#KIPAR4
014400  005037  172352                          CLR     @#KIPAR5
014404  005037  172354                          CLR     @#KIPAR6
014410  012737  007600  172356                  MOV     #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
014416  012737  000001  177572                  MOV     #1,     @#SR0   ;ENABLE MEMORY MANAGEMENT
014424  000207                                  RTS     PC              ;RETURN

                                        ;;*************************************************************
                                        ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
                                        ;;*************************************************************
014426  012767  000001  165110  INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
014434  005067  165106                  CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
014440  005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
014442  016705  165140                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
014446  005767  164134                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
014452  001514                          BEQ     10$             ;BRANCH IF NO MEM MGMT
014454  005037  172344                  CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
014460  012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
014464  036767  165054  165036  1$:     BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
014472  001015                          BNE     2$              ;BRANCH IF MATCH
014474  036767  165046  165030          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
014502  001011                          BNE     2$              ;BRANCH IF MATCH
014504  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
014512  006367  165026                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
014516  006167  165024                  ROL     BITPT+2         ;...HI POINTER.
014522  100360                          BPL     1$              ;BR IF MORE.
014524  000000                          HALT            ;FATAL ERROR!!!  NO 4K BANK FOUND?
014526  036767  165012  165046  2$:     BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
014534  001004                          BNE     3$              ;BR IF LAST BANK.
014536  036767  165004  165040          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
014544  001405                          BEQ     4$              ;BR IF NOT LAST BANK.
014546  016705  165026          3$:     MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
014552  042767  020000  165016          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
014560  013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
014566  016767  164752  164754          MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
014574  016767  164746  164750          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
014602  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
014606  001505                          BEQ     21$             ;BRANCH IF NOT 8K.
014610  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
014616  006367  164726                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
```



```
014622  006167  164724                  ROL     TMPPT+2         ;...HI POINTER.
014626  100473                          BMI     20$             ;BR IF NO MORE.
014630  036767  164714  164672          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
014636  001004                          BNE     6$              ;BRANCH IF A MATCH.
014640  036767  164706  164664          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
014646  001760                          BEQ     5$              ;BRANCH IF NO MEMORY
014650  036767  164674  164724  6$:     BIT     TMPPT,  LADMAP  ;CHECK IF LAST BANK.
014656  001004                          BNE     7$              ;BRANCH IF A MATCH
014660  036767  164666  164716          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
014666  001455                          BEQ     21$             ;BR IF NOT LAST BANK.
014670  016705  164704          7$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
014674  052767  020000  164674          BIS     #20000, TMPLAD  ;MAKE SURE LAST ADDRESS IS IN BANK 3.
014702  000447                          BR      21$             ;BR TO FINISH UP.

014704  036767  164634  164616  10$:    BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED.
014712  001006                          BNE     11$             ;BR IF MATCH.
014714  062702  020000                  ADD     #20000, R2      ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
014720  106367  164620                  ASLB    BITPT           ;UPDATE BANK POINTER TO NEXT BANK.
014724  100367                          BPL     10$             ;BR IF MORE BANKS.
014726  000000                          HALT            ;FATAL ERROR!!!  NO 4K BANK FOUND?
014730  016767  164610  164612  11$:    MOV     BITPT, TMPPT    ;COPY BANK POINTER.
014736  036767  164602  164636          BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
014744  001021                          BNE     12$             ;BR IF LAST BANK.
014746  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR 8K BLOCK SIZE.
014752  001423                          BEQ     21$             ;BRANCH IF SMALLER BLOCK SIZE.
014754  106367  164570                  ASLB    TMPPT           ;POINT TO NEXT BANK.
014760  100416                          BMI     20$             ;BRANCH IF OVERFLOW.
014762  036767  164562  164540          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
014770  001412                          BEQ     20$             ;BRANCH IF NOT TO BE TESTED.
014772  112767  000011  164557          MOVB    #11,    FLAG8K  ;SET 8K BLOCK SIZE FLAG.
015000  036767  164544  164574          BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK.
015006  001403                          BEQ     20$             ;BR IF NOT LAST BANK.
015010  016705  164564          12$:    MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
015014  000402                          BR      21$             ;SKIP MASK RESET.
015016  012705  017777          20$:    MOV     #MASK4K,R5      ;RESET MASK TO 4K BLOCK SIZE.
015022  056767  164516  164520  21$:    BIS     BITPT,  TMPPT   ;SET TMPPT FOR FLAGING LAST BANK.
015030  056767  164512  164514          BIS     BITPT+2,TMPPT+2
015036  036767  164502  164524          BIT     BITPT,  FADMAP  ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
015044  001004                          BNE     22$             ;BR IF FIRST BANK.
015046  036767  164474  164516          BIT     BITPT+2,FADMAP+2 ;CHECK HI 64K.
015054  001450                          BEQ     INITEX          ;BR IF NOT FIRST BANK.
015056  016702  164502          22$:    MOV     TMPFAD, R2      ;RESET ADDRESS POINTER TO FIRST ADR.
015062  000445                          BR      INITEX

015064  016705  164516          INITDN: MOV     BLKMSK, R5      ;RESET R5 TO CURRENT BLOCK MASK.
015070  005002                          CLR     R2              ;INIT ADDRESS POINTER.
015072  005767  163510                  TST     MMAVA           ;CHECK FOR MEM MGMT
015076  001411                          BEQ     31$             ;BRANCH IF NO MEM MGMT
015100  012767  100000  164440          MOV     #BIT15, BITPT+2 ;SET POINTER TO TOP BIT
015106  005067  164432                  CLR     BITPT
015112  012737  007600  172344          MOV     #7600,  @#KIPAR2 ;SET PAR TO TOP OF MEM
015120  000403                          BR      32$             ;BRANCH TO COMMON AREA

015122  012767  000400  164414  31$:    MOV     #BIT8,  BITPT   ;SET UP BANK POINTER
015130  012767  015152  164416  32$:    MOV     #33$,   MMORE   ;SET "MMDOWN" EXIT ADDRESS.
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 8K!
 3236  015152  036767  164366  164422  33$:     BIT     BITPT,  LADMAP  ;CHECK FOR NON BOUNDRY LAST ADDR.
 3237  015160  001001                            BNE     34$             ;BR IF LAST BANK FLAG FOUND.
 3238  015162  036767  164360  164414            BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
 3239  015170  001402                            BEQ     INITEX          ;BR IF NO LAD FLG FOUND.
 3240  015172  016702  164376            34$:    MOV     LSTADR, R2      ;SET UP R2.
 3241  015176  010467  164352            INITEX: MOV     R4,     MMORE   ;PUT RETURN PC INTO "MMORE"
 3242  015202  000204                            RTS     R4              ;RETURN
 3243
 3244                                    ;;**********************************************************
 3245                                    ;*  COMMON UPWARDS ADDRESSING ROUTINE
 3246                                    ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
 3247                                    ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
 3248                                    ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
 3249                                    ;;**********************************************************
 3250  015204  036767  164340  164370  MMUP:   BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 3251  015212  001122                            BNE     10$             ;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  163052                    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$:     MOV     LADMSK, R5      ;RESET MASK.
 3271  015326  042767  020000  164242            BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
 3272  015334  016767  164204  164206  4$:      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 8K.
 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$:      ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
 3278  015372  006367  164152                    ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
 3279  015376  006167  164150                    ROL     TMPPT+2         ;...HI POINTER.
 3280  015402  100513                            BMI     30$             ;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, TMPLAD  ;SET VIRTUAL ADR TO BANK 3.
 3291  015456  000467                            BR      31$
 3292
 3293  015460  026702  164112            10$:    CMP     TMPLAD, R2      ;CHECK IF LAST ADR REACHED.
 3294  015464  001064                            BNE     31$             ;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$:    ADD     #20000, R2      ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
 3303  015514  106367  164044            22$:    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 TOFIND 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                                    ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
 3329  015660  005767  164412            32$:    TST     MPRX            ;CHECK FOR ANY PARITY REGISTERS PRESENT.
 3330  015664  001402                            BEQ     33$             ;BR IF NONE.
 3331  015666  004767  001744                    JSR     PC,     CKPMER  ;CHECK FOR PARITY MEMORY ERRORS.
 3332  015672  000207                    33$:    RTS     PC              ;STRAIGHT RETURN.
 3333
 3334                                    ;;**********************************************************
 3335                                    ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
 3336                                    ;*  FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
 3337                                    ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
 3338                                    ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
 3339                                    ;;**********************************************************
 3340  015674  036767  163644  163666  MMDOWN: BIT     BITPT,  FADMAP  ;CHECK FOR FIRST ADR FLAG.
 3341  015702  001004                            BNE     1$              ;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.
```

```
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$:   SUB     #200, @#KIPAR2  ;LOWER MEM MGMT PAR BY 4K
3350  015740  006067  163602                ROR     BITPT+2         ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
3351  015744  006067  163574                ROR     BITPT           ;...LO MAP.
3352  015750  103440                        BCS     10$             ;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 MORE  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     MMORE,  (SP)    ;RESET RETURN ADDRESS
3370  016052  000207                10$:   RTS     PC              ;RETURN
```



```
3371                                    .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
3372                                    ;;*************************************************************
3373                                    ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
3374                                    ;*  BITS 16 AND 17 ARE IN $TMP0.
3375                                    ;;*************************************************************
3376  016054  010200                    PHYADR: MOV     R2,     R0      ;VITRUAL INTO R0
3377  016056  005067  163076                    CLR     $TMP0           ;CLEAR TEMP SAVE OF HIGH BITS
3378  016062  005767  162520                    TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
3379  016066  001417                            BEQ     1$              ;BRANCH IF NO MEM MGMT
3380  016070  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3381  016072  013701  172344                    MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
3382  016076  006301                            ASL     R1              ;SHIFT IT 6 TIMES
3383  016100  006301                            ASL     R1
3384  016102  006301                            ASL     R1
3385  016104  006301                            ASL     R1
3386  016106  006301                            ASL     R1
3387  016110  006167  163044                    ROL     $TMP0           ;SAVE EXTRA BITS
3388  016114  006301                            ASL     R1
3389  016116  006167  163036                    ROL     $TMP0
3390  016122  060100                            ADD     R1,     R0      ;ADD SHIFTED PAR TO VIRTUAL
3391  016124  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
3392  016126  000207                    1$:     RTS     PC              ;RETURN
3393
3394                                    ;;*************************************************************
3395                                    ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
3396                                    ;;*************************************************************
3397  016130  005000                    BANKNO: CLR     R0              ;INIT R0
3398  016132  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3399  016134  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3400  016136  016701  163402                    MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
3401  016142  016702  163400                    MOV     BITPT+2,R2      ;...HI 64K.
3402  016146  006202                    1$:     ASR     R2              ;SHIFT POINTER...HI
3403  016150  006001                            ROR     R1              ;...LO
3404  016152  103403                            BCS     2$              ;BR WHEN POINTER FOUND.
3405  016154  105200                            INCB    R0              ;COUNT BANKS.
3406  016156  100373                            BPL     1$              ;BR IF NOT OVERFLOW.
3407  016160  000000                            HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
3408  016162                            2$:
3409  016162  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
3410  016164  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
3411  016166  000207                            RTS     PC              ;RETURN
3412
3413                                    ;;*************************************************************
3414                                    ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
3415                                    ;;*************************************************************
3416  016170                            SETCON:
3417  016170  004467  176232                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
3418  016174  010022                    2$:     MOV     R0,     (R2)+   ;MOV CONSTANT INTO MEMORY
3419  016176  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
3420  016200  001375                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
3421  016202  004767  176776                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
3422  016206  000207                            RTS     PC              ;RETURN
```

```
                                ;;*****************************************************************
                                ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
                                ;;*****************************************************************
016210  106112                ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
016212  106112                        ROLB    (R2)            ;(R2)=177775 OR 000002
016214  106112                        ROLB    (R2)            ;(R2)=177773 OR 000004
016216  106112                        ROLB    (R2)            ;(R2)=177767 OR 000010
016220  106112                        ROLB    (R2)            ;(R2)=177757 OR 000020
016222  106112                        ROLB    (R2)            ;(R2)=177737 OR 000040
016224  106112                        ROLB    (R2)            ;(R2)=177677 OR 000100
016226  106112                        ROLB    (R2)            ;(R2)=177777 OR 000000
016230  106122                        ROLB    (R2)+           ;(R2)=177577 OR 000200
016232  106112                        ROLB    (R2)            ;(R2)=177377 OR 000400
016234  106112                        ROLB    (R2)            ;(R2)=176777 OR 001000
016236  106112                        ROLB    (R2)            ;(R2)=175777 OR 002000
016240  106112                        ROLB    (R2)            ;(R2)=173777 OR 004000
016242  106112                        ROLB    (R2)            ;(R2)=167777 OR 010000
016244  106112                        ROLB    (R2)            ;(R2)=157777 OR 020000
016246  106112                        ROLL    (R2)            ;(R2)=137777 OR 040000
016250  106112                        ROLB    (R2)            ;(R2)=077777 OR 100000
016252  106122                        ROLB    (R2)+           ;(R2)=177777 OR 000000
016254  000207                        RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
                                ;;*****************************************************************
016256  012704  000020          W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS

016262  010022                  2$:     MOV     R0,(R2)+
016264  010022                          MOV     R0,(R2)+
016266  010022                          MOV     R0,(R2)+
016270  010022                          MOV     R0,(R2)+

016272  010322                          MOV     R3,(R2)+
016274  010322                          MOV     R3,(R2)+
016276  010322                          MOV     R3,(R2)+
016300  010322                          MOV     R3,(R2)+

016302  010022                          MOV     R0,(R2)+
016304  010022                          MOV     R0,(R2)+
016306  010022                          MOV     R0,(R2)+
016310  010022                          MOV     R0,(R2)+

016312  010322                          MOV     R3,(R2)+
016314  010322                          MOV     R3,(R2)+
016316  010322                          MOV     R3,(R2)+
016320  010322                          MOV     R3,(R2)+

016322  005304                          DEC     R4
016324  001356                          BNE     2$
016326  010046                          MOV     R0,     -(SP)   ;SAVE R0
016330  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
016332  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
016334  000207                          RTS     PC              ;RETURN
```



```
                                .SBTTL  RELOCATION SUBROUTINES.
                                ;;*****************************************************************
                                ;* ROUTINE TO RELOCATE PROGRAM CODE
                                ;;*****************************************************************
016336                          RELOC:
016336  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016340  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
016342  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
016344  012502                  4$:     MOV     (R5)+,  R2      ;GET FIRST LOCATION.
016346  012503                          MOV     (R5)+,  R3      ;GET FIRST LOCATION OF DESTINATION.
016350  012704  020000                  MOV     #20000, R4      ;SET UP 8K COUNTER.
016354  012223                  1$:     MOV     (R2)+,  (R3)+   ;MOV THE DATA.
016356  005304                          DEC     R4              ;COUNT THE WORDS.
016360  001375                          BNE     1$              ;BR IF MORE.
016362  012704  020000                  MOV     #20000, R4      ;RESET THE COUNTER.
016366  024243                  2$:     CMP     -(R2),  -(R3)   ;CHECK THE DATA JUST MOVED.
016370  001417                          BEQ     3$              ;BR IF DATA OK.
016372  011267  162526                  MOV     (R2),   $GDDAT  ;GET SOURCE DATA.
016376  011367  162524                  MOV     (R3),   $BDDAT  ;GET DESTINATION DATA.
016402  010267  162512                  MOV     R2,     $GDADR  ;GET SOURCE ADDRESS.
016406  010367  162510                  MOV     R3,     $BDADR  ;GET DESTINATION ADDRESS.
016412  004767  003222                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
016416  000023                          .WORD   23              ;ERROR TYPE CODE.
016420  000000                          HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
016422  162705  000004                  SUB     #4,     R5      ;ADJUST RETURN POINTER.
016426  000746                          BR      4$              ;GO BACK AND TRY AGAIN.
016430  005304                  3$:     DEC     R4              ;COUNT WORDS.
016432  001355                          BNE     2$              ;BR IF MORE.
016434  004567  005052                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
016440  026542                          .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                ;"PROGRAM RELOCATED TO "
016442  010346                          MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
016444  004767  006502                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
016450  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
016452  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
016454  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016456  000205                          RTS     R5              ;RETURN
                                ;;*****************************************************************
                                ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
                                ;;*****************************************************************
016460  022767  000003  152114  RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
016466  001401                          BEQ     1$              ;BR IF OK
016470  000000                          HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
016472                          1$:
016472  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
016474  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016476  005767  162104                  TST     MMAVA
016502  001465                          BEQ     10$
016504  012737  007600  172346          MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
016512  005000                          CLR     R0              ;INIT BANK POINTER...LO 64K
016514  012701  100000                  MOV     #BIT15, R1      ;...HI 64K.
016520  162737  000200  172346  2$:     SUB     #200,   @#KIPAR3        ;BACK DOWN ONE BANK.
016526  006001                          ROR     R1              ;MOVE POINTER...HI 64K.
016530  006000                          ROR     R0              ;..LO 64K.
016532  103500                          BCS     90$
016534  030167  162766                  BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
3533  016540  001003                        BNE     3$              ;BR IF AVAILABLE
3534  016542  030067  162756                BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
3535  016546  001764                        BEQ     2$              ;BR IF NO BANK FOUND.
3536  016550  013737  172346  172344  3$:   MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
3537  016556  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3538  016560  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3539  016562  162737  000200  172344  4$:   SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
3540  016570  006001                        RCR     R1              ;SHIFT POINTER.
3541  016572  006000                        ROR     R0              ;...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     R0,     MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
3546  016610  001764                        BEQ     4$              ;BR IF BANK DOESN'T EXIST.
3547  016612  052601                  6$:   BIS     (SP)+,  R1      ;GET SECOND BANK POINTER.
3548  016614  052600                        BIS     (SP)+,  R0      ;...LO 64K.
3549  016616  030067  161760                BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
3550  016622  001044                        BNE     90$             ;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     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
3555  016642  013737  172346  172342        MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
                ;*  PROGRAM SHOULD NOW BE EXICUTING OUT OF LAST TWO BANKS OF MEMORY.
3556
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$:  MOV     #BIT8,  R0      ;SET BANK POINTER TO TOP OF MEM.
3561  016662  005001                        CLR     R1              ;SET ADDRESS POINTER TO TOP.
3562  016664  162701  020000          11$:  SUB     #20000, R1      ;BACK DOWN ONE BANK.
3563  016670  006200                        ASR     R0              ;MOVE POINTER DOWN ONE BANK.
3564  016672  103420                        BCS     90$             ;BR IF OVERFLOW.
3565  016674  030067  162624                BIT     R0,     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     R0              ;MOV POINTER DOWN ONE BANK.
3569  016710  103411                        BCS     90$             ;BR IF OVERFLOW.
3570  016712  030067  162606                BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
3571  016716  001762                        BEQ     11$             ;BR TO START OVER IF NO LOWER BANK.
3572  016720  010046                        MOV     R0,     -(SP)   ;SAVE THE POINTER.
3573  016722  006300                        ASL     R0              ;RESET POINTER TO HI BANK.
3574  016724  052600                        BIS     (SP)+,  R0      ;SET BIT FOR LO BANK.
3575  016726  030067  161650                BIT     R0,     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                  13$:  .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
                ;*  PROGRAM NOW EXICUTING OUT OF TOP OF MEMORY.
3585
3586  016760  060106                        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,     @#ERRVEC ;ADJUST ERROR VECTOR.
```



```
3589  016772  060137  000024                ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3590  016776  060137  000114                ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3591  017002  026727  162132  177570        CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
3592  017010  001404                        BEQ     14$             ;BR IF HARDWARE SWITCH REGISTER.
3593  017012  060167  162122                ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
3594  017016  060167  162120                ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
3595  017022  062701  001622          14$:  ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
3596  017026  066721  161546          15$:  ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
3597  017032  005721                  16$:  TST     (R1)+           ;CHECK FOR INTERIM TERMINATOR.
3598  017034  001776                        BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
3599  017036  024127  177777                CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
3600  017042  001371                        BNE     15$             ;BR IF MORE IN TABLE.
3601  017044  010067  161532          30$:  MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
3602  017050  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
3603  017052  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3604  017054  066716  161520                ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
3605  017060  000207                        RTS     PC              ;RETURN
3606
3607                                  ;;**********  **********  **********
3608                                  ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
3609                                  ;;**********  **********  **********
3610  017062  032767  000003  161512  RELO:  BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
3611  017070  001401                        BEQ     1$              ;BR IF NO CONFLICT.
3612  017072  000000                        HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1!!!!
3613  017074  005767  161506          1$:   TST     MMAVA           ;CHECK FOR MEM MGMT.
3614  017100  001417                        BEQ     10$             ;BR IF NO MEMMGMT.
3615  017102  005037  172344                CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
3616  017106  012737  000200  172346        MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
3617  017114  004567  177216                JSR     R5,     RELOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
3618  017120  000000                        .WORD   0               ;SOURCE STARTING ADDRESS.
3619  017122  040000                        .WORD   40000           ;DESTINATION STARTING ADDRESS.
3620  017124  005037  172340                CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
3621  017130  012737  000200  172342        MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
                ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3622
3623  017136  000444                        BR      30$             ;BR TO COMMON EXIT.
3624
3625  017140  016746  161434          10$:  MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
3626  017144  011667  000004                MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
3627  017150  004567  177162                JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
3628  017154  000000                  20$:  .WORD   0               ;SOURCE STARTING ADDRESS.
3629  017156  000000                        .WORD   0               ;DESTINATION STARTING ADDRESS.
3630  017160  161607                        SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
                ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
3631
3632  017162  161667                        SUB     (SP),   SP      ;RESET THE STACK POINTER.
3633  017164  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3634  017166  012700  001622                MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
3635  017172  166620  000002          21$:  SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
3636  017176  005720                  22$:  TST     (R0)+           ;CHECK FOR TERMINATORS.
3637  017200  001776                        BEQ     22$             ;BR OVER TERMINATORS.
3638  017202  024027  177777                CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
3639  017206  001371                        BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
3640  017210  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
3641  017212  161637  000004                SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
3642  017216  161637  000024                SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
3643  017222  161637  000114                SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
3644  017226  026727  161706  177570        CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
```

```
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     RELOCF          ;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.
3653
3654                                  ;;*************************************************************
3655                                  ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
3656                                  ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
3657                                  ;*  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     @#SR0           ;DISABLE MEM MGMT.
3665  017312  012701  040000          2$:     MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
3666  017316  012702  002734                  MOV     #1500., R2      ;GET COUNTER.
3667  017322  014140                  3$:     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     @#SR0           ;ENABLE MEM MGMT.
3674  017346  000207                  4$:     RTS     PC              ;RETURN.
3675
3676                                  ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
3677  017350  005767  162144          SAVLDR: TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
3678  017354  001024                          BNE     4$              ;BRANCH IF ALREADY SAVED
3679  017356  012700  040000                  MOV     #40000, R0      ;GET END OF 8K
3680  017362  010001                          MOV     R0,     R1      ;GET END OF 8K
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$:     CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
3685  017400  012737  025114  000004          MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
3686  017406  010046                          MOV     R0,     -(SP)   ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
3687  017410  012702  002734                  MOV     #1500., R2      ;SET UP WORD COUNTER
3688  017414  014041          3$:     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                                  .SBTTL  PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
3694                                  ;;*************************************************************
3695                                  ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
3696                                  ;*  FIND OUT WHICH REGISTER DETECTED THE ERROR.
3697                                  ;*  THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
3698                                  ;;*************************************************************
3699  017430  011667  161466          PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
3700  017434  004567  004052                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3701  017440  026501                          .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
3702                                                                  ;"UNEXPECTED MEMORY PARITY TRAP."
3703  017442  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3704  017444  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3705  017446  016703  162156                  MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
3706  017452  005733          1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3707  017454  100415                          BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
3708  017456  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3709  017460  001374                          BNE     1$              ;BR IF MORE REGISTERS.
3710  017462  004767  002152                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3711                                  ;***ERROR***    NO REGISTER INDICATED ERROR
3712  017466  000024                          .WORD   24              ;ERROR TYPE CODE.
3713  017470  000417                          BR      4$              ;EXIT
3714  017472  005713          2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
3715  017474  001415                          BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
3716  017476  005733                          TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
3717  017500  100374                          BPL     2$              ;BR IF NO ERROR FLAG.
3718  017502  004567  004004                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3719  017506  026572                          .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
3720                                                                  ;"MORE THAN ONE ERROR FOUND."
3721  017510                          3$:
3722  017510  004767  000610          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3723  017514  004767  002120                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3724  017520  000025                          .WORD   25              ;ERROR TYPE CODE.
3725  017522  004767  000216                  JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
3726  017526  000761                          BR      2$              ;GO LOOK FOR MORE ERRORS.
3727  017530                          4$:
3728  017530  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
3729  017532  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
3730  017534  000002                          RTI                     ;RETURN.
3731
3732                                  ;;*************************************************************
3733                                  ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
3734                                  ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
3735                                  ;;*************************************************************
3736  017536  005767  162534          MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
3737  017542  001434                          BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
3738  017544  032777  000100  161366          BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
3739  017552  001030                          BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
3740  017554  005767  161020                  TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
3741  017560  001410                          BEQ     SETAE           ;BR IF PROG IN BANK 0.
3742  017562  032777  000040  161350          BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
3743  017570  001004                          BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
3744  017572  026727  161764  001000          CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
3745  017600  103415                          BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
```

```
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  1624[ ;        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,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3770  017702  004767  001732          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3771  017706  000026                  .WORD   26              ;ERROR TYPE CODE.
3772  017710  000411                  BR      3$
3773  017712  004767  000026          JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
3774  017716                  2$:
3775  017716  004767  000402          65$:    JSR     PC,     SPRNTQ  ;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:
3790  017744  010046                  MOV     R0,-(SP)        ;;PUSH R0 ON STACK
3791  017746  010146                  MOV     R1,-(SP)        ;;PUSH R1 ON STACK
3792  017750  010246                  MOV     R2,-(SP)        ;;PUSH R2 ON STACK
3793  017752  010346                  MOV     R3,-(SP)        ;;PUSH R3 ON STACK
3794  017754  010446                  MOV     R4,-(SP)        ;;PUSH R4 ON STACK
3795  017756  013746  000114          MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
3796  017762  013746  000116          MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
3797  017766  004567  003520          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3798  017772  026636                  .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
3799                                                          ;"SCANNING MEMORY FOR BAD PARITY."
3800  017774  012700  000001          MOV     #BIT0,  R0      ;SET BIT POINTER TO FIRST BANK.
3801  020000  005001                  CLR     R1              ;CLR HI 64K POINTER.
3802  020002  005002                  CLR     R2              ;INIT ADDRESS POINTER.
3803  020004  005004                  CLR     R4              ;INIT ERROR DETECTED FLAG.
3804  020006  004767  000256          JSR     PC,     CLRPAR  ;CLEAR THE PARITY REGISTERS.
3805  020012  012737  000116  000114  MOV     #116,   @#114   ;HALT IF ANOTHER PARITY TRAP.
3806  020020  005037  000116          CLR     @#116
3807  020024  005767  160556          TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
3808  020030  001406                  BEQ     1$              ;BR IF NO MEM MGMT.
3809  020032  013746  172344          MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
3810  020036  005037  172344          CLR     @#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          1$:     BIT     R0,     MEMMAP  ;CHECK IF THIS BANK OF MEM EXISTS.
3813  020052  001003                  BNE     2$              ;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$:     TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
3821  020074  100024                  BPL     6$              ;BR IF NO EPROR 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,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
3828  020114  004767  001520          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
3829  020120  000030                  .WORD   30              ;ERROR TYPE CODE.
3830  020122  111212                  MOVB    (R2),   (R2)    ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
3831  020124  005053                  CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
3832  020126  105712                  TSTB    (R2)            ;CHECK IF THE PARITY ERROR WAS CLEARED.
3833  020130  005733                  TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
3834  020132  100005                  BPL     6$              ;BR IF IT IS OK.
3835  020134  004567  003352          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
3836  020140  026700                  .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
3837                                                          ;"PARITY ERROR WILL NOT CLEAR."
3838  020142  005073  177776          CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
3839  020146  005713          6$:     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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3861  020242  025706                        .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
 3862  020244                            21$:
 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                                  ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
 3874                                  ;;**********.**********.**********.**********.**********
 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 **SYSMAC**.
 3890                                  ;;**********.**********.**********.**********.**********
 3891  020312  010267  160602          SPRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
 3892  020316  005067  160602                  CLR     $GDDAT          ;SHOULD BE DATA IS "0".
 3893  020322  000430                        BR      SPRNTB
 3894
 3895  020324  014367  16063C          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      SPRNT0
 3898
 3899  020336  011367  160616          SPRNTP: MOV     (R3),   $TMP0   ;GET THE PARITY REGISTER ADDRESS.
 3900  020342  010267  160552          SPRNT0: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
```



```
 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,     $TMP0   ;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,     $GDDAT  ;GET WHAT THE DATA WAS
 3912  020410  000207                        RTS     PC              ;RETURN TO ENTER ERROR ROUTINES
 3913
 3914                                  ;;**********.**********.**********.**********.**********
 3915                                  ;*  SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
 3916                                  ;*  R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
 3917                                  ;;**********.**********.**********.**********.**********
 3918  020412  005710                  TYPMAP: TST     (R0)            ;CHECK IF ANY MEMORY IN MAP...LO 64K.
 3919  020414  001007                        BNE     1$              ;BR IF MEMORY IN MAP.
 3920  020416  005760  000002                TST     2(R0)           ;...HI 64K.
 3921  020422  001004                        BNE     1$              ;BR IF MEMORY IN MAP.
 3922  020424  004567  003062                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3923  020430  026266                        .WORD   NOMEM           ;ADDRESS OF MESSAGE TO BE TYPED
 3924                                                                ;"NO MEMORY FOUND."
 3925  020432  000475                        BR      6$              ;EXIT
 3926  020434                            1$:
 3927  020434  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 3928  020436  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 3929  020440  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 3930  020442  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 3931  020444  012701  000001                MOV     #BIT0,  R1      ;SET UP BANK POINTER...LO 64K.
 3932  020450  005002                        CLR     R2              ;...HI 64K.
 3933  020452  012703  177777                MOV     #-1,    R3      ;SET UP ADDRESS POINTER TO -1.
 3934  020456  010304                        MOV     R3,     R4      ;HI BITS OF ADDRESS AS WELL.
 3935  020460  030110                  2$:   BIT     R1,     (R0)    ;CHECK THE MAP FOR THIS BANK.
 3936  020462  001014                        BNE     3$              ;BR IF THIS BANK PRESENT.
 3937  020464  030260  000002                BIT     R2,     2(R0)   ;CHECK HI 64K MAP.
 3938  020470  001011                        BNE     3$              ;BR IF THIS BANK PRESENT.
 3939  020472  105703                        TSTB    R3              ;CHECK FOR PREVIOUS PRINTOUT.
 3940  020474  001042                        BNE     5$              ;BR IF ALREADY TYPED "TO"
 3941  020476  162703  000001                SUB     #1,     R3      ;BACK UP TO LAST ADR OF PREVIOUS BANK.
 3942  020502  005604                        SBC     R4              ;...HI ADDRESS BITS.
 3943  020504  004567  003002                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3944  020510  025517                        .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
 3945  020512  000410                        BR      4$              ;GO TO TYPE THE ADDRESS.
 3946  020514  105703                  3$:   TSTB    R3              ;CHECK FOR PREVIOUS TYPEOUT.
 3947  020516  001431                        BEQ     5$              ;BR IF ALREADY TYPE "FROM".
 3948  020520  062703  000001                ADD     #1,     R3      ;POINT TO FIRST ADDRESS OF THIS BANK.
 3949  020524  005504                        ADC     R4              ;...HI BITS OF ADDRESS.
 3950  020526  004567  002760                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 3951  020532  025507                        .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
 3952  020534                            4$:
 3953  020534  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 3954  020536  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 3955  020540  006303                        ASL     R3              ;BIT 15 INTO C-BIT
 3956  020542  006104                        ROL     R4              ;BIT 15 INTO R4.
```

```
                                        ROR     R3              ;RESTORE BITS 14-0.
                                        MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
                                                                ;;TYPE ADDRESS BITS 21-15
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
                                        JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
                                        .BYTE   3               ;;TYPE 3 DIGIT(S)
                                        .BYTE   0               ;;SUPPRESS LEADING ZEROS
                                        MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
                                                                ;;TYPE ADDRESS BITS 14-0
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
                                        JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
                                        .BYTE   5               ;;TYPE 5 DIGIT(S)
                                        .BYTE   1               ;;TYPE LEADING ZEROS
                                        MOV     (SP)+,R4        ;;POP STACK INTO R4
                                        MOV     (SP)+,R3        ;;POP STACK INTO R3
                                5$:     ADD     #20000, R3      ;UPDATE TO NEXT BANK.
                                        ADC     R4              ;...HI ADDRESS BITS.
                                        ASL     R1              ;SHIFT POINTER...LO 64K.
                                        ROL     R2              ;...HI 64K.
                                        BCC     2$              ;BR IF MORE BANKS.
                                        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
                                6$:     RTS     PC              ;RETURN.

                                .SBTTL  SCOPE HANDLER ROUTINE

                                ;;**************************************************************
                                ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW14=1         LOOP ON TEST
                                ;*SW11=1         INHIBIT ITERATIONS
                                ;*SW09=1         LOOP ON ERROR
                                ;*SW08=1         LOOP ON TEST IN SWR<4:0>
                                ;*CALL
                                ;*      SCOPE           ;;SCOPE=IOT

                                $SCOPE:
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
                                        MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
                                        JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
                                        MOV     (R5)+,  R4      ;SAVE MINIMUM BLOCK MASK NEXT TEST.
                                        MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
                                1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
                                        BNE     $OVER           ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
                                $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
```



```
                                        MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
                                        MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
                                        TST     @#177060        ;;TIME OUT ON XOR?
                                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
                                        BR      $SVLAD          ;;GO TO THE NEXT TEST
                                5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
                                        MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
                                        BR      7$              ;;LOOP ON THE PRESENT TEST
                                6$:;#####END OF CODE FOR THE XOR TESTER#####
                                        BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
                                        BEQ     2$              ;;BR IF NO
                                        MOV     @$WR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
                                        BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
                                        CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
                                        BEQ     $OVER           ;;BR IF YES
                                2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
                                        BEQ     3$              ;;BR IF NO
                                        CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
                                        BHI     3$              ;;BR IF NO
                                        BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
                                        BEQ     4$              ;;BR IF NO
                                7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
                                        BR      $OVER
                                4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
                                        CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
                                        BR      1$              ;;ESCAPE TO THE NEXT TEST
                                3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
                                        BNE     1$              ;;BR IF YES
                                        TST     $PASS           ;;IF FIRST PASS OF PROGRAM
                                        BEQ     1$              ;;       INHIBIT ITERATIONS
                                        INC     $ICNT           ;;INCREMENT ITERATION COUNT
                                        CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
                                        BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
                                1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
                                        MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
                                $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
                                        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
                                        MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
                                        MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
                                        CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
                                        MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
                                $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
                                        MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
                                INSERT: CMP     R5,     (SP)    ;CHECK FOR LOOP ON TEST.
                                        BEQ     1$              ;BR IF START NEXT TEST.
                                        JMP     ENDINS          ;JMP IF LOOP ON LAST TEST.
                                1$:     MOV     #37777, BLKMSK  ;SET 8K BOUNDRY MASK.
                                        TST     $PASS           ;CHECK FOR PASS 0.
                                        BEQ     2$              ;BR IF PASS 0
                                        CMPB    $TSTNM, #21     ;CHECK IF IN SECTION 3.
                                        BHIS    3$              ;BR IF IN SECTION 3.
                                2$:     ASR     BLKMSK          ;RESET BOUNDRY TO 4K.
                                3$:     MOV     FSTADR, TMPFAD  ;GET FIRST ADDRESS.
                                        TST     RELOCF          ;CHECK IF PRG RELOCATED.
                                        BEQ     4$              ;BR IF NOT RELOCATED.
                                        BIT     #SW05,  @SWR    ;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, TMPLAD  ;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 PROGRAM.
 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 BOUNDRY.
 4095  021356  001416                          BEQ     11$             ;BR IF FIRST ADR ON BLOCK BOUNDRY.
 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 BOUNDRY.
 4099  021376  001006                          BNE     11$             ;BR IF NOT ON 4K BOUNDRY.
 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$:    BIT     R4,     TMPLAD  ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
 4103  021420  001414                          BEQ     12$             ;BR IF ON BLOCK BOUNDRY.
 4104  021422  040467  160150                  BIC     R4,     TMPLAD  ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
 4105  021426  032767  017777  160142          BIT     #MASK4K,TMPLAD  ;CHECK IF ADJUSTED TO 4K BOUNDRY.
 4106  021434  001006                          BNE     12$             ;BR IF NOT ON 4K BOUNDRY.
 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$:    CMP     TMPLAD, TMPFAD  ;CHECK IF ANY MEMORY LEFT.
 4114  021500  101406                          BLOS    15$             ;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  021532  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4128  021536  004767  003122                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
 4129  021542      003                          .BYTE   3               ;TYPE 3 DIGITS.
 4130  021543      001                          .BYTE   1               ;TYPE LEADING ZEROS.
 4131  021544  000427                          BR      ENDINS          ;RETURN TO SKIP TEST.
 4132  021546  062716  000004          16$:    ADD     #4,     (SP)    ;SKIP THE SKIP ON RETURN.
 4133  021552  062767  000004  157326          ADD     #4,     $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
 4134  021560  012767  017777  160000  20$:    MOV     #MASK4K,FADMSK  ;GET 4K MASK.
 4135  021566  016705  157772                  MOV     TMPFAD, R5      ;GET FIRST ADR.
 4136  021572  040567  157770          21$:    BIC     R5,     FADMSK  ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
 4137  021576  006305                          ASL     R5              ;MOVE LOWEST BIT UP ONE.
 4138  021600  001374                          BNE     21$             ;LOOP UNTIL OVERFLOW.
 4139  021602  012767  017777  157770          MOV     #MASK4K,LADMSK  ;SET MASK BITS
 4140  021610  016705  157762                  MOV     TMPLAD, R5      ;GET LAST ADR.
 4141  021614  040567  157760          22$:    BIC     R5,     LADMSK  ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
 4142  021620  006305                          ASL     R5              ;MOVE LOWEST BIT OF LAST ADR UP ONE.
 4143  021622  001374                          BNE     22$             ;LOOP UNTIL OVERFLOW.
 4144  021624  000207                  ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
 4145  021626  000004                  $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  021630  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4149  021634  004767  000524                  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=EMT AND N=ERROR ITEM NUMBER
 4163
 4164  021640                  $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  021640  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4168  021644  004767  000514                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
 4169  021650  062716  000002                  ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
 4170  021654  105267  157223          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
 4171  021660  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
 4172  021662  016777  157214  157252          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
 4173  021670  032777  002000  157242          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
 4174  021676  001403                          BEQ     1$              ;;NO - SKIP
 4175  021700  004567  001606                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4176  021704  001174                          .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
 4177  021706  005267  157200          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
 4178  021712  011667  157200                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
 4179  021716  162767  000002  157172          SUB     #2,$ERRPC
 4180  021724  116767  157166  157162          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
```

```
 4181  021732  032777  020000  157200          BIT     #BIT13,@SWR     ;;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4185  021752  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4186  021754                          20$:
 4187  021754  122767  000001  157242          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 4188  021762  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
 4189  021764  116767  157124  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
 4190  021772  004767  002044                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
 4191  021776     000                  21$:    .BYTE   0
 4192  021777     000                          .BYTE   0
 4193  022000  000777                  22$:    BR      22$             ;;APT ERROR LOOP
 4194  022002  005777  157100          2$:     TST     @SWR            ;;HALT ON ERROR
 4195  022006  100005                          BPL     3$              ;;SKIP IF CONTINUE
 4196  022010  000000                          HALT                    ;;HALT ON ERROR!
 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,@SWR     ;;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                  MOV     $ESCAPE.(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
 4207  022050                          5$:
 4208  022050  022737  014222  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
 4209  022056  001001                          BNE     6$              ;;BRANCH IF NO
 4210  022060  000000                          HALT                    ;;YES
 4211  022062                          6$:
 4212  022062  000207                          RTS     PC
 4213                                  ;;**************************************************************
 4214
 4215                                  .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
 4216
 4217                                  ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($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  022064                          $ERRTYP:
 4222  022064  004567  001422                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4223  022070  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4224  022072  010046                          MOV     R0,-(SP)                ;SAVE R0
 4225  022074  005000                          CLR     R0                      ;PICKUP THE ITEM INDEX
 4226  022076  156700  157012                  BISB    $ITEMB,R0
 4227  022102  001007                          BNE     1$                      ;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  156444  157356          SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
 4238  022136  005300                          DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
 4239  022140  006300                          ASL     R0                      ;       WORK FOR THE ERROR TABLE
 4240  022142  006300                          ASL     R0
 4241  022144  006300                          ASL     R0
 4242  022146  066700  157466                  ADD     .ERRTB, R0      ;FORM TABLE POINTER
 4243  022152  012067  000006                  MOV     (R0)+,2$                ;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     (R0)+,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" POINTER 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     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
 4257  022222  001451                          BEQ     9$                      ;BR IF NO DATA TO BE TYPED
 4258  022224  066701  156450                  ADD     RELOCF, R1      ;ADJUST POINTER
 4259  022230  012000                          MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
 4260  022232  066700  156342                  ADD     RELOCF, R0      ;ADJUST POINTER.
 4261  022236  105720          6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
 4262  022240  001006                          BNE     7$                      ;BR IF NOT 16-BIT OCTAL
 4263  022242  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(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     @#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     @(R1)+,-(SP)    ;;SAVE @(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     @#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  177777  17$:     CMPB    #-1,    -1(R0)  ;CHECK FOR 18-BIT ADDRESS FORMAT.
 4277  022302  001004                          BNE     18$             ;BR IF NOT 18-BIT ADDRESS FORMAT.
 4278  022304  013146                          MOV     @(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    @(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     @#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.
```

```
 4293  022342  022362                        .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
 4294  022344  000734                        BR      6$                      ;LOOP
 4295
 4296  022346  012601                9$:     MOV     (SP)+,R1                ;RESTORE R1
 4297  022350  012600                10$:    MOV     (SP)+,R0                ;RESTORE R0
 4298  022352  004567  001134              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4299  022356  001201                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4300  022360  000207                        RTS     PC                      ;RETURN
 4301  022362  000011                11$:    .ASCIZ  /       /       ;TAB CHARACTER.
 4302                                        .EVEN
 4303                                .SBTTL  TTY INPUT ROUTINE
 4304
 4305                                ;;**********************************************************
 4306                                .ENABL  LSB
 4307
 4308                                ;;**********************************************************
 4309                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
 4310                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
 4311                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
 4312                                ;*WHEN OPERATING IN TTY FLAG MODE.
 4313  022364  022767  000176  156546 $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
 4314  022372  001104                        BNE     15$             ;;BRANCH IF NO
 4315  022374  105777  156544              TSTB    @$TKS           ;;CHAR THERE?
 4316  022400  100101                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
 4317  022402  117746  156540              MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
 4318  022406  042716  177600              BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
 4319  022412  022726  000007              CMP     #7,(SP)+        ;;IS IT A CONTROL G?
 4320  022416  001072                        BNE     15$             ;;NO, RETURN TO USER
 4321  022420  126727  156510  000001      CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
 4322  022426  001466                        BEQ     15$             ;;BRANCH IF YES
 4323
 4324  022430  004567  001056              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4325  022434  023311                        .WORD   $CNTLG          ;ADDRESS OF MESSAGE TO BE TYPED
 4326  022436                        $GTSWR:
 4327  022436  004567  001050              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4328  022442  023316                        .WORD   $MSWR           ;ADDRESS OF MESSAGE TO BE TYPED
 4329  022444  016746  155526              MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
 4330                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
 4331                                ;*  WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4332  022450  013746  177776              MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4333  022454  004767  002230              JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
 4334  022460  004567  001026              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4335  022464  023327                        .WORD   $MNEW           ;ADDRESS OF MESSAGE TO BE TYPED
 4336  022466  005046                19$:    CLR     -(SP)           ;;CLEAR COUNTER
 4337  022470  005046                        CLR     -(SP)           ;;THE NEW SWR
 4338  022472  105777  156446          7$:     TSTB    @$TKS           ;;CHAR THERE?
 4339  022476  100375                        BPL     7$              ;;IF NOT TRY AGAIN
 4340
 4341  022500  117746  156442              MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
 4342  022504  042716  177600              BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 4343
 4344
 4345
 4346  022510  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
 4347  022514  001006                        BNE     10$             ;;BRANCH IF NOT
 4348  022516  004567  000770              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```



```
 4349  022522  023304                        .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
 4350  022524  062706  000006          20$:    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$:    ADD     #6,SP           ;;CLEAR UP STACK
 4360  022560                          14$:
 4361  022560  004567  000726              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4362  022564  001201                        .WORD   $CRLF           ;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,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
 4366  022604  000002                15$:    RTI                     ;;RETURN
 4367  022606  004767  001142          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
 4368  022612  021627  000060              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  042716  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                          18$:
 4382  022660  004567  000626              JSR     R5,     $PRINT  ;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                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
 4390                                ;*CALL:
 4391                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
 4392                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
 4393                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
 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    @$TKS           ;;WAIT FOR
 4399  022704  100375                        BPL     1$              ;;A CHARACTER
 4400  022706  117766  156234  000004      MOVB    @$TKB,4(SP)     ;;READ THE TTY
 4401  022714  042766  177600  000004      BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
 4402  022722  026627  000004  000023      CMP     4(SP),#23       ;;IS IT A CONTROL-S?
 4403  022730  001013                        BNE     3$              ;;BRANCH IF NO
 4404  022732  105777  156206          2S:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
```

```
 4405  022736  100375                        BPL     2$              ;;LOOP UNTIL ITS THERE
 4406  022740  117746  156202                MOVB    @$TKB,-(SP)     ;;GET CHARACTER
 4407  022744  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
 4408  022750  022627  000021                CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
 4409  022754  001366                        BNE     2$              ;;IF NOT DISCARD IT
 4410  022756  000750                        BR      1$              ;;YES, RESUME
 4411  022760  026627  000004  000140  3$:   CMP     4(SP),#140      ;;IS IT UPPER CASE?
 4412  022766  002407                        BLT     4$              ;;BRANCH IF YES
 4413  022770  026627  000004  000175        CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
 4414  022776  003003                        BGT     4$              ;;BRANCH IF YES
 4415  023000  042766  000040  000004        BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
 4416  023006  000002                4$:     RTI                     ;;GO BACK TO USER
 4417                                        ;;**************************************************************
 4418                                        ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
 4419                                        ;*CALL:
 4420                                        ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
 4421                                        ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
 4422                                        ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
 4423
 4424  023010  010346                $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
 4425  023012  005046                        CLR     -(SP)           ;;CLEAR THE RUBOUT KEY
 4426  023014  012703  023274        1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
 4427  023020  022703  023304        2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
 4428  023024  101467                        BLOS    4$              ;;BR IF YES
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4431  023026  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4432  023032  004767  177632                JSR     PC,     $RDCHR  ;GO TO THE SUBROUTINE
 4433  023036  112613                        MOVB    (SP)+,(R3)      ;;GET CHARACTER
 4434  023040  122713  000177        10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
 4435  023044  001024                        BNE     5$              ;;BR IF NO
 4436  023046  005716                        TST     (SP)            ;;IS THIS THE FIRST RUBOUT?
 4437  023050  001010                        BNE     6$              ;;BR IF NO
 4438  023052  112767  000134  000212        MOVB    #'\,9$          ;;TYPE A BACK SLASH
 4439  023060  004567  000426                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4440  023064  023272                        .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4441  023066  012716  177777                MOV     #-1,(SP)        ;;SET THE RUBOUT KEY
 4442  023072  005303                6$:     DEC     R3              ;;BACKUP BY ONE
 4443  023074  020327  023274                CMP     R3,#$TTYIN      ;;STACK EMPTY?
 4444  023100  103441                        BLO     4$              ;;BR IF YES
 4445  023102  111367  000164                MOVB    (R3),9$         ;;SETUP TO TYPEOUT THE DELETED CHAR.
 4446  023106  004567  000400                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4447  023112  023272                        .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4448  023114  000741                        BR      2$              ;;GO READ ANOTHER CHAR.
 4449  023116  005716                5$:     TST     (SP)            ;;RUBOUT KEY SET?
 4450  023120  001407                        BEQ     7$              ;;BR IF NO
 4451  023122  112767  000134  000142        MOVB    #'\,9$          ;;TYPE A BACK SLASH
 4452  023130  004567  000356                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4453  023134  023272                        .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
 4454  023136  005016                        CLR     (SP)            ;;CLEAR THE RUBOUT KEY
 4455  023140  122713  000025        7$:     CMPB    #25,(R3)        ;;IS CHARACTER A CTRL U?
 4456  023144  001004                        BNE     8$              ;;BR IF NO
 4457  023146  004567  000340                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4458  023152  023304                        .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
 4459  023154  000717                        BR      1$              ;;GO START OVER
 4460  023156  122713  000022        8$:     CMPB    #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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4464  023172  001201                        .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 4465  023174  004567  000312                JSR     R5,     $PRINT  ;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 CHACTER
 4468  023204                        4$:
 4469  023204  004567  000302                JSR     R5,     $PRINT  ;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4474  023224  023272                        .WORD   9$              ;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,     $PRINT  ;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                     ;;RETURN
 4486  023272     000                9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
 4487  023273     000                        .BYTE   0               ;;TERMINATOR
 4488  023274  000010                $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
 4489  023304  052536  005015  000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
 4490  023311     136  006507  000  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
 4491  023316  020515  053523  020122 $MSWR:  .ASCIZ  <15><12>/SWR = /
 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                $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
 4510  023342  016666  000004  000002        MOV     4(SP),2(SP)     ;;INPUT NUMBER
 4511  023350  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4512  023352  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 4513  023354  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 4514  023356                        1$:
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4515
 4516
```

```
 4517  023356  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4518  023362  004767  177422                JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
 4519  023366  012600                        MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
 4520  023370  010067  000102                MOV     R0,5$           ;;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                        BLT     4$
 4529  023420  006301                        ASL     R1              ;;*2
 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                     ;;RETURN
 4545  023466  005726                4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
 4546  023470  105010                        CLRB    (R0)            ;;SET A TERMINATOR
 4547  023472  004567  000014                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4548  023476  000000                5$:     .WORD   0
 4549  023500  004567  000006                JSR     R5,     $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                                  ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
 4556                                  ;*  CALL:       JSR     R5,     $PRINT
 4557                                  ;*              <MESSAGE VIRTUAL ADDRESS>
 4558                                  ;;**************************************************************
 4559  023512  012567  000016        $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
 4560  023516  066767  155056  000010         ADD     RELOCF, 1$      ;MAKE IT PHYSICAL.
 4561                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
 4562                                  ;*  WITHOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 4563  023524  013746  177776                MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 4564  023530  004767  000004                JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
 4565  023534  000000                1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
 4566  023536  000205                        RTS     R5              ;RETURN.
 4567
 4568                                  .SBTTL  TYPE ROUTINE
 4569
 4570                                  ;;**************************************************************
 4571                                  ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
 4572                                  ;*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      3$              ;;LEAVE
 4589  023552  010046                1$:     MOV     R0,-(SP)        ;;SAVE R0
 4590  023554  017600  000002                MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
 4591  023560  122767  000001  155436        CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
 4592  023566  001011                        BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
 4593  023570  132767  000100  155427        BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
 4594  023576  001405                        BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
 4595  023600  010067  000004                MOV     R0,61$          ;;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$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
 4601  023624  001005                        BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
 4602  023626  005726                        TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
 4603  023630  012600                60$:    MOV     (SP)+,R0        ;;RESTORE R0
 4604  023632  062716  000002                ADD     #2,(SP)         ;;ADJUST RETURN PC
 4605  023636  000002                        RTI                     ;;RETURN
 4606  023640  122716  000011        4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
 4607  023644  001431                        BEQ     8$
 4608  023646  122716  000200                CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
 4609  023652  001007                        BNE     5$
 4610  023654  005726                        TST     (SP)+           ;;POP  <CR><LF> EQUIV
 4611  023656  004567  177630                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 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     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
 4619                                                                ;;AND THE NULL CHAR.
 4620  023710  105366  000001        7$:     DECB    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                DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
 4624  023726  000770                        BR      7$              ;;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          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
 4635  023760  100375                            BPL     $TYPEC
 4636  023762  116677  000002  155162            MOVB    2(SP),@$TPB     ;;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      $ATY5
 4653  024042  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
 4654  024050                          $ATYC:
 4655  024050  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4656  024052  010146                            MOV     R1,-(SP)        ;;PUSH R1 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    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
 4662  024100  001425                            BEQ     3$              ;;IF NOT:  BR
 4663  024102  017600  000004                    MOV     @4(SP),R0       ;;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     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
 4668  024126  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
 4669  024130  001376                            BNE     2$
 4670  024132  166700  155062                    SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
 4671  024136  006200                            ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
 4672  024140  010067  155056                    MOV     R0,$MSGLGT      ;;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),R0               ;;GET ERROR #
 4684  024214  062766  000002  000004            ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
```



```
 4685  024222  012701  001344                    MOV     #$ASTAT,R1      ;;POINT TO TABLE START
 4686  024226  005711                  6$:     TST     (R1)            ;;END OF TABLE?
 4687  024230  100404                            BMI     8$              ;;IF SO:  BR
 4688  024232  020021                            CMP     R0,(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          8$:     CMP     $APTR,R1        ;;TABLE FULL?
 4693  024246  001402                            BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
 4694  024250  010021                            MOV     R0,(R1)+        ;;SET UP NEW ENTRY
 4695  024252  005211                  9$:     INC     (R1)            ;;BUMP ERROR COUNT
 4696  024254  105767  000150          10$:    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          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
 4701  024274  001375                            BNE     11$             ;;IF NOT:  WAIT
 4702  024276  017667  000004  154702            MOV     @4(SP),$FATAL   ;;GET ERROR #
 4703  024304  062766  000002  000004            ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
 4704  024312  005267  154666                    INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
 4705  024316  105067  000106          12$:    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)+,R0        ;;POP STACK INTO R0
 4710  024336  000207                            RTS     PC              ;;RETURN
 4711  024340                          $ATY6:
 4712  024340  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4713  024342  016700  155144                    MOV     $APTR,R0
 4714  024346  162700  001344                    SUB     #$ASTAT,R0      ;;GET SIZE OF STAT TABLE
 4715  024352  005767  154626          1$:     TST     $MSGTY          ;;SEE IF DONE LAST COMMUNICATION
 4716  024356  001375                            BNE     1$              ;;IF NOT:  WAIT
 4717  024360  010067  154636                    MOV     R0,$MSGLG       ;;SET MESSAGE LENGTH
 4718  024364  012767  001344  154626            MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
 4719  024372  012767  000002  154604            MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
 4720  024400  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 4721  024402  000207                            RTS     PC              ;;RETURN
 4722  024404                          $ATY7:
 4723  024404  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 4724  024406  012701  001344                    MOV     #$ASTAT,R1      ;;GET START OF TABLE
 4725  024412  005721                  1$:     TST     (R1)+           ;;END OF TABLE?
 4726  024414  100402                            BMI     2$              ;;IF SO:  BR
 4727  024416  005021                            CLR     (R1)+           ;;CLEAR ERROR COUNT
 4728  024420  000774                            BR      1$              ;;KEEP CLEARING
 4729  024422                          2$:
 4730  024422  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 4731  024424  000207                            RTS     PC              ;;RETURN
 4732  024426     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
 4733  024427     000          $LFLG:  .BYTE   0               ;;LOG FLAG
 4734  024430     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
 4735                                          .EVEN
 4736                                  APTSIZE=200
 4737                                  APTENV=001
 4738                                  APTSPOOL=100
 4739                                  APTCSUP=040
 4740                                  ;;**************************************************
```

```
                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

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

024432                          $TYPDS:
024432  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
024434  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
024436  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
024440  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
024442  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
024444  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
024450  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
024454  100004                          BPL     1$              ;;BR IF INPUT IS POS.
024456  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
024460  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
024466  016700  154106          1$:     MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
024472  012703  024654                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
024476  060003                          ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
024500  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
024504  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
024506  016001  024644                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
024512  160105          3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
024514  002402                          BLT     4$              ;;BR IF DONE
024516  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
024520  000774                          BR      3$
024522  060105          4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
024524  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
024526  001002                          BNE     5$              ;;FALL THROUGH IF 0
024530  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
024532  100407                          BMI     7$              ;;BR IF YES
024534  106316                  5$:     ASLB    (SP)            ;;MSD?
024536  103003                          BCC     6$              ;;BR IF NO
024540  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
024546  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
024552  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024556  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024560  005720                          TST     (R0)+           ;;JUST INCREMENTING
024562  020067  155054                  CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
024566  103746                          BLO     2$              ;GO DO THE NEXT DIGIT
024570  101002                          BHI     8$              ;;GO TO EXIT
024572  010502                          MOV     R5,R2           ;;GET THE LSD
024574  000764                          BR      6$              ;;GO CHANGE TO ASCII
024576  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
024600  100003                          BPL     9$              ;;BR IF NO
024602  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
024610  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
024612  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
024614  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
```



```
024616  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
024620  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
024622  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
024624  004567  176662                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
024630  024654                          .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
024632  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
024640  012616                          MOV     (SP)+,(SP)
024642  000002                          RTI                     ;;RETURN TO USER
024644  023420                  $DTBL:  10000.
024646  001750                          1000.
024650  000144                          100.
024652  000012                          10.
024654  000004                  $DBLK:  .BLKW   4
                                .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE

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

024664  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024670  116667  000001  000213          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024676  112667  000211                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024702  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024706  000406                          BR      $TYPON
024710  112767  000001  000173  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024716  112767  000006  000167          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024724  112767  000005  000156  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024732  010346                          MOV     R3,-(SP)        ;;SAVE R3
024734  010446                          MOV     R4,-(SP)        ;;SAVE R4
024736  010546                          MOV     R5,-(SP)        ;;SAVE R5
024740  116704  000147                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024744  005404                          NEG     R4
024746  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024752  110467  000134                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024756  116704  000127                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024762  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024766  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
```

```
 4853  024770  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C"
 4854  024772  000404                          BR      3$              ;;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    $OMODE          ;;TYPE THIS DIGIT?
 4861  025012  100017                          BPL     7$              ;;BR IF NO
 4862  025014  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
 4863  025020  001002                          BNE     4$              ;;TEST FOR 0
 4864  025022  005704                          TST     R4              ;;SUPPRESS THIS 0?
 4865  025024  001403                          BEQ     5$              ;;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                  MOVB    R3,8$           ;;SAVE FOR TYPING
 4870  025044  004567  176442                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4871  025050  025106                          .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
 4872  025052  105367  000032          7$:     DECB    $OCNT           ;;COUNT BY 1
 4873  025056  003346                          BGT     2$              ;;BR IF MORE TO DO
 4874  025060  002402                          BLT     6$              ;;BR IF DONE
 4875  025062  005204                          INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
 4876  025064  000763                          BR      2$              ;;GO DO THE LAST DIGIT
 4877  025066  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
 4878  025070  012604                          MOV     (SP)+,R4        ;;RESTORE R4
 4879  025072  012603                          MOV     (SP)+,R3        ;;RESTORE R3
 4880  025074  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
 4881  025102  012616                          MOV     (SP)+,(SP)
 4882  025104  000002                          RTI                     ;;RETURN
 4883  025106     000                  8$:     .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     2$              ;BRANCH IF NOT REPORTED
 4892  025122  005267  177770                  INC     1$              ;SET DOUBLE TRAP FLAG.
 4893  025126  011667  154034                  MOV     (SP),   $TMP3   ;SAVE THE BAD PC FOR TYPOUT.
 4894  025132  004767  174502                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 4895  025136  000031                          .WORD   31              ;ERROR TYPE CODE.
 4896  025140  000401                          BR      3$              ;SKIP HALT
 4897  025142  000000                  2$:     HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
 4898                                                                  ;BEFORE FIRST WAS PRINTED
 4899  025144  005067  177746          3$:     CLR     1$
 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  025152                          $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  001400                          BEQ     1$              ;BRANCH IF NO MEM MGMT
 4913  025176  032737  000001  177572          BIT     #1,     @#SR0   ;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     #KIPAR0, 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$:     MOV     #3,R1           ;DIGIT SHIFT COUNT
 4940  025304  062703  000060                  ADD     #60,    R3      ;MAKE IT AN ASCII DIGIT
 4941  025310  110367  000036                  MOVB    R3,     8$      ;LOAD DIGIT INTO MESSAGE
 4942  025314  004567  176172                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4943  025320  025352                          .WORD   8$              ;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,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4953  025346  027065                          .WORD   FILL2           ;ADDRESS OF MESSAGE TO BE TYPED
 4954  025350  000207                          RTS     PC              ;RETURN
 4955  025352     000                  8$:     .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 THROUGT
 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  046420  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  053061  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  020505
 5004  025704  000012
 5005  025706  005015  047516  050040  NOPES:  .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  052504  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  047506
 5022  026046  020122  047516  046522
 5023  026054  046101  051040  047125
 5024  026062  044516  043516  005015
 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  020115  000040
 5034  026142  020     047515  020123  MX4:    .ASCIZ  <15><12>'MOS PARITY '
 5035  026150  040     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  020505  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  020505  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  035123  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  026505  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
```



```
 5133  027130  030455  024460  000056
 5134  027136  047522  040524  044524  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
 5135  027144  043516  041040  052111
 5136  027152  042440  051122  051117
 5137  027160  052050  052123  030461
 5138  027166  030455  024462  000056
 5139  027174  047515  020123  042522  DM6:    .ASCIZ  'MOS REFRESH TEST ERROR (TST 30-31).'
 5140  027202  051106  051505  020110
 5141  027210  042524  052123  042440
 5142  027216  051122  051117  024040
 5143  027224  051524  020124  030063
 5144  027232  031455  024461  000056
 5145  027240  020063  047530  020122  DM7:    .ASCIZ  '3 XOR 9 PATTERN ERROR(TST13-16).'
 5146  027246  020071  040520  052124
 5147  027254  051105  020116  051105
 5148  027262  047522  024122  051524
 5149  027270  030524  026463  033061
 5150  027276  027051     000
 5151  027301     115  051101  044103  DM10:   .ASCIZ  "MARCHING 1'S AND 0'S ERROR(TST 27)."
 5152  027306  047111  020107  023461
 5153  027314  020123  047101  020104
 5154  027322  023460  020123  051105
 5155  027330  047522  024122  051524
 5156  027336  020124  033462  027051
 5157  027344     000
 5158  027345     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST17).'
 5159  027352  020131  042515  042515
 5160  027360  054522  040440  042104
 5161  027366  042522  051523  042440
 5162  027374  051122  051117  052050
 5163  027402  052123  033461  027051
 5164  027410     000
 5165  027411     104  052101  050111  DM12:   .ASCIZ  "DATIP WITH WRONG PARITY DIDN'T TRAP(TST17)."
 5166  027416  053440  052111  020110
 5167  027424  051127  047117  020107
 5168  027432  040520  044522  054524
 5169  027440  042040  042111  023516
 5170  027446  020124  051124  050101
 5171  027454  052050  052123  033161
 5172  027462  027051     000
 5173  027465     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
 5174  027472  050040  051101  052111
 5175  027500  020131  051124  050101
 5176  027506  042520  026104  041040
 5177  027514  052125  047040  020117
 5178  027522  042522  044507  052123
 5179  027530  051105  051440  047510
 5180  027536  051527  042440  051122
 5181  027544  051117  043040  040514
 5182  027552  027107     000
 5183  027555     120  051101  052111  DM14    .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
 5184  027562  020131  042522  044507
 5185  027570  052123  051105  047040
 5186  027576  052117  046440  050101
 5187  027604  042520  020104  051501
 5188  027612  041440  047117  051124
```

```
 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  053140
 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
```



```
 5245  030306  040522  027120     000
 5246  030313      50  041501  044524  DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
 5247  030320  047117  042440  040516
 5248  030326  046103  020105  040527
 5249  030334  047123  052047  051440
 5250  030342  052105  027051     000
 5251  030347      15  052012  040522  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
 5252  030354  050120  042105  052040
 5253  030362  020117  020064     000
 5254
 5255                                  ;;*************************************************************
 5256                                  ;DATA COLUMN HEADINGS
 5257                                  ;;*************************************************************
 5258
 5259  030367     120  004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
 5260  030374  004507  027523  004502
 5261  030402  040527  000123
 5262  030406  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
 5263  030414  050057  004503  040515
 5264  030422  051411  041057  053411
 5265  030430  051501     000
 5266  030433     126  050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
 5267  030440  027520  041520  046411
 5268  030446  004501  027523  000102
 5269  030454  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
 5270  030462  050057  004503  042522
 5271  030470  004507  040515     000
 5272  030475     126  050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
 5273  030502  027520  041520  046411
 5274  030510  052501  004524  042522
 5275  030516  004507  027523  004502
 5276  030524  040527  000123
 5277  030530  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
 5278  030536  050057  004503  052511
 5279  030544  004524  040515  051411
 5280  030552  041057  053411  051501
 5281  030560     000
 5282  030561     126  050057  004503  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
 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  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
 5289  030626  027520  041520  052011
 5290  030634  050122  050057  000103
 5291  030642  027526  041520  050011  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
 5292  030650  050057  004503  051124
 5293  030656  027520  041520  051011
 5294  030664  043505  053411  051501
 5295  030672     000
 5296  030673     126  050057  004503  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
 5297  030700  027520  041520  051011
 5298  030706  043505  053411  051501
 5299  030714     000
 5300  030715     122  043505  053411  DH30:   .ASCIZ  'REG    WAS     MA      WAS'
```

```
 5301  030722  051501  046411  004501
 5302  030730  040527  000123
 5303
 5304                                  ;;*****************************************************************
 5305                                  ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
 5306                                  ;;*****************************************************************
 5307  030734      000     377     000  DF1:    .BYTE   0,-1,0,0
 5308  030737      000
 5309  030740      000     377     377  DF2:    .BYTE   0,-1,-1,0,0
 5310  030743      000     000
 5311  030745      000     377     377  DF3:    .BYTE   0,-1,-1,-2,-2
 5312  030750      376     376
 5313  030752      000     377     377  DF14:   .BYTE   0,-1,-1,-1,0,0
 5314  030755      377     000     000
 5315  030760      000     377     000  DF21:   .BYTE   0,-1,0,-1,0,0
 5316  030763      377     000     000
 5317  030766      377     000     377  DF30:   .BYTE   -1,0,-1,-2
 5318  030771      376                          .EVEN
 5319
 5320
 5321          032110                  . = 32110                       ;THE LOADERS ARE SAVE HERE TO END OF 8K
 5322
 5323          000001                          .END
```



```
ABASE = 000000            384     425
ACDW1 = 000000            384     427
ACDW2 = 000000            384     428
ACPUOP= 000000            384     339
ADDW0 = 000000            384     429
ADDW1 = 000000            384     430
ADDW10= 000000            384     439
ADDW11= 000000            384     440
ADDW12= 000000            384     441
ADDW13= 000000            384     442
ADDW14= 000000            384     443
ADDW15= 000000            384     444
ADDW2 = 000000            384     431
ADDW3 = 000000            384     432
ADDW4 = 000000            384     433
ADDW5 = 000000            384     434
ADDW6 = 000000            384     435
ADDW7 = 000000            384     436
ADDW8 = 000000            384     437
ADDW9 = 000000            384     438
ADEVCT= 000000            384     390
ADEVM = 000000            384     426
AE    = 000001            179#   2433    2519    3750
AENV  = 000000            384     395
AENVM = 000000            384     396
AFATAL= 000000            384     387
AMADR1= 000000            384     412
AMADR2= 000000            384     416
AMADR3= 000000            384     419
AMADR4= 000000            384     422
AMAMS1= 000000            384     426
AMAMS2= 000000            384     414
AMAMS3= 000000            384     417
AMAMS4= 000000            384     420
AMSGAD= 000000            384     392
AMSGLG= 000000            384     393
AMSGTY= 000000            384     386
AMTYP1= 000000            384     407
AMTYP2= 000000            384     415
AMTYP3= 000000            384     418
AMTYP4= 000000            384     421
APASS = 000000            384     389
APRIOR= 000000            384
APTCSU= 000040           4598    4736 #
APTENV= 000001           4187    4591    4659    4737#
APTSIZ= 000200            676    4736#
APTSPD= 000100           4593    4651    4739#
ASWREG= 000000            384     397
ATESTN= 000000            384     398
AUNIT = 000000            384     391
AUSER = 000000            384     393
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*
```

```
                    2409    2411    2467    2469    3148*   3149*   3156    3158    3161*   3162*   3165    3167    3172
                    3173    3192    3195*   3198    3199    3213    3214    3215    3217    3226*   3227*   3231*   3236
                    3238    3256*   3260*   3262    3264    3266    3268    3272    3273    3303*   3305    3307    3310
                    3317    3319    3324    3325    3340    3342    3350*   3351*   3353    3355    3360*   3362    3364
                    3366    3400    3401
BIT0    = 000001     123#    235     251     1039    1213    1369    1380    1470    1491    2456    2485    2501    3148
                    3767    3800    3831    4073
BIT00   = 000001     113#    123
BIT01   = 000002     112#    122
BIT02   = 000004     111#    121
BIT03   = 000010     110#    120
BIT04   = 000020     109#    119
BIT05   = 000040     108#    118
BIT06   = 000100     107#    117
BIT07   = 000200     106#    116
BIT08   = 000400     105#    115     4022
BIT09   = 001000     104#    114     4032    4201
BIT1    = 000002     122#
BIT10   = 002000     103#    4173
BIT11   = 004000     102#    4039
BIT12   = 010000     101#
BIT13   = 020000     100#    3174    3201    3274    3311    4181
BIT14   = 040000      99#     217    4008
BIT15   = 100000      98#    2485    3226    3527
BIT2    = 000004     121#
BIT3    = 000010     120#
BIT4    = 000020     119#
BIT5    = 000040     118#
BIT6    = 000100     117#
BIT7    = 000200     116#
BIT8    = 000400     115#    3231    3560
BIT9    = 001000     114#
BLKMSK  001606       547#    3151    3222    3254    4059*   4064*
BPTVEC= 000014       131
BYTMES  025437      1033    4973#
CASFLG  001616       551#     937*    940*   1483
CASREG  001620       552#    1485*
CKPMER  017636      1755    2417    3331    3759#
CLRPAR  020270      1133    1208    3804    3813#
CONST   026456      1458    5073#
CR      = 000015      38#    4637    4647
CRLF    = 000200      39#     966    4608    4647
CTRLS   005072      1273    1283#
DDISP   = 177570      45#     361     864
DF1     030734       719    5307#
DF14    030752       774     779     784     806     811     816    5313#
DF2     030740       724     734     739     744     749     754     795     821     826    5309#
DF21    030760       800    5315#
DF3     030745       729     759     764     769     790    5311#
DF30    030765       831     836    5317#
DH1     030367       717    5259#
DH12    030433       762     767    5266#
DH14    030454       772     782    5269#
DH15    030475       777    5272#
DH2     030406       722     727     732     737     742     747     752     757     788     793    5262#
DH21    030530       798    5277#
```



```
DH23    030561       804    5282#
DH24    030621       809    5288#
DH25    030642       814    5291#
DH26    030673       819     824    5296#
DH30    030715       829    5300#
DIDBH   012772      2711#
DIDBL   012700      2662#
DIDO    012610      2613#
DIPDO   013070      2761#
DISPLA  001142       361#    864*    872*    881*   3594*   3647*   4054*   4172*
DISPRE  000174       195#    870
DM1     027010       716     776    5119#
DM10    027301       751    5157#
DM11    027345       756    5158#
DM12    027411       761    5165#
DM13    027465       766    5173#
DM14    027555       771    5183#
DM16    027654       781    5194#
DM17    027733       786    5202#
DM2     027044       721     726    5124#
DM20    030031       792    5213#
DM21    030063       797    5218#
DM23    030132       803    5225#
DM24    030177       808    5232#
DM25    030253       813    5240#
DM26    030273       818    5243#
DM27    030313       823    5246#
DM31    030347       833    5251#
DM4     027100       731    5129#
DM5     027136       736    5134#
DM6     027174       741    5139#
DM7     027240       746    5145#
DONE    013776      3038#
DPDBH   013054      2659#
DPDBL   013162      2810#
DSWR    = 177570      44#     360     863
DT1     001546       573#     718
DT12    001674       577#     763     768
DT14    001706       579#     773     783
DT15    001720       581#     778
DT2     001660       575#     723     728     733     738     743     748     753     758     789     794
DT21    001736       584#     799
DT23    001754       587#     805
DT24    001772       590#     810
DT25    002002       592#     815
DT26    002016       594#     820     825
DT30    002030       596#     830
DT31    002042       598#     835
EMTVEC= 000030       133#
ENDINS  021624      4059    4131    4144#
ERRTRP  025114       206    1004    1112    3665    4889#
ERRVEC= 000004       126#     205     861     862-    873*    927*    938*    964*   1004*   1027*   1037*   1097*   1112*
                    3568*   3641*   3681*   3685-   4013    4014*   4016*   4019*
FADMAP  001570       540#    1421*   1422*   3215    3217    3340    3342    3364    3366    4073*   4100    4101    4109
                    4111
FADMES  026311      1374    5054#
```

```
FADMSK  001566   538#  3368  4134*  4136*
FILL2   027005  4953  5113#
FLAGBK  001557   532#  3207*  3297*  3300*  3318*
FROM    025507  1046  3951  4981#
FSTADR  001562   535#  1357  1455*  3744  4065
GMPR    004072  1013  1093#
GMPRA   004120  1098#  1110
GMPRB   004134  1097  1104#
GMPRC   004142  1102  1106#
GMPRD   004202  1095  1118#
GNS     = ****** U   194   905
HT      = 000011    36#  4606  4647
IMPCHE= 177746   186#   939*
IMPCK   003272   926   927   936#
INITDN  015064  1528  1564  1591  1666  1676  3122#
INITEX  015176  3218  3220  3239  3241#
INITMM  014426  1511  1554  1602  1631  1640  1708  1722  1758  1779  1802  1827  1837  1880
                1929  1940  1983  2032  2043  2091  2217  2228  2276  2408  2555  2616  2665
                2714  2764  2813  2862  2906  2958  2977  2990  3010  3023  3148#  3417
INSERT  021124  4056#
INSUFF  025524  1015  4984#
IOTVEC= 000020   171#
KIPAR0= 172340   166#   242   261*  3133*  3554*  3620*  4922
KIPAR1= 172342   167#   267*  3134*  3555*  3621*
KIPAR2= 172344   168#   975   995*  1072*  1181*  1185  2484  3135*  3154*  3160*  3171  3228*  3258*
                3276  3349*  3381  3536*  3539*  3554  3615*  3809  3810*  3849*  3854*
KIPAR3= 172346   169#  1069  1072  1073*  3136*  3171*  3176*  3276*  3277*  3525*  3528*  3536  3555
                3616*
KIPAR4= 172350   170#  3137*
KIPAR5= 172352   171#  3138*
KIPAR6= 172354   172#  3139*
KIPAR7= 172356   173#  3140*
KIPDR0= 172300   155#   237  3125*
KIPDR1= 172302   156#  3126*
KIPDR2= 172304   157#  3127*
KIPDR3= 172306   158#  3128*
KIPDR4= 172310   159#  3129*
KIPDR5= 172312   160#  3130*
KIPDR6= 172314   161#  3131*
KIPDR7= 172316   162#  3132*
LADMAP  001602   546#  1449*  1450*  1473*  1474*  3165  3167  3184  3186  3199  3208  3236  3238
                3250  3252  3266  3268  3285  3287  3307  3317  3319  4107  4108  4109  4111
LADMES  026376  1401  5064#
LADMSK  001600   544#  3169  3188  3210  3270  3289  3309  3321  4139*  4141*
LDDISP  001522   508#   830*
LF      = 000012    37#  4641  4647
LMAD    001520   507#  3059  3670*  3677  3691*
LSTADR  001574   541#  1353*  1413*  1417  1428  1467*  1468*  1469*  1470*  1471  3240  4074  4081
MAMF    017536  1488  2546  3736#
MAMF1   017622  3750#  3752
MAMF2   017634  3737  3739  3745  3754#
MANUAL  005366  1119  1274  1343  1353#  1448
MANUL1  005424  1362  1369#
MANUL2  006026  1364  1465#
MAPMEM  003324   938   952#
MAPRB   004250  1136  1143#
```



```
MASK4K= 017777   183#   966   991  1065  1419  1471  3212  3323  3842  4098  4105  4134  4139
MEMMAP  001524   200   509#   953   954  1192  1194  1197  1394  1396  1435  1437  1440  1442
                3532  3534  3543  3545  3565  3570  3812  3814
MEMMES  025421   962  4977#
MMAMES  025354   932  4963#
MMAVA   000606   233   290#   924*   930*   972   993  1024*  1030*  1067  1137  1179  1451  1465
                1489  2481  3152  3224  3255  3347  3378  3523  3613  3662  3671  3807  3847
                4911
MMDOWN  015674  1540  1576  1598  1672  1688  3234  3340#
MMINIT  014300   929  1029  1140  1493  3124#
MMORE   001554   528#  3232*  3233*  3241*  3326  3369
MMUP    015204  1524  1560  1616  1636  1651  1712  1733  1769  1792  1815  1831  1874  1917
                1933  1977  2020  2036  2084  2204  2221  2269  2389  2545  2580  2630  2679
                2725  2778  2827  2877  2959  2983  3003  3016  3036  3250#  3421
MMVEC = 000250   144#
MPAMEM  004210  1114  1133#
MPEND   002074   617#
MPPATS  002050   565   606#
MPRX    002276   563   697#  1093  1109  1113  1176  1248  1261  1272  1288  1334  1342  2401
                3329  3736  3759
MPR0    002076   562   632#  1096  1150  1212  1283  1328  1337
MPR1    002106   636#
MPR10   002216   672#
MPR11   002226   676#
MPR12   002236   680#
MPR13   002246   684#
MPR14   002256   688#  1187  1253  1255  1256  1262  1265*
MPR15   002266   692#  1189  1257  1259  1260  1266  1269*
MPR2    002116   640#
MPR3    002126   644#
MPR4    002136   648#
MPR5    002146   652#
MPR6    002156   656#
MPR7    002166   660#
MPR8    002176   664#
MPR9    002206   668#
MTMAP   025462  1210  4977#
MTOE    025572  3719  5087#
MTR     025605  1116  4993#
MX1     026071  1234  5026#
MX2     026110  1242  5029#
MX3     026123  1218  5031#
MX4     026142  1224  5034#
MX5     026160  1230  5037#
MX6     026176  1271  5040#
NOMEM   026266  3923  5050#
NOMTST  026737  4077  5105#
NOPES   025706  3861  5055#
OFFLG   001560   533#  2428*  2538*
PARVEC= 000114   180#  2442*  2446*  2453*  3590*  3643*  3746*  3747*
PBTRP   012052   564  2453#
PESRV   017430   566  3699#
PEWNC   026700  3836  5099#
PHYADR  016054  1512  1529  1555  1565  1592  1603  3376#
PIRQ    = 177772    43#
PIRQVE= 000240   137#
```

```
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
                3601*  3610  3650*  3651*  4084  4085
PROREL  025760  5012#
PR0     = 000000   60#
PR1     = 000040   61#
PR2     = 000100   62#
PR3     = 000140   63#
PR4     = 000200   64#
PR5     = 000240   65#
PR6     = 000300   66#
PR7     = 000340   67#
PS      = 177776   40#   41
PSCAN   017744  3725  3773  3778  3789#
PSW     = 177776   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
RELOCF  000600   269   288#   913*  1487  3233  3583*  3587*  3596  3604  3625  3649*  3740  4066
                4237  4258  4260  4560  4764
RELTOP  016460  3056  3517#
REL0    017062   276  3059  3610#
RESCHK  005252  1289  1328#
RESLDR  017270   262  3062  3658#
RESRVD  001516   505#   551  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  1814  3426#
RU      = 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   026635  3798  5093#
SELECT  002646   199   846#
SELFLG  001556   531#   844*   846*  1361
SETAE   017502  2416  3741  3743  3746#
SETCON  016170  1778  1801  2497  3416#
SKPMES  026763  4121  5107#
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#
SPRNTO  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  1998  2008  2049  2056  2063  2070  2097  2106
                2113  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#
```



```
SPRNT3  020362  2623  2672  2722  2771  2820  2870  3907#
SR0     = 177572   148#   235   251*   928*  1028*  1491  3141*  3664*  3673*  4913
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  1094  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*  2950
TIMOUT  003622   203  1023#  1085
TKVEC = 000060   135#
TMAP    004562  1184  1238#
TMPFAD  001564   537#  3219  3344  4065*  4070  4072*  4094  4096*  4097*  4098  4113  4135
TMPLAD  001576   543#  3170*  3188*  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*
TO      025617  1056  1078  3944  4983#
TPVEC = 000064   136#
TRAPVE= 000034   134#
TRTVEC= 000014   129#
TSTMAP  001530   512#  3156  3158  3180  3192  3192  3205  3262  3264  3281  3283  3305  3315
                3353  3355  3362  4024  4083*  4084*  4085*  4088  4090*  4091*  4100*  4101*  4107*
```

```
                    4108*  4115   4117
TST1    006166      1480   1492   1504#
TST10   007040      1738   1742   1751#
TST11   007122      1757   1774#
TST12   007206      1797#
TST13   007270      1819#
TST14   007614      1823   1921#
TST15   010144      1925   2024#
TST16   010766      2028   2229#
TST17   011610      2213   2398#
TST2    006312      1550#
TST20   012462      2405   2550#
TST21   012576      2607#
TST22   012666      2611   2656#
TST23   012760      2660   2705#
TST24   013056      2709   2755#
TST25   013150      2759   2804#
TST26   013242      2808   2853#
TST27   013336      2857   2903#
TST3    006406      1587#
TST30   013552      2974#
TST31   013664      3007#
TST32   014004       569   1508   3041#
TST4    006512      1627#
TST5    006602      1662#
TST6    006676      1704#  1744
TST6A   006704      1707#  1745
TST7    006726      1718#  1744
TYPMAP  020412      1005   1246   3918#
UNEXPT  026501      3701   5077#
UP    = 000000       175#  3125   3126   3127   3128   3132
WWP     001612       549#  1153   1157   1161*  2432   2439
WWPBYT  011652      2409#
WWPB0   011616      2401#
WWPB1   011712      2418#  2544
WWPB2   011756      2430#  2540
WWPB3   012326      2500   2509#
WWPB4   012416      2450   2466   2535#
WWPB5   012442      2415   2421   2542#
W3X9    016256      1828   1930   2033   2218   3449#
$APTHD  001330       456    462#
$APTR   001512       499#  4692   4713
$ASTAT  001344       472#   499   4681   4685   4711   4714   4718   4724
$ASTEN  001510       498#
$ATYC   024050      4652   4654#
$ATY1   024024      4650#
$ATY3   024032      4596   4651#
$ATY4   024042      4190   4653#
$ATY6   024340      4711#
$ATY7   024404      4722#
$AUTOB  001134       357#   902*  4325   4495
$BASE   001260       425#
$BDADR  001122       352#   587    590    592   3497*  3699*
$BDDAT  001126       354#   573    575    581    584    587    596   3495*  3911*
$BELL   001174       376#  4176   4213
$CDW1   001264       427#
```



```
$CDW2   001266       428#
$CHARC  024020      4613*  4623*  4630   4639*  4644#
$CKSWR  022364      4005   4149   4168   4200   4313#
$CMTAG  001100       340#   849    850    858
$CM3  = 000000       370#
$CM4  = 000004       370#   371#   372#   373#   374#
$CNTLG  023311      4325   4490#
$CNTLU  023304      4349   4458   4489#
$CPUOP  001232       399#
$CRLF   001201       378#  1007   1251   3065   4185   4213   4223   4248   4254   4299   4362   4464   4489
                    4553   4613   4647
$DBLK   024654      4765   4801   4809#
$DDW0   001270       429#
$DDW1   001272       430#
$DDW10  001314       439#
$DDW11  001316       440#
$DDW12  001320       441#
$DDW13  001322       442#
$DDW14  001324       443#
$DDW15  001326       444#
$DDW2   001274       431#
$DDW3   001276       432#
$DDW4   001300       433#
$DDW5   001302       434#
$DDW6   001304       435#
$DDW7   001306       436#
$DDW8   001310       437#
$DDW9   001312       438#
$DEVCT  001214       390#
$DEVM   001262       426#
$DDAGN  014254      3082   3098   3102   3107   3109#
$DTBL   024644      4769   4805#
$ENDAD  014222       215    886   3100#  4208
$ENDCT  014156       858   3084#
$ENDMG  014260      3087   3111#
$ENULL  014275      3094   3114#
$ENV    001224       395#   893    914   3053   3106   4187   4591   4659   4698
$ENVM   001225       396#   876   4593   4598   4661
$EOP    014126      3046   3048   3075#
$EOPCT  014150       858*  3081#  3085
$ERFLG  001103       343#  3992   4028   4030   4036*  4146   4170*  4213
$ERMAX  001115       349#  4030   4053*  4146
$ERROR  021640      1299   1310   1317   1350   1518   1535   1571   1610   1646   1683   1728   1764   1787
                    1810   1841   1851   1858   1865   1868   1897   1906   1947   1954   1961   1968   1991
                    2000   2009   2050   2057   2064   2071   2098   2107   2116   2123   2132   2141   2148
                    2157   2166   2173   2182   2191   2235   2242   2249   2256   2283   2292   2301   2308
                    2317   2326   2333   2342   2351   2358   2367   2376   2425   2448   2464   2472   2493
                    2506   2523   2532   2571   2624   2673   2723   2772   2821   2871   2917   2926   2948
                    2997   3030   3498   3710   3723   3770   3776   3828   4164#  4894
$ERRPC  001116       350#   573    575    577    579    581    584    587    590    592    594   4178*  4179*
                    4180   4213   4229   4236
$ERRTB  002340       567    714#
$ERRTY  022064      4183   4221#
$ERTTL  001112       347#  3824*  4177*  4213
$ESCAP  001172       375#  4052*  4204   4206   4213
$ETABL  001224       394#
```

```
$ETEND  001330   445#   468
$FATAL  001206   387#  4702*
$FFLG   024430  4650*  4653*  4696   4705*  4734#
$FILLC  001156   368#  4616   4647
$FILLS  001155   367#  4647
$GDADR  001120   351#   573    575    577    579    581    584    587    596   3496*  3891*  3900*  3903*
                3904*  3908*  3909*
$GDDAT  001124   353#   573    575    577    581    584    587   3494*  3892*  3910*
$GET42  014212  3095#
$GTSWR  022436   900   4326#
$HD     000000    12
$HIBTS  001330   463#
$HIOCT  023510  1385   1398   1409   1412*  1414   1425   4540*  4552#
$ICNT   001104   344#  4043*  4044   4046*  4145
$ILLUP  000756   295    311    330#
$INTAG  001135   358#  4363   4495
$ITEMB  001114   348#  4180*  4189   4213   4226
$LF     001202   379#  4213   4479   4489   4553   4647
$LFLG   024427  4681   4706*  4733#
$LPADR  001106   345#  1486*  1487*  1744*  4034*  4050*  4055   4133*  4145
$LPERR  001110   346#  4034   4051*  4145   4203
$MADR1  001236    4 2#
$MADR2  001242   416#
$MADR3  001246   419#
$MADR4  001252   422#
$MAIL   001204   385#   464    468    875    893   4049   4187   4591
$MAMS1  001234   406#
$MAMS2  001240   414#
$MAMS3  001244   417#
$MAMS4  001250   420#
$MBADR  001332   464#
$MFLG   024426  4651*  4657   4707*  4732#
$MNEW   023327  4335   4493#
$MSGAD  001220   393#  4667*  4670   4719*
$MSGLG  001222   393#  4672*  4717*
$MSGTY  001204   386#  4665   4673*  4700   4704*  4715   4719*
$MSWR   023316  4328   4491#
$MTYP1  001235   407#
$MTYP2  001241   415#
$MTYP3  001245   418#
$MTYP4  001251   421#
$MXCNT  021626  4047   4145#
$NULL   001154   366#  4618   4647
$NWTST= 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   2547#
                2582#  2584   2631#  2633   2680#  2682   2730#  2732   2779#  2781   2828#  2830   2879#
                2881   2931#  2963   3004#
$OCNT   025110  4842*  4872*  4885#
$OMODE  025112  4837*  4841*  4846   4849*  4860*  4887#
$OVER   021112  4009   4027   4035   4045   4054#
$PASS   001212   389#   875*  3078*  3079*  3088   3111   4041   4060   4086   4146
$PASTM  001336   466#
$PRINT  023512   325    888    931    961   1006   1014   1032   1045   1055   1077   1115   1209   1217
                1223   1229   1233   1241   1250   1270   1373   1400   1445   1457   3064   3086   3093
                3505   3700   3718   3797   3835   3860   3922   3943   3950   4076   4120   4175   4184
                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
                4870   4870   4942   4952
$PWRAD  000752   328#
$PWRDN  000610   295#   327    856
$PWRMG  000746   326#
$PWRUP  000662   305    311#
$QUES   001200   377#  4213   4383   4470   4489   4550   4553   4647
$RDCHR  022670  4396#  4432
$RDLIN  023010  4424#  4518
$RDOCT  023340  1379   1406   1463   4509#
$RDSZ   000010  4417#
$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   2975   3008   3039
                4001#
$SETUP= 000130   188#   655    856    858    859    886    890   3077   4002   4165   4197   4208   4308
                4495
$STUP   177777   188#
$SVLAD  021056  4017   4048#
$SVPC   000010   213#   218
$SWR    167400     1#     12     17     18     19     20     21     22     23     24    329    374    375
                 376    859   1510   1553   1590   1630   1665   1707   1721   1754   1777   1800   1825
                1927   2030   2215   2401   2553   2613   2662   2711   2761   2810   2859   2906   2977
                3010   3072   3077   3096   3110   3111   3993   3994   3995   3996   3997   4008   4020
                4022   4023   4028   4029   4030   4037   4038   4039   4051   4054   4145   4156   4157
                4158   4159   4160   4173   4181   4194   4201   4213
$SWREG  001226   397#   678
$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#  4336   4317   4341   4400   4406
$TKS    001144   362#  4336   4315   4339   4365*  4398   4404
$TMP0   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*  1383   1411
$TMP3   001166   373#   588    950*   973*   982*   984*   986*  1412  4893*
$TN     000032     1#     12   1495   1510#  1541   1553#  1578   1580#  1618   1630#  1653   1665#  1694
                1707#  1714   1716   1721#  1734   1747   1754#  1757   1771   1777#  1794   1800#  1816
                1819   1825#  1916   1921   1927#  2021   2024   2030#  2206   2209   2215#  2390   2401#
                2547   2583#  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  001132   280*   342#  1743*  3042*  3992   4026   4048*  4049   4054   4062   4124   4146   4172
                4213
$TTYIN  023274  4426   4427   4443   4466   4484   4488#
$TYPAD  023152  1049   1080   1082   3509   4279   4904#
$TYPDS  024432  3092   4274   4753#
$TYPE   023540  4564   4585#  4677
$TYPEC  023754  4337   4616   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#
```

```
$UNIT   001216            391#
$UNITM  001340            467#
$USWR   001230            398#
$VECT1  001254            423#
$VECT2  001256            424#
$VERPC  001514            504#   575   577   579   581   584   587   590   592   594  4236*  4237*
$XTSTR  020654           4011#
$OFILL  025111           4836*  4840*  4850  4886#
$40CAT= ****** U         4008   4183
      = 032110            190#   194#   205#   213   214#   216#   218#   219#   307   331   339#   380   452
                          453#   459#   457#   534#   697#   853   908  3111  3115  4145  4146  4213  4306
                         4488#  4489   4495   4553  4647  4735#  4809#  5321#
.CONST  001610            548#  1464*  1707  1721
.EIGHT  001642            568#  4786
.ERRTB  001640            567#  4242
.MPPAT  001634            565#  1754
.MPRX   001630            563#  2431   2438  2461  2499  3705  3749  3764  3819  3877
.MPRO   001626            562#  2455
.PBTRP  001632            564#  2442
.PESRV  001636            566#  2446   2453  3746
.RESRV  001624            561#
.STACK  001622            560#  1482
.TST32  001644            569#  4079
.$ASTA= ****** U         4651   4654
.$X     001330            452#   457
.3X9    007302           1825#
```



```
ABORT      1#   260  3164  3197  3235  3407  3500  3519  3577  3612  3661
CKSWR      1#  4002  4176  4185  4197
CKWD       1#  1307  1515  1532  1607  1643  1680  1725  1761  1784  1807  1841  1848  1855  1862
        1885  1894  1903  1944  1951  1958  1965  1988  1997  2036  2047  2054  2061  2068  2095
        2104  2113  2120  2129  2138  2145  2154  2163  2170  2179  2188  2232  2239  2246  2253
        2280  2289  2298  2305  2314  2323  2330  2339  2348  2355  2364  2373  2490  2520  2568
        2621  2670  2720  2769  2818  2868  2914  2923  2945  2994  3027
CKWD2      1#  1514  1605  1723  1760  1839  1847  1854  1861  1883  1893  1902  1942  1950  1957
        1984  1986  1996  2005  2045  2053  2060  2067  2093  2103  2112  2119  2128  2137  2144
        2153  2162  2169  2178  2187  2230  2238  2245  2252  2278  2288  2297  2304  2313  2322
        2329  2338  2347  2354  2363  2372  2567  2992  3025
COMMEN     1#   138#   838  1476
ENDCOM     1#    12   138#   842  1480
ERROR     32#
ESCAPE     1#   138#
GETPRI     1#   138#
GETSWR     1#   138#   859#
GTSWR      1#   897
LDPDR      1#  3124  3126  3127  3128  3133
MORETA    333#   447
MULT       1#   138#
NEWTST     1#   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        1#   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      1#   931   961  1096  1014  1031  1045  1055  1077  1115  1209  1217  1223  1229  1232
        1241  1373  1399  1445  1456  3063  3505  3700  3718  3797  3835  3860  3922  4076  4119
        4942  4952
PUSH       1#   136#   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      1#  4429
RDDEC      1#
RDLIN      1#  4514
RDOCT      1#  1376  1403  1460
REPORT     1#   138#
RESREG     1#
SAVREG     1#
SCOPE     33#
SCOPEX  3987#  4056
SCOPIN  3987#  4006
SETPRI     1#   138#
SETUP      1#   138#   847
SIMTRP     1#   897  1237  1376  1403  1460  3089  3960  3968  4002  4125  4146  4165  4197  4231
        4264  4277  4284  4330  4429  4515  4561
SKIP       1#   136#  1757
SLASH      1#   136#   621   631
SPACE    138#
STARS      1#   138#   211   220   225   293   309   335   380   383   449   454   458   471   501
         503   554   558   570   572   602   604   942   951  1019  1022  1088  1091  1122  1131
        1142  1146  1203  1206  1277  1281  1323  1326  1366  1368  1495  1503  1541  1549  1578
        1586  1618  1626  1653  1661  1690  1693  1694  1703  1714  1717  1747  1750  1771  1773
        1794  1796  1816  1818  1833  1835  1876  1878  1918  1920  1936  1938  1979  1981  2021
        2023  2038  2040  2086  2088  2206  2208  2223  2225  2271  2273  2390  2397  2547  2549
        2582  2606  2631  2655  2680  2704  2730  2754  2779  2803  2828  2852  2879  2902  2961
        2973  3004  3006  3066  3117  3123  3145  3147  3244  3249  3334  3339  3372  3375  3394
        3396  3413  3415  3423  3425  3446  3448  3478  3480  3514  3516  3607  3609  3654  3658
```

```
        3684    3698    3732    3735    3756    3758    3784    3788    3872    3874    3887    3890    3914    3917    3989
        4152    4213    4305    4328    4383    4417    4497    4554    4558    4570    4649    4740    4812    4959    4962
        5116    5118    5255    5257    5304    5306
SWRSU     1#    138#    659#
SWROR3    1#    2074    2079    2194    2199    2259    2264    2379    2384    3473
$1REGS  1495#   1497    1543    1530    1620    1655    1697
TYPADR    1#    1048    1059    1061    3508    4278
TYPBIN    1#     138#
TYPBN     1#
TYPBYT    1#    4123    4281
TYPDEC    1#     138#   3088    4270
TYPDS     1#    3089    4271
TYPE      1#     325     868     931     961    1006    1014    1032    1045    1055    1077    1115    1209    1217    1223
        1229    1233    1241    1373    1400    1445    1457    3064    3086    3093    3505    3700    3718    3797    3835
        3860    3922    3943    3950    4076    4120    4175    4184    4222    4245    4247    4251    4253    4292    4298
        4324    4326    4334    4348    4360    4381    4439    4446    4452    4457    4463    4465    4468    4473    4478
        4547    4549    4611    4800    4870    4942    4952
TYPNAM    1#     138#    882
TYPNUM    1#     138#
TYPOC     1#    1237    4231    4264    4330
TYPOCS    1#     138#   3958    3966
TYPOCT    1#     138#   1236    4229    4263    4329
TYPON     1#
TYPOS     1#    3960    3968    4125    4284
TYPTXT    1#     138#
$CKWD     1#    1308    1348    1516    1533    1569    1608    1644    1681    1726    1762    1785    1808    1842    1849
        1856    1863    1886    1895    1903    1945    1952    1959    1966    1989    1998    2007    2048    2055    2062
        2069    2096    2105    2114    2121    2130    2139    2146    2155    2164    2171    2180    2189    2233    2240
        2247    2254    2281    2290    2299    2306    2315    2324    2331    2340    2349    2356    2365    2374    2423
        2491    2521    2530    2569    2622    2671    2721    2770    2819    2869    2915    2924    2946    2995    3028
$INDN     1#    1526    1564    1591    1660    1676
$INMM     1#    1511    1554    1602    1631    1640    1708    1722    1758    1779    1802    1827    1837    1880    1929
        1940    1983    2032    2043    2091    2217    2228    2276    2408    2555    2616    2665    2714    2764    2813
        2862    2906    2977    2990    3010    3023    3416
$MMDN     1#    1538    1574    1596    1670    1686
$MMUP     1#    1522    1558    1614    1634    1649    1710    1731    1767    1790    1813    1829    1872    1915    1931
        1975    2016    2034    2082    2202    2219    2267    2387    2577    2628    2677    2727    2776    2825    2875
        2981    3001    3014    3034    3419
$$CKWD    1#    1298    1309    1316    1349    1517    1534    1570    1609    1645    1682    1727    1763    1786    1809
        1843    1850    1857    1864    1887    1896    1905    1946    1953    1960    1967    1990    1999    2008    2049
        2056    2063    2070    2097    2106    2115    2122    2131    2140    2147    2156    2165    2172    2181    2190
        2234    2241    2248    2255    2282    2291    2300    2307    2316    2325    2332    2341    2350    2357    2366
        2375    2424    2447    2463    2471    2492    2505    2522    2531    2570    2623    2672    2722    2771    2820
        2870    2916    2925    2947    2996    3029    3721    3769    3774    3826
$$CMRE  333#
$$CMTM  333#    370     371     372     373
$$ESCA    1#    138#
$$NEWT    1#    138#   1495    1541    1578    1616    1653    1694    1714    1747    1771    1794    1816    1918    2021
        2206    2390    2547    2582    2631    2680    2730    2779    2828    2879    2961    3004
$$SETM  875#
$$SKIP    1#    138#   1757
.CHBHD  2961#   2963
.C11    2390#   2392
.DIDBH  2680#   2682
.DIDBL  2631#   2633
.DIDO   2582#   2584
```



```
.DIPDO  2730#   2732
.DPDBH  2828#   2830
.DPDBL  2779#   2781
.EQUAT    1#     28
.ERROR    1#    1299    1310    1317    1350    1518    1535    1571    1610    1646    1683    1728    1764    1787    1810
        1844    1851    1858    1865    1888    1897    1906    1947    1954    1961    1968    1991    2000    2009    2050
        2057    2064    2071    2098    2107    2116    2123    2132    2141    2148    2157    2166    2173    2182    2191
        2235    2242    2249    2256    2283    2292    2301    2308    2317    2326    2333    2342    2351    2358    2367
        2376    2425    2448    2464    2472    2493    2506    2523    2532    2571    2624    2673    2723    2772    2821
        2871    2917    2926    2948    2997    3030    3498    3710    3723    3770    3776    3828    4894
.HEADE    1#     2
.KT11     1#     140
.MARHD  2879#   2881
.SCOPE    1#    1504    1550    1587    1627    1662    1704    1718    1751    1774    1797    1819    1921    2024    2209
        2393    2550    2607    2656    2705    2755    2804    2853    2903    2974    3007    3038
.SETUP    1#     188
.SWRHI    1#      13
.SWRLO    1#     25#
.TM7    1714#   1716
.$ACT1    1#     209
.$APTB    1#     381#
.$APTH    1#     447
.$APTY    1#    4647
.$ASTA    1#     469
.$CATC    1#     188
.$CMTA    1#     333
.$DB2D    1#
.$DB2O    1#
.$DIV     1#
.$EOP     1#    2066
.$ERRO    1#    4150
.$ERRT    1#    4213
.$MULT    1#
.$POWE    1#     291
.$RAND    1#
.$RDDE    1#
.$RDOC    1#    4495
.$READ    1#    4303
.$R2AZ    1#
.$SAVE    1#
.$SB2D    1#
.$SB2O    1#
.$SCOP    1#    3987
.$SIZE    1#
.$SUPR    1#
.$TRAP    1#
.$TYPB    1#
.$TYPD    1#    4740
.$TYPE    1#    4568
.$TYPO    1#    4810
.$40CA    1#
.1170     1#

. ABS.  032110      000
```

ERRORS DETECTED: 0

CZQMCF.BIN,CZQMCF.LST/CRF/SOL/NL:TOC=CZQMCF.SML,CZQMCF.P11
RUN-TIME: 21 28 1 SECONDS
RUN-TIME RATIO: 130/51=2.5
CORE USED: 39K (77 PAGES)