MS11-K

0–124K MEMORY EXERCISER
CZQMCE0

AH-9047E-MC

FICHE 1 OF 1

MAR 1978

digital

MADE IN USA

```
                    =============

        PRODUCT CODE:          AC-9D45E-MC

        PRODUCT NAME:          CZQMCEO 0-124K MEM EXER 16K

        PRODUCT DATE:          FEB 1978

        MAINTAINER:            DIAGNOSTIC ENGINEERING
```

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

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

The following are trademarks of Digital Equipment Corporation:

```
DIGITAL    PDP      UNIBUS     MASSBUS

DEC        DECUS    DECTAPE
```

REVISION HISTORY
================

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

TABLE OF CONTENTS

1.0 GENERAL PROGRAM INFORMATION.

1.1 Program Purpose (Abstract)

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

1.2 System Requirements

A. Hardware Requirements

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

B. Software Requirements

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

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

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

1.3 Related Documents and Standards

A. Programming practices - Document No. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC Package - MAINDEC-11-DZQAC-C2-D
C. MF11-U/UP Core Memory System Maintenance Manual
Document No. DEC-11-HMFMA-B-D
D. Applicable Circuit Schematics:
G235 - 16K X-Y DRIVE
G114 - 16K SENSE/INHIBIT
M8293 - 16K UNIBUS TIMING
M7259 - PARITY CONTROL

1.4 Diagnostic Hierarchy Prerequisites

Before running this program, a CPU diagnostic should be run to verify the functionality of the processor and PDP-11 instruction set.

If memory management is to be used, then the KT11 diagnostic should also be run before this program.

PDP-11/05	- MAINDEC-11-DZQKC
PDP-11/20	- MAINDEC-11-DZQKC
PDP-11/34	- MAINDEC-11-DFKTH
PDP-11/40	- MAINDEC-11-DBQEA OR MAINDEC-11-DCQKC
PDP-11/45	- MAINDEC-11-DCQKC
PDP-11/60	- MAINDEC-11-DQKDA
KT11-C	- MAINDEC-11-DCKTA THRU DCKTF
KT11-D	- MAINDEC-11-DBKTA THRU DBKTF

1.5 Assumptions

This program assumes the correct operation of the CPU and, if used, the memory management option.

2.0 OPERATING INSTRUCTIONS

2.1 Loading and Starting Procedures

2.1.1 Load the program using any standard absolute loader.

2.1.2 Starting address 200:

Normal program execution.

2.1.3 Starting address 204:

Allows the operator to input, via teletype conversation, first and last addresses to be exercised, and a data pattern to be used in tests 6 and 7.

2.1.4 Starting Address 210:

Restart program using previously selected parameters.

2.1.5 Starting Address 214:

Restore loaders and halt. This routine is capable of relocating the program back to banks 0 and 1 if the program was halted while running the top two banks of memory. There are special procedures required for this situation.

A. If memory addresses 0-1000 have not been exercised, either through parameter selection (SA=204) or by running with SW05=1, then:

```
    Load Address 214,
    Press START.
```

B. If running without memory management, then:

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

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

```
    Load Address 777707 (PC)
    Deposit 214
    Press CONTinue
```

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

```
    Load Address     772340  (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

```
-------------------------------------------------
I   I   I   I   I   I   I   I   I   I   I   I   I   I   I   I   I
!PE!  !   !   !        ADDRESS            !   !  !WP!   !AE!
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	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 MCS PARITY REGISTER

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

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

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

6.5.2 Toggle-in-Program #2

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

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

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

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

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

7.0 PROGRAM FUNCTIONAL FLOW CHARTS

Attached

8.0 PROGRAM LISTING

Attached

```
        FLOW CHART
     ****************
CZQMCED  D-124K MEM EXER 16K

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

TABLE OF CONTENTS
****

TABLE OF CONTENTS
****

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

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

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

```
            -----------------
           /   PROGRAM MAP    \YES
          /POINTING TO BANKS 0\---------
         /        & 1?         \       I
         -----------------------       I
                    I NO               I
                    V     RELO(59)     I
        *************************      I
        ** RELOCATE PROGRAM TO **      I
        **     BANKS 0 & 1     **      I
        **                     **      I
        *************************      I
                    I<-------------------
                    V
            -----------------
           /  RESTART FLAG   \YES           ************
          /      (R5=0)?      \----------->*START1(16)*
         /                     \            ************
         -----------------------
                    I NO
                    V     RESLDR(59)
        *************************
        **                     **
        **   RESTORE LOADERS   **
        **                     **
        *************************
                    I
                    V
               ***********
.=572          **HALT   **
               ***********
```

```
      ***********                             ***********                             ***********
      **$PWRDN **                             **$PWRUP **                             **$ILLUP **
      ***********                             ***********                             ***********
           I                                       I                                       I
           V                                       V                                       V
***********************                 ***********************                       ***********
*  $ILLUP -> VECTOR   *                 *  WAIT LOOF FOR TTY  *                       **HALT   **
*   SAVE REGISTERS    *                 *  RESTORE REGISTERS  *                       ***********
*  $PWRDN -> VECTOR   *                 *  $PWRDN -> VECTOR   *
***********************                 ***********************
           I                                       I
           V                                       V     $PRINT(63)
      ***********                          ***.****************
      **HALT   **                         /  TYPE POWER FAIL /
      ***********                        /     MESSAGE      /
                                        /                  /
                                        *******************
                                                   I
                                                   V
                                              ***********
                                              **RETURN **
                                              ***********
```

```
                  ***********************
                  *  STANDAPD '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 **                 SA=200         **START  **
                                  ***********                                ***********
                                       I                                          I
                               *****************                       *****************
                               * SET FLAG FOR  *                       *CLEAR FLAG FOR *
                               *   SELECTING   *                       *   SELECTING   *
                               *  PARAMETERS   *                       *  PARAMETERS   *
                               *****************                       *****************
                                       I                                          I
                                       ------------------------------------------
                                                           I
                                                STARTA     V
                                                ***********************
                                                *  CLEAR COMMON TAGS  *
                                                *   INIT SP INIT PF   *
                                                *       VECTOR        *
                                                ***********************
                                                           I
                                                           V
                                                ***********************
                                                *   SET UP SOFTWARE   *
                                                *SWITCH REGISTER IF NO*
                                                *    HARDWARE SWR     *
                                                ***********************
                                                           I
                                                           V
                                                   ********************
                                                  /TYPE PROGRAM TITLE/
                                                 /  (ONCE ONLY)     /
                                                /                  /
                                                *******************
                                                           I
                                                           V
                                                   -----------------
                                                  /   HAS PROGRAM   \YES
                                                 /    RELOCATED?     \----------->************
                                                /                     \           *RESTAR(O2)*
                                                -----------------------           ************
                                                           I NO
                                                           V
                                               *************************
                                               **                     **
                                               **    SAVE LOADERS     **
                                               **                     **
                                               *************************
                                                           I
                                                           I
                                                           I
                                                           V
```

```
   -----------------
  / CLEAR 'MMAVA'.  \YES
 / MEMORY MANAGEMENT \--------------------
/ EXITS AND DESIRED?  \                  I
-----------------------                  V
           I NO                  ***********************
           I                  *SET UP MEM MGMT. SET *
           I                  *  'MMAVA' FLAG TYPE  *
           I                  *  'KT11 AVAILABLE'   *
           I                  ***********************
           I                             I
           I<-----------------------------
           I
           V
   *****************
   *               *
   *TURN OFF CACHE *
   *               *
   *****************
           I
           I
           V
```

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

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

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

```
           ----------------------
          / TYPE "TO" FLAG        \NO
         /    SET? R0=0            \--------------
        /                           \            I
        ------------------------------           I
                    I YES                        I
                    V                            I
           ***********************               I
          /                      /               I
         /    TYPE "TO"         /                I
        /                      /                 I
        **********************                   I
                    I                            I
                    V                            I
           ***********************               I
          /                      /               I
         /  TYPE ADDRESS-1      /                I
        /                      /                 I
        **********************                   I
                    I<----------------------------
                    V
               ***********
               **HALT   **
               ***********
               ************
               *GMPR(08)  *
               ************
         GMPR       I
         ***********************
         *SET UP TABLE POINTERS*
         *SET UP TIMEOUT VECTOR*
         *                     *
         ***********************
------------------->I
I        GMPRA      V
I        ***********************
I        *  CLEAR FLAG. TEST   *  TIMEOUT TRAP
I        *  REGISTER ADR FOR   *-->>>>>>>>>>>>-----------
I        *       TIMEOUT       *                        I
I        ***********************           GMPRB        V
I                   I                      ************************
I                   V                      * ADJUST SP. SET FLAG  *
I        ***********************           *TO INDICATE REGISTER  *
I        *SAVE REGISTER ADR IN *           *     NOT PRESENT      *
I        * SECOND TABLE (MPRX) *           ************************
I        *                     *                        I
I        ***********************                        I
I                   I<-----------------------------------
I           GMPRC   V
I           -----------------
I    NO    /CLEAR MAP. END OF\
----------/  TABLE REACHED?   \
         /                     \
         -----------------------
                    I YES
                    V
         ***********************
         *  TERMINATE SECOND   *
         * TABLE (MPRX) RESET  *
         *   TIMEOUT VECTOR    *
         ***********************
                    I
                    V
            -----------------
           /   ANY PARITY    \NO
          / REGISTERS FOUND?  \-------------------------
         /                     \                        I
         -----------------------                        V
                    I YES                  ********************
                    I                     / TYPE "NO MEMORY  /
                    I                    / PARITY REGISTERS /
                    I                   /     FOUND".      /
                    I                  ********************
                    I                            I
                    I                            V
                    I                      ************
                    I                      *MANUAL(15)*
                    I                      ************
                    V
          ************************
          * INIT ALL REGISTERS   *
          *   SET UP POINTERS    *
          *                      *
          ************************
                     I
          MAPRB      V
          ************************
          *WRITE WRONG PARITY IN*
          * EACH BANK OF MEMORY *
          *                      *
          ************************
                     I
                     V
          ************************
          * FIND WHICH REGISTER  *
          *CONTROLS WHICH BANK.  *
          *                      *
          ************************
                     I
          TMAP       V
          ************************
          *TYPE PARITY REGISTER  *
          *       ADDRESS        *
          *                      *
          ************************
                     I
                     V      TYPMAP(62)
         *************************
         ** TYPE MAP OF MEMORY  **
         ** CONTROLLED BY EACH  **
         **      REGISTER       **
         *************************
                     I
                     I
                     I
                     I
                     I
                     I
                     I
                     I
                     I
                     I
                     V
         CTRLS
         ***********************
         *                     *
-------->*CLEAR PAPITY REGISTER*
I        *                     *
I        ***********************
I                   I
I                   V                                         $ERROR(63)
I           -----------------                 *************************
I          /                 \NO              **   ERROR: REGISTER   **
I         /  ALL BITS CLEAR   \------------->**    DOESN'T CLEAR    **
I        /                     \              **                     **
I        -----------------------              *************************
I                   I YES                                 I
I                   V<-------------------------------------
I        ***********************
I        *    SET AND CLEAR    *
I        *NON-RESERVED BITS OF *
I        *      REGISTERS      *
I        ***********************
I                   I
I                   V                                         $ERROR(63)
I           -----------------                 *************************
I          /                 \NO              ** ERROR: BIT STUCK IN **
I         / SET AND CLEAR OK? \------------->**      REGISTER       **
I        /                     \              **                     **
I        -----------------------              *************************
I                   I YES                                 I
I                   V<-------------------------------------
I           -----------------
I    YES   /                 \
----------/  MORE REGISTERS?  \
         /                     \
         -----------------------
                    I
         RESCHK     V
         ***********************
         *   SET BITS IN ALL   *
         *REGISTERS. DO "RESET"*
         *                     *
         ***********************
                    I
                    V                                         $ERROR(63)
            -----------------                 *************************
           /  ALL REGISTERS  \NO              ** ERROR: RESET FAILED **
          /     CLEARED?      \------- ----->**  TO CLEAR REGISTER  **
         /                     \              **                     **
         -----------------------              *************************
                    I YES                                 I
                    I<-------------------------------------
                    I
                    V
```

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

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

```
                   TST1             INITMM(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V
I                  ***********************
I                  *   WRITE PHYSICAL    *
I                  *ADDRESS VALUE IN EACH*
I                  *    WORD LOCATION    *
I                  ***********************
I                             I
I                             V     MMUP(56)
I                 *************************
I   MORE MEMORY   **    UPDATE ADDRESS   **
------------------**       POINTERS      **
                  **                     **
                  *************************
                              IDONE
                              V     INITDN(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V                                         $ERROR(63)
I                      ----------------                   *************************
I                     /    DOES EACH     \NO              **ERROR: ADDRESS VALUE **
I                    /   LOCATION HAVE    \---------------**   NOT IN LOCATION   **
I                   /   ADDRESS VALUE?     \              **                     **
I                   ------------------------              *************************
I                             I YES                                   I
I                             I<------------------- -----------------I
I                             V     MMDOWN(56)
I                 *************************
I   MORE MEMORY   **    UPDATE ADDRESS   **
-----------------*        POINTERS       **
                 **                      -*
                 *************************
                              IDONE
                              I
                              I
                              I
                              V
                   TST2             INITMM(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
------------------------------>I
I                              V
I                  ***********************
I                  *   WRITE PHYSICAL    *
I                  *ADDRESS VALUE IN EACH*
I                  *    BYTE LOCATION    *
I                  ***********************
I                              I
I                              V     MMUP(56)
I                 *************************
I   MORE MEMORY   **    UPDATE ADDRESS   **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                               V      INITDN(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
------------------------------>I
I                              V                                        $ERROR(63)
I                     ------------------                     *************************
I                    / DOES EACH BYTE   \NO                  **ERROR: ADDRESS VALUE **
I                   /   LOCATION HAVE    \------------------>**NOT IN BYTE LOCATION **
I                  /   ADDRESS VALUE?     \                  **                     **
I                  -----------------------                   *************************
I                              I YES                                     I
I                              I<----------------------------------------I
I                              V      MMDOWN(56)
I                 *************************
I   MORE MEMORY   **    UPDATE ADDRESS   **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                               I
                               I
                               I
                               V
                   TST3             INITDN(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V
I                  ************************
I                  *     WRITE ONE'S      *
I                  *  COMPLEMENT OF ADR   *
I                  * INTO WORD LOCATION   *
I                  ************************
I                             I
I                             V     MMDOWN(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              V     INITMM(55)
                  ***********.*************
                  ** INITIALI_E ADDRESS  **
                  **      PC'NTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V                                        $ERROR(63)
I                     -----------------              *************************
I                    / DOES EACH WORD  \NO           **ERROR: COMPLEMENT OF **
I                   /HAVE COMPLEMENT OF \----------->**ADR. NOT IN WORD LOC.**
I                  /     ADR. VALUE?     \           **                     **
I                  ------------------------          *************************
I                             I YES                              I
I                             I<---------------------------------I
I                             V     MMUP(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              I
                              I
                              V
```

```
                    TST4              INITMM(55)
                   **************************
                   ** INITIALIZE ADDRESS   **
                   **       POINTERS       **
                   **                      **
                   **************************
-- ---------------------------->I
I                               V
I                    ************************
I                    * WRITE BANK # VALUE   *
I                    *   INTO EACH BYTE     *
I                    *      LOCATION        *
I                    ************************
I                               I
I                               V      MMUP(56)
I                  **************************
I   MORE MEMORY    **    UPDATE ADDRESS    **
-------------------**       POINTERS       **
                   **                      **
                   **************************
                                IDONE
                                V     INITMM(55)
                   **************************
                   ** INITIALIZE ADDRESS   **
                   **       POINTERS       **
                   **                      **
                   **************************
------------------------------->I
I                               V                                     $ERROR(63)
I                      ------------------              **************************
I                     / DOES EACH BYTE   \NO           ** ERROR: BANK # VALUE  **
I                    /  HAVE ITS BANK #   \----------->**   NOT IN LOCATION    **
I                   /       VALUE?         \           **                      **
I                  --------------------------          **************************
I                               I YES                               I
I                               I<------------------------ ---------I
I                               V      MMUP(56)
I                  **************************
I   MORE MEMORY    **    UPDATE ADDRESS    **
-------------------**       POINTERS       **
                   **                      **
                   **************************
                                IDONE
                                I
                                I
                                I
                                V
                   TST5             INITDN(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
-----------------------------)I
I                             V
I                  **********************
I                  *WRITE 1'S COMPLEMENT *
I                  * OF BANK NUMBER INTO *
I                  *    BYTE LOCATION    *
I                  **********************
I                             I
I                             V      MMDOWN(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              V     INITDN(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
-----------------------------)I
I                             V                                        $ERROR(63)
I                     ------------------             *************************
I                    / DOES EACH BYTE  \NO           **ERROR: COMPLEMENT OF **
I                   /HAVE COMPLEMENT OF  \----------)** BANK # NOT IN BYTE  **
I                  /     BANK VALUE?     \           **        LOC.         **
I                  ------------------------          *************************
I                             I YES                              I
I                             I<---------------------------------I
I                             V     MMDOWN(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------ -----------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              I
                              I
                              V
```

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

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

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

```
                              ---------------------------------------------------------
                              I
                   TST11      V     SETCON(57)
                  *************************
                  **PUT ALL ONE'S IN ALL **
                  **       MEMORY        **
                  **                     **
                  *************************
                              I
                              V     INITMM(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
------------------------------>I
I                             V     ROTATE(57)
I                 *************************
I                 **   CLEAR C-BIT AND   **
I                 **ROTATE IT THROUGH TWO**
I                 **        BYTES        **
I                 *************************
I                             I
I                             V                                         SERROR(63)
I                     ------------------             *************************
I                    / C-BIT CLEAR AND \NO           **  ERROR: ROTATING 0  **
I                   /   -1 IN MEMORY    \----------->**       FAILED.       **
I                  /      LOCATION?      \           **                     **
I                  ------------------------          *************************
I                             I YES                              I
I                             I<---------------------------------I
I                             V     MMUP(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              I
                              V
                   TST12            SETCON(57)
                  **************************
                  **PUT ALL ZEROS IN ALL  **
                  **       MEMORY         **
                  **                      **
                  **************************
                              I
                              V     INITMM(55)
                  **************************
                  ** INITIALIZE ADDRESS   **
                  **      POINTERS        **
                  **                      **
                  **************************
-----------------------------)I
I                             V     ROTATE(57)
I                 **************************
I                 **SET C-BIT AND ROTATE  **
I                 **IT THROUGH TWO BYTES  **
I                 **                      **
I                 **************************
I                             I
I                             V                                          $ERROR(63)
I                     ------------------                                **************************
I                    / C-BIT SET AND 0 \NO                              **  ERROR: ROTATING 1   **
I                   /IN MEMORY LOCATION?\-----------)                   **       FAILED         **
I                  /                     \                              **                      **
I                  ------------------------                             **************************
I                             I YES                                 I
I                             I<----------------------------------I
I                             V     MMLP(56)
I                 **************************
I   MORE MEMORY   **   UPDATE ADDRESS     **
------------------**      POINTERS        **
                  **                      **
                  **************************
                              IDONE
                              I
                              I
                              I
                              V
                   TST13            INITMM(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V     W3X9(57)
I                 *************************
I                 **   WRITE 256. WORD   **
I                 **     BLOCKS WITH     **
I                 ** 0,0,0,0,-1,-1,-1,-1 **
I                 *************************
I                             I
I                             V     MMUP(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              V     INITMM(55)
                  *************************
                  ** INITIALIZE ADDRESS  **
                  **      POINTERS       **
                  **                     **
                  *************************
----------------------------->I
I                             V                                        $ERROR(63)
I                     ------------------             *************************
I                    /256. WORD BLOCKS \NO           **   ERROR: 3 XOR 9    **
I                   /   WRITTEN WITH    \----------->**   PATTERN FAILURE   **
I                  / 0,0,0,0,-1,-1,-1,-1 \           **                     **
I                  -----------------------           *************************
I                             I YES                              I
I                             I<---------------------------------I
I                             V     MMUP(56)
I                 *************************
I   MORE MEMORY   **   UPDATE ADDRESS    **
------------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              I
                              I
                              I
                              I
                              V
```

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

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

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

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

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

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

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

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

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

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

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

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

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

```
                                                 TST23            INITMM(55)
                                                *************************
                                                ** INITIALIZE ADDRESS  **
                                                **       POINTERS      **
                                                **                     **
                                                *************************
                              ----------------------------->I
                              I                             V
                              I                  ***********************
                              I                  *   PUT INSTRUCTION   *
                              I                  *   'MOVB R3,-(R2)'   *
                              I                  * AND RTS INTO MEMORY *
                              I                  ***********************
                              I                             I
                              I                             V
                              I                 *************************
                              I                 **JSR TO ADDRESS UNDER **                      ***********
                              I                 **       TEST          **-->>>---------------->**MAUT   **
                              I                 **                     **                      ***********
                              I                 *************************                           I
                              I                                                                     V
                              I                                                         ***********************
                              I                                                         * EXICUTE INSTRUCTION *
                              I                                                         *  IN MEMORY ADDRESS  *
                              I                                                         *  UNDER TEST (MAUT)  *
                              I                                                         ***********************
                              I                                                                     I
                              I                                                                     V
                              I                                                               ***********
                              I                             ------------------------------<<<--**RETURN **
                              I                                                               ***********
                              I                             I
                              I                             I
                              I                             I
                              I                             V                                        $ERROR(63)
                              I                     -----------------               *************************
                              I                   /      DID THE     \NO           ** ERROR: INSTRUCTION  **
                              I                  /INSTRUCTION EXICUTE\------ ---->**DIDN'T MODIFY ITSELF.**
                              I                 /       PROPERLY?     \            **                     **
                              I                 ------------------------           *************************
                              I                             I YES                              I
                              I                             I<----------------------------------
                              I                             V     MMUP(56)
                              I                 *************************
                              I   MORE MEMORY   **  UPDATE ADDRESS     **
                              ------------------**       POINTERS      **
                                                **                     **
                                                *************************
                                                            IDONE
                                                            I
                                                            I
                                                            I
                                                            V
                                                TST24            INITMM(55)
                                               *************************
                                               ** INITIALIZE ADDRESS  **
                                               **      POINTERS       **
                                               **                     **
                                               *************************
                             ----------------------------->I
                             I                             V
                             I                  ***********************
                             I                  *   PUT INSTRUCTION   *
                             I                  *     'NEG (R2)'      *
                             I                  * AND RTS INTO MEMORY *
                             I                  ***********************
                             I                             I
                             I                             V
                             I                 *************************
                             I                 **JSR TO ADDRESS UNDER **                      ***********
                             I                 **       TEST          **--)))---------------->**MAUT   **
                             I                 **                     **                      ***********
                             I                 *************************                           I
                             I                                                                     V
                             I                                                          ***********************
                             I                                                          * EXICUTE INSTRUCTION *
                             I                                                          *  IN MEMORY ADDRESS  *
                             I                                                          *  UNDER TEST (MAUT)  *
                             I                                                          ***********************
                             I                                                                     I
                             I                                                                     V
                             I                                                                ***********
                             I                             ------------------------------<<<--**RETURN **
                             I                             I                                  ***********
                             I                             I
                             I                             I
                             I                             V                                        $ERROR(63)
                             I                     ------------------             *************************
                             I                    /     DID THE      \NO          ** ERROR: INSTRUCTION  **
                             I                   /INSTRUCTION EXICUTE\----------->**DIDN'T MODIFY ITSELF.**
                             I                  /      PROPERLY?      \           **                     **
                             I                  ------------------------          *************************
                             I                             I YES                              I
                             I                             I<----------------------------------
                             I                             V     MMUP(56)
                             I                 *************************
                             I   MORE MEMORY   **   UPDATE ADDRESS    **
                             ------------------**      POINTERS       **
                                               **                     **
                                               *************************
                                                           IDONE
                                                           I
                                                           I
                                                           I
                                                           V
```

TEST 25: EXICUTE DATIP, DATOB (LO BYTE)

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

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

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

```
             ^                       ----------------
             I                      /  PASS COUNT =   \NO
             I                     /    0 OR 3?        \-------------------------I
             I                    /                      \                       I
             I                    -----------------------                        V
             I                               I YES                       *****************
             I                               V                           *UPDATE TO NEXT *
             I                       ----------------                    *    ADDRESS    *
             I          ***    NO   /                 \                  *               *
             I          *A*<-------/    DONE PASS?     \                 *****************
             I          ***       /                      \                       I
             I                    -----------------------                        I
             I                               I YES                               I
             I                               V                                   I
             I                       *****************                           I
             I                       *INCREMENT PASS *                           I
             I                       *    COUNTER    *                           I
             I                       *               *                           I
             I                       *****************                           I
             I                               I                                   I
             I    $MMUP                      V                                   I
**************************           ----------------                            I
**MMUP: UPDATE ADDRESS  **    YES   /DONE FOUR PASSES \                          I
**       POINTERS       ** <-------/    ON A 4K?       \                         I
**                      **        /                      \                       I
**************************        -----------------------                        I
             IDONE                           I NO                                I
             I                               V                                   I
             I                       *****************                           I
             V                       * SWAP BYTES IN *                           I
 ***********************             *CHECK REGISIER *                           I
 *   NO MORE MEMORY.   *             *               *                           I
 *  GO TO NEXT TEST.   *             *****************                           I
 *                     *                     I                                   I
 ***********************                     I                                   I
                                             I                                   I
                                             I<----------------------------------I
                                             I
                                             V
```

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

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

```
I---------------------------->I
I                             V
I                  ***********************
I                  *   INITIALIZE DATA   *
I                  *PATTERN FOR CHECKING *
I                  *                     *
I                  ***********************
I                             I
I         ------------------->I
I         I                   V                                       $ERROR(63)
I         I           -----------------              *************************
I         I          / IS DATA READ =  \NO           **   ERROR: REFRESH    **
I         I         /  DATA WRITTEN?    \----------->**  CIRCUITRY FAILED   **
I         I        /                     \           **                     **
I         I        -----------------------           *************************
I         I                   I YES                              I
I         I                   I<---------------------------------I
I         I                   V
I         I        ***********************
I         I        *                     *
I         I        *COMPLIMENT CHECK DATA*
I         I        *                     *
I         I        ***********************
I         I                   I
I         I                   V
I         I           -----------------
I         I    NO    /                 \
I         I---------/  END OF BLOCK     \
I                  /                     \
I                  -----------------------
I                             I YES
I                             V     MMUP(56)
I                 *************************
I MORE MEMORY     **   UPDATE ADDRESS    **
I-----------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              V
```

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

```
I----------------------------->I
I                              V
I                  ***********************
I                  *   INITIALIZE DATA   *
I                  *PATTERN FOR CHECKING *
I                  *                     *
I                  ***********************
I                              I
I         -------------------->I
I         I                    V                                        $ERROR(63)
I         I           -----------------              *************************
I         I          / IS DATA READ =  \NO           **   ERROR: REFRESH    **
I         I         /   DATA WRITTEN?   \----------->** CIRCUITRY FAILED    **
I         I        /                     \           **                     **
I         I        -----------------------           *************************
I         I                    I YES                               I
I         I                    I<----------------------------------I
I         I                    V
I         I        ***********************
I         I        *                     *
I         I        *COMPLIMENT CHECK DATA*
I         I        *                     *
I         I        ***********************
I         I                    I
I         I                    V
I         I           -----------------
I         I   NO     /                 \
I         I---------/   END OF BLOCK    \
I                  /                     \
I                  -----------------------
I                              I YES
I                              V     MMUP(56)
I                 *************************
I MORE MEMORY     **   UPDATE ADDRESS    **
I-----------------**      POINTERS       **
                  **                     **
                  *************************
                              IDONE
                              I
                              V
```

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

```
                              $EOP
                              ***********************
                              *STANDARD 'SYSMAC' END*
                              *   OF PASS ROUTINE   *
                              *                     *
                              ***********************
                                         I
                                         V
                                 ********************
                                / TYPE END OF PASS /
                               /     MESSAGE      /
                              /                  /
                              *******************
                                         I
                                         V
                                 -----------------
************             NO     /                 \YES                 ***********
*START1(16)*<------------------/ MONITOR PRESENT?  \------------------>**EXIT   **
************                  /                     \                  ***********
                              -----------------------
```

```
                                                                H06
CZQMCEO  0-124K MEM EXER 16K                                            DECFLO VER  00.07 10-JAN-78 13:19  PAGE 55
MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES                                                                               SEQ 0072
              ***********                             ***********                             ***********
              **MMINIT **                             **INITMM **                             **INITDN **
              ***********                             ***********                             ***********
                   I                                       I                                       I
        MMINIT     V                            INITMM     V                            INITDN     V
        ***********************                 ***********************                 ***********************
        *SET UP ALL THE MEMORY*                 *LOCATE FIRST BANK IN *                 *   SET UP ADDRESS    *
        *MANAGEMENT REGISTERS.*                 *      TEST MAP       *                 *  POINTERS TO LAST   *
        *                     *                 *                     *                 *  POSSIBLE ADDRESS   *
        ***********************                 ***********************                 ***********************
                   I                                       I                                       I
                   V                                       V                                       V     MMDOWN(56)
              ***********                          ------------------                  *************************
              **RETURN **                         /  LAST BANK AS   \NO                **SEARCH DOWNWARDS FOR **
              ***********                        /       WELL?       \----------       **TOP BANK IN TEST MAP **
                                                /                     \        I       **                     **
                                                -----------------------        I       *************************
                                                           I YES               I                   I
                                                           V                   I                   V
                                                ***********************        I           -----------------
                                                *                     *        I          / LAST BANK FLAG  \NO
                                                *SET UP LAST ADR MASK *        I         /       SET?        \----------
                                                *                     *        I        /                     \        I
                                                ***********************        I        -----------------------        I
                                                           I<-------------------                   I YES               I
                                                           V                                       V                   I
                                                   ------------------                   ***********************        I
                                                  /                 \NO                 *PUT LAST ADDRESS INTO*        I
                                                 /  8K BLOCK SIZE?   \----------        *   ADDRESS POINTER   *        I
                                                /                     \        I        *                     *        I
                                                -----------------------        I        ***********************        I
                                                           I YES               I                   I                   I
                                                           V                   I                   I                   I
                                                ***********************        I                   I                   I
                                                * SET UP SECOND BANK  *        I                   I                   I
                                                *      POINTERS       *        I                   I                   I
                                                *                     *        I                   I                   I
                                                ***********************        I                   I                   I
                                                           I<-------------------                   I                   I
                                                           V                                       I                   I
                                                   ------------------                              I                   I
                                                  / FIRST BANK FLAG \NO                            I                   I
                                                 /       SET?        \-------->------------------->I<-------------------
                                                /                     \        I        INITEX     V
                                                -----------------------        I        ***********************
                                                           I YES               I        * SAVE RETURN PC FOR  *
                                                           V                   I        *    LOOP ADDRESS     *
                                                ***********************        I        *                     *
                                                *  PUT FIRST ADDRESS  *        I        ***********************
                                                *INTO ADDRESS POINTER *        I                   I
                                                *                     *        I                   V
                                                ***********************        I              ***********
                                                           I                   I              **RETURN **
                                                           I                   I              ***********
                                                           ------------------->I
```

```
               ***********                                                        ***********
               **MMUP   **                                                        **MMDOWN **
               ***********                                                        ***********
                    I                                                                  I
           MMUP     V                                                          MMDOWN  V
            -----------------                                                  -----------------
           /                 \YES                                             /                 \YES
          /   LAST BANK?      \--------------------------                    /    FIRST BANK?    \----------------------
         /                     \                        I                   /                      \                  I
        ------------------------                        I                   ------------------------                  I
                    I NO                                V                              I NO                           V
                    I                           -----------------                      I                    - ------------------
                    I                          /                 \NO                   I                   /                 \NO
                    I                         /   LAST ADDRESS?   \-----               I                  /  FIRST ADDRESS    \-----
                    I                        /                     \    I              I                /                      \    I
                    I                        -----------------------    I              I                -------------------------   I
                    I                                   I YES           I              I                              I YES         I
                    V                                   V     CKPMER(61 I              V                              I             I
        ************************            **************************  I   ************************                  I             I
        *FIND NEXT BANK (NEXT  *  NONE      ** CHECK FOR NON-TRAP   **  I   * FIND NEXT LOWER 4K   *  NONE FOUND      I             I
        *TWO BANKS IF 8K BLOCK*------------>**MEMORY PARITY ERRORS.**   I   *        BANK          *----------------->I             I
        *        SIZE)         * FOUND      **                      **  I   *                      *                  I             I
        ************************            **************************  I   ************************                  V             I
                    I                                   I               I              I                         ***********        I
                    V                                   V               I              V                         **RETURN **        I
            -----------------                      ***********          I      -----------------                 ***********        I
           /                 \NO                   **RETURN **          I     /                 \NO                                 I
          /   LAST BANK?      \-----------         ***********          I    /    FIRST BANK?    \-------                             I
         /                     \         I                              I   /                      \    I                             I
        ------------------------         I                              I   ------------------------    I                             I
                    I YES                I                              I              I YES            I                             I
                    V                    I                              I              V                I                             I
        ************************         I                              I   ************************    I                             I
        * SET UP LAST ADDRESS  *         I                              I   * SET UP LAST ADDRESS  *    I                             I
        *        MASK          *         I                              I   *        MASK          *    I                             I
        *                      *         I                              I   *                      *    I                             I
        ************************         V                              V   ************************    V                             V
                    I<---------------------------------------------------              I<----------------------------------------------
                    V                                                                  V
        ************************                                            ************************
        *ADJUST RETURN ADDRESS*                                             *ADJUST RETURN ADDRESS*
        *       TO LOOP        *                                            *       TO LOOP        *
        *                      *                                            *                      *
        ************************                                            ************************
                    I                                                                  I
                    V                                                                  V
               ***********                                                        ***********
               **RETURN **                                                        **RETURN **
               ***********                                                        ***********
```

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

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

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

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

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

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

```
                      ***********                                                 ***********
                      **RELO   **                                                 **RESLDR **
                      ***********                                                 ***********
                           I                                                           I
                   RELO    V                                                RESLDR     V
                   -----------------                                        ***********************
            NO     /     MEMORY     \YES                                    * MOVE "LOADER" FROM  *
            -----/    MANAGEMENT?    \-----                                 * END OF 8K TO TOP OF *
            I   /                     \   I                                 *       MEMORY        *
            I   -----------------------   I                                 ***********************
            I                             I                                            I
            V     RELOC(58)               V                                            V
*************************      ***********************                            ***********
**RELOCATE PROGRAM BACK**      *   SET UP MEM MGMT   *                            **RETURN **
**     TO BANKS 0+1    **      *REGISTERS TO POINT TO*                            ***********
**                     **      *      BANKS 0+1      *
*************************      ***********************
            I                             I
            V                             V     RELOC(58)
 ***********************      *************************
 *ADJUST ALL PERTINENT *      **RELOCATE PROGRAM BACK**
 *  ADDRESS POINTERS   *      **     TO BANKS 0+1    **
 *                     *      **                     **
 ***********************      *************************
            I                             I
            -------------->I<--------------
                           I
                           V
                      ***********
                      **RETURN **
                      ***********
```

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

```
***********                                        ***********
**SPRNT  **--->I                                   **TYPMAP **
***********    I                                   ***********
               I                                        I
               I                                TYPMAP  V
***********    I                                ------------------            *********************
**SPRNTQ **--->I                               /   MAP CONTAIN   \NO        /  TYPE EMPTY MAP  /
***********    I                              /      FLAGS?       \--------->/    MESSAGE      /
               I                             /                     \       /                  /
               I                             ------------------------      *******************
***********    I                                        I YES                         I
**SPRNTR **--->I                                        V                             I
***********    I                                ********************                  I
               I                               /TYPE FIRST + LAST /                   I
               I                              / ADDRESS OF BANKS /                    I
***********    I                             /      FOUND       /                     I
**SPRNTO **--->I                             *******************                      I
***********    I                                        I<----------------------------
               I                                        V
               I                                   ***********
***********    I                                   **RETURN **
**SPRNT1 **--->I                                   ***********
***********    I
               I
               I
***********    I
**SPRNT3 **--->I
***********    I
               I
               I
***********    I
**SPRNT2 **--->I
***********    I
               I
               V
    ***********************
    * ROUTINES TO SET UP  *
    *   DATA FOR ERROR    *
    *      TYPEOUTS.      *
    ***********************
               I
               V
          ***********
          **RETURN **
          ***********
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

                         ***********
                         **.END   **
                         ***********
```

```
BANKNO      57*   57
CKPMEP      24    35    56    61*   61
CLRPAR      60*   60
CTRLS       14
DONE        53
ERRTRP      63    63*
ERRTYP      63    63*
EXIT        54
GMPR        08    12    12
GMPRA       12
GMPRB       12
GMPRC       12
HALT        03    04    04    08    11    58    63
INITDN      17    18    19    21    21    55*   55
INITEX      55
INITMM      17    18    19    20    20    22    23    24    25    26    27    27    28    29    29    30    31    31
            32    33    33    34    35    39    40    41    42    43    44    45    46    49    49    51    51    55*
            55    57
MAMF        38    60*   60
MANUAL      12    15*   15
MANUL1      15
MANUL2      16
MAPRB       13
MAUT        40    41    42    43    44    45
MMDOWN      17    18    19    21    21    55    56*   56
MMINIT      09    55*   55
MMUP        17    18    19    20    20    22    23    24    25    26    27    27    28    29    29    30    31    31
            32    33    33    34    38    39    40    41    42    43    44    45    49    50    51    52    56*   56
            57
PBTRP       36
PESRV       60*   60
PHYADR      57*   57
PSCAN       60    61    61*   61
RELO        03    53    59*   59
RELOC       58*   58    58    58    59    59
RELTOP      53    58*   58
RESCHK      14
RESLDR      03    53    59*   59
REST1       02
REST2       02
RESTAR      02*   02    06
RESTOR      02*
RETURN      04    40    41    42    43    44    45    55    55    56    56    56    56    57    57    57    57    57
            58    58    59    59    60    60    60    61    61    62    62    63    63    63    63    63    63    63
            63    63    63    63
ROTATE      25    26    57*   57
SELECT      06*
SETAE       35    60*
SETCON      25    26    35    57*   57
SPRNT       62*
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  CZQMCEO   O-124K MEMORY EXERCISER, 16K VER
        ;*COPYRIGHT (C) 1975,1977
        ;*DIGITAL EQUIPMENT CORP.
        ;*MAYNARD, MASS. 01754
        ;*
        ;*PROGRAM BY BRUCE BURGESS/KEN CHAPMAN
        ;*
        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
        ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
        ;*

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

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

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

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

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

```
                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  003352                        JMP     TIMOUT           ;TYPE OUT MEMORY MAP, BYTE BY BYTE.

                000004                                .=ERRVEC
        000004  025060                                .WORD   ERRTRP
        000006  000000                                .WORD   0

                                              .SBTTL  ACT11 HOOKS

                                              ;;*************************************************************
                                              ;HOOKS REQUIRED BY ACT11
                000010                                $SVPC=.                  ;SAVE PC
                000046                                .=46
        000046  014174                                $ENDAD                   ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                000052                                .=52
        000052  040000                                .WORD   BIT14            ;;2)SET LOC.52 TO BIT14
                000010                                .=$SVPC                  ;; RESTORE PC
        000300                                  .=300
                                ;;**************************************************************
                                ;*   THE FOLLOWING ROUTINES ARE LOCATED IN THE VECTOR AREA (0-1000) SO THAT
                                ;*    THEY CAN BE PROTECTED BY SELECTING SW05 (SEE DOCUMENT FOR USE OF SW05).
                                ;*    THE CODE CAN ALSO BE RUN FROM ANY BANK OF MEMORY, ASSUMING MEMORY
                                ;*    MANAGEMENT IS DISABLED BY "CONSOLE START".
                                ;;**************************************************************
000300  005005                  RESTAR: CLR     R5                      ;CLEAR FLAG TO INDICATE RESTART.
000302  000401                          BR      REST1                   ;GO RESTORE PROGRAM BEFORE RESTARTING.
000304  010705                  RESTOR: MOV     PC,     R5              ;PUT DATA INTO FLAG FOR RESTORE.
000306  012706  001100          REST1:  MOV     #STACK, SP              ;SET UP THE STACK POINTER.
000312  005767  001206                  TST     MEMMAP                  ;CHECK IF THE MEMORY HAS BEEN MAPPED.
000316  001002                          BNE     REST2                   ;BR IF MEMORY MAPPED.
000320  000167  002330                  JMP     STARTA                  ;GO START
000324  005767  000256          REST2:  TST     MMAVA                   ;CHECK IF MEM MGMT AVAILABLE.
000330  001470                          BEQ     10$                     ;BR IF NO MEM MGMT.
000332  032737  000001  177572          BIT     #BIT0,  @#SR0           ;CHECK IF MEM MGMT ACTIVE.
000340  001034                          BNE     2$                      ;BR IF MEM MGMT ALREADY SET UP.
000342  012700  172300                  MOV     #KIPDR0,R0              ;POINT TO FIRST MEM MGMT DDATA REG.
000346  012701  000010                  MOV     #8.,    R1              ;SET UP COUNTER.
000352  012720  077406          1$:     MOV     #077406,(R0)+           ;MAP FIRST 28K 1-FOR-1.
000356  005301                          DEC     R1                      ;COUNT REGESTERS.
000360  001374                          BNE     1$                      ;BR IF MORE REG.
000362  012700  172340                  MOV     #KIPAR0,R0              ;POINT TO FIRST MEM MGMT ADDRESS REG.
000366  005020                          CLR     (R0)+                   ;PAR0 MAPPED INTO BANK0.
000370  012720  000200                  MOV     #200,   (R0)+           ;PAR1 MAPPED INTO BANK1.
000374  012720  000400                  MOV     #400,   (R0)+           ;PAR2 MAPPED INTO BANK2.
000400  012720  000600                  MOV     #600,   (R0)+           ;PAR3 MAPPED INTO BANK3.
000404  012720  001000                  MOV     #1000,  (R0)+           ;PAR4 MAPPED INTO BANK4.
000410  012720  001200                  MOV     #1200,  (R0)+           ;PAR5 MAPPED INTO BANK5.
000414  012720  001400                  MOV     #1400,  (R0)+           ;PAR6 MAPPED INTO BANK6.
000420  012720  007600                  MOV     #7600,  (R0)+           ;PAR7 MAPPED INTO BANK37.
000424  012737  000001  177572          MOV     #BIT0,  @#SR0           ;ENABLE MEM MGMT.
000432  005000                  2$:     CLR     R0                      ;INIT TEMP PAR REG.
000434  016701  000142                  MOV     PRGMAP, R1              ;GET THE  ROGRAM MAP...LO 64K.
000440  016702  000140                  MOV     PRGMAP+2,R2             ;...HI 64K.
000444  006202                  3$:     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  016260                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    $TSTNM
 000562  000167  005272                JMP     START1            ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
 000566  004767  016442        22$:    JSR     PC,      RESLDR   ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
 000572  000000                        HALT                      ;HALT AFTER RESTORING THE LOADERS.
 000574  000167  002054                JMP     STARTA            ;CONTINUE WILL RESTART THE PROGRAM.
                               ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
                               ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
                               ;*  CIRCUMSTANCES.
 000600  000000                RELOCF: .WORD   0        ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
 000602  000000  000000        PRGMAP: .WORD   0,0      ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
 000606  000000                MMAVA:  .WORD   0        ;MEMORY MANAGEMENT AVAILABLE FLAG.
```

```
                                .SBTTL  POWER DOWN AND UP ROUTINES

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

                                ;;***************************************************************
                                ;POWER UP ROUTINE
000662  012737  000756  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
000670  016706  000066                  MOV     $SAVR6,SP       ;;GET SP
000674  005067  000062                  CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
000700  005267  000056          1$:     INC     $SAVR6          ;;WAIT FOR THE INC
000704  001375                          BNE     1$              ;;OF  WORD
000706  012677  000226                  MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
000712  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
000714  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
000716  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
000720  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
000722  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
000724  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
000726  012737  000610  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
000734  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
000742  004567  022510                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
000746  025605                  $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"
        0C1157      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
```

```
001212  000000          $PASS:   .WORD   APASS    ;;PASS COUNT
001214  000000          $DEVCT:  .WORD   ADEVCT   ;;DEVICE COUNT
001216  000000          $UNIT:   .WORD   AUNIT    ;;I/O UNIT NUMBER
001220  000000          $MSGAD:  .WORD   AMSGAD   ;;MESSAGE ADDRESS
001222  000000          $MSGLG:  .WORD   AMSGLG   ;;MESSAGE LENGTH
001224                  $ETABLE:                  ;;APT ENVIRONMENT TABLE
001224     000          $ENV:    .BYTE   AENV     ;;ENVIRONMENT BYTE
001225     000          $ENVM:   .BYTE   AENVM    ;;ENVIRONMENT MODE BITS
001226  000000          $SWREG:  .WORD   ASWREG   ;;APT SWITCH REGISTER
001230  000000          $USWR:   .WORD   AUSWR    ;;USER SWITCHES
001232  000000          $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
001234     000          $MAMS1:  .BYTE   AMAMS1   ;;HIGH ADDRESS,M.S. BYTE
001235     000          $MTYP1:  .BYTE   AMTYP1   ;;MEM. TYPE,BLK#1
                        ;*                        MEM.TYPE BYTE  --  (HIGH BYTE)
                        ;*                                 900 NSEC CORE=001
                        ;*                                 300 NSEC BIPOLAR=002
                        ;*                                 500 NSEC MOS=003
001236  000000          $MADR1:  .WORD   AMADR1   ;;HIGH ADDRESS,BLK#1
                        ;*                          MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001240     000          $MAMS2:   BYTE   AMAMS2   ;;HIGH ADDRESS,M.S. BYTE
001241     000          $MTYP2:  .BYTE   AMTYP2   ;;MEM.TYPE,BLK#2
001242  n00000          $MADR2:  .WORD   AMADR2   ;;MEM.LAST ADDRESS,BLK#2
001244     000          $MAMS3:  .BYTE   AMAMS3   ;;HIGH ADDRESS,M.S.BYTE
001245     000          $MTYP3:  .BYTE   AMTYP3   ;;MEM.TYPE,BLK#3
001246  000000          $MADR3:  .WORD   AMADR3   ;;MEM.LAST ADDRESS,BLK#3
001250     000          $MAMS4:  .BYTE   AMAMS4   ;;HIGH ADDRESS,M.S.BYTE
001251     000          $MTYP4:  .BYTE   AMTYP4   ;;MEM.TYPE,BLK#4
001252  000000          $MADR4:  .WORD   AMADR4   ;;MEM.LAST ADDRESS,BLK#4
001254  000000          $VECT1:  .WORD   AVECT1   ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
001256  000000          $VECT2:  .WORD   AVECT2   ;;INTERRUPT VECTOR#2BUS PRIORITY#2
001260  000000          $BASE:   .WORD   ABASE    ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
001262  000000          $DEVM:   .WORD   ADEVM    ;;DEVICE MAP
001264  000000          $CDW1:   .WORD   ACDW1    ;;CONTROLLER DESCRIPTION WORD#1
001266  000000          $CDW2:   .WORD   ACDW2    ;;CONTROLLER DESCRIPTION WORD#2
001270  000000          $DDW0:   .WORD   ADDW0    ;;DEVICE DESCRIPTOR WORD#0
001272  000000          $DDW1:   .WORD   ADDW1    ;;DEVICE DESCRIPTOR WORD#1
001274  000000          $DDW2:   .WORD   ADDW2    ;;DEVICE DESCRIPTOR WORD#2
001276  000000          $DDW3:   .WORD   ADDW3    ;;DEVICE DESCRIPTOR WORD#3
001300  000000          $DDW4:   .WORD   ADDW4    ;;DEVICE DESCRIPTOR WORD#4
001302  000000          $DDW5:   .WORD   ADDW5    ;;DEVICE DESCRIPTOR WORD#5
001304  000000          $DDW6:   .WORD   ADDW6    ;;DEVICE DESCRIPTOR WORD#6
001306  000000          $DDW7:   .WORD   ADDW7    ;;DEVICE DESCRIPTOR WORD#7
001310  000000          $DDW8:   .WORD   ADDW8    ;;DEVICE DESCRIPTOR WORD#8
001312  000000          $DDW9:   .WORD   ADDW9    ;;DEVICE DESCRIPTOR WORD#9
001314  000000          $DDW10:  .WORD   ADDW10   ;;DEVICE DESCRIPTOR WORD#10
001316  000000          $DDW11:  .WORD   ADDW11   ;;DEVICE DESCRIPTOR WORD#11
001320  000000          $DDW12:  .WORD   ADDW12   ;;DEVICE DESCRIPTOR WORD#12
001322  000000          $DDW13:  .WORD   ADDW13   ;;DEVICE DESCRIPTOR WORD#13
0013^4  000000          $DDW14:  .WORD   ADDW14   ;;DEVICE DESCRIPTOR WORD#14
001326  000000          $DDW15:  .WORD   ADDW15   ;;DEVICE DESCRIPTOR WORD#15
```

```
001330                          $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.

001330                          $APTHD:
001330  000000                  $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
001332  001204                  $MB?DR: .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                         LMAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
 001522  000000                         LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE
 001524                                 MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
 001524  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001526  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001530                                 TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
 001530  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001532  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001534                                 SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
                                                                ; TEST EACH BANK ONCE.
 001534  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001536  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001540                                 PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
 001540  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001542  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001544                                 BITPT:                  ;POINTER TO CURRENT 4K BANK OF MEMORY
 001544  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001546  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001550                                 TMPPT:                  ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
 001550  000000                                 .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
 001552  000000                                 .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
 001554  000000                         MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
                                                                ;SET UP BY "INITMM" AND "INITON" ROUTINEES.
                                                                ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
 001556     000                         SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
 001557     000                         FLAGBK: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP".
 001560     000                         OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
         001562                                 .EVEN
 001562  000000                         FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
                                                                ;FIRST ADDRESS IS USER SELECTABLE.
 001564  000000                         TMPFAD: .WORD   0       ;ADJUSTED FIRST ADDRESS.
 001566  000000                         FADMSK: .WORD   0       ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
                                                                ; TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
 001570  000000  000000                 FADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
 001574  000000                         LSTADR: .WORD   0       ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
                                                                ;LAST ADDRESS IS USER SELECTABLE.
 001576  000000                         TMPLAD: .WORD   0       ;ADJUSTED LAST ADDRESS.
 001600  000000                         LADMSK: .WORD   0       ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
                                                                ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
 001602  000000  000000                 LADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
 001606  000000                         BLKMSK: .WORD   0       ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
 001610  000000                         .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
 001612  000004                         WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND
 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                          .MPRO:  MPRO            ;MEMORY PARITY REGISTER TABLE ADDRESS.
001630  002276                          .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
001632  012026                          .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
001634  002050                          .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
001636  017374                          .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  013760                          .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  0C1120          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,$TMPO,$GDADR,0
001714  001120  000000
001720  001514  001116  001120          DT15:   $VERPC,$ERRPC,$GDADR,$TMPO,$GDDAT,$BDDAT,0
001726  001160  001124  001126
001734  000000
001736  001514  001116  001160          DT21:   $VERPC,$ERRPC,$TMPO,$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,$BDADR,$TMPO,$TMP1,0
002010  001160  001162  000000
002016  001514  001116  001160          DT26:   $VERPC,$ERRPC,$TMPO,$TMP1,0
002024  001162  000000
002030  001160  001162  001120          DT30:   $TMPO,$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
```

```
002066  107030                      107030                        ;ODD,ODD
002070  152525                      152525                        ;ODD,EVEN
002072  000000                      0                             ;EXTRA PATTERN HOLDER FOR
                                                                  ;FUTURE USE
002074  000000              MPEND:  0                             ;TABLE TERMINATOR


                            .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
                            ;//////////////////////////////////////////////////////////////////////
                            ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
                            ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
                            ;*
                            ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
                            ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
                            ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
                            ;*  THE DEVICE ADDRESS,  EACH BIT REPRESENTING A 4K BLOCK.  I.E.
                            ;*  FIRST WORD BIT0 = 0 - 4K,  BIT1 = 4 - 8K,...  BIT15 = 60 - 64K
                            ;*  SECOND WORD BIT0 = 64 - 68K,... BIT14 = 120 - 124K.
                            ;//////////////////////////////////////////////////////////////////////
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                            $ERRTB:
                                        ;*   ITEM 1
      002340   026754                           DM1             ;PARITY REGISTER DATA ERROR.
      002342   030333                           DH1             ;PC,REG,S/B,WAS
      002344   001646                           DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
      002346   030700                           DF1             ;16,18,16,16
                                        ;*   ITEM 2
      002350   027010                           DM2             ;ADDRESS TEST ERROR(TST1-5).
      002352   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002354   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002356   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 3
      002360   027010                           DM2             ;ADDRESS TEST ERROR(TST1-5).
      002362   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002364   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002366   030711                           DF3             ;16,18,18,8,8
                                        ;*   ITEM 4
      002370   027044                           DM4             ;CONSTANT DATA ERROR(TST6-10).
      002372   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002374   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002376   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 5
      002400   027102                           DM5             ;ROTATING BIT ERROR(TST11-12).
      002402   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002404   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002406   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 6
      002410   027140                           DM6             ;MOS REFRESH TEST ERROR (TST30-31).
      002412   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002414   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002416   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 7
      002420   027204                           DM7             ;3 XOR 9 PATTERN ERROR(TST13-16).
      002422   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002424   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
      002426   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 10
      002430   027245                           DM10            ;MARCHING 1'S AND 0'S ERROR(TST27).
      002432   030352                           DH2             ;V/PC,P/PC,MA,S/B,WAS
      002434   001660                           DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT.$BDDAT
      002436   030704                           DF2             ;16,18,18,16,16
                                        ;*   ITEM 11
```

```
    756  002440  027311                        DM11              ;PARITY MEMORY ADDRESS ERROR(TST17).
    757  002442  030352                        DH2               ;V/PC,P/PC,MA,S/B,WAS
    758  002444  001660                        DT2               ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
    759  002446  030711                        DF3               ;16,18,18,8,8
    760                              ;*  ITEM 12
    761  002450  027355                        DM12              ;DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
    762  002452  030377                        DH12              ;V/PC,P/PC,MA,S/B
    763  002454  001674                        DT12              ;$VERPC,$ERRPC,$GDADR,$GDDAT
    764  002456  030711                        DF3               ;16,18,18,8
    765                              ;*  ITEM 13
    766  002460  027431                        DM13              ;WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
    767  002462  030377                        DH12              ;V/PC,P/PC,MA,S/B
    768  002464  001674                        DT12              ;$VERPC,$ERRPC,$GDADR,$GDDAT
    769  002466  030711                        DF3               ;16,18,18,8
    770                              ;*  ITEM 14
    771  002470  027521                        DM14              ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
    772  002472  030420                        DH14              ;V/PC,P/PC,REG,MA
    773  002474  001706                        DT14              ;$VERPC,$ERRPC,$TMP0,$GDADR
    774  002476  030716                        DF14              ;16,18,18,18
    775                              ;*  ITEM 15
    776  002500  026754                        DM1               ;PARITY REGISTER DATA ERROR.
    777  002502  030441                        DH15              ;V/PC,P/PC,MAUT,REG,S/B,WAS
    778  002504  001720                        DT15              ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
    779  002506  030716                        DF14              ;16,18,18,18,16,16
    780                              ;*  ITEM 16
    781  002510  027620                        DM16              ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
    782  002512  030420                        DH14              ;V/PC,P/PC,REG,MA
    783  002514  001706                        DT14              ;$VERPC,$ERRPC,$TMP0,$GDADR
    784  002516  030716                        DF14              ;16,18,18,18
    785                              ;*  ITEM 17
    786  002520  027677                        DM17              ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
    787                                                          ; TRAPPED(TST21).
    788  002522  030352                        DH2               ;V/PC,P/PC,MA,S/B,WAS
    789  002524  001660                        DT2               ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
    790  002526  030711                        DF3               ;16,18,18,8,8
    791                              ;*  ITEM 20
    792  002530  027775                        DM20              ;RANDOM DATA ERROR(TST20).
    793  002532  030352                        DH2               ;V/PC,P/PC,MA,S/B,WAS
    794  002534  001660                        DT2               ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
    795  002536  030704                        DF2               ;16,18,18,16,16
    796                              ;*  ITEM 21
    797  002540  030027                        DM21              ;INSTRUCTION EXICUTION ERROR(TST21-26).
    798  002542  030474                        DH21              ;V/PC,P/PC,IUT,MA,S B,WAS
    799  002544  001736                        DT21              ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
    800  002546  030724                        DF21              ;16,18,16,18,16,16
    801                              ;*  ITEM 22                 ;NOT USED
    802                              ;*  ITEM 23
    803  002550  030076                        DM23              ;PROGRAM CODE CHANGED WHEN RELOCATED.
    804  002552  030525                        DH23              ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
    805  002554  001754                        DT23              ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT
    806  002556  030716                        DF14              ;16,18,18,18,16,16
    807                              ;*  ITEM 24
    808  002560  030143                        DM24              ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
    809  002562  030565                        DH24              ;V/PC,P/PC,TRP/PC
    810  002564  001772                        DT24              ;$VERPC,$ERRPC,$BDADR
    811  002566  030716                        DF14              ;16,18,18
```

```
                                        ;*   ITEM 25
002570  030217                                  DM25            ;TRAPPED TO 114.
002572  030606                                  DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
002574  002002                                  DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
002576  030716                                  DF14            ;16,18,18,18,16
                                        ;*   ITEM 26
002600  030237                                  DM26            ;FAILED TO TRAP.
002602  030637                                  DH26            ;V/PC,P/PC,REG,WAS
002604  002016                                  DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
002606  030704                                  DF2             ;16,18,18,16
                                        ;*   ITEM 27
002610  030257                                  DM27            ;(ACTION ENABLE WASN'T SET).
002612  030637                                  DH26            ;V/PC,P/PC,REG,WAS
002614  002016                                  DT26            ;$VERPC,$ERRPC,$TMP0,$BDDAT
002616  030704                                  DF2             ;16,18,18,16
                                        ;*   ITEM 30
002620  000000                                  0               ;NO MESSAGE.
002622  030661                                  DH30            ;REG,WAS,MA,WAS
002624  002030                                  DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
002626  030732                                  DF30            ;18,16,18,8
                                        ;*   ITEM 31
002630  030313                                  DM31            ;TRAPPED TO 4
002632  000000                                  0               ;NO HEADER
002634  002042                                  DT31            ;$TMP3
002636  030732                                  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     #SWR,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  011214  011204          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  014174  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
        003052  001434                          BEQ     68$             ;;BRANCH IF YES
        003054  004567  020376                  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
```

```
003070  126727  176130  000001           CMPB    $ENV,#1            ;;ARE WE RUNNING UNDER APT?
003076  001411                           BEQ     70$                ;;BRANCH IF YES
003100  026727  176034  000176           CMP     SWR,#SWPEG         ;;SOFTWARE SWITCH REG SELECTED?
003106  001010                           BNE     71$                ;;BRANCH IF NO
                                 ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
                                 ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
003110  013746  177776                   MOV     @#PSW,  -(SP)      ;PUT THE PROCESSOR STATUS ON THE STACK
003114  004767  017262                   JSR     PC,     $GTSWR     ;GO TO THE SUBROUTINE
003120  000403                           BR      71$
003122  112767  000001  176004   70$:    MOVB    #1,$AUTOB          ;;SET AUTO-MODE INDICATOR
003130                           71$:
003130  000405                           BR      68$                ;;GET OVER THE ASCIZ
                                 ;;69$:  .ASCIZ  <CRLF>'CZQMCEO'<CRLF>
003144                           68$:
003144  010700                           MOV     PC,     R0         ;GET CURRENT PROGRAM COUNTER.
003146  022700  003146                   CMP     #.,     R0         ;CHECK IF THE PROGRAM IS RELOCATED.
003152  001402                           BEQ     10$                ;BR IF PROGRAM NOT RELOCATED.
003154  000167  175120                   JMP     RESTAR             ;GO TRY TO RELOCTED BEFORE CONTINUING.
003160  012767  000003  175414   10$:    MOV     #3,     PRGMAP     ;INITIALIZE PROGRAM MAP....LO 64K.
003166  005067  175412                   CLR     PRGMAP+2           ;...HI 64K.
003172  005067  175402                   CLR     RELOCF             ;INIT THE RELOCATION FACTOR.
003176  004767  014112                   JSR     PC,SAVLDR          ;GO SAVE LOADERS

                                 ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
003202  005067  175400                   CLR     MMAVA              ;CLEAR MEM MGMT AVAILABLE FLAG
003206  032777  010000  175724           BIT     #SW12,  @SWR       ;CHECK FOR INHIBIT KT11 SWITCH
003214  001014                           BNE     IMPCK              ;BRANCH IF SET
003216  012737  003246  000004           MOV     #IMPCK,@#ERRVEC    ;SET UP TIMEOUT TRAP VECTOR
003224  005037  177572                   CLR     @#SR0              ;CLEAR MEM MGMT STATUS REG
003230  004767  011010                   JSR     PC,     MMINIT     ;MEM MGMT INITIALIZATION ROUTINE.
003234  005267  175346                   INC     MMAVA              ;SET MEM MGMT AVAILABLE FLAG
003240  004567  020212                   JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
003244  025320                           .WORD   MMAMES             ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"KT11 AVAILABLE"

                                 ;*      CHECK IF CACHE PRESENT, IF SO TURN IT OFF!!!
003246  012706  001100           IMPCK:  MOV     #STACK, SP
003252  005067  176340                   CLR     CASFLG             ;CLEAR CACHE PRESENT FLAG
003256  012737  003300  000004           MOV     #MAPMEM,@#ERRVEC
003264  052767  000014  174454           BIS     #14,    IMPCHE
003272  012767  000001  176316           MOV     #1,     CASFLG     ;SET CACHE PRESENT FLAG

                                 ;;*************************************************************
                                 ;*  ROUTINE TO MAP ALL OF MEMORY.
                                 ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
                                 ;*      R0 = MEMMAP POINTER...LO 64K.
                                 ;*      R1 = MEMMAP POINTER...HI 64K.
                                 ;*      R2 = ADDRESS POINTER
                                 ;*      R3 = BANK POINTER...LO 64K.
                                 ;*      R4 = BANK POINTER...HI 64K.
                                 ;*      R5 = SCRATCH REGISTER.
                                 ;;*************************************************************
003300  012706  001100           MAPMEM: MOV     #STACK, SP         ;RESET THE STACK
003304  012700  001524                   MOV     #MEMMAP,R0         ;SET UP MEMORY MAP POINTER...LO 64K.
003310  012701  001526                   MOV     #MEMMAP+2,R1       ;...HI 64K.
003314  005010                           CLR     (R0)               ;CLR MEMORY MAP...LO 64K.
```

```
003316  005011                         CLR     (R1)                ;...HI 64K.
003320  005002                         CLR     R2                  ;SET ADDRESS POINTER TO 0
003322  012703  000001                 MOV     #1,      R3         ;SETUP 4K BANK POINTER...LO 64K.
003326  005004                         CLR     R4                  ;...HI 64K.
003330  005067  175632                 CLR     $TMP3               ;INIT TEMPORARY HIGH ADDRESS BITS.
003334  004567  020116                 JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
003340  025365                         .WORD   MEMMES              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"MEMORY MAP:"
003342  012737  003456  000004         MOV     #2$, @#ERRVEC       ;SET UP TIMEOUT VECTOR
003350  011222                  1$:    MOV     (R2),    (R2)+      ;READ+WRITE ALL MEMORY
003352  032702  017777                 BIT     #MASK4K,R2          ;CHECK FOR 4K BOUNDRY
003356  001374                         BNE     1$                  ;BRANCH IF MORE IN BANK
003360  050310                         BIS     R3,      (R0)       ;SET FLAG FOR BANK...LO 64K.
003362  050411                         BIS     R4,      (R1)       ;...HI 64K.
003364  010267  175574                 MOV     R2,      $TMP2      ;SAVE ADDRESS POINTER.
003370  005367  175570                 DEC     $TMP2               ;ADJUST TO LAST ADR, LAST BANK.
003374  005767  175206                 TST     MMAVA               ;CHECK FOR MEM MGMT.
003400  001432                         BEQ     3$                  ;BR IF NO MEM MGMT.
003402  042767  160000  175554         BIC     #160000,$TMP2       ;CLEAR BANK BITS ON RELATIVE ADDRESS.
003410  013705  172344                 MOV     @#KIPAR2,R5         ;SAVE KIPAR2.
003414  005067  175546                 CLR     $TMP3               ;MAKE SURE HI BITS ARE INIT.
003420  006305                         ASL     R5                  ;SHIFT IT 6 PLACES.
003422  006305                         ASL     R5
003424  006305                         ASL     R5
003426  006305                         ASL     R5
003430  006305                         ASL     R5
003432  006167  175530                 R^.     $TMP3
003436  006305                         ASL     R5
00344C  006167  175522                 ROL     $TMP3
003444  060567  175514                 ADD     R5,      $TMP2      ;MAKE LAST ADR PHYSICAL.
003450  005567  175512                 ADC     $TMP3
003454  000404                         BR      3$                  ;GO TO UPDATE POINTERS.

                                ;*  TIMEOUT TRAPS TO HERE
003456  022626                  2$:    CMP     (SP)+,   (SP)+      ;RESTORE THE STACK POINTER
003460  052702  017777                 BIS     #MASK4K,R2          ;LAST ADDRESS OF 4K BANK
003464  005202                         INC     R2                  ;FIRST ADDRESS OF NEXT BANK.
003466  005767  175114          3$:    TST     MMAVA               ;CHECK FOR MEM MGMT
003472  001411                         BEQ     4$                  ;BRANCH IF NO MEM MGMT
003474  062737  000200  172344         ADD     #200, @#KIPAR2      ;UPDATE THIRD PAR
003502  012702  040000                 MOV     #40000, R2          ;POINT TO START OF THIRD PAR
003506  006303                         ASL     R3                  ;UPDATE LO BANK POINTER.
003510  006104                         ROL     R4                  ;UPDATE HI BANK POINTER
003512  100316                         BPL     1$                  ;BRANCH IF MORE MEMORY TO MAP.
003514  000402                         BR      5$                  ;EXIT WHEN DONE.

003516  106303                  4$:    ASLB    R3                  ;UPDATE MAP POINTER
003520  100313                         BPL     1$                  ;BRANCH IF NOT YET DONE
003522  012737  025060  000004  5$:    MOV     #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
003530  004767  014622                 JSR     PC,      TYPMAP     ;GO TYPE THE MAP.
003534  004567  017716                 JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
003540  001201                         .WORD   $CRLF               ;ADDRESS OF MESSAGE TO BE TYPED
003542  011067  175766                 MOV     (R0),    SAVTST     ;SET UP TEST MAP...LO 64K.
003546  011167  175764                 MOV     (R1),    SAVTST+2   ;...HI 64K.
003552  011000                         MOV     (R0),    R0         ;GET LOW MEM MAP
003554  042700  177760                 BIC     #177760, R0         ;MASK ALL BUT BOTTOM 4 BANKS
```

```
003560  020027  000017                    CMP     R0,         #17       ;CHECK THAT BOTTOM 16K IS ALL THERE!
003564  001530                            BEQ     GMPR                  ;BRANCH IF BOTTOM 16K EXISTS
003566  004567  017664                    JSR     R5,         $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
003572  025470                            .WORD   INSUFF                ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"FIRST 16K OF MEMORY NOT ALL THERE!"
003574  000000                    6$:     HALT                          ;FATAL ERROR HALT...
                                                                        ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
                                  ;;*************************************************************
                                  ;*  SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
                                  ;*    DATI, DATIP, DATO, AND DATOB.
                                  ;;*************************************************************
003576  012706  001100            TIMOUT: MOV     #STACK,     SP        ;SET UP THE STACK POINTER.
003602  005067  175000                    CLR     MMAVA                 ;CLEAR MEM MGMT AVAILABLE FLAG.
003606  032777  010000  175324            BIT     #SW12,      @SWR      ;CHECK IF MEM MGMT TO BE INHIBITED.
003614  001011                            BNE     1$                    ;BR IF NO MEM MGMT.
003616  012737  003640  000004            MOV     #1$,        @#ERRVEC  ;SET TIMEOUT FOR MEM MGMT CHECK.
003624  005037  177572                    CLR     @#SR0                 ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
003630  004767  010410                    JSR     PC,         MMINIT    ;INIT ALL MEM MGMT REGISTERS.
003634  005267  174746                    INC     MMAVA                 ;SET MEM MGMT AVAILABLE FLAG.
003640                            1$:
003640  004567  017612                    JSR     R5,         $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
003644  025403                            .WORD   BYTMES                ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"BYTE MEMORY MAP:"
003646  005000                            CLR     R0                    ;SET UP TYPE OUT FLAG.
003650  005002                            CLR     R2                    ;SET ADDRESS POINTER TO ZERO.
003652  012737  003716  000004            MOV     #20$,       @#ERRVEC  ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
003660  105712                    10$:    TSTB    (R2)                  ;DO DATI ONLY.
003662  032702  000001                    BIT     #BIT0,      R2        ;CHECK FOR WORD ADDRESS.
003666  001001                            BNE     11$                   ;BR IF ODD BYTE ADDRESS.
003670  011212                            MOV     (R2),       (R2)      ;DO DATI, DATO...NOP FOR READ ONLY MAP.
003672  151212                    11$:    BISB    (R2),       (R2)      ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
003674  005700                            TST     R0                    ;CHECK FOR PREVIOUS TYPOUT.
003676  001023                            BNE     30$                   ;BR IF ALREADY TYPED "FROM".
003700  004567  017552                    JSR     R5,         $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
003704  025453                            .WORD   FROM                  ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"FROM"
003706  010246                            MOV     R2,         -(SP)     ;PUT THE DATA ON THE STACK.
003710  004767  021202                    JSR     PC,         $TYPAD    ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
003714  000413                            BR      29$                   ;GO TO ADDRESS POINTER UPDATE.
                                  ;*  TIME OUTS COME HERE.
003716  022626                    20$:    CMP     (SP)+,      (SP)+     ;POP TWO OFF STACK.
003720  005700                            TST     R0                    ;CHECK FOR PREVIOUS TYPOUT.
003722  001411                            BEQ     30$                   ;BR IF ALREADY TYPED "TO".
003724  004567  017526                    JSR     R5,         $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
003730  025463                            .WORD   TO                    ;ADDRESS OF MESSAGE TO BE TYPED
                                                                        ;"TO"
003732  005302                            DEC     R2                    ;BACK UP ONE BYTE.
003734  010246                            MOV     R2,         -(SP)     ;PUT THE DATA ON THE STACK.
003736  004767  021154                    JSR     PC,         $TYPAD    ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
003742  005202                            INC     R2                    ;RESET ADDRESS POINTER.
003744  005100                    29$:    COM     R0                    ;RESET PREVIOUS TYPOUT FLAG.
003746  005202                    30$:    INC     R2                    ;UPDATE ADDRESS POINTER TO NEXT BYTE.
003750  001423                            BEQ     31$                   ;EXIT IF ZERO REACHED.
003752  032702  017777                    BIT     #MASK4K,R2            ;CHECK FOR 4K BANK BOUNDRY.
003756  001340                            BNE     10$                   ;BR IF MORE THIS 4K BANK.
003760  005767  174622                    TST     MMAVA                 ;CHECK IF MEM MGMT IS AVAILABLE.
```

```
003764  001735                           BEQ     10$                      ;BR IF NO MEM MGMT.
003766  022737  007600  172346           CMP     #7600,  @#KIPAR3         ;CHECK FOR END OF LAST 4K BANK.
003774  001411                           BEQ     31$                      ;EXIT WHEN ALL DONE.
003776  012702  060000                   MOV     #60000, R2               ;RESET VIRTUAL ADOPFSS POINTER.
004002  013737  172346  172344           MOV     @#KIPAR3,@#KIPAR2        ;SAVE MEM MGMT REG FOR TYPEOUT.
004010  062737  000200  172346           ADD     #200,   @#KIPAR3         ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
004016  000720                           BR      10$                      ;BR BACK TO DO NEXT BANK.
004020  005700                   31$:    TST     R0                       ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
004022  001407                           BEQ     32$                      ;BR TO EXIT IF TYPING ALL DONE.
004024  004567  017426                   JSR     R5,     $PRINT           ;GO PRINT OUT THE FOLLOWING MESSAGE.
004030  025463                           .WORD   TO                       ;ADDRESS OF MESSAGE TO BE TYPED
                                                                          ;"TO"
004032  005302                           DEC     R2                       ;BACK ADDRESS POINTER UP ONE BYTE.
004034  010246                           MOV     R2,     -(SP)            ;PUT THE DATA ON THE STACK.
004036  004767  021054                   JSR     PC,     $TYPAD           ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
004042  000000                   32$:    HALT            ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
                                                         ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
004044  000654                           BR      TIMOUT                   ;LOOP BACK AND DO AGAIN UPON CONTINUE.

                                 .SBTTL  MAP PARITY REGISTERS
                                 ;;*****************************************************************
                                 ;*  SEARCH FOR PARITY REGISTERS PRESENT AND TYPE ADDRESSES OF THOSE FOUND
                                 ;*  THAT ARE FUNCTIONAL AND HAVE CORRESPONDING PARITY MEMORY
                                 ;;*****************************************************************

004046  012704  002276           GMPR:   MOV     #MPRX,  R4               ;SET UP POINTER TO PARITY REG EXIST TABLE.
004052  032777  000100  175060           BIT     #SW06,  @SWR             ;CHECK FOR INHIBIT PARITY SWITCH.
004060  001036                           BNE     GMPRD                    ;BR IF INHIBIT PARITY.
004062  012703  002076                   MOV     #MPRO,  R3               ;SET UP TABLE POINTER
004066  012737  004110  000004           MOV     #GMPRB, @#ERRVEC         ;SET UP TIMEOUT TRAP SERVICE
004074  042713  000001           GMPRA:  BIC     #1,     (R3)             ;CLEAR FLAG BIT IN TABLF
004100  005773  000000                   TST     @(R3)                    ;DOES THIS MEMORY PARITY REGISTER EXIST.
                                 ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO "GMPRB".
004104  012324                           MOV     (R3)+,  (R4)+            ;SAVE IT IN THE PARITY REG EXIST TABLE.
004106  000403                           BR      GMPRC                    ;SKIP TIMEOUT SERVICE CODE
                                 ;*  TIMEOUT COMES HERE
004110  022626                   GMPRB:  CMP     (SP)+,  (SP)+            ;RESTORE STACK POINTER
004112  052723  000001                   BIS     #1,     (R3)+            ;SET FLAG TO INDICATE REGISTER NOT PRESENT
004116  005023                   GMPRC:  CLR     (R3)+                    ;CLEAR THE MAP...LO 64K.
004120  005023                           CLR     (R3)+                    ;...HI 64K.
004122  005023                           CLR     (R3)+                    ;...AND THE MASK.
004124  020327  002276                   CMP     R3,     #MPRX            ;HAVE WE CHECKED ALL REGISTERS?
004130  103761                           BLO     GMPRA                    ;NO - GO BACK TO CHECK NEXT ONE
004132  005014                           CLR     (R4)                     ;SET TERMINATOR IN PARITY REG EXIST TABLE.
004134  012737  025060  000004           MOV     #ERRTRP,@#ERRVEC         ; RESTORE TRAPCATCHER
004142  005767  176130                   TST     MPRX                             ;ANY PARITY REGISTERS PRESENT?
004146  001006                           BNE     MPAMEM                           ;YES - GO TEST CONTROLS PRESENT
004150  004567  017302                   JSR     R5,     $PRINT           ;GO PRINT OUT THE FOLLOWING MESSAGE.
004154  025551                           .WORD   MTR                      ;ADDRESS OF MESSAGE TO BE TYPED
                                                                          ;"NO MEMORY PARITY REGISTERS FOUND"
004156  005014                   GMPRD:  CLR     (R4)                     ;MAKE SURE TABLE IS CLEAR.
004160  000167  001156                   JMP     MANUAL                           ;AND SKIP ALL CONTROLS TESTING
```

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

004164  004767  014044                            MPAMEM: JSR     PC,       CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
004170  012767  000001  175346                            MOV     #1,       BITPT   ;INITIALIZE 4K POINTER
004176  005067  175344                                    CLR     BITPT+2           ;CLEAR HI 64K POINTER
004202  012702  014000                                    MOV     #14000, R2        ;SET ADR POINTER TO 14000.
004206  005767  174374                                    TST     MMAVA             ;CHECK FOR MEM MGMT
004212  001404                                            BEQ     MAPRB             ;BRANCH IF NO MEM MGMT
004214  012702  054000                                    MOV     #54000, R2        ;SET ADR POINTER TO PAR2
004220  004767  010020                                    JSR     PC,       MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.

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

004224  005067  175310                            MAPRB:  CLR     PMEMAP            ;CLEAR THE PARITY MEMORY MAP
004230  005067  175306                                    CLR     PMEMAP+2
004234  012703  002076                            1$:     MOV     #MPR0,  R3        ;INITIALIZE TABLE ADDRESS
004240  032713  000001                            2$:     BIT     #1,     (R3)      ;IS THIS REGISTER PRESENT?
004244  001052                                            BNE     3$                ;NO - GET THE NEXT ONE
004246  013773  001612  000000                            MOV     @#WWP,  @(R3)     ;YES - SET WRITE WRONG PARITY
                                                                                        ;AND CLEAR REST OF REGISTER
004254  011212                                            MOV     (R2),   (R2)      ;WRITE WRONG PARITY
004256  005712                                            TST     (R2)              ;READ WRONG PARITY
004260  043773  001612  000000                            BIC     @#WWP,  @(R3)     ;CLEAR WRITE WRONG PARITY
004266  005773  000000                                    TST     @(R3)             ;OTHERWISE, CHECK TO SEE IF THIS
                                                                                        ;CONTROL REGISTER GOT A PARITY
                                                                                        ;ERROR
004272  100014                                            BPL     6$                ;BRANCH IF IT DIDN'T AND CHECK
004274  032773  007740  000000                            BIT     #7740,  @(R3)     ;IS IT A CORE PAR. REG.
004302  001404                                            BEQ     5$                ;BRANCH IF NOT.
004304  012763  070032  000006                            MOV     #70032, 6(R3)     ;IF IT IS SET UP MASK
004312  000413                                            BR      7$                ;AND BRANCH TO SET BITS.
004314  012763  077772  000006            5$:     MOV     #77772, 6(R3)     ;IF MOS SET UP MASK
004322  000407                                            BR      7$                ;AND BRANCH TO SET BIT.
004324  032773  007740  000000            6$:     BIT     #7740,  @(R3)     ;IF ANY BITS ARE SET
004332  001417                                            BEQ     3$                ;THEN CSR IS MS11-K.
004334  012763  070000  000006                            MOV     #70000, 6(R3)     ;IF MS11-K SET MASK.
004342  056763  175176  000002            7$:     BIS     BITPT,  2(R3)     ;SET FLAG IN MAP FOR THIS PARITY REGISTER
004350  056763  175172  000004                            BIS     BITPT+2,4(R3)
004356  056767  175162  175154                            BIS     BITPT,  PMEMAP    ;SET FLAG IN PARITY MAP
004364  056767  175156  175150                            BIS     BITPT+2,PMEMAP+2
004372  062703  000010                    3$:     ADD     #10,    R3        ;STEP UP TO NEXT REGISTER
004376  020327  002276                                    CMP     R3,     #MPRX     ;ARE WE DONE WITH TABLE?
```

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

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

004536  004767  013472          TMAP:   JSR     PC,     CLRPAR     ;INITIALIZE ALL PARITY REGISTERS PRESENT
004542  004567  016710                  JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
004546  025426                          .WORD   MTMAP              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"PARITY MEMORY MAP:"
004550  012703  002076                  MOV     #MPR0,  R3         ;INITIALIZE TABLE POINTER
004554  032713  000001          1$:     BIT     #BIT0,  (R3)       ;CHECK IF THIS REGISTER IS PRESENT.
004560  001046                          BNE     2$                 ;BR IF NOT PRESENT.
004562  022763  070032  000006          CMP     #70032, 6(R3)
004570  001004                          BNE     3$
004572  004567  016660                  JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
004576  026067                          .WORD   MX3                ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"CORE PARITY"
004600  000417                          BR      5$
004602  022763  077772  000006  3$:     CMP     #77772, 6(R3)
004610  001004                          BNE     4$
004612  004567  016640                  JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
004616  026106                          .WORD   MX4                ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"MOS PARITY"
004620  000407                          BR      5$
004622  022763  070000  000006  4$:     CMP     #70000, 6(R3)
004630  001003                          BNE     5$
004632  004567  016620                  JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
004636  026124                          .WORD   MX5                ;ADDRESS OF MESSAGE TO BE TYPED
                                                                   ;"MS11-K CSR"
004640                          5$:
```

```
004640  004567  016612                  JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
004644  026035                          .WORD   MX1              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                 ;"REGISTER AT"
004646  011346                          MOV     (R3),-(SP)       ;;SAVE (R3) FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
004650  013746  177776                  MOV     @#PSW,  -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
004654  004767  017774                  JSR     PC,     $TYPOC   ;GO TO THE SUBROUTINE
004660  004567  016572                  JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
004664  026054                          .WORD   MX2              ;ADDRESS OF MESSAGE TO BE TYPED
                                                                 ;"CONTROLS"
004666  010300                          MOV     R3,     R0       ;SET UP R0 FOR TYPMAP ROUTINE.
004670  005720                          TST     (R0)+            ;UPDATE POINTER TO MAP.
004672  004767  013460                  JSR     PC,     TYPMAP   ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
004676  062703  000010          2$:     ADD     #10,    R3       ;UPDATE TO NEXT REGISTER IN TABLE.
004702  020327  002276                  CMP     R3,     #MPRX    ;ARE WE ALL DONE WITH TABLE?
004706  103722                          BLO     1$               ;BRANCH IF MORE REGISTERS
004710  004567  016542                  JSR     R5,     $PRINT   ;THE REASON I'M OUTPUTTING THIS CRLF
004714  001201                          $CRLF                    ;IS TO GIVE THE PRINTER ENOUGH TIME TO
                                                                 ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
004716  022737  070000  002264          CMP     #70000,@#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
004724  001006                          BNE     7$               ;IF NO BRANCH
004726  043727  002260  001540          BIC     @#MPR14+2,#PMEMAP ;IF YES THEN CLEAR THE BITS IN
004734  043737  002262  001540          BIC     @#MPR14+4,@#PMEMAP ;THE PARITY MEMORY MAP.
004742  022737  070000  002274  7$:     CMP     #70000, @#MPR15+6        ;DO WE HAVE A MS11-K
004750  001031                          BNE     9$               ;IF NO GO TO TESTS NOW.
004752  043737  002270  001540          BIC     @#MPR15+2,@#PMEMAP       ;IF YES I AM GOING TO
004760  043737  002272  001542          BIC     @#MPR15+4,@#PMEMAP+2     ;CLEAR THE PARITY INDICATORS
004766  012705  002276                  MOV     #MPRX,  R5       ;FOR THAT PORTION OF MEMORY.
004772  021537  002256          6$:     CMP     (R5),@#MPR14     ;SEARCH FOR THIS MS11-K  CSR IN
004776  001004                          BNE     8$               ;AND IF ITS THERE DELETE IT
005000  005015                          CLR     (R5)
005002  052737  000001  002256          BIS     #1,@#MPR14
005010  022537  002266          8$:     CMP     (R5)+,  @#MPR15  ;SEARCH FOR MS11-K CSR IN
005014  001366                          BNE     6$               ;THE AVAILABILITY TABLE,
005016  005045                          CLR     -(R5)            ;AND CLEAR ITS ADDRESS FROM THE TABLE
005020  052737  000001  002266          BIS     #1,     @#MPR15  ;SET BIT0 IN ADDRESS IN CSR TABLE
005026  004567  016424                  JSR     R5,     $PRINT   ;OUTPUT MESSAGE TO RUN MS11-K TEST.
005032  026142                          .WORD   MX6
005034  005737  002276          9$:     TST     @#MPRX           ;ARE THERE ANY PARITY REGISTERS TO TEST?
005040  001002                          BNE     CTRLS            ;IF SO TEST THE BITS IN THE REGISTERS,
005042  000167  000274                  JMP     MANUAL           ;IF NO JUMP OVER REGISTER TESTS.

                                .SBTTL  TEST PARITY REGISTERS
                                ;;*************************************************************
                                ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
                                ;*  CAN BE SET AND CLEARED.
                                ;*  THIS IS A ONCE ONLY TEST.
                                ;;*************************************************************

005046  012703  002076          CTRLS:  MOV     #MPR0,  R3       ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
005052  011302                  1$:     MOV     (R3),   R2       ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
005054  062703  000010                  ADD     #10,    R3       ;UPDATE POINTER TO NEXT PAR. REG. ADD.
005060  032702  000001                  BIT     #1,     R2       ;IS THIS REGISTER BEING USED?
005064  001372                          BNE     1$               ;GO TO NEXT IF NOT
005066  020327  002276                  CMP     R3,     #MPRX    ;ARE WE AT END OF TABLE
```

```
005072  003055                                  BGT     RESCHK              ;GO TO NEXT TEST IF YES
005074  005763  177776                          TST     -?(R3)              ;TEST MHSK FOR PARITY REGISTER
005100  001764                                  BEQ     1$                  ;IF = 0,THEN DO NOT TEST
005102  016367  177776  174406                  MOV     -2(R3),  RESRVD     ;GET MASK FOR REGISTER WE ARE WORKING ON
005110  012700  000001                          MCV     #1,      R0         ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
005114  005012                                  CLR     (P2)                ;INITIALIZE THE PARITY REGISTER
005116  011201                                  MOV     (R2),    R1         ;READ THE CONTENTS OF THE PARITY REGISTER
005120  046701  174372                          BIC     RESRVD,  R1         ;CLEAR BITS WHICH ARE RESERVED
005124  001405                                  BEQ     2$                  ;CHECK OTHER BITS - BRANCH IF OK
005126  004767  013124                  64$:    JSR     PC,      SPRNT      ;SET UP VALUES FOR ERROR PRINTING.
005132  004767  014446                          JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
005136  000001                                  .WORD   1                   ;ERROR TYPE CODE.
005140  030067  174352                  2$:     BIT     R0,      RESRVD     ;IS THIS BIT RESERVED?
005144  001025                                  BNE     3$                  ;YES - DON'T TEST IT
005146  010012                                  MOV     R0,      (R2)       ;NO - SET THIS BIT IN THE PARITY REGISTER
005150  011201                                  MOV     (R2),    R1         ;READ & SAVE CONTENTS OF THE  PARITY REGISTER
005152  005012                                  CLR     (R2)                ;CLEAR THE PARITY REGISTER
005154  046701  174336                          BIC     RESRVD,  R1         ;CLEAR BIT LOCATIONS THAT ARE RESERVED
005160  020001                                  CMP     R0,      R1         ;COMPARE THE CHECK WC D WITH THE DATA READ.
005162  001405                                  BEQ     66$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
005164  004767  013116                  65$:    JSR     PC,      SPRNTO     ;SET UP VALUES FOR ERROR PRINTING.
005170  004767  014410                          JSP     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
005174  000001                                  .WORD   1                   ;ERROR TYPE CODE.
005176                                  66$:
005176  011201                                  MOV     (R2),    R1         ;READ THE CONTENTS OF THE PARITY REGISTER
005200  046701  174312                          BIC     RESRVD,  R1         ;CLEAR BITS WHICH ARE RESERVED
005204  001405                                  BEQ     3$                  ;CHECK OTHER BITS - BRANCH IF OK
005206  004767  013044                  67$:    JSR     PC,      SPRNT      ;SET UP VALUES FOR ERROR PRINTING.
005212  004767  014366                          JSR     PC,      $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
005216  000001                                  .WORD   1                   ;ERROR TYPE CODE.
005220  006300                          3$:     ASL     R0                  ;ROTATE TO GET NEXT BIT TO BE TESTED
005222  103346                                  BCC     2$                  ;BRANCH IF NOT DONE WITH ALL BITS
005224  000712                                  BR      1$                  ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.

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

005226  012704  002076                  RESCHK: MOV     #MPR0,   R4         ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
005232  010403                          1$:     MOV     R4,      R3
005234  062704  000010                          ADD     #10,     R4
005240  032713  000001                  BIT     #1,     (R3)     ;IS THIS REGISTER BEING USED
005244  001372                                  BNE     1$                  ;BRANCH IF NO
005246  012773  177777  000000                  MOV     #-1,     @(R3)      ;SET ALL BITS TO A 1
005254  022704  002276                          CMP     #MPRX,R4            ;ARE WE AT THE END OF THE TABLE
005260  002764                                  BLT     1$                  ;IF YES THEN WE ARE READY TO TEST
005262  000005                                  RESET                       ;RESET THE WORLD
005264  012703  002076                          MOV     #MPR0,   R3         ;LOAD INITIAL ADDRESS FOR POINTER
005270  011302                          2$:     MOV     (R3),    R2         ;STORE PARITY REGISTER ADDRESS
005272  062703  000010                          ADD     #10,     R3
005276  032702  000001                          BIT     #1,      R2
005302  001372                                  BNE     2$
005304  022703  002276                          CMP     #MPRX,   R3
005310  002014                                  BGE     MANUAL
005312  011201                                  MOV     (R2),    R1         ;GET CONTENTS OF REGISTER
```

```
 005314  005012                           CLR     (R2)
 005316  042701  077772                   BIC     #77772, R1      ;CLEAR BITS NOT EFFECTED  BY RESET
 005322  005701                           TST     P1              ;CHECK IF REST  WERE CLEARED BY  RESET
 005324  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 005326  004767  012724          64$:     JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
 005332  004767  014246                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 005336  000001                           .WORD   1               ;ERROR TYPE CODE.
 005340                          65$:
 005340  000753                           BR      2$              ;BRANCH BACK TO CHECK NEXT REGISTER


 005342  012700  000014          MANUAL:  MOV     #12.,   R0      ;SET COUNTER TO CLEAR 12 WORDS.
 005346  012701  001562                   MOV     #FSTADR,R1      ;STARTING AT FSTADR.
 005352  005021                  1$:      CLR     (R1)+           ;CLEAR THE LOCATIONS.
 005354  005300                           DEC     R0              ;COUNT.
 005356  001375                           BNE     1$              ;BR IF MORE.
 005360  105767  174172                   TSTB    SELFLG          ;CHECK FOR SELECT PARAMETERS STARTUP.
 005364  001005                           BNE     MANUL1          ;BR IF PARAMETERS TO BE SELECTED.
 005366  016767  173572  174200           MOV     $TMP2,  LSTADR  ;SET UP VIRTUAL LAST ADDRESS.
 005374  000167  000402                   JMP     MANUL2          ;SKIP PARAMETER SELECTION SECTION.
```

```
                                   .SBTTL  USER PARAMETER SELECTION SECTION
                                   ;;*********************************************************
                                   ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                   ;;*********************************************************
005400  012700  000001             MANUL1: MOV     #BIT0,  R0       ;SET UP BANK POINTER.
005404  005001                             CLR     R1               ;...HI 64K.
005406  005002                             CLR     R2               ;CLEAR ADDRESS POINTER.
005410  005003                             CLR     R3               ;...HI ADDRESS BITS.
005412  004567  016040                     JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
005416  026255                             .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**.
005420  013746  177776                     MOV     @#PSW,  -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
005424  004767  015654                     JSR     PC,     $RDOCT   ;GO TO THE SUBROUTINE
005430  042716  000001                     BIC     #BIT0,  (SP)     ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
005434  005067  174074                     CLR     SAVTST           ;INIT TEST MAP...LO 64K.
005440  005067  174072                     CLR     SAVTST+2         ;...HI 64K.
005444  062702  020000             1$:     ADD     #20000, R2       ;UPDATE ADDRESS POINTER TO NEXT BANK.
005450  005503                             ADC     R3
005452  020367  015776                     CMP     R3,     $HIOCT   ;CHECK HI ADDRESS BITS.
005456  103403                             BLO     2$               ;BR IF NOT HI ENOUGH YET.
005460  101006                             BHI     3$               ;BR IF PAST SELECTED ADDRESS.
005462  020216                             CMP     R2,     (SP)     ;CHECK THE LO ADDRESS BITS.
005464  101004                             BHI     3$               ;BR IF PAST SELECTED ADDRESS.
005466  006300                     2$:     ASL     R0               ;UPDATE POINTER...LO 64K.
005470  006101                             ROL     R1               ;...HI 64K.
005472  100364                             BPL     1$               ;BR BACK TO CHECK NEXT BANK.
005474  000507                             BR      17$              ;BR IF OVERFLOW.
005476  030067  174022             3$:     BIT     R0,     MEMMAP   ;CHECK IF BANK EXISTS.
005502  001003                             BNE     4$               ;BR IF BANK EXISTS.
005504  030167  174016                     BIT     R1,     MEMMAP+2 ;CHECK HI 64K.
005510  001501                             BEQ     17$              ;BR IF ADDRESS IN UN-MAPPED BANK.
005512  016704  015736             4$:     MOV     $HIOCT, R4       ;SAVE FIRST ADR HI BITS.
005516                             10$:
005516  004567  015734                     JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
005522  026342                             .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**.
005524  013746  177776                     MOV     @#PSW,  -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
005530  004767  015550                     JSR     PC,     $RDOCT   ;GO TO THE SUBROUTINE
005534  005716                             TST     (SP)             ;CHECK IF ADR 0 SELECTED (DEFAULT).
005536  001010                             BNE     11$              ;BR IF NOT 0 (DEFAULT)
005540  005767  015710                     TST     $HIOCT           ;CHECK HI BITS.
005544  001005                             BNE     11$              ;BR IF NOT 0 (DEFAULT).
005546  016716  173412                     MOV     $TMP2,  (SP)     ;SET UP DEFAULT LAST ADR.
005552  016767  173410  015674             MOV     $TMP3,  $HIOCT
005560  012667  174010             11$:    MOV     (SP)+,  LSTADR   ;GET THE DATA.
005564  020467  015664                     CMP     R4,     $HIOCT   ;CHECK FOR LAST ADR BELOW FIRST ADR.
005570  101352                             BHI     10$              ;BB IF LAST BELOW FIRST.
005572  103403                             BLO     12$              ;BR IF LAST ABOVE FIRST.
005574  021667  173774                     CMP     (SP),   LSTADR   ;CHECK FOR LAST BELOW FIRST.
005600  101346                             BHI     10$              ;BR IF LAST BELOW FIRST.
005602  032716  017777             12$:    BIT     #MASK4K,(SP)     ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005606  001404                             BEQ     13$              ;BR IF ON BOUNDRY.
```

```
005610  010067  173754                    MOV     R0,      FADMAP   ;SET UP FIRST ADDRESS MAP.
005614  010167  173752                    MOV     R1,      FADMAP+2
005620  050067  173710          13$:      BIS     R0,      SAVTST   ;SET FLAG IN TEST MAP...LO 64K.
005624  050167  173706                    BIS     R1,      SAVTST+2 ;...HI 64K.
005630  020367  015620          14$:      CMP     R3,      $HIOCT   ;CHECK FOR PAST LAST ADR.
005634  103404                            BLO     15$               ;BR IF BELOW LAST ADR.
005636  101020                            BHI     16$               ;BR IF GONE PAST LAST ADR.
005640  020267  173730                    CMP     R2,      LSTADR   ;CHECK FOR PAST LAST ADR.
005644  101015                            BHI     16$               ;BR IF GONE PAST LAST ADR.
005646  062702  020000          15$:      ADD     #20000,  R2       ;UPDATE ADDRESS POINTER.
005652  005503                            ADC     R3                ;...HI BITS.
005654  006300                            ASL     R0                ;UPDATE BANK POINTER...LO 64K.
005656  006101                            ROL     R1                ;...HI 64K.
005660  100415                            BMI     17$               ;BR IF OVERFLOW.
005662  030067  173636                    BIT     R0,      MEMMAP   ;CHECK IF THIS BANK EXISTS.
005666  001354                            BNE     13$               ;BR IF BANK EXISTS.
005670  030167  173632                    BIT     R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
005674  001351                            BNE     13$               ;BR IF BANK EXISTS.
005676  000754                            BR      14$               ;BR IF BANK DOESN'T EXIST.
005700  030067  173620          16$:      BIT     R0,      MEMMAP   ;CHECK IF THIS BANK EXISTS.
005704  001010                            BNE     20$               ;BR IF IT EXISTS.
005706  030167  173614                    BIT     R1,      MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
005712  001005                            BNE     20$               ;BR IF IT EXISTS.
005714  005726                  17$:      TST     (SP)+             ;ADJUST THE STACK.
005716  004567  015534                    JSR     R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
005722  026365                            .WORD   BADADR            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"?ADDRESS IN UNMAPPED BANK?"
005724  000606                            BR      MANUAL            ;LOOP BACK TO THE BEGINNING.
005726  010067  173650          20$:      MOV     R0,      LADMAP   ;SET UP MAP FOR LAST ADDRESS.
005732  010167  173646                    MOV     R1,      LADMAP+2
005736  005767  172644          21$:      TST     MMAVA             ;CHECK FOR MEMORY MANAGEMENT.
005742  001404                            BEQ     22$               ;BR IF NO MEM MGMT.
005744  042716  160000                    BIC     #160000,(SP)      ;ADJUST FSTADR TO VITRUAL BANK 0.
005750  062716  040000                    ADD     #40000, (SP)      ;...TO VIRTUAL BANK 2.
005754  012667  173602          22$:      MOV     (SP)+,   FSTADR   ;SAVE FISRT ADDRESS OFF THE STACK.
005760                          30$:
005760  004567  015472                    JSR     R5,      $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
005764  026422                            .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**.
005766  013746  177776                    MOV     @#PSW,   -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
005772  004767  015306                    JSR     PC,      $RDOCT   ;GO TO THE SUBROUTINE
005776  012667  173606                    MOV     (SP)+,   .CONST   ;SAVE THE CONSTANT
006002  005767  172600          MANUL2:   TST     MMAVA             ;CHECK IF MEM MGMT IS AVAILABLE.
006006  001406                            BEQ     31$               ;BR IF NO MEM MGMT.
006010  042767  160000  173556            BIC     #160000,LSTADR    ;ADJUST LSTADR TO VIRTUAL BANK 0.
006016  062767  040000  173550            ADD     #40000,  LSTADR   ;...VIRTUAL BANK 2.
006024  062767  000002  173542  31$:      ADD     #2,      LSTADR   ;ADJUST LAST ADDRESS UP ONE WORD.
006032  042767  000001  173534            BIC     #BIT0,   LSTADR   ;MAKE SURE IT IS A WORD ADDRESS.
006040  032767  017777  173526            BIT     #MASK4K,LSTADR    ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
006046  001004                            BNE     START1            ;BR IF NOT ON BOUNDRY.
006050  005067  173526                    CLR     LADMAP            ;CLEAR OUT THE LAST ADDRESS MAP.
006054  005067  173524                    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
                                        ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

 1475  006060  016706  173536          START1: MOV     .STACK, SP      ;SET STACK POINTER
 1476  006064  005767  173526                  TST     CASFLG          ;CHECK CACHE PRESENT FLAG
 1477  006070  001403                          BEQ     1$              ;BRANCH IF NO CACHE
 1478  006072  052777  000014  173520          BIS     #14,    @CASREG ;TURN OFF CACHE
 1479  006100  012767  006060  173000  1$:     MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
 1480  006106  066767  172466  172772          ADD     RELOCF, $LPADR
 1481  006114  004767  011362                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
 1482  006120  005767  172462                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
 1483  006124  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
 1484  006126  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
 1485  006134  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
 1486  006136  004767  006102                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
```

```
                                    .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
                                    ;;**************************************************************
                                    ;*TEST 1         WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
                                    ;*       R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
                                    ;*       R1 = DATA READ FROM MEMORY (WAS)
                                    ;*       R2 = VIRTUAL ADDRESS
                                    ;*       R3 = NOT USED
                                    ;*       R4 = NOT USED
                                    ;*       R5 = BLOCK BOUNDRY BIT MASK.
                                    ;;**************************************************************
006142                              TST1:
006142  004567  012426                      JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006146  000001                              .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
                                                                    ; REQUIRED FOR THIS TEST.
006150  000167  005604                      JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                    ; AVAILABLE FOR TEST.
                                    ;*  UPWARDS WORD ADDRESSING.
006154  004467  006212                      JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006160  004767  007634              1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006164  010012                      2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
006166  012201                              MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
006170  020001                              CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006172  001405                              BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006174  004767  012132              64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
006200  004767  013400                      JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006204  000002                              .WORD   2               ;ERROR TYPE CODE.
006206                              65$:
006206  062700  000002                      ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
006212  030502                              BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006214  001363                              BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006216  004767  006726                      JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                    ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                    ;*  DOWNWARDS WORD ADDRESSING.
006222  004467  006602                      JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006226  004767  007566              3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006232  162700  000002              4$:     SUB     #2,     R0      ;DEC DATA BY 2
006236  014201                              MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
006240  020001                              CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
006242  001405                              BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006244  004767  012036              66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
006250  004767  013330                      JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
006254  000002                              .WORD   2               ;ERROR TYPE CODE.
006256                              67$:
006256  030502                              BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006260  001364                              BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
006262  004767  007352                      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.
                                ;;*******************************************************
006266                          TST2:
006266  004567  012302                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006272  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                ;*  UPWARDS BYTE ADDRESSING.
006274  004467  006072                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006300  004767  007514          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006304  110022                  2$:     MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
006306  005200                          INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
006310  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
006312  001374                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
006314  004767  006630                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
                                ;*  DOWNWARDS BYTE ADDRESSING.
006320  004467  006504                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006324  004767  007470          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006330  005300                  4$:     DEC     R0              ;DEC DATA BY 1
006332  114201                          MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
006334  120001                          CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
006336  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
006340  004767  011742          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
006344  004767  013234                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
    50  000003                          .WORD   3               ;ERROR TYPE CODE.
    52                          65$:
     2  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
     4  001365                          BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
     6  004767  007256                  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.
                                ;;*******************************************************
006362                          TST3:
006362  004567  012206                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
006366  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                ;*  DOWNWARDS WORD ADDRESSING.
006370  004467  006434                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006374  004767  007420          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
006400  005100                          COM     R0              ;COMPLEMENT THE ADR
006402  062700  000002          2$:     ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
006406  010042                          MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
006410  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```

```
006412  001373                           BNE     2$                  ;BRANCH IF MORE IN CURRENT BLOCK.
006414  004767  007220                   JSR     PC,       MMDOWN    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                 ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
                                 ;*  UPWARDS WORD ADDRESSING.
006420  004467  005746                   JSR     R4,       INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006424  004767  007370           3$:     JSR     PC,       PHYADR    ;GET PHYSICAL ADDRESS INTO R0
006430  005100                           COM     R0                  ;COMPLEMENT IT
006432                           4$:
006432  012201                           MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
006434  020001                           CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
006436  001405                           BEQ     65$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
006440  004767  011665           64$:    JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
006444  004767  013134                   JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
006450  000002                           .WORD   2                   ;ERROR TYPE CODE.
006452                           65$:
006452  162700  000002                   SUB     #2,       R0        ;COUNT DOWN WITH ADDRESS
006456  030502                           BIT     R5,       R2        ;CHECK FOR END OF A BLOCK.
006460  001364                           BNE     4$                  ;BRANCH IF MORE IN CURRENT BLOCK.
006462  004767  006462                   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.
                                 ;;*********************************************************
006466                           TST4:
006466  004567  012102                   JSR     R5,       $SCOPE    ;GO TO SCOPE ROUTINE.
006472  000000                           .WORD   0                   ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                 ;*  UPWARDS BYTE ADDRESSING.
006474  004467  005672                   JSR     R4,       INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006500  004767  007370           1$:     JSR     PC,       BANKNO    ;GET THE BANK NUMBER INTO R0
006504  110022                   2$:     MOVB    R0,       (R2)+     ;WRITE BANK # INTO ALL ADDRESSES
006506  030502                           BIT     R5,       R2        ;CHECK FOR END OF A BLOCK.
006510  001375                           BNE     2$                  ;BRANCH I, MORE IN CURRENT BLOCK.
006512  004767  006432                   JSR     PC,       MMUP      ;FIND NEXT BLOCK AND LOOP TO 1$.

                                 ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
                                 ;*  UPWARDS BYTE ADDRESSING.
006516  004467  005650                   JSR     R4,       INITMM    ;INITIALIZE THE MEMORY ADDRESS POINTERS.
006522  004767  007346           3$:     JSR     PC,       BANKNO    ;GET THE BANK NUMBER INTO R0
006526  112201                   4$:     MOVB    (R2)+,    R1        ;READ THE DATA OUT OF MEMORY
006530  020001                           CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
006532  001405                           BEQ     65$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
006534  004767  011554           64$:    JSR     PC,       SPRNT1    ;SET UP VALUES FOR ERROR PRINTING.
006540  004767  013040                   JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
006544  000003                           .WORD   3                   ;ERROR TYPE CODE.
006546                           65$:
006546  030502                           BIT     R5,       R2        ;CHECK FOR END OF A BLOCK.
006550  001366                           BNE     4$                  ;BRANCH IF MORE IN CURRENT BLOCK.
006552  004767  006372                   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.
                                ;;*************************************************************
 006556                         TST5:
 006556  004567  012012                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 006562  000000                         .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
                                ;*  DOWNWARDS BYTE ADDRESSING.
 006564  004467  006240                 JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 006570  004767  007300         1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
 006574  005100                         COM     R0              ;1'S COMPLEMENT OF BANK #
 006576  110042                 2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
 006600  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 006602  001375                         BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
 006604  004767  007030                 JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;*  CHECK THAT DATA WRITTEN CAN BE READ.
                                ;*  DOWNWARDS BYTE ADDRESSING.
 006610  004467  006214                 JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 006614  004767  007254         3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
 006620  005100                         COM     R0              ;SET 1'S COMPLEMENT OF BANK #
 006622  114201                 4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
 006624  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 006626  001405                         BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 006630  004767  011452         64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
 006634  004767  012744                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 0C6640  000003                         .WORD   3               ;ERROR TYPE CODE.
 006642                         65$:
 006642  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 006644  001366                         BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
 006646  004767  006766                 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.
                                        ;;**************************************************************
        006652                          TST6:
        006652  004567  011716                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
        006656  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
        006660  016700  172724          TST6P:  MOV     .CONST, R0      ;GET USER CONSTANT
        006664  004467  005502                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
        006670  010022                  1$:     MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
        006672  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
        006674  001375                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
        006676  004767  006246                  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.
                                        ;;**************************************************************
        006702                          TST7:
        006702  004567  011666                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
        006706  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
        006710  016700  172674                  MOV     .CONST, R0      ;GET USER CONSTANT
        006714  004467  005452                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
        006720                          1$:
        006720  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
        006722  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        006724  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
        006726  004767  011400          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
        006732  004767  012646                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        006736  000004                          .WORD   4               ;ERROR TYPE CODE.
        006740                          65$:
        006740  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
        006742  001366                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
        006744  004767  006200                  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.
        006750  032777  000400  172162          BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
        006756  001416                          BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
        006760  017746  172154                  MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
        006764  042716  177740                  BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
        006770  022726  000006                  CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
        006774  001007                          BNE     TST10           ;BRANCH IF NOT TEST 6
        006776  162767  000001  172076          SUB     #1,     $TSTNM  ;RESET TEST NUM
        007004  162767  000030  172074          SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
```

```
007012  000722                          BR      TST6A           ;GO TO TEST 6

                                ;;**************************************************************
                                ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
                                        ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
                                ;;**************************************************************
007014                          TST10:
007014  004567  011554                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007020  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
007022  016704  172606                  MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
007026  004767  010550          1$:     JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
007032  012400                          MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
007034  001420                          BEQ     TST11           ;;BR IF END OF TABLE.
007036  004467  005330                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007042  010012                  2$:     MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
007044  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007046  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007050  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007052  004767  011254          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007056  004767  012522                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007062  000004                          .WORD   4               ;ERROR TYPE CODE.
007064                          65$:
007064  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007066  001365                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
007070  004767  006054                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
007074  000754                          BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
                                        ;;************************************************************
                                        ;*TEST 11        ROTATE A "0" BIT THROUGH A FIELD OF ONES.
                                        ;;************************************************************
007076                                  TST11:
007076  004567  011472                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007102  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
007104  012700  177777                          MOV     #-1,    R0      ;SET CHECK WORD
007110  004767  007020                          JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
007114  004467  005252                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007120  000241                          1$:     CLC                     ;CLEAR CARRY BIT IN PSW
007122  004767  007026                          JSR     PC,     ROTATE
007126  016201  177776                          MOV     -2(R2), R1      ;GET RESULT
007132  103402                                  BCS     63$             ;BRANCH IF 'C' BIT WAS SET
007134  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007136  001405                                  BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007140  004767  011166                  63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007144  004767  012434                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007150  000005                                  .WORD   5               ;ERROR TYPE CODE.
007152                                  64$:
007152  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007154  001361                                  BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007156  004767  005766                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;;************************************************************
                                        ;*TEST 12        ROTATE A "1" BIT THROUGH A FIELD OF ZEROS
                                        ;;************************************************************
007162                                  TST12:
007162  004567  011406                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007166  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
007170  005000                                  CLR     R0              ;SET CHECK WORD
007172  004767  006736                          JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
007176  004467  005170                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007202  000261                          1$:     SEC                     ;SET 'C' BIT IN PSW
007204  004767  006744                          JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
007210  016201  177776                          MOV     -2(R2), R1      ;GET RESULT
007214  103002                                  BCC     63$             ;BRANCH IF 'C' IS CLEAR
007216  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007220  001405                                  BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007222  004767  011104                  63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007226  004767  012352                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007232  000005                                  .WORD   5               ;ERROR TYPE CODE.
007234                                  64$:
007234  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007236  001361                                  BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007240  004767  005704                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                ;;************************************************************
                                ;*TEST 13        3 XOR 9 TEST PATTERN.
                                ;;************************************************************
007244                          TST13:
007244  004567  011324                  JSR     R5,       $SCOPE   ;GO TO SCOPE ROUTINE.
007250  000777                          .WORD   777                ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                   ; REQUIRED FOR THIS TEST.
007252  000167  000312                  JMP     TST14              ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                   ; AVAILABLE FOR TEST.
007256  005000                  .3X9:   CLR     R0                 ;SET UP TEST DATA
007260  012703  177777                  MOV     #-1,      R3       ;SET COM DATA REG
007264  004467  005102                  JSR     R4,       INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007270  004767  006726          1$:     JSR     PC,       W3X9     ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007274  030502                          BIT     R5,       R2       ;CHECK FOR END OF A BLOCK.
007276  001374                          BNE     1$                 ;BRANCH IF MORE IN CURRENT BLOCK.
007300  004767  005644                  JSR     PC,       MMUP     ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;************************************************************
                                ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
                                ;;************************************************************
007304  005000                          CLR     R0                 ;SET CHECK WORD
007306  004467  005060                  JSR     R4,       INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007312  012704  000100          11$:    MOV     #64.,     R4       ;SET 256. WORD COUNTER
007316                          12$:
007316  012201                          MOV     (R2)+,    R1       ;GET THE DATA FROM MEMORY UNDER TEST.
007320  020001                          CMP     R0,       R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
007322  001405                          BEQ     65$                ;BRANCH OVER ERROR CALL IF GOOD DATA.
007324  0C4767  011002          64$:    JSR     PC,       SPRNT2   ;SET UP VALUES FOR ERROR PRINTING.
007330  C04767  012250                  JSR     PC,       $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
007334  000007                          .WORD   7                  ;ERROR TYPE CODE.
007336                          65$:
007336  012201                          MOV     (R2)+,    R1       ;GET THE DATA FROM MEMORY UNDER TEST.
007340  020001                          CMP     R0,       R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
007342  001405                          BEQ     67$                ;BRANCH OVER ERROR CALL IF GOOD DATA.
007344  004767  010762          66$:    JSR     PC,       SPRNT2   ;SET UP VALUES FOR ERROR PRINTING.
007350  004767  012230                  JSR     PC,       $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
007354  000007                          .WORD   7                  ;ERROR TYPE CODE.
007356                          67$:
007356  012201                          MOV     (R2)+,    R1       ;GET THE DATA FROM MEMORY UNDER TEST.
007360  020001                          CMP     R0,       R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
007362  001405                          BEQ     69$                ;BRANCH OVER ERROR CALL IF GOOD DATA.
007364  004767  010742          68$:    JSR     PC,       SPRNT2   ;SET UP VALUES FOR ERROR PRINTING.
007370  004767  012210                  JSR     PC,       $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
007374  000007                          .WORD   7                  ;ERROR TYPE CODE.
007376                          69$:
007376  012201                          MOV     (R2)+,    R1       ;GET THE DATA FROM MEMORY UNDER TEST.
007400  020001                          CMP     R0,       R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
007402  001405                          BEQ     71$                ;BRANCH OVER ERROR CALL IF GOOD DATA.
007404  004767  010722          70$:    JSR     PC,       SPRNT2   ;SET UP VALUES FOR ERROR PRINTING.
007410  004767  012170                  JSR     PC,       $ERROR   ;*** FRROR *** (GO TYPE A MESSAGE)
007414  000007                          .WORD   7                  ;ERROR TYPE CODE.
007416                          71$:
007416  005100                          COM     R0                 ;COMPLEMENT CHECK WORD
007420  005304                          DEC     R4                 ;DECREMENT 256. WORD COUNTER
007422  001335                          BNE     12$
007424  005100                          COM     R0                 ;COMPLEMENT CHECK WORD
```

```
007426  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007430  001330                  BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
007432  004767  005512          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                        ;;*************************************************************
                        ;*  CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
                        ;;*************************************************************
007436  005000                  CLR     R0
007440  004467  004726          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007444  012704  000100  21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007450  012703  000004  22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
007454                  23$:
007454  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007456  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007460  001405                  BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007462  004767  010644  72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007466  004767  012112          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007472  000007                  .WORD   7               ;ERROR TYPE CODE.
007474                  73$:
007474  005100                  COM     R0              ;COMPLEMENT CHECK WORD
007476  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
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     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007506  004767  010620  74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007512  004767  012066          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007516  000007                  .WORD   7               ;ERROR TYPE CODE.
007520                  75$:
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     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007532  004767  010574  76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007536  004767  012042          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007542  000007                  .WORD   7               ;ERROR TYPE CODE.
007544                  77$:
007544  005303                  DEC     R3              ;DECREMENT 4 WORD COUNTER
007546  001342                  BNE     23$             ;BR IF NOT DONE.
007550  005100                  COM     R0              ;COMPLEMENT CHECK WORD
007552  005304                  DEC     R4              ;DECREMENT 256. WORD COUNTER
007554  001335                  BNE     22$             ;BR IF NOT DONE.
007556  005100                  COM     R0              ;COMPLEMENT CHECK WORD
007560  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007562  001330                  BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
007564  004767  005360          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 14       COMPLEMENT 3 XOR 9 TEST PATTERN
                                ;;*****************************************************************
007570                          TST14:
007570  004567  011000                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007574  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007576  000167  000316                  JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007602  012700  177777                  MOV     #-1,    R0      ;SET UP TEST DATA
007606  005003                          CLR     R3              ;SET COM DATA REG
007610  004467  004556                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007614  004767  006402          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007620  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007622  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007624  004767  005320                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.


                                ;;*****************************************************************
                                ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
007630  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007634  004467  004532                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007640  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007644                          12$:
007644  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007646  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007650  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007652  004767  010454          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007656  004767  011722                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007662  000007                          .WORD   7               ;ERROR TYPE CODE.
007664                          65$:
007664  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007666  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007670  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007672  004767  010434          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007676  004767  011702                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007702  000007                          .WORD   7               ;ERROR TYPE CODE.
007704                          67$:
007704  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007706  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007710  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007712  004767  010414          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007716  004767  011662                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007722  000007                          .WORD   7               ;ERROR TYPE CODE.
007724                          69$:
007724  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007726  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007730  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007732  004767  010374          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007736  004767  011642                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007742  000007                          .WORD   7               ;ERROR TYPE CODE.
007744                          71$:
007744  005100                          COM     R0              ;COMPLEMENT CHECK WORD
007746  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
007750  001335                          BNE     12$
```

```
007752  005100                      COM     R0              ;COMPLEMENT CHECK WORD
007754  030502                      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007756  001330                      BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
007760  004767  005164              JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.

                            ;;*******************************************************
                            ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
                            ;;*******************************************************
007764  012700  177777              MOV     #-1,    R0      ;SET UP CHECK WORD.
007770  004467  004376              JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007774  012704  000100      21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
010000  012703  000004      22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
010004                      23$:
010004  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010006  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010010  001405                      BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010012  004767  010314      72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010016  004767  011562              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010022  000007                      .WORD   7               ;ERROR TYPE CODE.
010024                      73$:
010024  005100                      COM     R0              ;COMPLEMENT CHECK WORD
010026  005142                      COM     -(R2)           ;COMPLEMENT TEST DATA
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     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010036  004767  010270      74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010042  004767  011536              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010046  000007                      .WORD   7               ;ERROR TYPE CODE.
010050                      75$:
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     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010062  004767  010244      76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010066  004767  011512              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010072  000007                      .WORD   7               ;ERROR TYPE CODE.
010074                      77$:
010074  005303                      DEC     R3              ;DECREMENT 4 WORD COUNTER
010076  001342                      BNE     23$             ;BR IF NOT DONE.
010100  005100                      COM     R0              ;COMPLEMENT CHECK WORD
010102  005304                      DEC     R4              ;DECREMENT 256. WORD COUNTER
010104  001335                      BNE     22$             ;BR IF NOT DONE.
010106  005100                      COM     R0              ;COMPLEMENT CHECK WORD
010110  030502                      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010112  001330                      BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
010114  004767  005030              JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                        ;;*******************************************************************
                                        ;*TEST 15        MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
                                        ;;*******************************************************************
010120                                  TST15:
010120  004567  010450                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
010124  000777                                  .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                        ; REQUIRED FOR THIS TEST.
010126  000167  000610                          JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
010132  012700  000401                          MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN
010136  012703  177777                          MOV     #-1,    R3      ;SET COM DATA REG
010142  004467  004224                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010146  004767  006050                  1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
010152  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
010154  001374                                  BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
010156  004767  004766                          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;;*******************************************************************
                                        ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                                        ;;*******************************************************************
010162  012700  000401                          MOV     #401,   R0      ;RESET PARITY "ALL ZEROS" PATTERN.
010166  012703  177777                          MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
010172  004467  004174                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010176  012704  000100                  11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
010202                                  12$:
010202  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010204  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010206  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010210  004767  010116                  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010214  004767  011364                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010220  000007                                  .WORD   7               ;ERROR TYPE CODE.
010222                                  65$:
010222  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010224  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010226  001405                                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010230  004767  010076                  66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010234  004767  011344                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010240  000007                                  .WORD   7               ;ERROR TYPE CODE.
010242                                  67$:
010242  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010244  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010246  001405                                  BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010250  004767  010056                  68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010254  004767  011324                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010260  000007                                  .WORD   7               ;ERROR TYPE CODE.
010262                                  69$:
010262  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
010264  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
010266  001405                                  BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
010270  004767  010036                  70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
010274  004767  011304                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
010300  000007                                  .WORD   7               ;ERROR TYPE CODE.
010302                                  71$:
010302  010046                                  MOV     R0,     -(SP)   ;SAVE R0
010304  010300                                  MOV     R3,     R0      ;PUT R3 INTO R0
010306  012603                                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
```

```
010310  005304                          DEC     R4                      ;COUNT 256. WORDS
010312  001333                          BNE     12$                     ;BRANCH IF MORE
010314  010046                          MOV     R0,     -(SP)           ;SAVE R0
010316  010300                          MOV     R3,     R0              ;PUT R3 INTO R0
010320  012603                          MOV     (SP)+,  R3              ;PUT SAVED R0 INTO R3
010322  030502                          BIT     R5,     R2              ;CHECK FOR END OF A BLOCK.
010324  001324                          BNE     11$                     ;BRANCH IF MORE IN CURRENT BLOCK.
010326  004767  004616                  JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 11$.

                                ;;*****************************************************************
                                ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
                                ;;*****************************************************************
010332  012700  000401                  MOV     #401,   R0              ;SET UP PARITY "ALL ZEROS" PATTERN.
010336  012703  177777                  MOV     #-1,    R3              ;SET UP ALL ONES PATTERN.
010342  004467  004024                  JSR     R4,     INITMM          ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010346  012704  000100          21$:    MOV     #64.,R4                 ;SET 256. WORD COUNTER
010352                          22$:
010352  012201                          MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010354  020001                          CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010356  001405                          BEQ     73$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010360  004767  007746          72$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010364  004767  011214                  JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MFSSAGE)
010370  000007                          .WORD   7                       ;ERROR TYPE CODE.
010372                          73$:
010372  005100                          COM     R0                      ;COMPLEMENT CHECK WORD
010374  005142                          COM     -(R2)                   ;COMPLEMENT TEST DATA
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     75$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010404  004767  007722          74$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010410  004767  011170                  JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010414  000007                          .WORD   7                       ;ERROR TYPE CODE.
010416                          75$:
010416  005100                          COM     R0                      ;COMPLEMENT CHECK WORD
010420  005142                          COM     -(R2)                   ;RESTORE DATA
010422  012201                          MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010424  020001                          CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATP READ.
010426  001405                          BEQ     77$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010430  004767  007676          76$:    JSR     PC,     SPRNT2          ,SET UP VALUES FOR ERROR PRINTING.
010434  004767  011144                  JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010440  000007                          .WORD   7                       ;ERROR TYPE CODE.
010442                          77$:
010442  012201                          MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010444  020001                          CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010446  001405                          BEQ     79$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010450  004767  007656          78$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010454  004767  011124                  JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010460  000007                          .WORD   7                       ;ERROR TYPE CODE.
010462                          79$:
010462  005100                          COM     R0                      ;COMPLEMENT CHECK WORD
010464  005142                          COM     -(R2)                   ;COMPLEMENT TEST DATA
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     81$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010474  004767  007632          80$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010500  004767  011100                  JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
010504  000007                  .WORD   7                       ;ERROR TYPE CODE.
010506                   81$:
010506  005100                  COM     R0                      ;COMPLEMENT CHECK WORD
010510  005142                  COM     -(R2)                   ;RESTORE DATA
010512  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010514  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010516  001405                  BEQ     83$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010520  004767  007606   82$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010524  004767  011054          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010530  000007                  .WORD   7                       ;ERROR TYPE CODE.
010532                   83$:
010532  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010534  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA KEAD.
010536  001405                  BEQ     85$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010540  004767  007566   84$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010544  004767  011034          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010550  000007                  .WORD   7                       ;ERROR TYPE CODE.
010552                   85$:
010552  005100                  COM     R0                      ;COMPLEMENT CHECK WORD
010554  005142                  COM     -(R2)                   ;COMPLEMENT TEST DATA
010556  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010560  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010562  001405                  BEQ     87$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010564  004767  007542   86$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010570  004767  011010          JSR     PC,     $ERROR          ;*** ERROR *** (GO T/PE A MESSAGE)
010574  000007                  .WORD   7                       ;ERROR TYPE CODE.
010576                   87$:
010576  005100                  COM     R0                      ;COMPLEMENT CHECK WORD
010600  005142                  COM     -(R2)                   ;RESTORE DATA
010602  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010604  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010606  001405                  BEQ     89$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010610  004767  007516   88$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010614  004767  010764          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010620  000007                  .WORD   7                       ;ERROR TYPE CODE.
010622                   89$:
010622  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010624  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010626  001405                  BEQ     91$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010630  004767  007476   90$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010634  004767  010744          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010640  000007                  .WORD   7                       ;ERROR TYPE CODE.
010642                   91$:
010642  005100                  COM     R0                      ;COMPLEMENT CHECK WORD
010644  005142                  COM     -(R2)                   ;COMPLEMENT TEST DATA
010646  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010650  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
010652  001405                  BEQ     93$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010654  004767  007452   92$:   JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
010660  004767  010720          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
010664  000007                  .WORD   7                       ;ERROR TYPE CODE.
010666                   93$:
010666  005100                  COM     R0                      ;COMPLEMENT CHECK WORD
C10670  005142                  COM     -(R2)                   ;RESTORE DATA
010672  012201                  MOV     (R2)+,  R1              ;GET THE DATA FROM MEMORY UNDER TEST.
010674  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
010676  001405                  BEQ     95$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
010700  004767  007426  94$:    JSR     PC,       SPRNT2        ;SET UP VALUES FOR ERROR PRINTING.
010704  004767  010674          JSR     PC,       $ERROR        ;*** ERROR *** (GO TYPE A MESSAGE)
010710  000007                  .WORD   7                       ;ERROR TYPE CODE.
010712                  95$:
010712  010046                  MOV     R0,       -(SP)         ;SAVE R0
010714  010300                  MOV     R3,       R0            ;PUT R3 INTO R0
010716  012603                  MOV     (SP)+,    R3            ;PUT SAVED R0 INTO R3
010720  005304                  DEC     R4                      ;DECREMENT 256. WORD COUNTER
01C722  001213                  BNE     22$                     ;BRANCH IF MORE.
010724  010046                  MOV     R0,       -(SP)         ;SAVE R0
010726  010300                  MOV     R3,       R0            ;PUT R3 INTO R0
010730  012603                  MOV     (SP)+,    R3            ;PUT SAVED R0 INTO R3
010732  030502                  BIT     R5,       R2            ;CHECK FOR END OF A BLOCK.
010734  001204                  BNE     21$                     ;BRANCH IF MORE IN CURRENT BLOCK.
010736  004767  004206          JSR     PC,       MMUP          ;FIND NEXT BLOCK AND LOOP TO 21$.

                        ;;*************************************************************
                        ;*TEST 16       COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
                        ;;*************************************************************
010742                  TST16:
010742  004567  007626          JSR     R5,       $SCOPE        ;GO TO SCOPE ROUTINE.
010746  000777                  .WORD   777                     ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
010750  000167  000610          JMP     TST17                   ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
010754  012700  177777          MOV     #-1,      R0            ;SET UP ALL ONES PATTERN
010760  012703  000401          MOV     #401,     R3            ;SET UP PARITY "ALL ZEROS" PATTERN
010764  004467  003402          JSR     R4,       INITMM        ;INITIALIZE THE MEMORY ADDRESS POINTERS.
010770  004767  005226  1$:     JSR     PC,       W3X9          ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
010774  030502                  BIT     R5,       R2            ;CHECK FOR END OF A BLOCK.
010776  001374                  BNE     1$                      ;BRANCH IF MORE IN CURRENT BLOCK.
011000  004767  004144          JSR     PC,       MMUP          ;FIND NEXT BLOCK AND LOOP TO 1$.

                        ;;*************************************************************
                        ;*  CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN' WRITTEN ABOVE.
                        ;;*************************************************************
011004  012700  177777          MOV     #-1,      R0            ;SET UP ALL ONES PATTERN
011010  012703  000401          MOV     #401,     R3            ;SET UP PARITY "ALL ZEROS" PATTERN
011014  004467  003352          JSR     R4,       INITMM        ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011020  012704  000100  11$:    MOV     #64 ,     R4            ;SET 256. WORD COUNTER
011024                  12$:
011024  012201                  MOV     (R2)+,    R1            ;GET THE DATA FROM MEMORY UNDER TEST.
011026  020001                  CMP     R0,       R1            ;COMPARE THE CHECK WORD WITH THE DATA READ.
011030  001405                  BEQ     65$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
011032  004767  007274  64$:    JSR     PC,       SPRNT2        ;SET UP VALUES FOR ERROR PRINTING.
011036  004767  010542          JSR     PC,       $ERROR        ;*** ERROR *** (GO TYPE A MESSAGE)
011042  000007                  .WORD   7                       ;ERROR TYPE CODE.
011044                  65$:
011044  012201                  MOV     (R2)+,    R1            ;GET THE DATA FROM MEMORY UNDER TEST.
011046  020001                  CMP     R0,       R1            ;COMPARE THE CHECK WORD WITH THE DATA READ.
011050  001405                  BEQ     67$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
011052  004767  007254  66$:    JSR     PC,       SPRNT2        ;SET UP VALUES FOR ERROR PRINTING.
011056  004767  010522          JSR     PC,       $ERROR        ;*** ERROR *** (GO TYPE A MESSAGE)
011062  000007                  .WORD   7                       ;ERROR TYPE CODE.
011064                  67$:
```

```
011064  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011066  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011070  001405                  BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011072  004767  007234  68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011076  004767  010502          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011102  000007                  .WORD   7               ;ERROR TYPE CODE.
011104                  69$:
011104  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011106  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011110  001405                  BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011112  004767  007214  70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011116  004767  010462          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011122  000007                  .WORD   7               ;ERROR TYPE CODE.
011124                  71$:
011124  010046                  MOV     R0,     -(SP)   ;SAVE R0
011126  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
011130  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
011132  005304                  DEC     R4              ;COUNT 256. WORDS
011134  001333                  BNE     12$             ;BRANCH IF MORE
011136  010046                  MOV     R0,     -(SP)   ;SAVE R0
011140  010300                  MOV     R3,     R0      ;PUT R3 INTO R0
011142  012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
011144  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
011146  001324                  BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
011150  004767  003774          JSR     PC,     MMLUP   ;FIND NEXT BLOCK AND LOOP TO 11$.

                        ;;*************************************************************
                        ;*  CHECK, COM, CHECK, COM  CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
                        ;;*************************************************************
011154  012700  177777          MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
011160  012703  000401          MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
011164  004467  003202          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
011170  012704  000100  21$:    MOV     #64.,R4         ;SET 256. WORD COUNTER
011174                  22$:
011174  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011176  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011200  001405                  BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011202  004767  007124  72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011206  004767  010372          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011212  000007                  .WORD   7               ;ERROR TYPE CODE.
011214                  73$:
011214  005100                  COM     R0              ;COMPLEMENT CHECK WORD
011216  005142                  COM     -(R2)           ;COMPLEMENT TEST DATA
011220  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011222  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011224  001405                  BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011226  004767  007100  74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011232  004767  010346          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011236  000007                  .WORD   7               ;ERROR TYPE CODE.
011240                  75$:
011240  005100                  COM     R0              ;COMPLEMENT CHECK WORD
011242  005142                  COM     -(R2)           ;RESTORE DATA
011244  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011246  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011250  001405                  BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011252  004767  007054  76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
011256  004767  010322            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011262  000007                    .WORD   7                   ;ERROR TYPE CODE.
011264                     77$:
011264  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011266  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011270  001405                    BEQ     79$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011272  004767  007034     78$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011276  004767  010302            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011302  000007                    .WORD   7                   ;ERROR TYPE CODE.
011304                     79$:
011304  005100                    COM     R0                  ;COMPLEMENT CHECK WORD
011306  005142                    COM     -(R2)               ;COMPLEMENT TEST DATA
011310  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011312  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011314  001405                    BEQ     81$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011316  004767  007010     80$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011322  004767  010256            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011326  000007                    .WORD   7                   ;ERROR TYPE CODE.
011330                     81$:
011330  005100                    COM     R0                  ;COMPLEMENT CHECK WORD
011332  005142                    COM     -(R2)               ;RESTORE DATA
011334  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011336  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011340  001405                    BEQ     83$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011342  004767  006764     82$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011346  004767  010232            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011352  000007                    .WORD   7                   ;ERROR TYPE CODE.
011354                     83$:
011354  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011356  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011360  001405                    BEQ     85$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011362  004767  006744     84$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011366  004767  010212            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011372  000007                    .WORD   7                   ;ERROR TYPE CODE.
011374                     85$:
011374  005100                    COM     R0                  ;COMPLEMENT CHECK WORD
011376  005142                    COM     -(R2)               ;COMPLEMENT TEST DATA
011400  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011402  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011404  001405                    BEQ     87$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011406  004767  006720     86$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011412  004767  010166            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011416  000007                    .WORD   7                   ;ERROR TYPE CODE.
011420                     87$:
011420  005100                    COM     R0                  ;COMPLEMENT CHECK WORD
011422  005142                    COM     -(R2)               ;RESTORE DATA
011424  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011426  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011430  001405                    BEQ     89$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
011432  004767  006674     88$:   JSR     PC,       SPRNT2    ;SET UP VALUES FOR ERROR PRINTING.
011436  004767  010142            JSR     PC,       $ERROR    ;*** ERROR *** (GO TYPE A MESSAGE)
011442  000007                    .WORD   7                   ;ERROR TYPE CODE.
011444                     89$:
011444  012201                    MOV     (R2)+,    R1        ;GET THE DATA FROM MEMORY UNDER TEST.
011446  020001                    CMP     R0,       R1        ;COMPARE THE CHECK WORD WITH THE DATA READ.
011450  001405                    BEQ     91$                 ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

```
011452  004767  006654          90$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011456  004767  010122                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011462  000007                          .WORD   7               ;ERROR TYPE CODE.
011464                          91$:
011464  005100                          COM     R0              ;COMPLEMENT CHECK WORD
011466  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
011470  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011472  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011474  001405                          BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011476  004767  006630          92$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011502  004767  010C76                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011506  000007                          .WORD   7               ;ERROR TYPE CODE.
011510                          93$:
011510  005100                          COM     R0              ;COMPLEMENT CHECK WORD
011512  005142                          COM     -(R2)           ;RESTORE DATA
011514  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
011516  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
011520  001405                          BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
011522  004767  006604          94$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
011526  004767  010052                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
011532  000007                          .WORD   7               ;ERROR TYPE CODE.
011534                          95$:
011534  010046                          MOV     R0,     -(SP)   ;SAVE R0
011536  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
011540  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
011542  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
011544  001213                          BNE     22$             ;BRANCH IF MORE.
011546  010046                          MOV     R0,     -(SP)   ;SAVE R0
011550  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
011552  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
011554  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
011556  001204                          BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
011560  004767  003364                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

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

```
012004  016737  167626  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
012012  004767  006270          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
012016  004767  007562                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012022  000012                          .WORD   12              ;ERROR TYPE CODE.
012024  000562                          BR      WWPB4           ;SKIP TRAP SERVICE.

                                ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
012026  016737  167604  000114  PBTRP:  MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
012034  022626                          CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
012036  016703  167564                  MOV     .MPRO,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
012042  032713  000001          21$:    BIT     #BITO,  (R3)    ;CHECK IF THIS REGISTER EXISTS.
012046  001003                          BNE     22$             ;BR IF IT DOESN'T EXIST.
012050  017301  000000                  MOV     @(R3),  R1      ;GET THE CONTENTS.
012054  100413                          BMI     23$             ;BR IF ERROR FLAG SET.
012056  062703  000010          22$:    ADD     #10,    R3      ;MOVE POINTER TO NEXT REG.
012062  020367  167542                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
012066  103765                          BLO     21$             ;BR IF MORE REGISTERS.
012070  004767  006212          64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
012074  004767  007504                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012100  000013                          .WORD   13              ;ERROR TYPE CODE.
012102  000533                          BR      WWPB4           ;EXIT AFTER ERROR.
012104  036763  167434  000002  23$:    BIT     BITPT,  2(R3)   ;CHECK THE MAP FOR THIS REGISTER.
012112  001011                          BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
012114  036763  167426  000004          BIT     BITPT+2,4(R3)   ;CHECK THE HI 64K.
012122  001005                          BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
012124  004767  006152          65$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
012130  004767  007450                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012134  000014                          .WORD   14              ;ERROR TYPE CODE.
012136                          24$:
012136  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
012140  010200                          MOV     R2,     R0      ;GET THE ADDRESS POINTER.
012142  042700  003777                  BIC     #3777,  R0      ;CLEAR LOW ADDRESS BITS.
012146  000300                          SWAB    R0              ;SHIFT 6 PLACES RIGHT.
012150  006300                          ASL     R0
012152  006300                          ASL     R0
012154  005767  166426                  TST     MMAVA           ;CHECK FOR MEM MGMT.
012160  001404                          BEQ     25$             ;BR IF NO MEM MGMT.
012162  042700  177600                  BIC     #177600,R0      ;CLEAR BANK BITS
012166  063700  172344                  ADD     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
012172  052700  100001          25$:    BIS     #BIT15+BITO,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
012176  016367  000006  167312          MOV     6(R3),  RESRVD  ;GET APPROPIATE MASK.
012204  046700  167306                  BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
012210  046701  167302                  BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
                                ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
012214  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012216  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012220  004767  006056          66$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
012224  004767  007354                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012230  000015                          .WORD   15              ;ERROR TYPE CODE.
012232                          67$:
012232  005073  000000                  CLR     @(R3)           ;CLEAR REG INCLUDING ACTION ENABLE.
012236  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
012240  062703  000010          26$:    ADD     #10,    R3      ;UPDATE POINTER TO NEXT PARITY REG + MAP.
012244  020367  167360                  CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
012250  101014                          BHI     WWPB3           ;BR IF END OF TABLE REACHED.
012252  032713  000001                  BIT     #BITO,  (R3)    ;CHECK IF NEXT REG EXISTS.
```

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

```
                                   ;;*******************************************************
                                   ;*TEST 20       RANDOM DATA  TESTING THRU PROGRAM CODE RELOCATION.
                                   ;;*******************************************************
012436                             TST20:
012436  004567  006132                      JSR     R5,      $SCOPE   ;GO TO SCOPE ROUTINE.
012442  000000                              .WORD   0                 ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
012444  010703                     RANTST:  MOV     PC,      R3       ;GET CURRENT PROGRAM COUNTER.
012446  042703  007777                      BIC     #7777,   R3       ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
012452  004467  001714                      JSR     R4,      INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012456  010246                     1$:      MOV     R2,      -(SP)    ;SAVE MEMORY POINTER.
012460  010346                              MOV     R3,      -(SP)    ;SAVE "DATA" POINTER.
012462  012322                     2$:      MOV     (R3)+,   (R2)+    ;MOV CODE INTO TEST MEMORY.
012464  032703  007777                      BIT     #7777,   R3       ;CHECK FOR END OF "DATA TABLE"
012470  001002                              BNE     3$                ;BRANCH IF MORE
012472  162703  010000                      SUB     #10000,  R3       ;RESET POINTER TO START OF "RANDOM DATA"
012476  030502                     3$:      BIT     R5,      R2       ;CHECK FOR END OF BLOCK
012500  001370                              BNE     2$                ;BRANCH IF MORE.
012502  012603                              MOV     (SP)+,   R3       ;RESET "DATA" POINTER.
012504  012602                              MOV     (SP)+,   R2       ;RESET MEMORY POINTER.
012506  012300                     4$:      MOV     (R3)+,   R0       ;GET S/B DATA.
012510  012201                              MOV     (R2)+,   R1       ;GET THE DATA FROM MEMORY UNDER TEST.
012512  020001                              CMP     R0,      R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
012514  001405                              BEQ     65$               ;BRANCH OVER ERROR CALL IF GOOD DATA.
012516  004767  005610             64$:     JSR     PC,      SPRNT2   ;SET UP VALUES FOR ERROR PRINTING.
012522  004767  007056                      JSR     PC,      $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
012526  000020                              .WORD   20                ;ERROR TYPE CODE.
012530                             65$:
012530  032703  007777                      BIT     #7777,   R3       ;CHECK FOR END OF "DATA TABLE"
012534  001002                              BNE     5$                ;BR IF MORE.
012536  162703  010000                      SUB     #10000,  R3       ;RESET POINTER TO TOP OF "DATA TABLE".
012542                             5$:
012542  030502                              BIT     R5,      R2       ;CHECK FOR END OF A BLOCK.
012544  001360                              BNE     4$                ;BRANCH IF MORE IN CURRENT BLOCK.
012546  004767  002376                      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 EXAMFLE 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.
                                       ;;*************************************************************
012552                                 TST21:
012552  004567  006016                         JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012556  000003                                 .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                       ; REQUIRED FOR THIS TEST.
012560  000167  000056                         JMP     TST22           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                       ; AVAILABLE FOR TEST.
012564  012703  010412                 DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
012570  012704  000205                         MOV     #205,   R4      ;GET 'RTS R5'
012574  010400                                 MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
012576  004467  001570                         JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012602  010322                         1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012604  010412                         2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012606  004542                                 JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
012610  012201                                 MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012612  020001                                 CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
012614  001405                                 BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
012616  004767  005504                 64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
012622  004767  006756                         JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
012626  000021                                 .WORD   21              ;ERROR TYPE CODE.
012630                                 65$:
012630  010322                                 MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
012632  030502                                 BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
012634  001363                                 BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
012636  004767  002306                         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 MEMOFY 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.
                              ;;*************************************************************
012642                        TST22:
012642  004567  005726                JSR     R5,     $SCOPE   ;GO TO SCOPE ROUTINE.
012646  000003                        .WORD   3                ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                               ; REQUIRED FOR THIS TEST.
012650  000167  000060                JMP     TST23            ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                               ; AVAILABLE FOR TEST.
012654  012703  110412        DIDBL:  MOV     #110412,R3       ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
012660  012704  000205                MOV     #205,   R4       ;GET 'RTS R5'
012664  012700  110605                MOV     #110605,R0       ;SET UP S/B DATA AFTER EXECUTION.
012670  004467  001476                JSR     R4,     INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012674  010322                1$:     MOV     R3,     (R2)+    ;PUT IUT INTO FIRST LOC OF BLOCK.
012676  010412                2$:     MOV     R4,     (R2)     ;PUT 'RTS R5' FOLLOWING IUT.
012700  004542                        JSR     R5,     -(R2)    ;GO EXECUTE THE IUT.
012702  012201                        MOV     (R2)+,  R1       ;GET THE DATA FROM THE MEM ADR UNDER TEST.
012704  020001                        CMP     R0,     R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
012706  001405                        BEQ     65$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
012710  004767  005412        64$:    JSR     PC,     SPRNT3   ;SET UP VALUES FOR ERROR PRINTING.
012714  004767  006664                JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
012720  000021                        .WORD   21               ;ERROR TYPE CODE.
012722                        65$:
012722  010322                        MOV     R3,     (R2)+    ;PUT THE IUT INTO THE NEXT LOCATION.
012724  030502                        BIT     R5,     R2       ;CHECK FOR END OF A BLOCK.
012726  001363                        BNE     2$               ;BRANCH IF MORE IN CURRENT BLOCK.
012730  004767  002214                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.
                                ;;*******************************************************************
012734                          TST23:
012734  004567  005634                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
012740  000003                          .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                ; REQUIRED FOR THIS TEST.
012742  000167  000064                  JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
012746  012703  110342          DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
012752  012704  000205                  MOV     #205,   R4      ;GET 'RTS R5'
012756  012700  161342                  MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
012762  004467  001404                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
012766  010322                  1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
012770  010412                  2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
012772  004562  177776                  JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
012776  005302                          DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
013000  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013002  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013004  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013006  004767  005314          64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013012  004767  006566                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013016  000021                          .WORD   21              ;ERROR TYPE CODE.
013020                          65$:
013020  010322                          MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013022  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013024  001361                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013026  004767  002116                  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.
                                        ;;*************************************************************
 013032                                 TST24:
 013032   004567   005536                      JSR     R5,     $SCOPE   ;GO TO SCOPE ROUTINE.
 013036   000003                               .WORD   3                ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                        ; REQUIRED FOR THIS TEST.
 013040   000167   000060                      JMP     TST25            ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
 013044   012703   005412               DIPDO: MOV     #005412,R3       ;GET 'NEG (R2)' INSTRUCTION (IUT).
 013050   012704   000205                      MOV     #205,   R4       ;GET 'RTS R5'
 013054   012700   172366                      MOV     #172366,R0       ;SET UP S/B DATA AFTER EXECUTION.
 013060   004467   001306                      JSR     R4,     INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 013064   010322                        1$:    MOV     R3,     (R2)+    ;PUT IUT INTO FIRST LOC OF BLOCK.
 013066   010412                        2$:    MOV     R4,     (R2)     ;PUT 'RTS R5' FOLLOWING IUT.
 013070   004542                               JSR     R5,     -(R2)    ;GO EXECUTE THE IUT.
 013072   012201                               MOV     (R2)+,  R1       ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 013074   020001                               CMP     R0,     R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
 013076   001405                               BEQ     65$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
 013100   004767   005222               64$:   JSR     PC,     SPRNT3   ;SET UP VALUES FOR ERROR PRINTING.
 013104   004767   006474                      JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
 013110   000021                               .WORD   21               ;ERROR TYPE CODE.
 013112                                 65$:
 013112   010322                               MOV     R3,     (R2)+    ;PUT THE IUT INTO THE NEXT LOCATION.
 013114   030502                               BIT     R5,     R2       ;CHECK FOR END OF A BLOCK.
 013116   001363                               BNE     2$               ;BRANCH IF MORE IN CURRENT BLOCK.
 013120   004767   002024                      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.
                                        ;;*********************************************************************
013124                                  TST25:
013124  004567  005444                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013130  000003                                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                        ; REQUIRED FOR THIS TEST.
013132  000167  000060                          JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                        ; AVAILABLE FOR TEST.
013136  012703  142242                  DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
013142  012704  000205                          MOV     #205,   R4      ;GET 'RTS R5'
013146  012700  142000                          MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
013152  004467  001214                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013156  010322                          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
013160  010412                          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
013162  004542                                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
013164  012201                                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013166  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013170  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013172  004767  005130                  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
013176  004767  006402                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013202  000021                                  .WORD   21              ;ERROR TYPE CODE.
013204                                  65$:
013204  010322                                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
013206  030502                                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013210  001363                                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
013212  004767  001732                          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.
                                   ;;*************************************************************
013216                             TST26:
013216  004567  005352                     JSR     R5,     $SCOPE   ;GO TO SCOPE ROUTINE.
013222  000003                             .WORD   3                ;MINIMUM BLOCK SIZE OF 2 WORDS
                                                                    ; REQUIRED FOR THIS TEST.
013224  000167  000062                     JMP     TST27            ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                    ; AVAILABLE FOR TEST.
013230  012703  152212             DPDBH:  MOV     #152212,R3       ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
013234  012704  000205                     MOV     #205,   R4       ,GET 'RTS R5'
013240  012700  157212                     MOV     #157212,R0       ;SET UP S/B DATA AFTER EXECUTION.
013244  004467  001122                     JSR     R4,     INITMM   ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013250  010322                     1$:     MOV     R3,     (R2)+    ;PUT IUT INTO FIRST LOC OF BLOCK.
013252  010412                     2$:     MOV     R4,     (R2)     ;PUT 'RTS R5' FOLLOWING IUT.
013254  004542                             JSR     R5,     -(R2)    ;GO EXECUTE THE IUT.
013256  005302                             DEC     R2               ;RESET R2 TO POINT TO IUT.
013260  012201                             MOV     (R2)+,  R1       ;GET THE DATA FROM THE MEM ADR UNDER TEST.
013262  020001                             CMP     R0,     R1       ;COMPARE THE CHECK WORD WITH THE DATA READ.
013264  001405                             BEQ     65$              ;BRANCH OVER ERROR CALL IF GOOD DATA.
013266  004767  005034             64$:    JSR     PC,     SPRNT3   ;SET UP VALl'ES FOR ERROR PRINTING.
013272  004767  006306                     JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
013276  000021                             .WORD   21               ;ERROR TYPE CODE.
013300                             65$:
013300  010322                             MOV     R3,     (R2)+    ;PUT THE IUT INTO THE NEXT LOCATION.
013302  030502                             BIT     R5,     R2       ;CHECK FOR END OF A BLOCK.
013304  001362                             BNE     2$               ;BRANCH IF MORE IN CURRENT BLOCK.
013306  004767  001636                     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.
                                        ;;*****************************************************************
013312                                  TST27:
013312  004567  005256                          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013316  000000                                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013320  004467  001046                          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013324  010267  166264                          MOV     R2,TEMP         ;SAVE BANK STARTING ADDRESS
013330  005003                          1$:     CLR     R3              ;CLEAR PASS COUNTER
013332  012700  000377                          MOV     #000377,R0      ;SETUP TO WRITE PATTERN
013336  010022                          2$:     MOV     R0,(R2)+        ;WRITE PATTERN
013340  030502                                  BIT     R5,R2           ;END OF 4K?
013342  001375                                  BNE     2$              ;CONTINUE WRITING IF NO.
013344  014201                          3$:     MOV     -(R2),R1        ;GET DATA WRITEN
013346  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013350  001405                                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013352  004767  004754                  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013356  004767  006222                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013362  000010                                  .WORD   10              ;ERROR TYPE CODE.
013364                                  65$:
013364  000300                          4$:     SWAB    R0              ;SWAP BYTES OF DATA
013366  010012                                  MOV     R0,(R2) ;WRITE SWAPPED WORD
013370  011201                                  MOV     (R2),R1         ;GET DATA WRITEN
013372  020001                                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013374  001405                                  BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013376  004767  004730                  66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013402  004767  006176                          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013406  000010                                  .WORD   10              ;ERROR TYPE CODE.
013410                                  67$:
013410  000300                                  SWAB    R0              ;PUT DATA BACK TO ORINGINAL
013412  005703                                  TST     R3              ;IF ON PASS 0 OR PASS 3
013414  001403                                  BEQ     5$              ;WE ARE ADDRESSING DOWN
013416  020327  000003                          CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
013422  001010                                  BNE     6$              ;UPWARD
```

```
013424  030502          5$:     BIT     R5,R2                   ;DONE A PASS?
013426  001346                  BNE     3$                      ;IF NO CONTINUE
013430  005203                  INC     R3                      ;IF YES INCREMENT PASS COUNTER
013432  022703  000004          CMP     #4,R3                   ;ARE WE DONE ALL PASSES FOR THIS 4K?
013436  001427                  BEQ     9$                      ;IF YES BRANCH
013440  000300                  SWAB    R0                      ;ELSE SET UP NEW READ WORD
013442  000404                  BR      7$                      ;GO TO START OF ADDRESS UP
013444  062702  000002  6$:     ADD     #2,R2                   ;UPDATE TO NEXT ADDRESS
013450  030502                  BIT     R5,R2                   ;DONE A PASS
013452  001411                  BEQ     8$                      ;IF YES BRANCH
013454  011201          7$:     MOV     (R2),R1                 ;GET DATA WRITTEN
013456  020001                  CMP     R0,     R1              ;COMPARE THE CHECK WORD WITH THE DATA READ.
013460  001405                  BEQ     69$                     ;BRANCH OVER ERROR CALL IF GOOD DATA.
013462  004767  004644  68$:    JSR     PC,     SPRNT2          ;SET UP VALUES FOR ERROR PRINTING.
013466  004767  006112          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
013472  000010                  .WORD   10                      ;ERROR TYPE CODE.
013474                  69$:
013474  000733                  BR      4$
013476  005203          8$:     INC     R3                      ;INCREMENT      PASS COUNTER
013500  000300                  SWAB    R0                      ;SET UP NEW READ WORD
013502  020327  000002          CMP     R3,#2                   ;ADDRESSING UP?
013506  001316                  BNE     3$                      ;IF NO GO TO DOWN SEQUENCE
n13510  016702  166100          MOV     TEMP,R2                 ;IF YES RESET ADDRESS TO START
013514  000757                  BR      7$                      ;GO TO UP SEQUENCE
013516  004467  000650  9$:     JSR     R4,INITMM               ;INITIALIZE MEMORY ADDRESS POINTERS
013522  004767  001422          JSR     PC,MMUP                 ;UPDATE TO NEW BANK IF EXISTS

                        ;;*************************************************************
                        ;*TEST 30        WRITE CHECKERBOARD STARTING WITH '125252' DATA.
                        ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
                        ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
                        ;* DETERIORATE BETWEEN REFRESH CYCLES.
                        ;*
                        ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
                        ;*      R1=DATA READ FROM MEMORY(WAS)
                        ;*      R2=VIRTUAL ADDRESS
                        ;*      R3=SMALL LOOP COUNTER FOR STALL
                        ;*      R4=NUMBER OF TIMES SMALL LOOP DONE
                        ;*      R5=BLOCK BOUNDARY BIT MASK.
                        ;;*************************************************************
013526                  TST30:
013526  004567  005042          JSR     R5,     $SCOPE          ;GO TO SCOPE ROUTINE.
013532  000000                  .WORD   0                       ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013534  004467  000632          JSR     R4,     INITMM          ;INITIALIZE THE MEMORY ADDRESS POINTERS
013540  012700  125252          MOV     #125252 ,R0             ;SETUP DATA PATTERN
013544  010022          1$:     MOV     R0      ,(R2)+          ;WRITE A WORD
013546  005100                  COM     R0                      ;COMPLEMENT DATA
013550  030502                  BIT     R5,     R2              ;CHECK FOR END OF A BLOCK.
013552  001374                  BNE     1$                      ;BRANCH IF MORE IN CURRENT BLOCK.
013554  004767  C .370          JSR     PC,     MMUP            ;FIND NEXT BLOCK AND LOOP TO 1$.
013560  005003                  CLR     R3                      ;SET UP COUNTER FOR STALL
013562  012704  000046          MOV     #46,    R4              ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
013566  005303          2$:     DEC     R3
013570  001376                  BNE     2$
013572  005304                  DEC     R4
013574  001374                  BNE     2$
```

```
013576  004467  000570                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013602  012700  125252                  MOV     #125252,R0      ;INIT DATA FOR CHECKING
013606                          3$:
013606  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
013610  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ
013612  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013614  004767  004512          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013620  004767  005760                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013624  000006                          .WORD   6               ;ERROR TYPE CODE.
013626                          65$:
013626  005100                          COM     R0
013630  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013632  001365                          BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
013634  004767  001310                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
                                ;;*****************************************************************
                                ;*TEST 31       WRITE CHECKERBOARD STARTING WITH 052525 DATA
                                ;;*****************************************************************
013640                          TST31:
013640  004567  004730                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
013644  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
013646  004467  000520                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013652  012700  052525                  MOV     #052525,R0      ;SETUP DATA PATTERN
013656  010022                  1$:     MOV     R0,     (R2)+   ;WRITE A WORD
013660  005100                          COM     R0
013662  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013664  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
013666  004767  001256                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
013672  005003                          CLR     R3              ;SET COUNTER FOR LOOP
013674  012704  000046                  MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
013700  005303                  2$:     DEC     R3
013702  001376                          BNE     2$
013704  005304                          DEC     R4
013706  001374                          BNE     2$
013710  004467  000456                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
013714  012700  052525                  MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
013720                          3$:
013720  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
013722  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
013724  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
013726  004767  004400          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
013732  004767  005646                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
013736  000006                          .WORD   6               ;ERROR TYPE CODE.
013740                          65$:
013740  005100                          COM     R0
013742  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
013744  001365                          BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
013746  004767  001176                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
                                      .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
 013752                               DONE:
 013752  004567  004616                        JSR     R5,     $SCOPE   ;GO TO SCOPE ROUTINE.
 013756  000000                                .WORD   0                ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 013760  005067  165204               TST32:   CLR     $TIMES           ;RESET ITERATION COUNTER FOR RESTARTING TEST.
 013764  105067  165112                        CLRB    $TSTNM           ;RESET TEST NUMBER.
 013770  036767  164606  165536       1$:      BIT     PRGMAP, SAVTST   ;CHECK IF PROGRAM IS IN TEST AREA.
 013776  001004                                BNE     2$               ;BR IF IT PROG IN MEM TO BE TESTED.
 014000  036767  164600  165530                BIT     PRGMAP+2,SAVTST+2 ;CHECK HI 64K
 014006  001434                                BEQ     $EOP             ;BR IF PROG NOT IN MEM TO BE TESTED.
 014010  032777  000200  165122       2$:      BIT     #SW07,  @SWR     ;CHECK FOR INHIBIT RELOCATION SWITCH.
 014016  001030                                BNE     $EOP             ;SKIP RELOCATION IF SWITCH SET.
 014020  022767  000003  164554                CMP     #3,     PRGMAP   ;CHECK IF PROGRAM IN FIRST 8K.
 014026  001012                                BNE     4$               ;BR IF NOT IN FIRST 8K.
 014030  005737  000042                        TST     @#42             ;CHECK FOR A ACT11.
 014034  001014                                BNE     5$               ;BR IF A ACT11.
 014036  105737  001224                        TSTB    @#$ENV           ;CHECK FOR APT
 014042  001011                                BNE     5$               ;IF APT DO NOT RELOCATE
 014044  004767  002354                        JSR     PC,     RELTOP   ;RELOCATE PROGRAM TO TOP OF MEMORY.
 014050  000167  172004               3$:      JMP     START1           ;LOOP BACK AND RUN ALL TESTS AGAIN.

 014054  004767  002746               4$:      JSR     PC,     RELO     ;RELOCATE PROGRAM BACK TO FIRST 8K.
 014060  005737  000042                        TST     @#42             ;TEST FOR XXDP
 014064  001402                                BEQ     6$               ;IF NOT RUNNING UNDER MON. DONT
 014066  004767  003142               5$:      JSR     PC,     RESLDR   ;RESTORE LOADERS.
 014072                               6$:
 014072  004567  007360                        JSR     R5,     $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
 014076  001201                                .WORD   $CRLF            ;ADDRESS OF MESSAGE TO BE TYPED
```

```
                                        ;;*************************************************************

                                        .SBTTL  END OF PASS ROUTINE

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

 014100                                 $EOP:
 014100  000240                                 NOP
 014102  005067  165062                         CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
 014106  005267  165100                         INC     $PASS           ;;INCREMENT THE PASS NUMBER
 014112  042767  100000  165072                 BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
 014120  005327                                 DEC     (PC)+           ;;LOOP?
 014122  000001                         $EOPCT: .WORD   1
 014124  003035                                 BGT     $DOAGN          ;;YES
 014126  012737                                 MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
 014130  000001                         $ENDCT: .WORD   1
 014132  014122                                 $EOPCT
 014134  004567  007316                         JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 014140  014224                                 .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
 014142  016746  165044                         MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 014146  013746  177776                         MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 014152  004767  010220                         JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
 014156  004567  007274                         JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 014162  014241                                 .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
 014164                                 $GET42:

 014164  016700  163652                         MOV     42,     R0              ;;GET MONITOR ADDRESS
 014170  001413                                 BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
 014172  000005                                 RESET                   ;;CLEAR THE WORLD
 014174  004710                         $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
 014176  000240                                 NOP                     ;;SAVE ROOM
 014200  000240                                 NOP                     ;;FOR
 014202  000240                                 NOP                     ;;ACT11
 014204  023737  000042  000046                 CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
 014212  001402                                 BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
 014214  004767  003074                         JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
 014220                                 $DOAGN:
 014220  000167  171634                         JMP     START1          ;;RETURN*****
 014224  005015  047105  020104         $ENDMG: .ASCIZ  <15><12>/END PASS #/
 014232  040520  051523  021440
 014240     000
 014241     377     377     000         $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
                                        .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
                                        .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
                                        ;;*************************************************************
                                        ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
                                        ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
                                        ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
                                        ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
                                        ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                        ;;*************************************************************
```

```
014244                                  MMINIT:
014244  012737  077406  172300                  MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
014252  012737  077406  172302                  MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
014260  012737  077406  172304                  MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
014266  012737  077406  172306                  MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
014274  005037  172310                          CLR     @#KIPDR4
014300  005037  172312                          CLR     @#KI°DR5
014304  005037  172314                          CLR     @#KIPDR6
014310  012737  077406  172316                  MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
014316  005037  172340                          CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
014322  012737  000200  172342                  MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
014330  005037  172344                          CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
014334  005037  172346                          CLR     @#KIPAR3
014340  005037  172350                          CLR     @#KIPAR4
014344  005037  172352                          CLR     @#KIPAR5
014350  005037  172354                          CLR     @#KIPAR6
014354  012737  007600  172356                  MOV     #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
014362  012737  000001  177572                  MOV     #1,     @#SR0   ;ENABLE MEMORY MANAGEMENT
014370  000207                                  RTS     PC              ;RETURN


                                        ;;*************************************************************
                                        ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
                                        ;;*************************************************************
014372  012767  000001  165144          INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
014400  005067  165142                          CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
014404  005002                                  CLR     R2              ;SET ADDRESS POINTER TO 0
014406  016705  165174                          MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
014412  005767  164170                          TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
014416  001514                                  BEQ     10$             ;BRANCH IF NO MEM MGMT
014420  C05037  172344                          CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
014424  012702  040000                          MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
014430  036767  165110  165072          1$:     BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
014436  0( 1015                                 BNE     2$              ;BRANCH IF MATCH
014440  036767  165102  165064                  BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
014446  001011                                  BNE     2$              ;BRANCH IF MATCH
014450  062737  000200  172344                  ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
014456  006367  165062                          ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
014462  006167  165060                          ROL     BITPT+2         ;...HI POINTER.
014466  100360                                  BPL     1$              ;BR IF MORE.
014470  000000                                  HALT            ;FATAL ERROR!!!  NO 4K BANK FOUND?
014472  036767  165046  165102          2$:     BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
014500  001004                                  BNE     3$              ;BR IF LAST BANK.
014502  036767  165040  165074                  BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
014510  001405                                  BEQ     4$              ;BR IF NOT LAST BANK.
014512  016705  165062                  3$:     MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
014516  042767  020000  165052                  BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
014524  013737  172344  172346          4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
014532  016767  165006  165010                  MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
014540  016767  165002  165004                  MOV     BITPT+2,TMPPT+2 ;...HI 64K.
014546  032705  020000                          BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
014552  001505                                  BEQ     21$             ;BRANCH IF NOT 8K.
014554  062737  000200  172346          5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
014562  006367  164762                          ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
014566  006167  164760                          ROL     TMPPT+2         ;...HI POINTER.
014572  100473                                  BMI     20$             ;BR IF NO MORE.
```

```
014574  036767  164750  164726            BIT     TMPPT,   TSTMAP   ;CHECK IF BANK TO BE TESTED.
014602  001004                            BNE     6$                ;BRANCH IF A MATCH.
014604  036767  164742  164720            BIT     TMPPT+2,TSTMAP+2  ;CHECK FOR HI 64K BANKS.
014612  001760                            BEQ     5$                ;BRANCH IF NO MEMORY
014614  036767  164730  164760  6$:       BIT     TMPPT,   LADMAP   ;CHECK IF LAST BANK.
014622  001004                            BNE     7$                ;BRANCH IF A MATCH
014624  036767  164722  164752            BIT     TMPFT+2,LADMAP+2  ;CHECK HI 64K
014632  001455                            BEQ     21$               ;BR IF NOT LAST BANK.
014634  016705  164740          7$:       MOV     LADMSK, R5        ;RESET MASK TO FIND LAST ADR.
014640  052767  020000  164730            BIS     #20000, TMPLAD    ;MAKE SURE LAST ADDRESS IS IN BANK 3.
014646  000447                            BR      21$               ;BR TO FINISH UP.

014650  036767  164670  164652  10$:      BIT     BITPT,   TSTMAP   ;CHECK IF THIS BANK TO BE TESTED.
014656  001006                            BNE     11$               ;BR IF MATCH.
014660  062702  020000                    ADD     #20000, R2        ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
014664  106367  164654                    ASLB    BITPT             ;UPDATE BANK POINTER TO NEXT BANK.
014670  100367                            BPL     10$               ;BR IF MORE BANKS.
014672  000000                            HALT             ;FATAL ERROR!!!  NO 4K BANK FOUND?
014674  016767  164644  164646  11$:      MOV     BITPT, TMPPT      ;COPY BANK POINTER.
014702  036767  164636  164672            BIT     BITPT,   LADMAP   ;CHECK IF LAST BANK.
014710  001021                            BNE     12$               ;BR IF LAST BANK.
014712  032705  020000                    BIT     #BIT13, R5        ;CHECK FOR 8K BLOCK SIZE.
014716  001423                            BEQ     21$               ;BRANCH IF SMALLER BLOCK SIZE.
014720  106367  164624                    ASLB    TMPPT             ;POINT TO NEXT BANK.
014724  100416                            BMI     20$               ;BRANCH IF OVERFLOW.
014726  036767  164616  164574            BIT     TMPPT,   TSTMAP   ;CHECK IF BANK TO BE TESTED.
014734  001412                            BEQ     20$               ;BRANCH IF NOT TO BE TESTED.
014736  112767  000011  164613            MOVB    #11,     FLAG8K   ;SET 8K BLOCK SIZE FLAG.
014744  036767  164600  164630            BIT     TMPPT,   LADMAP   ;CHECK FOR LAST BANK.
014752  0C1403                            BEQ     20$               ;BR IF NOT LAST BANK.
014754  0 6705  164620          12$:      MOV     LADMSK, R5        ;RESET MASK TO FIND LAST ADR.
014760  0)0402                            BR      21$               ;SKIP MASK RESET.
014762  0[2705  017777          20$:      MOV     #MASK4K,R5        ;RESET MASK TO 4K BLOCK SIZE.
014766  0;6767  164552  164554  21$:      BIS     BITPT,   TMPPT    ;SET TMPPT FOR FLAGING LAST BANK.
014774  (56767  164546  164550            BIS     BITPT+2,TMPPT+2
015002  J36767  164536  164560            BIT     BITPT,   FADMAP   ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
015010  001004                            BNE     22$               ;BR IF FIRST BANK.
015012  036767  164530  164552            BIT     BITPT+2,FADMAP+2  ;CHECK HI 64K.
015020  001450                            BEQ     INITEX            ;BR IF NOT FIRST BANK.
015022  016702  164536          22$:      MOV     TMPFAD, R2        ;RESET ADDRESS POINTER TO FIRST ADR.
015026  000445                            BR      INITEX

015030  016705  164552          INITDN:   MOV     BLKMSK, R5        ;RESET R5 TO CURRENT BLOCK MASK.
015034  005002                            CLR     R2                ;INIT ADDRESS POINTER.
015036  005767  163544                    TST     MMAVA             ;CHECK FOR MEM MGMT
015042  001411                            BEQ     31$               ;BRANCH IF NO MEM MGMT
015044  012767  100000  164474            MOV     #BIT15, BITPT+2   ;SET POINTER TO TOP BIT
015052  005067  164466                    CLR     BITPT
015056  012737  007600  172344            MOV     #7600,   @#KIPAR2 ;SET PAR TO TOP OF MEM
015064  000403                            BR      32$               ;BRANCH TO COMMON AREA

015066  012767  000400  164450  31$:      MOV     #BIT8,   BITPT    ;SET UP BANK POINTER
015074  012767  015116  164452  32$:      MOV     #33$,    MMORE    ;SET "MMDOWN" EXIT ADDRESS.
015102  066767  163472  164444            ADD     RELOCF,  MMORE    ;ADD OFFSET
015110  004767  000524                    JSR     PC,      MMDOWN   ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
015114  000000                            HALT             ;FATAL ERROR!!!  NO MEM INDICATED IN MEM MAP ABOVE 8K!
```

```
015116  036767  164422  164456  33$:    BIT     BITPT,  LADMAP  ;CHECK FOR NON BOUNDRY LAST ADDR.
015124  001004                          BNE     34$             ;BR IF LAST BANK FLAG FOUND.
015126  036767  164414  164450          BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
015134  001402                          BEQ     INITEX          ;BR IF NO LAD FLG FOUND.
015136  016702  164432          34$:    MOV     LSTADR, R2      ;SET UP R2.
015142  010467  164406          INITEX: MOV     R4,     MMORE   ;PUT RETURN PC INTO "MMORE"
015146  000204                          RTS     R4              ;RETURN

                                        ;;***********************************************************
                                        ;*  COMMON UPWARDS ADDRESSING ROUTINE
                                        ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
                                        ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
                                        ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
                                        ;;***********************************************************
015150  036767  164374  164424  MMUP:   BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
015156  001122                          BNE     10$             ;BR IF LAST BANK.
015160  036767  164366  164416          BIT     TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
015166  001116                          BNE     10$             ;BR IF LAST BANK.
015170  016705  164412                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
015174  005767  163406                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
015200  001515                          BEQ     20$             ;BRANCH IF NO MEM MGMT
015202  012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
015206  062737  000200  172344  1$:     ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
015214  006367  164324                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
015220  006167  164322                  ROL     BITPT+2         ;...HI POINTER.
015224  100577                          BMI     32$             ;BR IF ALL DONE.
015226  036767  164312  164274          BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK EXISTS
015234  001004                          BNE     2$              ;BRANCH IF MATCH
015236  036767  164304  164266          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
015244  001760                          BEQ     1$              ;BRANCH IF NO MATCH
015246  036767  164272  164326  2$:     BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
015254  001004                          BNE     3$              ;BRANCH IF LAST BANK FLAG.
015256  036767  164264  164320          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
015264  001405                          BEQ     4$              ;BR IF NOT LAST BANK.
015266  016705  164306          3$:     MOV     LADMSK, R5      ;RESET MASK.
015272  042767  020000  164276          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
015300  016767  164240  164242  4$:     MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
015306  016767  164234  164236          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
015314  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
015320  001530                          BEQ     31$             ;BRANCH IF NOT.
015322  013737  172344  172346          MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
015330  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
015336  006367  164206                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
015342  006167  164204                  ROL     TMPPT+2         ;...HI POINTER.
015346  100513                          BMI     30$             ;BR IF NO MORE.
015350  036767  164174  164152  6$:     BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
015356  001004                          BNE     7$              ;BRANCH IF A MATCH.
015360  036767  164166  164144          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
015366  001760                          BEQ     5$              ;BRANCH IF NO MEMORY
015370  036767  164154  164204  7$:     BIT     TMPPT,LADMAP    ;CHECK FOR LAST BANK FLAG.
015376  001004                          BNE     8$              ;BRANCH IF A MATCH
015400  036767  164146  164176          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
015406  001475                          BEQ     31$             ;BR IF NO LAST BANK FLAG.
015410  016705  164164          8$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADDRESS.
015414  052767  02000C  164154          BIS     #20000, TMPLAD  ;SET VIRTUAL ADR TO BANK 3.
015422  000467                          BR      31$
```

```

015424  026702  164146          10$:     CMP     TMPLAD, R2        ;CHECK IF LAST ADR REACHED.
015430  001064                           BNE     31$               ;BR IF MORE.
015432  000474                           BR      32$               ;BR IF ALL DONE.

015434  106267  164117          20$:     ASRB    FLAG8K            ;SHIFT 8K FLAG
015440  001407                           BEQ     22$               ;BR IF NOT 8K BLOCK.
015442  103455                           BCS     30$               ;BR IF ANOTHER 4K.
015444  105067  164107                   CLRB    FLAG8K            ;CLEAR OUT ALL FLAGS.
015450  162702  040000                   SUB     #40000, R2        ;BACK UP 8K.
015454  062702  020000          21$:     ADD     #20000, R2        ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
015460  106367  164060          22$:     ASLB    BITPT             ;UPDATE POINTER.
015464  100457                           BMI     32$               ;BRANCH WHEN END IS REACHED.
015466  036767  164052  164034           BIT     BITPT,   TSTMAP   ;CHECK IF THIS BANK EXISTS.
015474  001767                           BEQ     21$               ;BRANCH IF NO MATCH.
015476  036767  164042  164076           BIT     BITPT,   LADMAP   ;CHECK FOR LAST BANK FLAG.
015504  001402                           BEQ     23$               ;BR IF NO MATCH.
015506  016705  164066                   MOV     LADMSK, R5        ;RESET MASK TO FIND LAST ADR.
015512  016767  164026  164030  23$:     MOV     BITPT,   TMPPT    ;SET UP TMP POINTER.
015520  032705  020000                   BIT     #BIT13, R5        ;CHECK FOR 8K BLOCK SIZE.
015524  001426                           BEQ     31$               ;BRANCH IF SMALLER BLOCK SIZE.
015526  106367  164016                   ASLB    TMPPT             ;POINT TO NEXT BANK.
015532  100421                           BMI     30$               ;BRANCH IF OVERFLOW.
015534  036767  164010  163766           BIT     TMPPT,   TSTMAP   ;CHECK IF BANK TO BE TESTED.
015542  001415                           BEQ     30$               ;BRANCH IF NOT TO BE TESTED.
015544  036767  163774  164030           BIT     BITPT,   LADMAP   ;CHECK FOR LAST BANK FLAG.
015552  112767  000011  163777           MOVB    #11,     FLAG8K   ;SET 8K BLOCK FLAG.
015560  036767  163760  164014           BIT     BITPT,   LADMAP   ;CHECK FOR LAST BANK FLAG.
015566  001403                           BEQ     30$               ;BR IF NO FLAG.
015570  016705  164004                   MOV     LADMSK, R5        ;RESET MASK TOFIND LAST ADR.
015574  000402                           BR      31$
015576  012705  017777          30$:     MOV     #MASK4K,R5        ;SET MASK TO 4K.
015602  056767  163736  163740  31$:     BIS     BITPT,   TMPPT    ;SET TMPPT FOR FINDING LAST ADR.
015610  056767  163732  163734           BIS     BITPT+2,TMPPT+2
015616  016716  163732                   MOV     MMORE,   (SP)     ;FUDGE RETURN ADDRESS TO LOOP.
015622  000207                           RTS     PC                ;RETURN
                                ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
015624  005767  164446          32$:     TST     MPRX              ;CHECK FOR ANY PARITY REGISTERS PRESENT.
015630  001402                           BEQ     33$               ;BR IF NONE.
015632  004767  001744                   JSR     PC,      CKPMER   ;CHECK FOR PARITY MEMORY ERRORS.
015636  000207                  33$:     RTS     PC                ;STRAIGHT RETURN.

                                ;;*************************************************************
                                ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
                                ;*  FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
                                ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS.
                                ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE..
                                ;;*************************************************************
015640  036767  163700  163722  MMDOWN:  BIT     BITPT,   FADMAP   ;CHECK FOR FIRST ADR FLAG.
015646  001004                           BNE     1$                ;BR IF FIRST ADR IN THIS BANK.
015650  036767  163672  163714           BIT     BITPT+2,FADMAP+2  ;CHECK FOR FIRST ADR FLAG.
015656  001404                           BEQ     2$                ;BR IF NO FLAG
015660  026702  163700          1$:      CMP     TMPFAD, R2        ;CHECK IF FIRST ADDRESS REACHED.
015664  001052                           BNE     9$                ;BR IF MORE.
015666  000453                           BR      10$               ;BR IF ALL DONE.
015670  005767  162712          2$:      TST     MMAVA             ;CHECK IF MEM MGMT IS AVAILABLE
```

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

```
                                .SBTTL  SUBROUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
                                ;;*************************************************************
                                ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN RO (BOTTOM 16 BITS).
                                ;*  BITS 16 AND 17 ARE IN $TMPO.
                                ;;*************************************************************
016020  010200                  PHYADR: MOV     R2,     RO      ;VITRUAL INTO RO
016022  005067  163132                  CLR     $TMFO           ;CLEAR TEMP SAVE OF HIGH BITS
016026  005767  162554                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
016032  001417                          BEQ     1$              ;BRANCH IF NO MEM MGMT
016034  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016036  013701  172344                  MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
016042  006301                          ASL     R1              ;SHIFT IT 6 TIMES
016044  006301                          ASL     R1
016046  006301                          ASL     R1
016050  006301                          ASL     R1
016052  006301                          ASL     R1
016054  006167  163100                  ROL     $TMPO           ;SAVE EXTRA BITS
016060  006301                          ASL     R1
016062  006167  163072                  ROL     $TMPO
016066  060100                          ADD     R1,     RO      ;ADD SHIFTED PAR TO VIRTUAL
016070  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016072  000207                  1$:     RTS     PC              ;RETURN

                                ;;*************************************************************
                                ;*  SUBROUTINE TO PUT BANK NUMBER INTO RO.
                                ;;*************************************************************
016074  005000                  BANKNO: CLR     RO              ;INIT RO
016076  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016100  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016102  016701  163436                  MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
016106  016702  163434                  MOV     BITPT+2,R2      ;...HI 64K.
016112  006202                  1$:     ASR     R2              ;SHIFT POINTER...HI
016114  006001                          ROR     R1              ;...LO
016116  103403                          BCS     2$              ;BR WHEN POINTER FOUND.
016120  105200                          INCB    RO              ;COUNT BANKS.
016122  100373                          BPL     1$              ;BR IF NOT OVERFLOW.
016124  000000                          HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
016126                          2$:
016126  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
016130  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
016132  000207                          RTS     PC              ;RETURN

                                ;;*************************************************************
                                ;*  SUBROUTINE TO WRITE THE CONSTANT IN RO INTO ALL OF MEMORY.
                                ;;*************************************************************
016134                          SETCON:
016134  004467  176232                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
016140  010022                  2$:     MOV     RO,     (R2)+   ;MOV CONSTANT INTO MEMORY
016142  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
016144  001375                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
016146  004767  176776                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
016152  000207                          RTS     PC              ;RETURN
```

```
                                ;;***************************************************************
                                ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
                                ;;***************************************************************
016154  106112                  ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
016156  106112                          ROLB    (R2)            ;(R2)=177775 OR 000002
016160  106112                          ROLB    (P2)            ;(R2)=177773 OR 000004
C16162  106112                          ROLB    (R2,            ;(R2)=17776, OR 000010
016164  106112                          ROLB    (R2)            ;(R2)=177757 OR 000020
016166  106112                          ROLB    (R2)            ;(R2)=177737 OR 000040
016170  106112                          ROLB    (R2)            ;(R2)=177677 OR 000100
016172  106112                          ROLB    (R2)            ;(R2)=177777 OR 000000
016174  106122                          ROLB    (R2)+           ;(R2)=177577 OR 000200
016176  106112                          ROLB    (R2)            ;(R2)=177377 OR 000400
016200  106112                          ROLB    (R2)            ;(R2)=176777 OR 001000
016202  106112                          ROLB    (R2)            ;(R2)=175777 OR 002000
016204  106112                          ROLB    (R2)            ;(R2)=173777 OR 004000
016206  106112                          ROLB    (R2)            ;(R2)=167777 OR 010000
016210  106112                          ROLB    (R2)            ;(R2)=157777 OR 020000
016212  106112                          ROLB    (R2)            ;(R2)=137777 OR 040000
016214  106112                          ROLB    (R2)            ;(R2)=077777 OR 100000
016216  106122                          ROLB    (R2)+           ;(R2)=177777 OR 000000
016220  000207                          RTS     PC              ;RETURN

                                ;;***************************************************************
                                ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
                                ;;***************************************************************
016222  012704  000020          W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS

016226  010022                  2$:     MOV     R0,(R2)+
016230  010022                          MOV     R0,(R2)+
016232  010022                          MOV     R0,(R2)+
016234  010022                          MOV     R0,(R2)+

016236  010322                          MOV     R3,(R2)+
016240  010322                          MOV     R3,(R2)+
016242  010322                          MOV     R3,(R2)+
016244  010322                          MOV     R3,(R2)+

016246  010022                          MOV     R0,(R2)+
016250  010022                          MOV     R0,(R2)+
016252  010022                          MOV     R0,(R2)+
016254  010022                          MOV     R0,(R2)+

016256  010322                          MOV     R3,(R2)+
016260  010322                          MOV     R3,(R2)+
016262  010322                          MOV     R3,(R2)+
016264  010322                          MOV     R3,(R2)+

016266  005304                          DEC     R4
016270  001356                          BNE     2$
016272  010046                          MOV     R0,     -(SP)   ;SAVE R0
016274  010300                          MOV     R3,     R0      ;PUT R3 INTO R0
016276  012603                          MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
016300  000207                          RTS     PC              ;RETURN
```

```
                                        .SBTTL  RELOCATION SUBROUTINES.
                                        ;;*************************************************************
                                        ;* ROUTINE TO RELOCATE PROGRAM CODE
                                        ;;*************************************************************
016302                                  RELOC:
016302  010246                                  MOV     R2,-(SP)                ;;PUSH R2 ON STACK
016304  010346                                  MOV     R3,-(SP)                ;;PUSH R3 ON STACK
016306  010446                                  MOV     R4,-(SP)                ;;PUSH R4 ON STACK
016310  012502                          4$:     MOV     (R5)+,  R2              ;GET FIRST LOCATION.
016312  012503                                  MOV     (R5)+,  R3              ;GET FIRST LOCATION OF DESTINATION.
016314  012704  020000                          MOV     #20000, R4              ;SET UP 8K COUNTER.
016320  012223                          1$:     MOV     (R2)+,  (R3)+           ;MOV THE DATA.
016322  005304                                  DEC     R4                      ;COUNT THE WORDS.
016324  001375                                  BNE     1$                      ;BR IF MORE.
016326  012704  020000                          MOV     #20000, R4              ;RESET THE COUNTER.
016332  024243                          2$:     CMP     -(R2),  -(R3)           ;CHECK THE DATA JUST MOVED.
016334  001417                                  BEQ     3$                      ;BR IF DATA OK.
016336  011267  162562                          MOV     (R2),   $GDDAT          ;GET SOURCE DATA.
016342  011367  162560                          MOV     (R3),   $BDDAT          ;GET DESTINATION DATA.
016346  010267  162546                          MOV     R2,     $GDADR          ;GET SOURCE ADDRESS.
016352  010367  162544                          MOV     R3,     $BDADR          ;GET DESTINATION ADDRESS.
016356  004767  003222                          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
016362  000023                                  .WORD   23                      ;ERROR TYPE CODE.
016364  000000                                  HALT            ;FATAL ERROR!!!  RELOCATION FAILED.
016366  162705  000004                          SUB     #4,     R5              ;ADJUST RETURN POINTER.
016372  000746                                  BR      4$                      ;GO BACK AND TRY AGAIN.
0'6374  005304                          3$:     DEC     R4                      ;COUNT WORDS.
016376  001355                                  BNE     2$                      ;BR IF MORE.
01.400  004567  005052                          JSR     R5,     $PRINT          ;GO PRINT OUT THE FOLLOWING MESSAGE.
016404  026506                                  .WORD   PRELOC                  ;ADDRESS OF MESSAGE TO BE TYPED
                                                                                ;"PROGRAM RELOCATED TO "
016406  010346                                  MOV     R3,     -(SP)           ;PUT THE DATA ON THE STACK.
016410  004767  006502                          JSR     PC,     $TYPAD          ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
016414  012604                                  MOV     (SP)+,R4                ;;POP STACK INTO R4
016416  012603                                  MOV     (SP)+,R3                ;;POP STACK INTO R3
016420  012602                                  MOV     (SP)+,R2                ;;POP STACK INTO R2
016422  000205                                  RTS     R5                      ;RETURN
                                        ;;*************************************************************
                                        ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
                                        ;;*************************************************************
016424  022767  000003  162150          RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
016432  001401                                  BEQ     1$                      ;BR IF OK
016434  000000                                  HALT            ;FATAL EPROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
016436                                  1$:
016436  010046                                  MOV     R0,-(SP)                ;;PUSH R0 ON STACK
016440  010146                                  MOV     R1,-(SP)                ;;PUSH R1 ON STACK
016442  005767  162140                          TST     MMAVA
016446  001465                                  BEQ     10$
016450  012737  007600  172346                  MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
016456  005000                                  CLR     R0              ;INIT BANK POINTER...LO 64K
016460  012701  100000                          MOV     #BIT15, R1      ;...HI 64K.
016464  162737  000200  172346  2$:     SUB     #200,   @#KIPAR3        ;BACK DOWN ONE BANK.
016472  006001                                  ROR     R1                      ;MOVE POINTER...HI 64K.
016474  006000                                  ROR     R0                      ;..LO 64K.
016476  103500                                  BCS     90$
016500  030167  163022                          BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
3523 016504 001003                          BNE     3$                 ;BR IF AVAILABLE
3524 016506 030067 163012                   BIT     R0,      MEMMAP    ;CHECK FOR BANK EXISTS.
3525 016512 001764                          BEQ     2$                 ;BR IF NO BANK FOUND.
3526 016514 013737 172346 172344   3$:      MOV     @#KIPAR3,@#KIPAR2  ;COPY PAR
3527 016522 010046                          MOV     R0,-(SP)           ;;PUSH R0 ON STACK
3528 016524 010146                          MOV     R1,-(SP)           ;;PUSH R1 ON STACK
3529 016526 162737 000200 172344   4$:      SUB     #200,    @#KIPAR2  ;BACK DOWN WITH LOW PAR.
3530 016534 006001                          ROR     R1                 ;SHIFT POINTER.
3531 016536 006000                          ROR     R0                 ;...LO 64K.
3532 016540 103457                          BCS     90$                ;BR IF OVERFLOW.
3533 016542 030167 162760          5$:      BIT     R1,      MEMMAP+2  ;       CHECK IF BANK EXISTS...HI 64K.
3534 016546 001003                          BNE     6$                 ;BR IF BANK EXISTS.
3535 016550 030067 162750                   BIT     R0,      MEMMAP    ;CHECK IF BANK EXISTS...LO 64K.
3536 016554 001764                          BEQ     4$                 ;BR IF BANK DOESN'T EXIST.
3537 016556 052601                 6$:      BIS     (SP)+,   R1        ;GET SECOND BANK POINTER.
3538 016560 052600                          BIS     (SP)+,   R0        ;....0 64K.
3539 016562 030067 162014                   BIT     R0,      PRGMAP    ;CHECK FOR CONFLICT.
3540 016566 001044                          BNE     90$                ;ABORT IF DESTINATION OVERLAYS SOURCE.
3541 016570 004567 177506                   JSR     R5,      RELOC     ;GO RELOCATE PROGRAM.
3542 016574 000000                          .WORD   0                  ;SOURCE FIRST ADDRESS.
3543 016576 040000                          .WORD   40000              ;DESTINATION FIRST ADDRESS.
3544 016600 013737 172344 172340            MOV     @#KIPAR2,@#KIPAR0  ;RELOCATE LO BANK
3545 016606 013737 172346 172342            MOV     @#KIPAR3,@#KIPAR1  ;RELOCATE HI BANK.
3546                               ;*  PROGRAM SHOULD NOW BE EXICUTING OUT OF LAST TWO BANKS OF MEMORY.
3547 016614 010167 161764                   MOV     R1,      PRGMAP+2  ;RESET PROGRAM MAP.
3548 016620 000473                          BR      30$                ;BR TO COMMON EXIT.
3549
3550 016622 012700 000400          10$:     MOV     #BIT8,   R0        ;SET BANK POINTER TO TOP OF MEM.
3551 016626 005001                          CLR     R1                 ;SET ADDRESS POINTER TO TOP.
3552 016630 162701 020000          11$:     SUB     #20000,  R1        ;BACK DOWN ONE BANK.
3553 016634 006200                          ASR     R0                 ;MOVE POINTER DOWN ONE BANK.
3554 016636 103420                          BCS     90$                ;BR IF OVERFLOW.
3555 016640 030067 162660                   BIT     R0,      MEMMAP    ;CHECK IF THIS BANK EXISTS.
3556 016644 001771                          BEQ     11$                ;BR IF NON-EXISTANT BANK.
3557 016646 162701 020000                   SUB     #20000,  R1        ;BACK DOWN TO NEXT BANK.
3558 016652 006200                          ASR     R0                 ;MOV POINTER DOWN ONE BANK.
3559 016654 103411                          BCS     90$                ;BR IF OVERFLOW.
3560 016656 030067 162642                   BIT     R0,      MEMMAP    ;CHECK IF THIS BANK EXISTS.
3561 016662 001762                          BEQ     11$                ;BR TO START OVER IF NO LOWER BANK.
3562 016664 010046                          MOV     R0,      -(SP)     ;SAVE THE POINTER.
3563 016666 006300                          ASL     R0                 ;RESET POINTER TO HI BANK.
3564 016670 052600                          BIS     (SF)+,   R0        ;SET BIT FOR LO BANK.
3565 016672 030067 161704                   BIT     R0,      PRGMAP    ;CHECK FOR A PROGRAM CONFLICT.
3566 016676 001401                          BEQ     12$                ;BR IF NO CONFLICT.
3567 016700                            90$:
3568 016700 000000                          HALT                       ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
3569 016702 010167 000006          12$:     MOV     R1,      13$       ;SET DATA FOR RELOCATION SUBROUTINE.
3570 016706 004567 177370                   JSR     R5,      RELOC     ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
3571 016712 000000                          .WORD   0                  ;SOURCE STARTING ADDRESS.
3572 016714 000000                 13$:     .WORD   0                  ;DESTINATION STARTING ADDRESS.
3573 016716 010167 161656                   MOV     R1,      RELOCF    ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
3574 016722 060107                          ADD     R1,      PC        ;JUMP TO RELOCATED PROGRAM
3575                               ;*  PROGRAM NOW EXICUTING OUT OF TOP OF MEMORY.
3576 016724 060106                          ADD     R1,      SP        ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
3577 016726 010167 161646                   MOV     R1,      RELOCF    ;SET THE RELOCATION FACTOR.
3578 016732 060137 000004                   ADD     R1,      @#ERRVEC  ;ADJUST ERROR VECTOR.
```

```
016736  060137  000024                  ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
016742  060137  000114                  ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
016746  026727  162166  177570          CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
016754  001404                          BEQ     14$             ;BR IF HARDWARE SWITCH REGISTER.
016756  060167  162156                  ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
016762  060167  162154                  ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
016766  062701  001622          14$:    ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
016772  066721  161602          15$:    ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
016776  005721                  16$:    TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
017000  001776                          BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
017002  024127  177777                  CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
017006  001371                          BNE     15$             ;BR IF MORE IN TABLE.
017010  010067  161566          30$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
017014  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
017016  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
017020  066716  161554                  ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
017024  000207                          RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
                                ;;*****************************************************************
017026  032767  000003  161546  RELO:   BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
017034  001401                          BEQ     1$              ;BR IF NO CONFLICT.
017036  000000                          HALT            ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1!!!!
017040  005767  161542          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
017044  001417                          BEQ     10$             ;BR IF NO MEMMGMT.
017046  005037  172344                  CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
017052  012737  000200  172346          MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
017060  004567  177216                  JSR     R5,     RELOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
017064  000000                          .WORD   0               ;SOURCE STARTING ADDRESS.
017066  040000                          .WORD   40000           ;DESTINATION STARTING ADDRESS.
017070  005037  172340                  CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
017074  012737  000200  172342          MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
                                ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
017102  000444                          BR      30$             ;BR TO COMMON EXIT.

017104  016746  161470          10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
017110  011667  000004                  MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
017114  004567  177162                  JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
017120  000000                  20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
017122  000000                          .WORD   0               ;DESTINATION STARTING ADDRESS.
017124  161607                          SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
                                ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
017126  161606                          SUB     (SP),   SP      ;RESET THE STACK POINTER.
017130  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
017132  012700  001622                  MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
017136  166620  000002          21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
017142  005720                  22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
017144  001776                          BEQ     22$             ;BR OVER TERMINATORS.
017146  024027  177777                  CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
017152  001371                          BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
017154  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
017156  161637  000004                  SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
017162  161637  000024                  SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
017166  161637  000114                  SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
017172  026727  161742  177570          CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
```

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

                                  ;;***********************************************************
                                  ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
                                  ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
                                  ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
                                  ;;***********************************************************
017234  016700  162260            RESLDR: MOV     LMAD,     R0         ;CHECK IF THE LOADERS WERE SAVED.
017240  001001                            BNE     1$                   ;BR IF LOADER AREA WAS SAVED.
017242  000000                            HALT              ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
017244  005767  161336            1$:     TST     MMAVA                ;CHECK FOR MEM MGMT.
017250  001402                            BEQ     2$                   ;SKIP IF NO MEM MGMT.
017252  005037  177572                    CLR     @#SR0                ;DISABLE MEM MGMT.
017256  012701  040000            2$:     MOV     #40000,   R1         ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
017262  012702  002734                    MOV     #1500.,   R2         ;GET COUNTER.
017266  014140                    3$:     MOV     -(R1),    -(R0)      ;MOVE THE LOADER AREA.
017270  005302                            DEC     R2                   ;COUNT HOW LONG THE AREA IS.
017272  001375                            BNE     3$                   ;BR IF NOT MORE TO MOVE.
017274  005067  162220                    CLR     LMAD                 ;CLEAR MONITOR SAVED FLAG
017300  005767  161302                    TST     MMAVA                ;CHECK FOR MEM MGMT.
017304  001402                            BEQ     4$                   ;BR IF NO MEM MGMT.
017306  005237  177572                    INC     @#SR0                ;ENABLE MEM MGMT.
017312  000207                    4$:     RTS     PC                   ;RETURN.

                                  ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
017314  005767  162200            SAVLDR: TST     LMAD                 ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
017320  001024                            BNE     4$                   ;BRANCH IF ALREADY SAVED
017322  012700  040000                    MOV     #40000,   R0         ;GET END OF 8K
017326  010001                            MOV     R0,       R1         ;GET END OF 8K
017330  012737  017342  000004            MOV     #2$,  @#ERRVEC       ;SET UP TIMEOUT VECTOR
017336  011020                    1$:     MOV     (R0),     (R0)+      ;SEARCH FOR END OF MEMORY
017340  000776                            BR      1$                   ;KEEP SEARCHING
017342  022626                    2$:     CMP     (SP)+,    (SP)+      ;RESTORE STACK POINTER
017344  012737  025060  000004            MOV     #ERRTRP,@#ERRVEC     ;RESET TIMEOUT VECTOR.
017352  010046                            MOV     R0,       -(SP)      ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
017354  012702  002734                    MOV     #1500.,   R2         ;SET UP WORD COUNTER
017360  014041                    3$:     MOV     -(R0),    -(R1)      ;SAVE THE LOADERS
017362  005302                            DEC     R2                   ;COUNT THE WORDS
017364  001375                            BNE     3$                   ;BRANCH IF MORE WORDS
017366  012667  162126                    MOV     (SP)+,    LMAD       ;SAVE LAST MEMORY ADDRESS
017372  000207                    4$:     RTS     PC                   ;RETURN
```

```
                                           .SBTTL  PARITY MEMORY TRAF ..RVICE AND SUBROUTINES.
                                           ;;************************************************************
                                           ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
                                           ;*  FIND OUT WHICH REGISTER DETECTED THE ERROR,
                                           ;*  THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
                                           ;;************************************************************
 017374  011667  161522                    PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
 017400  004567  004052                            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 017404  026445                                    .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
                                                                           ;"UNEXPECTED MEMORY PARITY TRAP."
 017406  010146                                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 017410  010346                                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 017412  016703  162212                            MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
 017416  005733                            1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
 017420  100415                                    BMI     3$              BR IF THIS REGISTER SHOWS THE ERROR.
 017422  005713                                    TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 017424  001374                                    BNE     1$              ;BR IF MORE REGISTERS.
 017426  004767  002152                            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
                                           ;***ERROR***    NO REGISTER INDICATED ERROR
 017432  000024                                    .WORD   24              ;ERROR TYPE CODE.
 017434  000417                                    BR      4$              ;EXIT
 017436  005713                            2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 017440  001415                                    BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
 017442  005733                                    TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
 017444  100374                                    BPL     2$              ;BR IF NO ERROR FLAG.
 017446  004567  004004                            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 017452  026536                                    .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
                                                                           ;"MORE THAN ONE ERROR FOUND."
 017454                                    3$:
 017454  004767  000610                    64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
 017460  004767  002120                            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 017464  000025                                    .WORD   25              ;ERROR TYPE CODE.
 017466  004767  000216                            JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
 017472  000761                                    BR      2$              ;GO LOOK FOR MORE ERRORS.
 017474                                    4$:
 017474  012603                                    MOV     (SP)+,R3        ;;POP STACK INTO R3
 017476  012601                                    MOV     (SP)+,R1        ;;POP STACK INTO R1
 017500  000002                                    RTI                     ;RETURN.

                                           ;;************************************************************
                                           ;ROUTINE TO ENABLE PARITY ERR.R ACTION ON MA/MF PARITY MEMORIES
                                           ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                           ;;************************************************************
 017502  005767  162570                    MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
 017506  001434                                    BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
 017510  032777  000100  161422                    BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
 017516  001030                                    BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
 017520  005767  161054                            TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
 017524  001410                                    BEQ     SETAE           ;BR IF PROG IN BANK 0.
 017526  032777  000040  161404                    BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
 017534  001004                                    BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
 017536  026727  162020  001000                    CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
 017544  103415                                    BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.
```

```
017546  016737  162064  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
017554  005037  000116                  CLR     @#PARVEC+2       ;PRIORITY LEVEL 0 ON TRAP
017560  010346                          MOV     R3,-(SP)         ;;PUSH R3 ON STACK
017562  016703  162042                  MOV     .MPRX,  R3       ;GET PARITY REGISTER TABLE POINTER.
017566  052733  000001          MAMF1:  BIS     #AE,    @(R3)+   ;SET ACTION ENABLE BIT IN PARITY REG
017572  005713                          TST     (R3)             ;CHECK FOR END OF TABLE.
017574  001374                          BNE     MAMF1            ;BR IF MORE PARITY REGISTERS.
017576  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
017600  000207                  MAMF2:  RTS     PC               ;RETURN.

                                ;;************************************************************
                                ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
                                ;;************************************************************
017602  005767  162470          CKPMER: TST     MPRX             ;CHECK IF ANY PARITY REGISTERS EXIST.
017606  001437                          BEQ     4$               ;BR IF NO PARITY REGISTERS.
017610  032777  000100  161322          BIT     #SW6,   @SWR     ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
017616  001033                          BNE     4$               ;BR IF PARITY ERROR CHECKING INHIBITED.
017620  010346                          MOV     R3,-(SP)         ;;PUSH R3 ON STACK
017622  016703  162002                  MOV     .MPRX,  R3       ;GET PARITY REG TABLE POINTER.
017626  005733                  1$:     TST     @(R3)+           ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017630  100023                          BPL     3$               ;BR IF NO ERROR
017632  032773  000001  177776          BIT     #BIT0,  @-2(R3)  ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
017640  001010                          BNE     2$               ;BR IF NO ACTION ENABLE.
017642  004767  000422          64$:    JSR     PC,     SPRNTQ   ;SET UP VALUES FOR ERROR PRINTING.
017646  004767  001732                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
017652  000026                          .WORD   26               ;ERROR TYPE CODE.
017654  000411                          BR      3$
017656  004767  000026                  JSR     PC,     PSCAN    ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
017662                          2$:
017662  004767  000402          65$:    JSR     PC,     SPRNTQ   ;SET UP VALUES FOR ERROR PRINTING.
017666  004767  001712                  JSR     PC,     $ERROR   ;*** ERROR *** (GO TYPE A MESSAGE)
017672  000027                          .WORD   27               ;ERROR TYPE CODE.
017674  004767  000010                  JSR     PC,     PSCAN    ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
017700  005713                  3$:     TST     (R3)             ;CHECK FOR TABLE TERMINATOR.
017702  001351                          BNE     1$               ;BR IF MORE.
017704  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
017706  000207                  4$:     RTS     PC               ;RETURN.

                                ;;************************************************************
                                ;*  THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY,
                                ;*  TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
                                ;*  LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
                                ;;************************************************************
```

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

```
020130  000402                          BR      11$                     ;BR TO CHECK FOR NEXT BANK.
020132  062702  020000          10$:    ADD     #20000, R2              ;SKIP BANKS THAT AREN'T THERE.
020136  005767  160444          11$:    TST     MMAVA                   ;CHECK FOR MEM MGMT.
020142  001413                          BEQ     12$                     ;BR IF NO MEM MGMT.
020144  062737  000200  172344          ADD     #200,   @#KIPAR2        ;UPDATE MEM MGMT REG TO NEXT 4K.
020152  012702  040000                  MOV     #40000, R2              ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
020156  006300                          ASL     R0                      ;UPDATE BANK POINTER.
020160  006101                          ROL     R1                      ;...HI 64K.
020162  100313                          BPL     1$                      ;BR IF MORE BANKS.
020164  012637  172344                  MOV     (SP)+,@#KIPAR2            ;;POP STACK INTO @#KIPAR2
020170  000402                          BR      20$                     ;GO CHECK IF ANY ERRORS FOUND.
020172  106300                  12$:    ASLB    R0                      ;UPDATE POINTER TO NEXT BANK.
020174  100306                          BPL     1$                      ;BR IF MORE BANKS.
020176  005704                  20$:    TST     R4                      ;CHECK IF ANY PARITY ERRORS DETECTED.
020200  001003                          BNE     21$                     ;BR IF ERRORS DETECTED.
020202  004567  003250                  JSR     R5,     $PRINT          ;GO PRINT OUT THE FOLLOWING MESSAGE.
020206  025652                          .WORD   NOPES                   ;ADDRESS OF MESSAGE TO BE TYPED
020210                          21$:
020210  012637  000116                  MOV     (SP)+,@#116             ;;POP STACK INTO @#116
020214  012637  000114                  MOV     (SP)+,@#114             ;;POP STACK INTO @#114
020220  012604                          MOV     (SP)+,R4                ;;POP STACK INTO R4
020222  012603                          MOV     (SP)+,R3                ;;POP STACK INTO R3
020224  012602                          MOV     (S?)+,R2                ;;POP STACK INTO R2
020226  012601                          MOV     (SP)+,R1                ;;POP STACK INTO R1
020230  012600                          MOV     (SP)+,R0                ;;POP STACK INTO R0
020232  000207                          RTS     PC                      ;RETURN.

                                ;;**************************************************************
                                ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
                                ;;**************************************************************
020234                          CLRPAR:
020234  010346                          MOV     R3,-(SP)                ;;PUSH R3 ON STACK
020236  016703  161366                  MOV     .MPRX,  R3              ;GET PARITY REGISTER TABLE POINTER.
020242  005713                  1$:     TST     (R3)                    ;CHECK FOR THE TABLE TERMINATOR.
020244  001402                          BEQ     2$                      ;BR IF DONE ALL PARITY REGISTERS.
020246  005033                          CLR     @(R3)+                  ;CLEAR THE PARITY REGISTER.
020250  000774                          BR      1$                      ;BR FOR MORE
020252                          2$:
020252  012603                          MOV     (SP)+,R3                ;;POP STACK INTO R3
020254  000207                          RTS     PC                      ;RETURN.

                                .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
                                ;;**************************************************************
                                ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMT^})
                                ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
                                ;;**************************************************************
020256  010267  160636          $PRNT:  MOV     R2,     $GDADR          ;SAVE THE ADDRESS UNDER TEST.
020262  005067  160636                  CLR     $GDDAT                  ;SHOULD BE DATA IS "0".
020266  000430                          BR      SPRNTB

020270  014367  160664          SPRNTQ: MOV     -(R3),  $TMP0           ;GET THE PARITY REGISTER ADDRESS.
020274  013367  160662                  MOV     @(R3)+, $TMP1           ;GET THE CONTENTS OF THE PARITY REG.
020300  000402                          BR      SPRNT0

020302  011367  160652          SPRNTP: MOV     (R3),   $TMP0           ;GET THE PARITY REGISTER ADDRESS.
020306  010267  160606          SPRNT0: MOV     R2,     $GDADR          ;GET THE MEMORY ADDRESS BEING TESTED
```

```
020312  000414                          BR      SPRNTA              ;BR TO COMMON SECTION.

020314  010267  160600          SPRNT1: MOV     R2,       $GDADR    ;GET THE MEMORY ADDRESS BEING TESTED
020320  005367  160574                  DEC     $GDADR              ;ADJUST IT FOR PRINTOUT.
020324  000407                          BR      SPRNTA              ;BR TO COMMON SECTION.

020326  010367  160626          SPRNT3: MOV     R3,       $TMP0     ;GET THE DATA IN R3.
020332  010267  160562          SPRNT2: MOV     R2,       $GDADR    ;GET THE MEMORY ADDRESS BEING TESTED
020336  162767  000002  160554          SUB     #2,       $GDADR    ;ADJUST IT FOR PRINTOUT.
020344  010067  160554          SPRNTA: MOV     R0,       $GDDAT    ;GET WHAT THE DATA SHOULD BE
020350  010167  160552          SPRNTB: MOV     R1,       $BDDAT    ;GET WHAT THE DATA WAS
020354  000207                          RTS     PC                  ;RETURN TO ENTER ERROR ROUTINES

                                ;;*************************************************************
                                ;*  SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
                                ;*  R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
                                ;;*************************************************************
020356  005710                  TYPMAP: TST     (R0)                ;CHECK IF ANY MEMORY IN MAP...LO 64K.
020360  001007                          BNE     1$                  ;BR IF MEMORY IN MAP.
020362  005760  000002                  TST     2(R0)               ;...HI 64K.
020366  001004                          BNE     1$                  ;BR IF MEMORY IN MAP.
020370  004567  003062                  JSR     R5,       $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
020374  026232                          .WORD   NOMEM               ;ADDRESS OF MESSAGE TO BE TYPED
                                                                    ;"NO MEMORY FOUND."
020376  000475                          BR      6$                  ;EXIT
020400                          1$:
020400  010146                          MOV     R1,-(SP)            ;;PUSH R1 ON STACK
020402  010246                          MOV     R2,-(SP)            ;;PUSH R2 ON STACK
020404  010346                          MOV     R3,-(SP)            ;;PUSH R3 ON STACK
020406  010446                          MOV     R4,-(SP)            ;;PUSH R4 ON STACK
020410  012701  000001                  MOV     #BIT0,    R1        ;SET UP BANK POINTER...LO 64K.
020414  005002                          CLR     R2                  ;...HI 64K.
020416  012703  177777                  MOV     #-1,      R3        ;SET UP ADDRESS POINTER TO -1.
020422  010304                          MOV     R3,       R4        ;HI BITS OF ADDRESS AS WILL.
020424  030110                  2$:     BIT     R1,       (R0)      ;CHECK THE MAP FOR THIS BANK.
020426  001014                          BNE     3$                  ;BR IF THIS BANK PRESENT.
020430  030260  000002                  BIT     R2,       2(R0)     ;CHECK HI 64K MAP.
020434  001011                          BNE     3$                  ;BR IF THIS BANK PRESENT.
020436  105703                          TSTB    R3                  ;CHECK FOR PREVIOUS PRINTOUT.
020440  001042                          BNE     5$                  ;BR IF ALREADY TYPED "TO"
020442  162703  000001                  SUB     #1,       R3        ;BACK UP TO LAST ADR OF PREVIOUS BANK.
C20446  005604                          SBC     R4                  ;...HI ADDRESS BITS.
020450  004567  003002                  JSR     R5,       $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
020454  025463                          .WORD   TO                  ;ADDRESS OF MESSAGE TO BE TYPED
020456  000410                          BR      4$                  ;GO TO TYPE THE ADDRESS.
020460  105703                  3$:     TSTB    R3                  ;CHECK FOR PREVIOUS TYPEOUT.
020462  001431                          BEQ     5$                  ;BR IF ALREADY TYPE "FROM".
020464  062703  000001                  ADD     #1,       R3        ;POINT TO FIRST ADDRESS OF THIS BANK.
020470  005504                          ADC     R4                  ;...HI BITS OF ADDRESS.
020472  004567  002760                  JSR     R5,       $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
020476  025453                          .WORD   FROM                ;ADDRESS OF MESSAGE TO BE TYPED
020500                          4$:
020500  010346                          MOV     R3,-(SP)            ;;PUSH R3 ON STACK
020502  010446                          MOV     R4,-(SP)            ;;PUSH R4 ON STACK
020504  006303                          ASL     R3                  ;BIT 15 INTO C-BIT
020506  006104                          ROL     R4                  ;BIT 15 INTO R4.
```

```
020510  006003                              ROR     R3                  ;RESTORE BITS 14-0.
020512  010446                              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**.
020514  013746  177776                      MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
020520  004767  004104                      JSR     PC,     $TYPOS      ;GO TO THE SUBROUTINE
020524     003                              .BYTE   3                   ;;TYPE 3 DIGIT(S)
020525     000                              .BYTE   0                   ;;SUPPRESS LEADING ZEROS
020526  010346                              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**.
020530  013746  177776                      MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
020534  004767  004070                      JSR     PC,     $TYPOS      ;GO TO THE SUBROUTINE
020540     005                              .BYTE   5                   ;;TYPE 5 DIGIT(S)
020541     001                              .BYTE   1                   ;;TYPE LEADING ZEROS
020542  012604                              MOV     (SP)+,R4            ;;POP STACK INTO R4
020544  012603                              MOV     (SP)+,R3            ;;POP STACK INTO R3
020546  062703  020000              5$:     ADD     #20000, R3          ;UPDATE TO NEXT BANK.
020552  005504                              ADC     R4                  ;...HI ADDRESS BITS.
020554  006301                              ASL     R1                  ;SHIFT POINTER...LO 64K.
020556  006102                              ROL     R2                  ;...HI 64K.
020560  103321                              BCC     2$                  ;BR IF MORE BANKS.
020562  012604                              MOV     (SP)+,R4            ;;POP STACK INTO R4
020564  012603                              MOV     (SP)+,R3            ;;POP STACK INTO R3
020566  012602                              MOV     (SP)+,R2            ;;POP STACK INTO R2
020570  012601                              MOV     (SP)+,R1            ;;POP STACK INTO R1
020572  000207                      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

020574                              $SCOPE:
                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                    ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
020574  013746  177776                      MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
020600  004767  001524                      JSR     PC,     $CKSWR      ;GO TO THE SUBROUTINE
020604  012504                              MOV     (R5)+,  R4          ;SAVE MINIMUM BLOCK MASK NEXT TEST.
020606  010516                              MOV     R5,     (SP)        ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
020610  032777  040000  160322      1$:     BIT     #BIT14,@SWR         ;;LOOP ON PRESENT TEST?
020616  001117                              BNE     $OVER               ;;YES IF SW14=1
                                    ;#####START OF CODE FOR THE XOR TESTER#####
020620  000416                      $XTSTR: BR      6$                  ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
```

```
 4003  020622  013746  000004                   MOV     @#ERRVEC,-(SP)   ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 4004  020626  012737  020646  000004           MOV     #5$,@#ERRVEC     ;;SET FOR TIMEOUT
 4005  020634  005737  177060                   TST     @#177060         ;;TIME OUT ON XOR?
 4006  020640  012637  000004                   MOV     (SP)+,@#ERRVEC   ;;RESTORE THE ERROR VECTOR
 4007  020644  000466                           BR      $SVLAD           ;;GO TO THE NEXT TEST
 4008  020646  022626                  5$:      CMP     (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
 4009  020650  012637  000004                   MOV     (SP,+,@#ERRVEC   ;;RESTORE THE ERROR VECTOR
 4010  020654  000426                           BR      7$               ;;LOOP ON THE PRESENT TEST
 4011  020656                          6$:;#####END OF CODE FOR THE XOR TESTER#####
 4012  020656  032777  000400  160254           BIT     #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
 4013  020664  001407                           BEQ     2$               ;;BR IF NO
 4014  020666  017746  160246                   MOV     @SWR,-(SP)       ;;SET DESIRED TEST NUM. FROM SWR
 4015  020672  042716  000340                   BIC     #$SWRMK,(SP)     ;;STRIP AWAY UNDESIRED BITS
 4016  020676  122667  160200                   CMPB    (SP)+,$TSTNM     ;;ON THE RIGHT TEST?
 4C17  020702  001465                           BEQ     $OVER            ;;BR IF YES
 4018  020704  105767  160173          2$:      TSTB    $ERFLG           ;;HAS AN ERROR OCCURRED?
 4019  020710  0C!421                           BEQ     3$               ;;BR IF NO
 4020  020712  126767  160177  160163           CMPB    $ERMAX,$ERFLG    ;;MAX. ERRORS FOR THIS TEST OCCURRED?
 4021  020720  101015                           BHI     3$               ;;BR IF NO
 4022  020722  032777  001000  160210           BIT     #BIT09,@SWR      ;;LOOP ON ERROR?
 4023  020730  001404                           BEQ     4$               ;;BR IF NO
 4024  020732  016767  160152  160146  7$:      MOV     $LPERR,$LPADR    ;;SET LOOP ADDRESS TO LAST SCOPE
 4025  020740  000446                           BR      $OVER
 4026  0207 2  105067  160135          4$:      CLRB    $ERFLG           ;;ZERO THE ERROR FLAG
 4027  020746  005067  160216                   CLR     $TIMES           ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
 4028  020752  000415                           BR      1$               ;;ESCAPE TO THE NEXT TEST
 4029  020754  032777  004000  160156  3$:      BIT     #BIT11,@SWR      ;;INHIBIT ITERATIONS?
 4030  020762  001011                           BNE     1$               ;;BR IF YES
 4031  020764  005767  160222                   TST     $PASS            ;;IF FIRST PASS OF PROGRAM
 4032  020770  001406                           BEQ     1$               ;;          INHIBIT ITERATIONS
 4033  020772  005267  160106                   INC     $ICNT            ;;INCREMENT ITERATION COUNT
 4034  020776  026767  160166  160100           CMP     $TIMES,$ICNT     ;;CHECK THE NUMBER OF ITERATIONS MADE
 4035  021004  002024                           BGE     $OVER            ;;BR IF MORE ITERATION REQUIRED
 4036  021006  012767  000001  160070  1$:      MOV     #1,$ICNT         ;;REINITIALIZE THE ITERATION COUNTER
 4037  021014  016767  000552  160146           MOV     $MXCNT,$TIMES    ;;SET NUMBER OF ITERATIONS TO DO
 4038  021022  105267  160054          $SVLAD:  INCB    $TSTNM           ;;COUNT TEST NUMBERS
 4039  021026  116767  160050  160154           MOVB    $TSTNM,$TESTN    ;;SET TEST NUMBER IN APT MAILBOX
 4040  021034  011667  160046                   MOV     (SP),$LPADR      ;;SAVE SCOPE LOOP ADDRESS
 4041  021040  011667  160044                   MOV     (SP),$LPERR      ;;SAVE ERROR LOOP ADDRESS
 4042  021044  005067  160122                   CLR     $ESCAPE          ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
 4043  021050  112767  000001  160037           MOVB    #1,$ERMAX        ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
 4044  021056  016777  160020  160056  $OVER:   MOV     $TSTNM,@DISPLAY  ;;DISPLAY TEST NUMBER
 4045  021064  016716  160016                   MOV     $LPADR,(SP)      ;;FUDGE RETURN ADDRESS
 4046  021070  020516                  INSERT:  CMP     R5,     (SP)     ;CHECK FOR LOOP ON TEST.
 4047  021072  001402                         ) BEQ     1$               ;BR IF START NEXT TEST.
 4048  021074  000167  000470                 { JMP     ENDINS           ;JMP IF LOOP ON LAST TEST.
 4049  021100  012767  037777  160500  1$:      MOV     #37777, BLKMSK   ;SET 8K BOUNDRY MASK.
 4050  021106  005767  160100                   TST     $PASS            ;CHECK FOR PASS 0.
 4051  021112  001404                           BEQ     2$               ;BR IF PASS 0
 4052  021114  126727  157762  000021           CMPB    $TSTNM, #21      ;CHECK IF IN SECTION 3.
 4053  021122  103002                           BHIS    3$               ;BR IF IN SECTION 3.
 4054  021124  006267  160456          2$:      ASR     BLKMSK           ;RESET BOUNDRY TO 4K.
 4055  021130  016767  160426  160426  3$:      MOV     FSTADR, TMPFAD   ;GET FIRST ADDRESS.
 4056  021136  005767  157436                   TST     RELOCF           ;CHECK IF PRG RELOCATED.
 4057  021142  001430                           BEQ     4$               ;BR IF NOT RELOCATED.
 4058  021144  032777  000040  157766           BIT     #SW05,  @SWR     ;CHECK IF LOC 0-776 TO BE PROTECTED.
 4059  021152  001424                           BEQ     4$                 ;BR IF SW NOT SET.
 4060  021154  026727  160404  001000           CMP     TMPFAD, #1000      ;CHECK IF NOT BEING TESTED.
 4061  021162  103020                           BHIS    4$                 ;BR IF ALREADY PROTECTED.
 4062  021164  012767  001000  160372           MOV     #1000,  TMPFAD     ;RESET FIRST ADDRESS.
 4063  021172  05276,  000001  160370           BIS     #BIT0,  FADMAP     ;SET FLAG IN FIRST BANK.
 4064  021200  026727  160370  001000           CMP     LSTADR, #1000      ;CHECK IF GONE PAST LAST ADR.
 4065  021206  101006                           BHI     4$                 ;BR IF ENOUGH MEMORY.
 4066  021210  004567  002242                   JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4067  021214  026703                           .WORD   NOMTST             ;ADDRESS OF MESSAGE TO BE TYPED
 4068                                                                      ;"NO MEMORY TESTED"
 4069  021216  016716  160422                   MOV     .TST32, (SP)       ;ADJUST RETURN ADR FOR ABORT.
 4070  021222  000207                           RTS     PC                 ;ABORT.
 4071  021224  016767  160344  160344   4$:     MOV     LSTADR, TMPLAD     ;GET LAST ADDRESS.
 4072  021232  016767  160276  160270           MOV     SAVTST, TSTMAP     ;GET TEST MAP, LO 64K.
 4073  021240  016767  160272  160264           MOV     SAVTST+2,TSTMAP+2  ;....HI 64K.
 4074  021246  046767  157330  160254           BIC     PRGMAP, TSTMAP     ;DON'T TEST OVER THE PROGRAM.
 4075  021254  046767  157324  160250           BIC     PRGMAP+2,TSTMAP+2
 4076  021262  005767  157724                   TST     $PASS              ;CHECK FOR FIRST PASS
 4077  021266  001011                           BNE     10$                ;BR IF NOT FIRST PASS.
 4078  021270  032767  000003  160232           BIT     #3,     TSTMAP     ;CHECK IF FIRST TWO BANKS AVAILABLE.
 4079  021276  001405                           BEQ     10$                ;NOT TESTING FIRST 2 BANKS.
 4080  021300  042767  177774  160222           BIC     #177774,TSTMAP     ;CLR ALL BUT FIRST 2 BANKS.
 4081  021306  005067  160220                   CLR     TSTMAP+2
 4082  021312  005704                   10$:    TST     R4                 ;CHECK FOR A MINIMUM BLOCK SIZE.
 4083  021314  001503                           BEQ     20$                ;BR IF NO MIN BLOCK SIZE.
 4084  021316  030467  160242                   BIT     R4,     TMPFAD     ;CHECK IF FIRST ADR ON BLOCK BOUNDRY.
 4085  021322  001416                           BEQ     11$                ;BR IF FIRST ADR ON BLOCK BOUNDRY.
 4086  021324  050467  160234                   BIS     R4,     TMPFAD     ;ADJUST FIRST ADR TO END OF BLOCK.
 4087  021330  005267  160230                   INC     TMPFAD             ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
 4088  021334  032767  017777  160222           BIT     #MASK4K,TMPFAD     ;CHECK IF FIRST ADR REACHED 4K BOUNDRY.
 4 39  021342  001006                           BNE     11$                ;BR IF NOT ON 4K BOUNDRY.
 4090  021344  046767  160220  160156           BIC     FADMAP, TSTMAP     ;DON'T TEST FIRST BANK.
 4091  021352  046767  160214  160152           BIC     FADMAP+2,TSTMAP+2
 4092  021360  030467  160212           11$:    BIT     R4,     TMPLAD     ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
 4093  021364  001414                           BEQ     12$                ;BR IF ON BLOCK BOUNDRY.
 4094  021366  040467  160204                   BIC     R4,     TMPLAD     ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
 4095  021372  032767  017777  160176           BIT     #MASK4K,TMPLAD     ;CHECK IF ADJUSTED TO 4K BOUNDRY.
 4096  021400  001006                           BNE     12$                ;BR IF NOT ON 4K BOUNDRY.
 4097  021402  046767  160174  160120           BIC     LADMAP, TSTMAP     ;SKIP TESTING LAST BANK.
 4098  021410  046767  160170  160114           BIC     LADMAP+2,TSTMAP+2
 4099  021416  036767  160146  160156   12$:    BIT     FADMAP, LADMAP     ;CHECK IF FIRST AND LAST IN SAME BANK.
 4100  021424  001004                           BNE     13$                ;BR IF IN SAME BANK.
 4101  021426  036767  160140  160150           BIT     FADMAP+2,LADMAP+2  ;....UPPER 64K.
 4102  021434  001404                           BEQ     14$                ;BR IF FIRST AND LAST NOT SAME BANK.
 4103  021436  026767  160134  160120   13$:    CMP     TMPLAD, TMPFAD     ;CHECK IF ANY MEMORY LEFT.
 4104  021444  101406                           BLOS    15$                ;BR IF NO MEMORY TO TEST.
 4105  021446  005767  160056           14$:    TST     TSTMAP             ;CHECK IF ANY BANKS LEFT TO TEST!!
 4106  021452  001017                           BNE     16$                ;BR IF TEST MAP NOT EMPTY.
 4107  021454  005767  160052                   TST     TSTMAP+2           ;CHECK FOR ANY BANKS.
 4108  021460  001014                           BNE     16$                ;BR IF TEST MAP NOT EMPTY.
 4109  021462                           15$:
 4110  021462  004567  001770                   JSR     R5,     $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
 4111  021466  026727                           .WORD   SKPMES             ;ADDRESS OF MESSAGE TO BE TYPED
 4112                                                                      ;"SKIPPING TEST #"
 4113  021470  005046                           CLR     -(SP)              ;CLEAR THE WORD ON THE STACK.
 4114  021472  116716  157404                   MOVB    $TSTNM, (SP)       ;PUT THE DATA ON THE STACK.
```

```
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021476  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021502  004767  003122                          JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
021506     003                                  .BYTE   3               ;TYPE 3 DIGITS.
021507     001                                  .BYTE   1               ;TYPE LEADING ZEROS.
021510  000427                                  BR      ENDINS          ;RETURN TO SKIP TEST.
021512  062716  000004                  16$:    ADD     #4,     (SP)    ;SKIP THE SKIP ON RETURN.
021516  062767  000004  157362                  ADD     #4,     $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
021524  012767  017777  160034          20$:    MOV     #MASK4K,FADMSK  ;GET 4K MASK.
021532  016705  160026                          MOV     TMPFAD, R5      ;GET FIRST ADR.
021536  040567  160024                  21$:    BIC     R5,     FADMSK  ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
021542  006305                                  ASL     R5              ;MOVE LOWEST BIT UP ONE.
021544  001374                                  BNE     21$             ;LOOP UNTIL OVERFLOW.
021546  012767  017777  160024                  MOV     #MASK4K,LADMSK  ;SET MASK BITS
021554  016705  160016                          MOV     TMPLAD, R5      ;GET LAST ADR.
021560  040567  160014                  22$:    BIC     R5,     LADMSK  ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
021564  006305                                  ASL     R5              ;MOVE LOWEST BIT OF LAST ADR UP ONE.
021566  001374                                  BNE     22$             ;LOOP UNTIL OVERFLOW.
021570  000207                          ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
021572  000004                          $MXCNT: 4                       ;;MAX. NUMBER OF ITERATIONS
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021574  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021600  004767  000524                          JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
                                        .SBTTL  ERROR HANDLER ROUTINE

                                        ;;*************************************************************
                                        ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                        ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                        ;*AND GO TO $ERRTYP ON ERROR
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW15=1         HALT ON ERROR
                                        ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1         BELL ON ERROR
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*CALL
                                        ;*      ERROR   N       ;;ERROR=EMT AND N=ERROR ITEM NUMBER

021604                                  $ERROR:
                                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021604  013746  177776                          MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021610  004767  000514                          JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
021614  062716  000002                          ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
021620  105267  157257                  7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
021624  001775                                  BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
021626  016777  157250  157306                  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
021634  032777  002000  157276                  BIT     #BIT10,@SWR     ;;BELL ON ERROR?
021642  001403                                  BEQ     1$              ;;NO - SKIP
021644  004567  001606                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021650  001174                                  .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
021652  005267  157234                  1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
021656  011667  157234                          MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
021662  162767  000002  157226                  SUB     #2,$ERRPC
021670  117767  157222  157216                  MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
```

```
021676  032777  020000  157234          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
021704  001005                          BNE     20$             ;;SKIP TYPEOUTS
021706  004767  000116                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
021712  004567  001540                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
021716  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
021720                          20$:
021720  122767  000001  157276          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
021726  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
021730  116767  157160  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
021736  004767  002044                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
021742     000                  21$:    .BYTE   0
021743     000                          .BYTE   0
021744  000777                  22$:    BR      22$             ;;APT ERROR LOOP
021746  005777  157166          2$:     TST     @SWR            ;;HALT ON ERROR
021752  100005                          BPL     3$              ;;SKIP IF CONTINUE
021754  000000                          HALT                    ;;HALT ON ERROR!
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
021756  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
021762  004767  000342                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
021766  032777  001000  157144  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
021774  001402                          BEQ     4$              ;;BR IF NO
021776  016716  157106                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
022002  005767  157164          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
022006  001402                          BEQ     5$              ;;BR IF NONE
022010  016716  157156                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
022014                          5$:
022014  022737  014174  000042          CMP     #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
022022  001001                          BNE     6$              ;;BRANCH IF NO
022024  000000                          HALT                    ;;YES
022026                          6$:
022026  000207                          RTS     PC
                                ;;*************************************************************

                                .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE

                                ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

022030                          $ERRTYP:
022030  004567  001422                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022034  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022036  010046                          MOV     R0,-(SP)                        ;SAVE R0
022040  005000                          CLR     R0                              ;PICKUP THE ITEM INDEX
022042  156700  157046                  BISB    $ITEMB,R0
022046  001007                          BNE     1$                              ;IF ITEM NUMBER IS ZERO, JUST
                                                                                ;TYPE THE PC OF THE ERROR
022050  016746  157042                  MOV     $ERRPC,-(SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                ;;ERROR ADDRESS
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022054  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022060  004767  002570                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022064  000513                          BR      10$                             ;GET OUT
022066  016767  157024  157420  1$:     MOV     $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
```

```
022074  166767  156500  157412          SUB     RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
022102  005300                          DEC     R0                      ;ADJUST THE INDEX SO THAT IT WILL
022104  006300                          ASL     R0                      ;       WORK FOR THE ERROR TABLE
022106  006300                          ASL     R0
022110  006300                          ASL     R0
022112  066700  157522                  ADD     .ERRTB, R0      ;FORM TABLE POINTER
022116  012067  000006                  MOV     (R0)+,2$                ;PICKUP "ERROR MESSAGE" POINTER
022122  001406                          BEQ     3$                      ;SKIP TYPEOUT IF NO POINTER
022124  004567  001326                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022130  000000                  2$:     .WORD   0                       ;"ERROR MESSAGE" POINTER GOES HERE
022132  004567  001320                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022136  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022140  012067  000006          3$:     MOV     (R0)+,4$                ;PICKUP "DATA HEADER" POINTER
022144  001406                          BEQ     5$                      ;SKIP TYPEOUT IF 0
022146  004567  001304                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022152  000000                  4$:     .WORD   0                       ;"DATA HEADER" POINTER GOES HERE
022154  004567  001276                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022160  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022162  010146                  5$:     MOV     R1,-(SP)                ;SAVE R1
022164  012001                          MOV     (R0)+,R1                ;PICKUP "DATA TABLE" POINTER
022166  001451                          BEQ     9$                      ;BR IF NO DATA TO BE TYPED
022170  066701  156404                  ADD     RELOCF, R1      ;ADJUST POINTER
022174  012000                          MOV     (R0)+,R0                ;PICKUP "DATA FORMAT" POINTER
022176  066700  156376                  ADD     RELOCF, R0      ;ADJUST POINTER.
022202  105720                  6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
022204  001006                          BNE     7$                      ;BR IF NOT 16-BIT OCTAL
022206  013146                          MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022210  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022214  004767  002434                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022220  000426                          BR      8$
022222  100406                  7$:     BMI     17$             ;BRANCH IF NOT DECIMAL
022224  013146                          MOV     @(R1)+, -(SP)   ;;SAVE @(R1)+ FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022226  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022232  004767  002140                  JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
022236  000417                          BR      8$              ;SKIP
022240  122760  177777  177777  17$:    CMPB    #-1,    -1(R0)  ;CHECK FOR 18-BIT ADDRESS FORMAT.
022246  001004                          BNE     18$             ;BR IF NOT 18-BIT ADDRESS FORMAT.
022250  013146                          MOV     @(R1)+, -(SP)   ;PUT THE DATA ON THE STACK.
022252  004767  002640                  JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
022256  000407                          BR      8$              ;SKIP
022260                          18$:
022260  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
022262  113116                          MOVB    @(R1)+, (SP)    ;PUT THE DATA ON THE STACK.
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022264  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022270  004767  002334                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
022274     003                          .BYTE   3               ;TYPE 3 DIGITS.
022275     001                          .BYTE   1               ;TYPE LEADING ZEROS.
022276  005711                  8$:     TST     (R1)                    ;IS THERE ANOTHER NUMBER?
022300  001404                          BEQ     9$                      ;BR IF NO
022302  004567  001150                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
022306  022326                                  .WORD   11$                 ;ADDRESS OF MESSAGE TO BE TYPED
022310  000734                                  BR      6$                          ;LOOP

022312  012601                          9$:     MOV     (SP)+,R1                    ;RESTORE R1
022314  012600                          10$:    MOV     (SP)+,R0                    ;RESTORE R0
022316  004567  001134                          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022322  001201                                  .WORD   $CRLF               ;ADDRESS OF MESSAGE TO BE TYPED
022324  000207                                  RTS     PC                          ;RETURN
022326  000011                          11$:    .ASCIZ  /       /           ;TAB CHARACTER.
                                                .EVEN
                                        .SBTTL  TTY INPUT ROUTINE

                                        ;;*****************************************************************
                                        .ENABL  LSB

                                        ;;*****************************************************************
                                        ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                        ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                        ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                        ;*WHEN OPERATING IN TTY FLAG MODE.
022330  022767  000176  156602  $CKSWR: CMP     #SWREG,SWR          ;;IS THE SOFT-SWR SELECTED?
022336  001104                          BNE     15$                 ;;BRANCH IF NO
022340  105777  156600                  TSTB    @$TKS               ;;CHAR THERE?
022344  100101                          BPL     15$                 ;;IF NO, DON'T WAIT AROUND
022346  117746  156574                  MOVB    @$TKB,-(SP)         ;;SAVE THE CHAR
022352  042716  177600                  BIC     #↑C177,(SP)         ;;STRIP-OFF THE ASCII
022356  022726  000007                  CMP     #7,(SP)+            ;;IS IT A CONTROL G?
022362  001072                          BNE     15$                 ;;NO, RETURN TO USER
022364  126727  156544  000001          CMPB    $AUTOB,#1           ;;ARE WE RUNNING IN AUTO-MODE?
022372  001466                          BEQ     15$                 ;;BRANCH IF YES

022374  004567  001056                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022400  023255                          .WORD   $CNTLG              ;ADDRESS OF MESSAGE TO BE TYPED
022402                          $GTSWR:
022402  004567  001050                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022406  023262                          .WORD   $MSWR               ;ADDRESS OF MESSAGE TO BE TYPED
022410  016746  155562                  MOV     SWREG,-(SP)         ;;SAVE SWREG FOR TYPEOUT
                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
                                ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022414  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
022420  004767  002230                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
022424  004567  001026                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022430  023273                          .WORD   $MNEW               ;ADDRESS OF MESSAGE TO BE TYPED
022432  005046                  19$:    CLR     -(SP)               ;;CLEAR COUNTER
022434  005046                          CLR     -(SP)               ;;THE NEW SWR
022436  105777  156502          7$:     TSTB    @$TKS               ;;CHAR THERE?
022442  100375                          BPL     7$                  ;;IF NOT TRY AGAIN

022444  117746  156476                  MOVB    @$TKB,-(SP)         ;;PICK UP CHAR
022450  042716  177600                  BIC     #↑C177,(SP)         ;;MAKE IT 7-BIT ASCII



022454  021627  000025          9$:     CMP     (SP),#25            ;;IS IT A CONTROL-U?
022460  001006                          BNE     10$                 ;;BRANCH IF NOT
022462  004567  000770                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
022466  023250                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
022470  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
022474  000756                          BR      19$             ;;LET'S TRY IT AGAIN


022476  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
022502  001023                          BNE     16$             ;;BRANCH IF NO
022504  005766  000004                  TST     4,SP            ;;YES, IS IT THE FIRST CHAR
022510  001403                          BEQ     11$             ;;BRANCH IF YES
022512  016677  000002  156420          MOV     2(SP),@SWR      ;;SAVE NEW SWR
022520  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
022524                          14$:
022524  004567  000726                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE
022530  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
022532  126727  156377  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
022540  001003                          BNE     15$             ;;BRANCH IF NOT
022542  012777  000100  156374          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
022550  000002                  15$:    RTI                     ;;RETURN
022552  004767  001142          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
022556  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
022562  002420                          BLT     18$             ;;BRANCH IF YES
022564  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
022570  003015                          BGT     18$             ;;BRANCH IF YES
022572  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
022576  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
022602  001403                          BEQ     17$             ;;BRANCH IF YES
022604  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
022606  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
022610  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
022612  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
022616  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
022622  000705                          BR      7$              ;;GET THE NEXT ONE
022624                          18$:
022624  004567  000626                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
022630  001200                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
022632  000716                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB



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

022634  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
022636  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
022644  105777  156274          1$:     TSTB    @$TKS           ;;WAIT FOR
022650  100375                          BPL     1$              ;;A CHARACTER
022652  117766  156270  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
022660  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
022666  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
022674  001013                          BNE     3$              ;;BRANCH IF NO
022676  105777  156242          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
```

```
022702  100375                           BPL     2$                   ;;LOOP UNTIL ITS THERE
022704  117746  156236                   MOVB    @$TKB,-(SP)          ;;GET CHARACTER
022710  042716  177600                   BIC     #↑C177,(SP)          ;;MAKE IT 7-BIT ASCII
022714  022627  000021                   CMP     (SP)+,#21            ;;IS IT A CONTROL-Q?
022720  001366                           BNE     2$                   ;;IF NOT DISCARD IT
022722  000750                           BR      1$                   ;;YES, RESUME
022724  026627  000004  000140   3$:     CMP     4(SF),#140           ;;IS IT UPPER CASE?
022732  002407                           BLT     4$                   ;;BRANCH IF YES
022734  026627  000004  000175           CMP     4(SP),#175           ;;IS IT A SPECIAL CHAR?
022742  003003                           BGT     4$                   ;;BRANCH IF YES
022744  042766  000040  000004           BIC     #40,4(SP)            ;;MAKE IT UPPER CASE
022752  000002                   4$:     RTI                          ;;GO BACK TO USER
                                 ;;**************************************************
                                 ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                 ;*CALL:
                                 ;*       RDLIN                        ;;INPUT A STRING FROM THE TTY
                                 ;*       RETURN HERE                  ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                 ;*                                    ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

022754  010346                   $RDLIN: MOV     R3,-(SP)             ;;SAVE R3
022756  005046                           CLR     -(SP)                ;;CLEAR THE RUBOUT KEY
022760  012703  023240           1$:     MOV     #$TTYIN,R3           ;;GET ADDRESS
022764  022703  023250           2$:     CMP     #$TTYIN+8.,R3        ;;BUFFER FULL?
022770  101467                           BLOS    4$                   ;;BR IF YES
                                 ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
                                 ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
022772  013746  177776                   MOV     @#PSW,  -(SP)        ;PUT THE PROCESSOR STATUS ON THE STACK
022776  004767  177632                   JSR     PC,     $RDCHR       ;GO TO THE SUBROUTINE
023002  112613                           MOVB    SP)+,(R3)            ;;GET CHARACTER
023004  122713  000177           10$:    CMPB    #177,(R3)            ;;IS IT A RUBOUT
023010  001024                           BNE     5$                   ;;BR IF NO
023012  005716                           TST     (SP)                 ;;IS THIS THE FIRST RUBOUT?
023014  001010                           BNE     6$                   ;;BR IF NO
023016  112767  000134  000212           MOVB    #'\,9$               ;;TYPE A BACK SLASH
023024  004567  000426                   JSR     R5,     $PRINT       ;GO PRINT OUT THE FOLLOWING MESSAGE.
023030  023236                           .WORD   9$                   ;ADDRESS OF MESSAGE TO BE TYPED
023032  012716  177777                   MOV     #-1,(SP)             ;;SET THE RUBOUT KEY
023036  005303                   6$:     DEC     R3                   ;;BACKUP BY ONE
023040  020327  023240                   CMP     R3,#$TTYIN           ;;STACK EMPTY?
023044  103441                           BLO     4$                   ;;BR IF YES
023046  111367  000164                   MOVB    (R3),9$              ;;SETUP TO TYPEOUT THE DELETED CHAR.
023052  004567  000400                   JSR     R5,     $PRINT       ;GO PRINT OUT THE FOLLOWING MESSAGE.
023056  023236                           .WORD   9$                   ;ADDRESS OF MESSAGE TO BE TYPED
023060  000741                           BR      2$                   ;;GO READ ANOTHER CHAR.
023062  005716                   5$:     TST     (SP)                 ;;RUBOUT KEY SET?
023064  001407                           BEQ     7$                   ;;BR IF NO
023066  112767  000134  000142           MOVB    #'\,9$               ;;TYPE A BACK SLASH
023074  004567  000356                   JSR     R5,     $PRINT       ;GO PRINT OUT THE FOLLOWING MESSAGE.
023100  023236                           .WORD   9$                   ;ADDRESS OF MESSAGE TO BE TYPED
023102  005016                           CLR     (SP)                 ;;CLEAR THE RUBOUT KEY
023104  122713  000025           7$:     CMPB    #25,(R3)             ;;IS CHARACTER A CTRL U"
023110  001004                           BNE     8$                   ;;BR IF NO
023112  004567  000340                   JSR     R5,     $PRINT       ;GO PRINT OUT THE FOLLOWING MESSAGE.
023116  023250                           .WORD   $CNTLU               ;ADDRESS OF MESSAGE TO BE TYPED
023120  000717                           BR      1$                   ;;GO START OVER
023122  122713  000022           8$:     CMPB    #22,(R3)             ;;IS CHARACTER A "↑R"?
```

```
023126  001014                                   BNE     3$                  ;;BRANCH IF NO
023130  105013                                   CLRB    (R3)                ;;CLEAR THE CHARACTER
023132  004567  000320                           JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
023136  001201                                   .WORD   $CRLF               ;ADDRESS OF MESSAGE TO BE TYPED
023140  004567  000312                           JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
023144  023240                                   .WORD   $TTYIN              ;ADDRESS OF MESSAGE TO BE TYPED
023146  000706                                   BR      2$                  ;;GO PICKUP ANOTHER CHACTER
023150                                   4$:
023150  004567  000302                           JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
023154  001200                                   .WORD   $QUES               ;ADDRESS OF MESSAGE TO BE TYPED
023156  000700                                   BR      1$                  ;;CLEAR THE BUFFER AND LOOP
023160  111367  000052                   3$:     MOVB    (R3),9$             ;;ECHO THE CHARACTER
023164  004567  000266                           JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
023170  023236                                   .WORD   9$                  ;ADDRESS OF MESSAGE TO BE TYPED
023172  122723  000015                           CMPB    #15,(R3)+           ;;CHECK FOR RETURN
023176  001272                                   BNE     2$                  ;;LOOP IF NOT RETURN
023200  105063  177777                           CLRB    -1(R3)              ;;CLEAR RETURN (THE 15)
023204  004567  000246                           JSR     R5,      $PRINT     ;GO PRINT OUT THE FOLLOWING MESSAGE.
023210  001202                                   .WORD   $LF                 ;ADDRESS OF MESSAGE TO BE TYPED
023212  005726                                   TST     (SP)+               ;;CLEAN RUBOUT KEY FROM THE STACK
023214  012603                                   MOV     (SP)+,R3            ;;RESTORE R3
023216  011646                                   MOV     (SP),-(SP)          ;;ADJUST THE STACK AND PUT ADDRESS OF THE
023220  016666  000004  000002                   MOV     4(SP),2(SP)         ;;       FIRST ASCII CHARACTER ON IT
023226  012766  023240  000004                   MOV     #$TTYIN,4(SP)
023234  000002                                   RTI                         ;;RETURN
023236     000                           9$:     .BYTE   0                   ;;STORAGE FOR ASCII CHAR. TO TYPE
023237     000                                   .BYTE   0                   ;;TERMINATOR
023240  000010                           $TTYIN: .BLKB   8.                  ;;RESERVE 8 BYTES FOR TTY INPUT
023250  052536  005015     000           $CNTLU: .ASCIZ  /↑U/<15><12>        ;;CONTROL "U"
023255     136  006507  000012           $CNTLG: .ASCIZ  /↑G/<15><12>        ;;CONTROL "G"
023262  005015  053523  020122           $MSWR:  .ASCIZ  <15><12>/SWR = /
023270  020075     000
023273     040  047040  053505           $MNEW:  .ASCIZ  /  NEW = /
023300  036440  000040
                                         .SBTTL  READ AN OCTAL NUMBER FROM THE TTY

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

023304  011646                           $RDOCT: MOV     (SP),-(SP)          ;;PROVIDE SPACE FOR THE
023306  016666  000004  000002                   MOV     4(SP),2(SP)         ;;INPUT NUMBER
023314  010046                                   MOV     R0,-(SP)            ;;PUSH R0 ON STACK
023316  010146                                   MOV     R1,-(SP)            ;;PUSH R1 ON STACK
023320  010246                                   MOV     R2,-(SP)            ;;PUSH R2 ON STACK
023322                                   1$:
                                         ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
                                         ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
```

```
023322  013746  177776                 MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
023326  004767  177422                 JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
023332  012600                         MOV     (SP)+,RC        ;;GET ADDRESS OF 1ST CHARACTER
023334  010067  000102                 MOV     R0,5$           ;;AND SAVE IT
023340  005001                         CLR     R1              ;;CLEAR DATA WORD
023342  005002                         CLR     R2
023344  112046                 2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
023346  001420                         BEQ     3$              ;;IF ZERO GET OUT
023350  122716  000060                 CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
023354  003026                         BGT     4$              ;;IS AN OCTAL DIGIT
023356  122716  000067                 CMPB    #'7,(SP)
023362  002423                         BLT     4$
023364  006301                         ASL     R1              ;;*2
023366  006102                         ROL     R2
023370  006301                         ASL     R1              ;;*4
023372  006102                         ROL     R2
023374  006301                         ASL     R1              ;;*8
023376  006102                         ROL     R2
023400  042716  177770                 BIC     #↑C7,(SP)       ;;STRIP THE ASCII JUNK
023404  062601                         ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
023406  000756                         BR      2$              ;;LOOP
023410  005726                 3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
023412  010166  000012                 MOV     R1,12(SP)       ;;SAVE THE RESULT
023416  010267  000032                 MOV     R2,$HIOCT
023422  C12602                         MOV     (SP)+,R2        ;;POP STACK INTO R2
023424  012601                         MOV     (SP)+,R1        ;;POP STACK INTO R1
023426  012600                         MOV     (SP)+,R0        ;;POP STACK INTO R0
023430  000002                         RTI                     ;;RETURN
023432  005726                 4$:     TST     (SP)+           ;;CLEAN PARTIAL FROM STACK
023434  105010                         CLRB    (R0)            ;;SET A TERMINATOR
023436  004567  000014                 JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023442  000000                 5$:     .WORD   0
023444  004567  000006                 JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023450  001200                         .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
023452  000723                         BR      1$              ;;TRY AGAIN
023454  000000                 $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE

                               ;;*******************************************************************
                               ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
                               ;*  CALL:         JSR     R5,     $PRINT
                               ;*                <MESSAGE VIRTUAL ADDRESS>
                               ;;*******************************************************************
023456  012567  000016         $PRINT: MOV     (R5)+,  1$      ;GET THE MESSAGE VIRTUAL ADDRESS.
023462  066767  155112  000010         ADD     RELOC,  1$      ;MAKE IT PHYSICAL.
                               ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
                               ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
023470  013746  177776                 MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
023474  004767  000004                 JSR     PC,     $TYPE   ;GO TO THE SUBROUTINE
023500  000000                 1$:     .WORD   0               ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
023502  000205                         RTS     R5              ;RETURN.

                               .SBTTL  TYPE ROUTINE

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

```
                                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*      TYPE
                                        ;*      MESADR
                                        ;*

023504  105767  155447          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
023510  100002                          BPL     1$              ;;BR IF YES
023512  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
023514  000430                          BR      3$              ;;LEAVE
023516  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
023520  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
023524  122767  000001  155472          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
023532  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
023534  132767  000100  155463          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
023542  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
023544  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
023550  004767  000222                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
023554  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
023556  132767  000040  155441  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
023564  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
023566  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
023570  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
023572  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
023574  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
023576  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
023602  000002                          RTI                     ;;RETURN
023604  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
023610  001431                          BEQ     8$
023612  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
023616  001007                          BNE     5$
023620  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
023622  004567  177630                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
023626  001201                          $CRLF
023630  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
023634  000754                          BR      2$              ;;GET NEXT CHARACTER
023636  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
023642  126726  155310          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
023646  001347                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
023650  016746  155300                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
023654  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
023660  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
023662  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
023666  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
023672  000770                          BR      7$              ;;LOOP

                                ;HORIZONTAL TAB PROCESSOR

023674  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
```

```
023700  004767  000014          9$:      JSR     PC,$TYPEC          ;;TYPE A SPACE
023704  132767  000007  000052           BITB    #7,$CHARCNT        ;;BRANCH IF NOT AT
023712  001372                           BNE     9$                 ;;TAB STOP
023714  005726                           TST     (SP)+              ;;POP SPACE OFF STACK
023716  000723                           BR      2$                 ;;GET NEXT CHARACTER
023720  105777  155224          $TYPEC:  TSTB    @$TPS              ;;WAIT UNTIL PRINTER IS READY
023724  100375                           BPL     $TYFEC
023726  116677  000002  155216           MOVB    2(SP),@$TPB        ;;LOAD CHAR TO BE TYPED INTO DATA REG.
023734  122766  000015  000002           CMPB    #CR,2(SP)          ;;IS CHARACTER A CARRIAGE RETURN?
023742  001003                           BNE     1$                 ;;BRANCH IF NO
023744  105067  000014                   CLRB    $CHARCNT           ;;YES--CLEAR CHARACTER COUNT
023750  000406                           BR      $TYPEX             ;;EXIT
023752  122766  000012  000002  1$:      CMPB    #LF,2(SP)          ;;IS CHARACTER A LINE FEED?
023760  001402                           BEQ     $TYPEX             ;;BRANCH IF YES
023762  105227                           INCB    (PC)+              ;;COUNT THE CHARACTER
023764  000000                  $CHARCNT:.WORD   0                  ;;CHARACTER COUNT STORAGE
023766  000207                  $TYPEX:  RTS     PC

                                .SBTTL  APT COMMUNICATIONS ROUTINE

                                ;;*****************************************************************
023770  112767  000001  000376  $ATY1:   MOVB    #1,$FFLG           ;;TO REPORT FATAL ERROR
023776  112767  000001  000366  $ATY3:   MOVB    #1,$MFLG           ;;TO TYPE A MESSAGE
024004  000403                           BR      $ATYC
024006  112767  000001  000360  $ATY4:   MOVB    #',$FFLG           ;;TO ONLY REPORT FATAL ERROR
024014                          $ATYC:
024014  010046                           MOV     R0,-(SP)           ;;PUSH R0 ON STACK
024016  010146                           MOV     R1,-(SP)           ;;PUSH R1 ON STACK
024020  105767  000346                   TSTB    $MFLG              ;;SHOULD TYPE A MESSAGE?
024024  001450                           BEQ     5$                 ;;IF NOT:  3R
024026  122767  000001  155170           CMPB    #APTENV,$ENV       ;;OPERATING UNDER APT?
024034  001031                           BNE     3$                 ;;IF NOT:  BR
024036  132767  000100  155161           BITB    #APTSPOOL,$ENVM    ;;SHOULD SPOOL MESSAGES?
024044  001425                           BEQ     3$                 ;;IF NOT:  BR
024046  017600  000004                   MOV     @4(SP),R0          ;;GET MESSAGE ADDR.
024052  062766  000002  000004           ADD     #2,4(SP)                   ;;BUMP RETURN ADDR.
024060  005767  155120          1$:      TST     $MSGTYPE           ;;SEE IF DONE W/ LAST XMISSION?
024064  001375                           BNE     1$                 ;;IF NOT:  WAIT
024066  010067  155126                   MOV     R0,$MSGAD          ;;PUT ADDR IN MAILBOX
024072  105720                  2$:      TSTB    (R0)+              ;;FIND END OF MESSAGE
024074  001376                           BNE     2$
024076  166700  155116                   SUB     $MSGAD,R0          ;;SUB START OF MESSAGE
024102  006200                           ASR     R0                 ;;GET MESSAGE LNGTH IN WORDS
024104  010067  155112                   MOV     R0,$MSGLGT         ;;PUT LENGTH IN MAILBOX
024110  012767  000004  155066           MOV     #4,$MSGTYPE        ;;TELL APT TO TAKE MSG.
024116  000413                           BR      5$
024120  017667  000004  000016  3$:      MOV     @4(SP),4$          ;;PUT MSG ADDR IN JSR LINKAGE
024126  062766  000002  000004           ADD     #2,4(SP)                   ;;BUMP RETURN ADDRESS
024134  016746  153636                   MOV     177776,-(SP)       ;;PUSH 177776 ON STACK
024140  004767  177340                   JSR     PC,$TYPE           ;;CALL TYPE MACRO
024144  000000                  4$:      .WORD   0
024146                          5$:
024146  105767  000221                   TSTB    $LFLG              ;;SHOULD LOG AN ERROR?
024152  001422                           BEQ     10$                ;;IF NOT:  BR
024154  017600  000004                   MOV     @4(SP),R0                  ;;GET ERROR #
024160  062766  000002  000004           ADD     #2,4(SP)                   ;;BUMP RETURN ADDR.
```

```
024166  012701  001344                   MOV     #$ASTAT,R1        ;;POINT TO TABLE START
024172  005711                   6$:     TST     (R1)              ;;END OF TABLE?
024174  100404                           BMI     8$                ;;IF SO:  BR
024176  020021                           CMP     R0,(R1)+          ;;PROPER ENTRY?
024200  001406                           BEQ     9$                ;;IF SO:  BR
024202  005721                           TST     (P1)+             ;;MOVE PAST COUNTER WORD
024204  000772                           BR      6$                ;;KEEP LOOKING
024206  026701  155300           8$:     CMP     $APTR,R1          ;;TABLE FULL?
024212  001402                           BEQ     10$               ;;IF SO:  BR -- NO MORE ROOM
024214  010021                           MOV     R0,(R1)+          ;;SET UP NEW ENTRY
024216  005211                   9$:     INC     (R1)              ;;BUMP ERROR COUNT
024220  105767  000150           10$:    TSTB    $FFLG             ;;SHOULD REPORT FATAL ERROR?
024224  001416                           BEQ     12$               ;;IF NOT:  BR
024226  005767  154772                   TST     $ENV              ;;RUNNING UNDER APT?
024232  001413                           BEQ     12$               ;;IF NOT:  BR
024234  005767  154744           11$:    TST     $MSGTYPE          ;;FINISHED LAST MESSAGE?
024240  001375                           BNE     11$               ;;IF NOT:  WAIT
024242  017667  000004  154736           MOV     @4(SP),$FATAL     ;;GET ERROR #
024250  062766  000002  000004           ADD     #2,4(SP)                  ;;BUMP RETURN ADDR.
024256  005267  154722                   INC     $MSGTYPE          ;;TELL APT TO TAKE ERROR
024262  105067  000106           12$:    CLRB    $FFLG             ;;CLEAR FATAL FLAG
024266  105067  000101                   CLRB    $LFLG             ;;CLEAR LOG FLAG
024272  105067  000074                   CLRB    $MFLG             ;;CLEAR MESSAGE FLAG
024276  012601                           MOV     (SP)+,R1          ;;POP STACK INTO R1
024300  012600                           MOV     (SP)+,R0          ;;POP STACK INTO R0
024302  000207                           RTS     PC                ;;RETURN
024304                           $ATY6:
024304  010046                           MOV     R0,-(SP)          ;;PUSH R0 ON STACK
024306  016700  155200                   MOV     $APTR,R0
024312  162700  001344                   SUB     #$ASTAT,R0        ;;GET SIZE OF STAT TABLE
024316  005767  154662           1$:     TST     $MSGTY            ;;SEE IF DONE LAST COMMUNICATION
024322  001375                           BNE     1$                ;;IF NOT:  WAIT
024324  010067  154672                   MOV     R0,$MSGLG         ;;SET MESSAGE LENGTH
024330  012767  001344  154662           MOV     #$ASTAT,$MSGAD    ;;SET MESSAGE ADDR.
024336  012767  000002  154640           MOV     #2,$MSGTY         ;;TELL APT TO TAKE STATS.
024344  012600                           MOV     (SP)+,R0          ;;POP STACK INTO R0
024346  000207                           RTS     PC                ;;RETURN
024350                           $ATY7:
024350  010046                           MOV     R0,-(SP)          ;;PUSH R0 ON STACK
024352  012701  001344                   MOV     #$ASTAT,R1        ;;GET START OF TABLE
024356  005721                   1$:     TST     (R1)+             ;;END OF TABLE?
024360  100402                           BMI     2$                ;;IF SO:  BR
024362  005021                           CLR     (R1)+             ;;CLEAR ERROR COUNT
024364  000774                           BR      1$                ;;KEEP CLEARING
024366                           2$:
024366  012600                           MOV     (SP)+,R0          ;;POP STACK INTO R0
024370  000207                           RTS     PC                ;;RETURN
024372     000                   $MFLG:  .BYTE   0                 ;;MESSG. FLAG
024373     000                   $LFLG:  .BYTE   0                 ;;LOG FLAG
024374     000                   $FFLG:  .BYTE   0                 ;;FATAL FLAG
        024376                           .EVEN
        000200                   APTSIZE=200
        000001                   APTENV=001
        000100                   APTSPOOL=100
        000040                   APTCSUP=040
                                 ;;********************************************************
```

```

                                        .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

024376                                  $TYPDS:
024376  010046                                  MOV     R0,-(SP)                ;;PUSH R0 ON STACK
024400  010146                                  MOV     R1,-(SP)                ;;PUSH R1 ON STACK
024402  010246                                  MOV     R2,-(SP)                ;;PUSH R2 ON STACK
024404  010346                                  MOV     R3,-(SP)                ;;PUSH R3 ON STACK
024406  010546                                  MOV     R5,-(SP)                ;;PUSH R5 ON STACK
024410  012746  020200                          MOV     #20200,-(SP)            ;;SET BLANK SWITCH AND SIGN
024414  016605  000020                          MOV     20(SP),R5               ;;GET THE INPUT NUMBER
024420  100004                                  BPL     1$                      ;;BR IF INPUT IS POS.
024422  005405                                  NEG     R5                      ;;MAKE THE BINARY NUMBER POS.
024424  112766  000055  000001                  MOVB    #'-,1(SP)               ;;MAKE THE ASCII NUMBER NEG.
024432  016700  154142          1$:     MOV     RELOCF, R0              ;GET RELOCATION FACTOR.
024436  012703  024620                          MOV     #$DBLK,R3               ;;SETUP THE OUTPUT POINTER
024442  060003                                  ADD     R0,     R3              ;ADD IN RELOCATION FACTOR.
024444  112723  000040                          MOVB    #' ,(R3)+               ;;SET THE FIRST CHARACTER TO A BLANK
024450  005002                          2$:     CLR     R2                      ;;CLEAR THE BCD NUMBER
024452  016001  024610                          MOV     $DTBL(R0),R1            ;;GET THE CONSTANT
024456  160105                          3$:     SUB     R1,R5                   ;;FORM THIS BCD DIGIT
024460  002402                                  BLT     4$                      ;;BR IF DONE
024462  005202                                  INC     R2                      ;;INCREASE THE BCD DIGIT BY 1
024464  000774                                  BR      3$
024466  060105                          4$:     ADD     R1,R5                   ;;ADD BACK THE CONSTANT
024470  005702                                  TST     R2                      ;;CHECK IF BCD DIGIT=0
024472  001002                                  BNE     5$                      ;;FALL THROUGH IF 0
024474  105716                                  TSTB    (SP)                    ;;STILL DOING LEADING 0'S?
024476  100407                                  BMI     7$                      ;;BR IF YES
024500  106316                          5$:     ASLB    (SP)                    ;;MSD?
024502  103003                                  BCC     6$                      ;;BR IF NO
024504  116663  000001  177777                  MOVB    1(SP),-1(R3)            ;;YES--SET THE SIGN
024512  052702  000060          6$:     BIS     #'0,R2                  ;;MAKE THE BCD DIGIT ASCII
024516  052702  000040          7$:     BIS     #' ,R2                  ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
024522  110223                                  MOVB    R2,(R3)+                ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
024524  005720                                  TST     (R0)+                   ;;JUST INCREMENTING
024526  020067  155110                          CMP     R0,     .EIGHT          ;CHECK THE TABLE INDEX
024532  103746                                  BLO     2$                      ;GO DO THE NEXT DIGIT
024534  101002                                  BHI     8$                      ;;GO TO EXIT
024536  010502                                  MOV     R5,R2                   ;;GET THE LSD
024540  000764                                  BR      6$                      ;;GO CHANGE TO ASCII
024542  105726                          8$:     TSTB    (SP)+                   ;;WAS THE LSD THE FIRST NON-ZERO?
024544  100003                                  BPL     9$                      ;;BR IF NO
024546  116663  177777  177776                  MOVB    -1(SP),-2(R3)           ;;YES--SET THE SIGN FOR TYPING
024554  105013                          9$:     CLRB    (R3)                    ;;SET THE TERMINATOR
024556  012605                                  MOV     (SP)+,R5                ;;POP STACK INTO R5
024560  012603                                  MOV     (SP)+,R3                ;;POP STACK INTO R3
```

```
024562  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
024564  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
024566  012600                            MOV     (SP)+,RC        ;;POP STACK INTO RO
024570  004567  176662                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
024574  024620                            .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
024576  016666  000002  000004            MOV     2(SP),4(SP)     ;;ADJUST THE STACK
024604  012616                            MOV     (SP,+,(SP)
024606  000002                            RTI                     ;;RETURN TO USER
024610  023420                  $DTBL:    10000.
024612  001750                            1000.
024614  000144                            100.
024616  000012                            10.
024620  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

024630  017646  000000          $TYPOS:   MOV     @(SP),-(SP)     ;;PICKUP THE MODE
024634  116667  000001  000213            MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
024642  112667  000211                    MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
024646  062716  000002                    ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
024652  000406                            BR      $TYPON
024654  112767  000001  000173  $TYPOC:   MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
024662  112767  000006  000167            MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
024670  112767  000005  000156  $TYPON:   MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
024676  010346                            MOV     R3,-(SP)        ;;SAVE R3
024700  010446                            MOV     R4,-(SP)        ;;SAVE R4
024702  010546                            MOV     R5,-(SP)        ;;SAVE R5
024704  116704  000147                    MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
024710  005404                            NEG     R4
024712  062704  000006                    ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
024716  110467  000134                    MOVB    R4,$OMODE       ;;SAVE IT FOR USE
024722  116704  000127                    MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
024726  016605  000012                    MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
024732  005003                            CLR     R3              ;;CLEAR THE OUTPUT WORD
```

```
024734  006105                          1$:     ROL     R5                      ;;ROTATE MSB INTO "C"
024736  000404                                  BR      3$                      ;;GO DO MSB
024740  006105                          2$:     ROL     R5                      ;;FORM THIS DIGIT
024742  006105                                  ROL     R5
024744  006105                                  ROL     R5
024746  010503                                  MOV     R5,R3
024750  006103                          3$:     ROL     R3                      ;;GET LSB OF THIS DIGIT
024752  105367  000100                          DECB    $OMODE                  ;;TYPE THIS DIGIT?
024756  100017                                  BPL     7$                      ;;BR IF NO
024760  042703  177770                          BIC     #177770,R3              ;;GET RID OF JUNK
024764  001002                                  BNE     4$                      ;;TEST FOR 0
024766  005704                                  TST     R4                      ;;SUPPRESS THIS 0?
024770  001403                                  BEQ     5$                      ;;BR IF YES
024772  005204                          4$:     INC     R4                      ;;DON'T SUPPRESS ANYMORE 0'S
024774  052703  000060                          BIS     #'0,R3                  ;;MAKE THIS DIGIT ASCII
025000  052703  000040                  5$:     BIS     #' ,R3                  ;;MAKE ASCII IF NOT ALREADY
025004  110367  000042                          MOVB    R3,8$                   ;;SAVE FOR TYPING
025010  004567  176442                          JSR     R5,     $PRINT          ;GO PRINT OUT THE FOLLOWING MESSAGE.
025014  025052                                  .WORD   8$                      ;ADDRESS OF MESSAGE TO BE TYPED
025016  105367  000032                  7$:     DECB    $OCNT                   ;;COUNT BY 1
025022  003346                                  BGT     2$                      ;;BR IF MORE TO DO
025024  002402                                  BLT     6$                      ;;BR IF DONE
025026  005204                                  INC     R4                      ;;INSURE LAST DIGIT ISN'T A BLANK
025030  000743                                  BR      2$                      ;;GO DO THE LAST DIGIT
025032  012605                          6$:     MOV     (SP)+,R5                ;;RESTORE R5
025034  012604                                  MOV     (SP)+,R4                ;;RESTORE R4
025036  012603                                  MOV     (SP)+,R3                ;;RESTORE R3
025040  016666  000002  000004                  MOV     2(SP),4(SP)             ;;SET THE STACK FOR RETURNING
025046  012616                                  MOV     (SP)+,(SP)
025050  000002                                  RTI                             ;;RETURN
025052     000                          8$:     .BYTE   0                       ;;STORAGE FOR ASCII DIGIT
025053     000                                  .BYTE   0                       ;;TERMINATOR FOR TYPE ROUTINE
025054     000                          $OCNT:  .BYTE   0                       ;;OCTAL DIGIT COUNTER
025055     000                          $OFILL: .BYTE   0                       ;;ZERO FILL SWITCH
025056  000000                          $OMODE: .WORD   0                       ;;NUMBER OF DIGITS TO TYPE
                                        ;ERROR TRAP SERVICE ROUTINE
025060  005727                          ERRTRP: TST     (PC)+                   ;CHECK IF PREV TRAP TO 4 REPORTED
025062  000000                          1$:     .WORD   0                       ;CONTAINS ERROR REPORTED FLAG
025064  001010                                  BNE     2$                      ;BRANCH IF NOT REPORTED
025066  005267  177770                          INC     1$                      ;SET DOUBLE TRAP FLAG.
025072  011667  154070                          MOV     (SP),   $TMP3           ;SAVE THE BAD PC FOR TYPOUT.
025076  00476.' 174502                          JSR     PC,     $ERROR          ;*** ERROR *** (GO TYPE A MESSAGE)
025102  000031                                  .WORD   31                      ;ERROR TYPE CODE.
025104  000401                                  BR      3$                      ;SKIP HALT
025106  000000                          2$:     HALT                            ;ERROR! SECOND TRAP TO 4 OCCURRED
                                                                                ;BEFORE FIRST WAS PRINTED
025110  005067  177746                  3$:     CLR     1$
025114  000002                                  RTI                             ;RETURN TO PROGRAM AND TRY TO RECOVER

                                        .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
                                        ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
025116                                  $TYPAD:
025116  010046                                  MOV     R0,-(SP)                ;;PUSH R0 ON STACK
025120  010146                                  MOV     R1,-(SP)                ;;PUSH R1 ON STACK
025122  010246                                  MOV     R2,-(SP)                ;;PUSH R2 ON STACK
025124  010346                                  MOV     R3,-(SP)                ;;PUSH 93 ON STACK
```

```
025126  016602  000012                    MOV     12(SP), R2       ;GET BASE ADDRESS
025132  005003                            CLR     R3               ;WORKING & INDEX REGISTER
025134  005767  153446                    TST     MMAVA            ;CHECK FOR MEM MGMT AVAILABLE
025140  001430                            BEQ     1$               ;BRANCH IF NO MEM MGMT
025142  032737  000001  177572            BIT     #1,      @#SR0   ;CHECK IF MEM MGMT ENABLED
025150  001424                            BEQ     1$               ;BRANCH IF MEM MGMT NOT ENABLED
025152  010201                            MOV     R2,      R1      ;COPY VIRTUAL ADR
025154  006101                            ROL     R1               ;SHUFFLE BITS 13,14,15 INTO 1,2,3
025156  006101                            ROL     R1
025160  006101                            ROL     R1
025162  006101                            ROL     R1
025164  006101                            ROL     R1
025166  042701  177761                    BIC     #177761, R1      ;CLR ALL EXCEPT BITS 1,2,3
025172  062701  172340                    ADD     #KIPAR0, R1      ;SET TO APPROPRIATE PAR
025176  011101                            MOV     (R1),    R1      ;GET CONTENTS OF PAR
025200  012700  000006                    MOV     #6,      R0      ;SET UP COUNTER
025204  006301                    4$:     ASL     R1               ;SHIFT PAR
025206  006103                            ROL     R3               ;SAVE OVERFLOW BITS
025210  077003                            SOB     R0,      4$      ;COUNT SIX SHIFTS
025212  042702  160000                    BIC     #160000, R2      ;SAVE BANK BITS
025216  060102                            ADD     R1,      R2      ;COMPUTE PHYSICAL ADDRESS
025220  005503                            ADC     R3               ;MAKE SURE CARRY ISN'T LOST!
025222  006302                    1$:     ASL     R2               ;FIRST DIGIT TO R3
025224  006103                            ROL     R3
025226  012700  000006                    MOV     #6,R0            ;DIGIT COUNT
025232  000404                            BR      3$               ;PRINT FIRST DIGIT
025234  006302                    2$:     ASL     R2
025236  006103                            ROL     R3
025240  005301                            DEC     R1
025242  001374                            BNE     2$
025244  012701  000003            3$:     MOV     #3,R1            ;DIGIT SHIFT COUNT
025250  062703  000060                    ADD     #60,     R3      ;MAKE IT AN ASCII DIGIT
025254  110367  000036                    MOVB    R3,      8$      ;LOAD DIGIT INTO MESSAGE
025260  004567  176172                    JSR     R5,      $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
025264  025316                            .WORD   8$               ;ADDRESS OF MESSAGE TO BE TYPED
025266  005003                            CLR     R3               ;CLEAR INDEX
025270  005300                            DEC     R0               ;DEC DIGIT COUNT
025272  001360                            BNE     2$
025274  012603                            MOV     (SP)+,R3         ;;POP STACK INTO R3
025276  012602                            MOV     (SP)+,R2         ;;POP STACK INTO R2
025300  012601                            MOV     (SP)+,R1         ;;POP STACK INTO R1
025302  012600                            MOV     (SP)+,R0         ;;POP STACK INTO R0
025304  012616                            MOV     (SP)+,   (SP)    ;ADJUST THE STACK TO CLEAR DATA
025306  004567  176144                    JSR     R5,      $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
025312  026751                            .WORD   FILL2            ;ADDRESS OF MESSAGE TO BE TYPED
025314  000207                            RTS     PC               ;RETURN
025316     000                    8$:     .BYTE   0                ;ONE DIGIT MESSAGE BUFFER
025317     000                            .BYTE   0                ;MESSAGE TERMINATOR

                                          .SBTTL  STANDARD PROGRAM MESSAGES
                                  ;;*********************************************************
                                  ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
                                  ;THE PROGRAM
                                  ;;*********************************************************
025320  005015  052113  030461    MMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
025326  024040  042515  047515
```

```
 4955  025334  054522  046440  047101
 4956  025342  043501  046505  047105
 4957  025350  024524  040440  040526
 4958  025356  046111  041101  042514
 4959  025364     000
 4960  025365     015  046412  046505  MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
 4961  025372  051117  020131  040515
 4962  025400  035120     000
 4963  025403     015  041012  052131  BYTMES: .ASCIZ  <15><12>'BYTE MEMORY MAP:'
 4964  025410  020105  042515  047515
 4965  025416  054522  046440  050101
 4966  025424  000072
 4967  025426  005015  040520  044522  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
 4968  025434  054524  046440  046505
 4969  025442  051117  020131  040515
 4970  025450  035120     000
 4971  025453     015  043012  047522  FROM:   .ASCIZ  <15><12>'FROM '
 4972  025460  020115     000
 4973  025463     040  047524  000040  TO:     .ASCIZ  ' TO '
 4974  025470  005015  047111  052523  INSUFF: .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
 4975  025476  043106  041511  042511
 4976  025504  052116  046440  046505
 4977  025512  051117  027131  027056
 4978  025520  044506  051522  020124
 4979  025526  033061  020113  047516
 4980  025534  020124  046101  020114
 4981  025542  044124  051105  020505
 4982  025550     000
 4903  025551     015  047012  020117  MTR:    .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
 4984  025556  040520  044522  054524
 4985  025564  051040  043505  051511
 4986  025572  042524  051522  043040
 4987  025600  052517  042116     000
 4988  025605     015  051012  051505  PWRMSG: .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
 4989  025612  040524  052122  047111
 4990  025620  020107  043101  042524
 4991  025626  020122  020101  047520
 4992  025634  042527  020122  040506
 4993  025642  046111  051125  006505
 4994  025650  000012
 4995  025652  005015  047516  050040  NOPES:  .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
 4996  025660  051101  052111  020131
 4997  025666  051105  047522  051522
 4998  025674  043040  052517  042116
 4999  025702  047440  020116  042515
 5000  025710  047515  054522  051440
 5001  025716  040503  006516  000012
 5002  025724  005015  051120  043517  PROREL: .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
 5003  025732  040522  020115  047516
 5004  025740  020127  042522  044523
 5005  025746  042504  020123  040502
 5006  025754  045503  040440  020124
 5007  025762  020060  047524  034040
 5008  025770     113
 5009  025771     015  044012  052111          .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
 5010  025776  041440  047117  044524
 5011  026004  052516  020105  047506
 5012  026012  020122  047516  046522
 5013  026020  046101  051040  047125
 5014  026026  044516  043516  005015
 5015  026034     000
 5016  026035     015  051012  043505  MX1:    .ASCIZ  <15><12>'REGISTER AT '
 5017  026042  051511  042524  020122
 5018  026050  052101  000040
 5019  026054  041440  047117  051124  MX2:    .ASCIZ  ' CONTROLS '
 5020  026062  046117  020123     000
 5021  026067     015  041412  051117  MX3:    .ASCIZ  <15><12>'CORE PARITY '
 5022  026074  020105  040520  044522
 5023  026102  054524  000040
 5024  026106  005015  047515  020123  MX4:    .ASCIZ  <15><12>'MOS PARITY '
 5025  026114  040520  044522  054524
 5026  026122  000040
 5027  026124  005015  051515  030461  MX5:    .ASCIZ  <15><12>'MS11-K CSR '
 5028  026132  045455  041440  051123
 5029  026140  000040
 5030  026142  051515  030461  045455  MX6:    .ASCIZ  'MS11-K MEMORY PRESENT!! TO COMPLETELY TEST RUN DZMML...'
 5031  026150  046440  046505  051117
 5032  026156  020131  051120  051505
 5033  026164  047105  020524  020041
 5034  026172  047524  041440  046517
 5035  026200  046120  052105  046105
 5036  026206  020131  042524  052123
 5037  026214  051040  047125  042040
 5038  026222  046532  046115  027056
 5039  026230  000056
 5040  026232  005015  047516  046440  NOMEM:  .ASCIZ  <15><12>'NO MEMORY FOUND.'
 5041  026240  046505  051117  020131
 5042  026246  047506  047125  027104
 5043  026254     000
 5044  026255     015  005012  044412  FADMES: .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
 5045  026262  050116  052125  040440
 5046  026270  046114  050040  051101
 5047  026276  046501  052105  051105
 5048  026304  020123  047111  047440
 5049  026312  052103  046101     056
 5050  026317     015  043012  051111          .ASCIZ  <15><12>'FIRST ADDRESS:  '
 5051  026324  052123  040440  042104
 5052  026332  042522  051523  020072
 5053  026340  000040
 5054  026342  005015  040514  052123  LADMES: .ASCIZ  <15><12>'LAST ADDRESS:    '
 5055  026350  040440  042104  042522
 5056  026356  051523  020072  020040
 5057  026364     000
 5058  026365     015  037412  042101  BADADR: .ASCIZ  <15><12>'?ADDRESS IN UNMAPPED BANK?'
 5059  026372  051104  051505  020123
 5060  026400  047111  052440  046516
 5061  026406  050101  042520  020104
 5062  026414  040502  045516  000077
 5063  026422  005015  042523  042514  CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
 5064  026430  052103  041440  047117
 5065  026436  052123  047101  035124
 5066  026444     000
```

```
026445       015  052412  042516   UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
026452    050130  041505  042524
026460    020104  042515  047515
026466    054522  050040  051101
026474    052111  020131  051105
026502    047522  000122
026506    005015  051120  043517   PRELOC: .ASCIZ  <15 <12>'PROGRAM RELOCATED TO '
026514    040522  020115  042522
026522    047514  040503  042524
026530    020104  047524  000040
026536    005015  047515  042522   MTOE:    .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
026544    052040  040510  020116
026552    047117  020105  040520
026560    044522  054524  042440
026566    051122  051117  043040
026574    052517  042116  000056
026602    005015  041523  047101   SCANM:   .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
026610    044516  043516  046440
026616    046505  051117  020131
026624    047506  020122  040502
026632    020104  040520  044522
026640    054524  000056
026644    005015  040520  044522   PEWNC:   .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
026652    054524  042440  051122
026660    051117  053440  046111
026666    020114  047516  020124
026674    046103  040505  027122
026702       000
026703       015  047012  020117   NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
026710    042515  047515  054522
026716    052040  051505  042524
026724    027104     000
026727       015  051412  044513   SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
026734    050120  047111  020107
026742    042524  052123  021440
026750       000
026751       377  000377           FILL2:  .ASCIZ  <377><377>

                                   .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
                                   ;;*****************************************************
                                   ;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
                                   ;;*****************************************************
026754    040520  044522  054524   DM1:     .ASCIZ  'PARITY REGISTER DATA ERROR.'
026762    051040  043505  051511
026770    042524  020122  040504
026776    040524  042440  051122
027004    051117  000056
027010    042101  051104  051505   DM2:     .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
027016    020123  042524  052123
027024    042440  051122  051117
027032    052050  052123  026461
027040    024465  000056
027044    047503  051516  040524   DM4:     .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
027052    052116  042040  052101
027060    020101  051105  047522
027066    024122  051524  033124
```

```
 5123  027074  030455  024460  000056
 5124  027102  047522  040524  044524  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
 5125  027110  043516  041040  052111
 5126  027116  042440  051122  051117
 5127  027124  052050  052123  030461
 5128  027132  030455  024462  000056
 5129  027140  047515  020123  042522  DM6:    .ASCIZ  'MOS REFRESH TEST ERROR (TST 30-31).'
 5130  027146  051106  051505  020110
 5131  027154  042524  052123  042440
 5132  027162  051122  051117  024040
 5133  027170  051524  020124  030063
 5134  027176  031455  024461  000056
 5135  027204  020063  047530  020122  DM7:    .ASCIZ  '3 XOR 9 PATTERN ERROR(TST13-16).'
 5136  027212  020071  040520  052124
 5137  027220  051105  020116  051105
 5138  027226  047522  024122  051524
 5139  027234  030524  026463  033061
 5140  027242  027051     000
 5141  027245     115  051101  044103  DM10:   .ASCIZ  "MARCHING 1'S AND 0'S ERROR(TST 27)."
 5142  027252  047111  020107  023461
 5143  027260  020123  047101  020104
 5144  027266  023460  020123  051105
 5145  027274  047522  024122  051524
 5146  027302  020124  033462  027051
 5147  027310     000
 5148  027311     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST17).'
 5149  027316  020131  042515  047515
 5150  027324  054522  040440  042104
 5151  027332  042522  051523  042440
 5152  027340  051122  051117  052050
 5153  027346  052123  033461  027051
 5154  027354     000
 5155  027355     104  052101  050111  DM12:   .ASCIZ  "DATIP WITH WRONG PARITY DIDN'T TRAP(TST17)."
 5156  027362  053440  052111  020110
 5157  027370  051127  047117  020107
 5158  027376  040520  044522  054524
 5159  027404  042040  042111  023516
 5160  027412  020124  051124  050101
 5161  027420  052050  052123  033461
 5162  027426  027051     000
 5163  027431     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
 5164  027436  050040  051101  052111
 5165  027444  020131  051124  050101
 5166  027452  042520  026104  041040
 5167  027460  052125  047040  020117
 5168  027466  042522  044507  052123
 5169  027474  051105  051440  047510
 5170  027502  051527  042440  051122
 5171  027510  051117  043040  040514
 5172  027516  027107     000
 5173  027521     120  051101  052111  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
 5174  027526  020131  042522  044507
 5175  027534  052123  051105  047040
 5176  027542  052117  046440  050101
 5177  027550  042520  020104  051501
 5178  027556  041440  047117  051124
 5179  027564  046117  044514  043516
 5180  027572  052040  044510  020123
 5181  027600  042101  051104  051505
 5182  027606  024123  051524  030524
 5183  027614  024467  000056
 5184  027620  047515  042522  052040  DM16:    .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
 5185  027626  040510  020116  047117
 5186  027634  020105  042522  044507
 5187  027642  052123  051105  044440
 5188  027650  042116  041511  052101
 5189  027656  042105  050040  051101
 5190  027664  052111  020131  051105
 5191  027672  047522  027122     000
 5192  027677     104  052101  020101  DM17:    .ASCIZ  "DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17)."
 5193  027704  044123  052517  042114
 5194  027712  023516  020124  040510
 5195  027720  042526  041440  040510
 5196  027726  043516  042105  053440
 5197  027734  042510  020116  040520
 5198  027742  044522  054524  042440
 5199  027750  051122  051117  052040
 5200  027756  040522  050120  042105
 5201  027764  052050  052123  033461
 5202  027772  027051     000
 5203  027775     122  047101  047504  DM20:    .ASCIZ  'RANDOM DATA ERROR(TST20).'
 5204  030002  020115  040504  040524
 5205  030010  042440  051122  051117
 5206  030016  052050  052123  030062
 5207  030024  027051     000
 5208  030027     111  051516  051124  DM21:    .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST21-26).'
 5209  030034  041525  044524  047117
 5210  030042  042440  042530  052503
 5211  030050  044524  047117  042440
 5212  030056  051122  051117  052050
 5213  030064  052123  030462  031055
 5214  030072  024466  000056
 5215  030076  051120  043517  040522  DM23:    .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
 5216  030104  020115  047503  042504
 5217  030112  041440  040510  043516
 5218  030120  042105  053440  042510
 5219  030126  020116  042522  047514
 5220  030134  040503  042524  027104
 5221  030142     000
 5222  030143     124  040522  050120  DM24:    .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
 5223  030150  042105  020054  052502
 5224  030156  020124  047516  051040
 5225  030164  043505  051511  042524
 5226  030172  020122  040510  020104
 5227  030200  051105  047522  020122
 5228  030206  044502  020124  042523
 5229  030214  027124     000
 5230  030217     124  040522  050120  DM25:    .ASCIZ  'TRAPPED TO 114.'
 5231  030224  042105  052040  020117
 5232  030232  030461  027064     000
 5233  030237     106  044501  042514  DM26:    .ASCIZ  'FAILED TO TRAP.'
 5234  030244  020104  047524  052040
```

```
5235  030252  040522  027120     000
5236  030257     050  041501  044524  DM27:   .ASCIZ  "(ACTION ENABLE WASN'T SET)."
5237  030264  047117  042440  040516
5238  030272  046102  020105  040527
5239  030300  047123  052047  051440
5240  030306  052105  027051     000
5241  030313     015  052012  040522  DM31:   .ASCIZ  <15,<12>'TRAPPED TO 4 '
5242  030320  050120  042105  052040
5243  030326  020117  020064     000
5244
5245                                  ;;*************************************************************
5246                                  ;DATA COLUMN HEADINGS
5247                                  ;;*************************************************************
5248
5249  030333     120  004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
5250  030340  004507  027523  004502
5251  030346  040527  000123
5252  030352  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
5253  030360  050057  004503  040515
5254  030366  051411  041057  053411
5255  030374  051501     000
5256  030377     126  050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
5257  030404  027520  041520  046411
5258  030412  004501  027523  000102
5259  030420  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
5260  030426  050057  004503  042522
5261  030434  004507  040515     000
5262  030441     126  050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
5263  030446  027520  041520  046411
5264  030454  052501  004524  042522
5265  030462  004507  027523  004502
5266  030470  040527  000123
5267  030474  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
5268  030502  050057  004503  052511
5269  030510  004524  040515  051411
5270  030516  041057  0534'1  051501
5271  030524     000
5272  030525     126  050057  004503  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B   , WAS'
5273  030532  027520  041520  051411
5274  030540  041522  046440  004501
5275  030546  051504  020124  040515
5276  030554  051411  041057  053411
5277  030562  051501     000
5278  030565     126  050057  004503  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
5279  030572  027520  041520  052011
5280  030600  050122  050057  000103
5281  030606  027526  041520  050011  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
5282  030614  050057  004503  051124
5283  030622  027520  041520  051011
5284  030630  043505  053411  051501
5285  030636     000
5286  030637     126  050057  004503  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
5287  030644  027520  041520  051011
5288  030652  043505  053411  051501
5289  030660     000
5290  030661     122  043505  053411  DH30:   .ASCIZ  'REG    WAS     MA      WAS'
```

```
030666  051501  046411  004501
030674  040527  000123

                                  ;****************************************************************
                                  ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
                                  ;****************************************************************
030700     000     377     000    DF1:     .BYTE   0,-1,0,0
030703     000
030704     000     377     377    DF2:     .BYTE   0,-1,-1,0,0
030707     000     000
030711     000     377     377    DF3:     .BYTE   0,-1,-1,-2,-2
030714     376     376
030716     000     377     377    DF14:    .BYTE   0,-1,-1,-1,0,0
030721     377     000     000
030724     000     377     000    DF21:    .BYTE   0,-1,0,-1,0,0
030727     377     000     000
030732     377     000     377    DF30:    .BYTE   -1,0,-1,-2
030735     376
                                           .EVEN

        032110                    . = 32110                         ;THE LOADERS ARE SAVE HERE TO END OF 8K

        000001                             .END
```

```
ABASE = 000000         384     425
ACDW1 = 000000         384     427
ACDW2 = 000000         384     428
ACPUOP= 000000         384     399
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#   2426     2512     3740
AENV  = 000000         384     395
AENVM = 000000         384     396
AFATAL= 000000         384     387
AMADR1= 000000         384     412
AMADR2= 000000         384     416
AMADR3= 000000         384     419
AMADR4= 000000         384     422
AMAMS1= 000000         384     406
AMAMS2= 000000         384     414
AMAMS3= 000000         384     417
AMAMS4= 000000         384     420
AMSGAD= 000000         384     392
AMSGLG= 000000         384     393
AMSGTY= 000000         384     386
AMTYP1= 000000         384     407
AMTYP2= 000000         384     415
AMTYP3= 000000         384     418
AMTYP4= 000000         384     421
APASS = 000000         384     389
APRIOR= 000000         384
APTCSU= 000040        4588    4729#
APTENV= 000001        4177    4581     4649     4727#
APTSIZ= 000200         876    4726#
APTSPO= 000100        4583    4651     4728#
ASWREG= 000000         384     397
ATESTN= 000000         384     388
AUNIT = 000000         384     391
AUSWR = 000000         384     398
AVECT1= 000000         384     423
AVECT2= 000000         384     424
BADADR  026365        1439    5058#
BANKNO  016074        1625    1634     1660     1670     3387#
BITPT   001544         522#   1127*    1128*    1164     1165     1166     1167     1175*    1176*    1185     1187     1190     1193*
```

```
                                 2402     2404     2460     2462     3138*    3139*    3146     3148     3151*    3152*    3155     3157     3162
                                 3163     3182     3185*    3188     3189     3203     3204     3205     3207     3216*    3217*    3221*    3226
                                 3228     3249*    3250*    3252     3254     3256     3258     3262     3263     3293*    3295     3297     3300
                                 3307     3309     3314     3315     3330     3332     3340*    3341*    3343     3345     3350*    3352     3354
                                 3356     3390     3391
BIT0   = 000001                   123#     235      251     1032     1206     1362     1373     1463     1484     2449     2478     2494     3138
                                 3757     3790     3921     4063
BIT00  = 000001                   113#     123
BIT01  = 000002                   112#     122
BIT02  = 000004                   111#     121
BIT03  = 000010                   110#     120
BIT04  = 000020                   109#     119
BIT05  = 000040 .                 108#     1.8
BIT06  = 000100                   107#     117
BIT07  = 000200                   106#     116
BIT08  = 000400                   105#     115     4012
BIT09  = 001000                   104#     114     4022     4191
BIT1   = 000002                   122#
BIT10  = 002000                   103#    4163
BIT11  = 004000                   102#    4029
BIT12  = 010000                   101#
BIT13  = 020000                   100#    3164     3191     3264     3301     4171
BIT14  = 040000                    99#     217     3998
BIT15  = 100000                    98#    2478     3216     3517
BIT2   = 000004                   121#
BIT3   = 000010                   120#
BIT4   = 000020                   119#
BIT5   = 000040                   118#
BIT6   = 000100                   117#
BIT7   = 000200                   116#
BIT8   = 000400                   115#    3221     3550
BIT9   = 001000                   114#
BLKMSK   001606                   547#    3141     3212     3244     4049*    4054*
BPTVEC=  000014                   130#
BYTMES   025403                  1026     4963#
CASFLG   001616                   551#     930*     933*    1476
CASREG   001620                   552#    1478*
CKPMER   017602                  1748     2410     3321     3749#
CLRPAR   020234                  1126     1201     3794     3865#
CONST    026422                  1451     5063#
CR     = 000015                    38#    4627     4637
CRLF   = 000200                    39#     906     4598     4637
CTRLS    005046                  1266     1276#
DDISP  = 177570                    45#     361      864
DF1      030700                   719     5297#
DF14     030716                   774      779      784      806      811      816     5303#
DF2      030704                   724      734      739      744      749      754      795      821      826     5299#
DF21     030724                   800     5305#
DF3      030711                   729      759      764      769      790     5301#
DF30     030732                   831      836     5307#
DH1      030333                   717     5249#
DH12     030377                   762      767     5256#
DH14     030420                   772      782     5259#
DH15     030441                   777     5262#
DH2      030352                   722      727      732      737      742      747      752      757      788      793     5252#
DH21     030474                   798     5267#
```

```
DH23     030525            804     5272#
DH24     030565            809     5278#
DH25     030606            814     5281#
DH26     030637            819      824     5286#
DH30     030661            829     5290#
DIDBH    012746           2704#
DIDBL    012654           2655#
DIDO     012564           2606#
DIPDO    013044           2754#
DISPLA   001142            361#     864*     872*     881*    3584*    3637*    4044*    4162*
DISPRE   000174            195#     872
DM1      026754            716      776     5109#
DM10     027245            751     5141#
DM11     027311            756     5148#
DM12     027355            761     5155#
DM13     027431            766     5163#
DM14     027521            771     5173#
DM16     027620            781     5184#
DM17     027677            786     5192#
DM2      027010            721      726     5114#
DM20     027775            792     5203#
DM21     030027            797     5208#
DM23     030076            803     5215#
DM24     030143            808     5222#
DM25     030217            813     5230#
DM26     030237            818     5233#
DM27     030257            823     5236#
DM31     030313            833     5241#
DM4      027044            731     5119#
DM5      027102            736     5124#
DM6      027140            741     5129#
DM7      027204            746     5135#
DONE     013752           3031#
DPDBH    013230           2852#
DPDBL    013136           2803#
DSWR   = 177570             44#     360      863
DT1      001646            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   021570           4048     4121     4134#
ERRTRP   025060            206      997     1105     3675     4879#
ERRVEC=  000004            126#     205      861      862*     873*     920*     931*     957*     997*    1020*    1030*    1090*    1105*
                          3578*    3631*    3671*    3675*    4003     4004*    4006*    4009*
FADMAP   001570            540#    1414*    1415*    3205     3207     3330     3332     3354     3356     4063*    4090     4091     4099
                          4101
FADMES   026255           1367     5044#
```

```
FADMSK  001566           538#    3358     4124*    4126*
FILL2   026751          4943     5103#
FLAG8K  001557           532#    3197*    3287*    3290*    3308*
FROM    025453          1039     3941     4971#
FSTADR  001562           535#    1350     1448*    3734     4055
GMPR    004046          1006     1086#
GMPRA   004074          1091#    1103
GMPRB   004110          1090     1097#
GMPRC   004116          1095     1099#
GMPRD   004156          1088     1111#
GNS     = ****** U       194      905
HT      = 000011          36#    4596     4637
IMPCHE= 177746           186#     932*
IMPCK   003246           919      920      929#
INITDN  015030          1521     1557     1584     1659     1669     3212#
INITEX  015142          3208     3210     3229     3231#
INITMM  014372          1504     1547     1595     1624     1633     1701     1715     1751     1772     1795     1820     1830     1873
                        1922     1933     1976     2025     2036     2084     2210     2221     2269     2401     2548     2609     2658
                        2707     2757     2806     2855     2899     2951     2970     2983     3003     3016     3138#    3407
INSERT  021070          4046#
INSUFF  025470          1008     4974#
IOTVEC= 000020           131#
KIPAR0= 172340           166#     242      261*    3123*    3544*    3610*    4912
KIPAR1= 172342           167#     267*    3124*    3545*    3611*
KIPAR2= 172344           168#     968      988*    1065*    1174*    1178     2477     3125*    3144*    3150*    3161     3218*    3248*
                        3266     3339*    3371     3526*    3529*    3544     3605*    3799     3800*    3839*    3844*
KIPAR3= 172346           169#    1062     1065     1066*    3126*    3161*    3166*    3266*    3267*    3515*    3518*    3526     3545
                        3606*
KIPAR4= 172350           170#    3127*
KIPAR5= 172352           171#    3128*
KIPAR6= 172354           172#    3129*
KIPAR7= 172356           173#    3130*
KIPDR0= 172300           155#     237     3115*
KIPDR1= 172302           156#    3116*
KIPDR2= 172304           157#    3117*
KIPDR3= 172306           158#    3118*
KIPDR4= 172310           159#    3119*
KIPDR5= 172312           160#    3120*
KIPDR6= 172314           161#    3121*
KIPDR7= 172316           162#    3122*
LADMAP  001602           546#    1442*    1443*    1466*    1467*    3155     3157     3174     3176     3189     3198     3226     3228
                        3240     3242     3256     3258     3275     3277     3297     3307     3309     4097     4098     4099     4101
LADMES  026342          1394     5054#
LADMSK  001600           544#    3159     3178     3200     3260     3279     3299     3311     4129*    4131*
LDDISP  001522           508#     880*
LF      = 000012          37#    4631     4637
LMAD    001520           507#    3649     3660*    3667     3681*
LSTADR  001574           541#    1356*    1406*    1410     1421     1460*    1461*    1462*    1463*    1464     3230     4064     4071
MAMF    017502          1481     2539     3726#
MAMF1   017566          3740#    3742
MAMF2   017600          3727     3729     3735     3744#
MANUAL  005342          1112     1267     1336     1349#    1441
MANUL1  005400          1355     1362#
MANUL2  006002          1357     1458#
MAPMEM  003300           931      945#
MAPRB   004224          1131     1141#
```

```
MASK4K= 017777             183#     959      984     1058     1412     1464     3202     3313     3832     4088     4095     4124     4129
MEMMAP  001524             230      509#     946      947     1185     1187     1190     1387     1389     1428     1430     1433     1435
                          3522     3524     3533     3535     3555     3560     3802     3804
MEMMES  025365             955     4960#
MMAMES  025320             925     4953#
MMAVA   000606             233      290#     917*     923*     965      986     1017*    1023*    1060     1130     1172     1444     1458
                          1482     2474     3142     3214     3245     3337     3368     3513     3603     3652     3661     3797     3837
                          4901
MMDOWN  015640            1533     1569     1591     1665     1681     3224     3330#
MMINIT  014244             922     1022     1133     1486     3114#
MMORE   001554             528#    3222*    3223*    3231*    3316     3359
MMUP    015150            1517     1553     1609     1629     1644     1705     1726     1762     1785     1808     1824     1867     1910
                          1926     1970     2013     2029     2077     2197     2214     2262     2382     2538     2573     2623     2672
                          2722     2771     2820     2870     2952     2976     2996     3009     3029     3240#    3411
MMVEC = 000250             144#
MPAMEM  004164            1107     1126#
MPEND   002074             617#
MPPATS  002050             565      606#
MPRX    002276             563      697#    1086     1102     1106     1169     1241     1254     1265     1281     1327     1335     2394
                          3319     3726     3749
MPR0    002076             562      632#    1089     1143     1205     1276     1321     1330
MPR1    002106             636#
MPR10   002216             672#
MPR11   002226             676#
MPR12   002236             680#
MPR13   002246             684#
MPR14   002256             688#    1180     1246     1248     1249     1255     1258*
MPR15   002266             692#    1182     1250     1252     1253     1259     1262*
MPR2    002116             640#
MPR3    002126             644#
MPR4    002136             648#
MPR5    002146             652#
MPR6    002156             656#
MPR7    002166             660#
MFR8    002176             664#
MPR9    002206             668#
MTMAP   025426            1203     4967#
MTOE    026536            3709     5077#
MTR     025551            1109     4983#
MX1     026035            1227     5016#
MX2     026054            1235     5019#
MX3     026067            1211     5021#
MX4     026106            1217     5024#
MX5     026124            1223     5027#
MX6     026142            1264     5030#
NOMEM   026232            3913     5040#
NOMTST  026703            4067     5095#
NOPES   025652            3851     4995#
OEFLG   001560             533#    2421*    2531*
PARVEC= 000114             180#    2435*    2439*    2446*    3580*    3633*    3736*    3737*
PBTRP   012026             564     2446#
PESRV   017374             566     3689#
PEWNC   026644            3826     5089#
PHYADR  016020            1505     1522     1548     1558     1585     1596     3366#
PIRQ  = 177772              43#
PIRQVE= 000240             137#
```

```
PMEMAP  001540      519#    1141*    1142*    1166*    1167*    1248*    1249*    1252*    1253*    2402     2404
PRELOC  026506     3496     5073#
PRGMAP  000602      253      259      274      289#     911*     912*    3036     3038     3042     3507     3539     3547*    3565
                   3591*    3600     3640*    3641*    4074     4075
PROREL  025724     5002#
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   017710     3715     3763     3768     3779#
PSW   = 177776       41#     899     1232     1371     1398     1455     3083     3952     3960     3994     4117     4138     4157
                   4189     4223     4256     4263     4276     4322     4421     4507     4553
PWRMSG  025605      326     4988#
PWRVEC= 000024      132#     295*     296*     305*     311*     323*     324*     856*     857*    3579*    3632*
RADTAB  001622      559#    3585     3624
RANTST  012444     2546#
RELOC   016302     3471#    3541     3570     3607     3617
RELOCF  000600      269      288#     913*    1480     3223     3573*    3577*    3586     3594     3615     3639*    3730     4056
                   4227     4248     4250     4550     4754
RELTOP  016424     3048     3507#
RELO    017026      276     3051     3600#
RESCHK  005226     1282     1321#
RESLDR  017234      282     3054     3649#
RESRVD  001516      505#     561     1285*    1289     1294     1299     1307     2479*    2480     2481     2510
RESTAR  000300      201      226#     328      910
RESTOR  000304      202      228#
REST1   000306      227      229#
REST2   000324      231      233#
RESVEC= 000010      127#
ROTATE  016154     1774     1797     3416#
RW    = 000006      176#    3115     3116     3117     3118     3122
SAVLDR  017314      914     3098     3667#
SAVTST  001534      515#    1001*    1002*    1374*    1375*    1416*    1417*    3036     3038     4072     4073
SCANM   026602     3788     5083#
SELECT  002646      199      846#
SELFLG  001556      531#     844*     846*    1354
SETAE   017546     2409     3731     3733     3736#
SETCON  016134     1771     1794     2400     3406#
SKPMES  026727     4111     5099#
SPRNT   020256     1291     1309     1342     2417     3881#
SPRNTA  020344     3891     3895     3900#
SPRNTB  020350     3883     3901#
SPRNTP  020302     2464     2485     2498     2515     3889#
SPRNTQ  020270     3712     3759     3765     3817     3885#
SPRNT0  020306     1302     1527     1563     1675     2440     2456     2524     3887     3890#
SPRNT1  020314     1638     3893#
SPRNT2  020332     1510     1602     1720     1756     1779     1802     1836     1843     1850     1857     1880     1889     1898
                   1939     1946     1953     1960     1983     1992     2001     2042     2049     2056     2063     2090     2099
                   2108     2115     2124     2133     2140     2149     2158     2165     2174     2183     2227     2234     2241
                   2248     2275     2284     2293     2300     2309     2318     2325     2334     2343     2350     2359     2368
                   2563     2909     2918     2940     2989     3022     3898#
```

```
SPRNT3  020326      2616    2665    2715    2764    2813    2863    3897#
SR0   = 177572       148#    235     251*    921*   1021*   1484    3131*   3654*   3663*   4903
SR1   = 177574       149#
SR2   = 177576       150#
SR3   = 172516       151#
STACK = 001100        31#    229     270     272     560     854     929     945    1016
START   002640       198     844#
STARTA  002654       232     284     845     847#
START1  006060       281    1465    1475#   1479    3049    3100
STKLMT= 177774        42#
SWR     001140       303     316*    360#    852     863*    865     871*    878*    895     918    1018    1087    1730
                    1732    2396    3040    3581    3583*   3634    3636*   3728    3732    3751    3998    4012    4014
                    4022    4029    4058    4163    4171    4184    4191    4303    4348*
SWREG   000176       196#    871     895    4303    4319
SW0   = 000001        95#
SW00  = 000001        85#     95
SW01  = 000002        64#     94
SW02  = 000004        83#     93
SW03  = 000010        82#     92
SW04  = 000020        81#     91
SW05  = 000040        80#     90    4058
SW06  = 000100        79#     89    1087    2396
SW07  = 000200        78#     88    3040
SW08  = 000400        77#     87    1730
SW09  = 001000        76#     86
SW1   = 000002        94#
SW10  = 002000        75#
SW11  = 004000        74#
SW12  = 010000        73#    918    1018
SW13  = 020000        72#
SW14  = 040000        71#
SW15  = 100000        70#
SW2   = 000004        93#
SW3   = 000010        92#
SW4   = 000020        91#
SW5   = 000040        90#   3732
SW6   = 000100        89#   3728    3751
SW7   = 000200        88#
SW8   = 000400        87#
SW9   = 001000        86#
TBITVE= 000014       128#
TEMP    001614       550#   2900*   2949
TIMOUT  003576       203    1016#   1078
TKVEC = 000060       135#
TMAP    004536      1177    1201#
TMPFAD  001564       537#   3209    3334    4055*   4060    4062*   4084    4086*   4087*   4088    4103    4125
TMPLAD  001576       543#   3160*   3179*   3261*   3280*   3283    4071*   4092    4094*   4095    4103    4130
TMPPT   001550       525#   3162*   3163*   3167*   3168*   3170    3172    3174    3176    3188*   3193*   3195    3198
                    3203*   3204*   3240    3242    3262*   3263*   3268*   3269*   3271    3273    3275    3277    3300*
                    3303*   3305    3314*   3315*
TO      025463      1049    1071    3934    4973#
TPVEC = 000064       136#
TRAPVE= 000034       134#
TRTVEC= 000014       129#
TSTMAP  001530       512#   3146    3148    3170    3172    3182    3195    3252    3254    3271    3273    3295    3305
                    3343    3345    3352    4072*   4073*   4074*   4075*   4078    4080*   4081*   4090*   4091*   4097*
```

```
                          4098*   4105    4107
TST1      006142          1483    1485    1497#
TST10     007014          1731    1735    1744#
TST11     007076          1750    1767#
TST12     007162          1790#
TST13     007244          1812#
TST14     007570          1816    1914#
TST15     010120          1918    2017#
TST16     010742          2021    2202#
TST17     011564          2206    2391#
TST2      006266          1543#
TST20     012436          2398    2543#
TST21     012552          2600#
TST22     012642          2604    2649#
TST23     012734          2653    2698#
TST24     013032          2702    2748#
TST25     013124          2752    2797#
TST26     013216          2801    2846#
TST27     013312          2850    2896#
TST3      006362          1580#
TST30     013526          2967#
TST31     013640          3000#
TST32     013760           569    1501    3034#
TST4      006466          1620#
TST5      006556          1655#
TST6      006652          1697#   1737
TST6A     006660          1700#   1738
TST7      006702          1711#   1737
TYPMAP    020356           998    1239    3908#
UNEXPT    026445          3691    5067#
UP      = 000000           175#   3115    3116    3117    3118    3122
WWP       001612           549#   1146    1150    1184*   2425    2432
WWPBYT    011626          2402#
WWPB0     011572          2394#
WWPB1     011666          2411#   2537
WWPB2     011732          2423#   2533
WWPB3     012302          2493    2502#
WWPB4     012372          2443    2459    2528#
WWPB5     012416          2408    2414    2535#
W3X9      016222          1821    1923    2026    2211    3439#
$APTHD    001330           456     462#
$APTR     001512           499#   4682    4703
$ASTAT    001344           472#    499    4671    4675    4701    4704    4708    4714
$ASTEN    001510           498#
$ATYC     024014          4642    4644#
$ATY1     023770          4640#
$ATY3     023776          4586    4641#
$ATY4     024006          4180    4643#
$ATY6     024304          4701#
$ATY7     024350          4712#
$AUTOB    001134           357#    902*   4311    4485
$BASE     001260           425#
$BDADR    001122           352#    587     590     592    3487*   3689*
$BDDAT    001126           354#    573     575     581     584     587     596    3485*   3901*
$BELL     001174           376#   4166    4203
$CDW1     001264           427#
```

```
$CDW2     001256          428#
$CHARC    023764         4603*   4613*   4620    4629*   4634#
$CKSWR    022330         3995    4139    4158    4190    4303#
$CMTAG    001100          340#    849     850     858
$CM3   =  000000          370#
$CM4   =  000004          370#    371#    372#    373#    374#
$CNTLG    023255         4315    4480#
$CNTLU    023250         4339    4448    4479#
$CPUOP    001232          399#
$CRLF     001201          378#   1000    1244    3057    4175    4203    4210    4238    4244    4289    4352    4454    4479
                         4543    4602    4637
$DBLK     024620         4755    4791    4799#
$DDW0     001270          429#
$DDW1     001272          430#
$DDW10    001314          439#
$DDW11    001316          440#
$DDW12    001320          441#
$DDW13    001322          442#
$DDW14    001324          443#
$DDW15    001326          444#
$DDW2     001274          431#
$DDW3     001276          432#
$DDW4     001300          433#
$DDW5     001302          434#
$DDW6     001304          435#
$DDW7     001306          436#
$DDW8     001310          437#
$DDW9     001312          438#
$DEVCT    001214          390#
$DEVM     001262          426#
$DOAGN    014220         3074    3090    3097    3099#
$DTBL     024610         4759    4795#
$ENDAD    014174          215     886    3092#   4198
$ENDCT    014130          858    3076#
$ENDMG    014224         3079    3101#
$ENULL    014241         3086    3104#
$ENV      001224          395#    893    3046    4177    4581    4649    4688
$ENVM     001225          396#    876    4583    4588    4651
$EOP      014100         3039    3041    3067#
$EOPCT    014122          858*   3073#   3077
$ERFLG    001103          343#   3982    4018    4020    4026*   4136    4160*   4203
$ERMAX    001115          349#   4020    4043*   4136
$ERROR    021604         1292    1303    1310    1345    1511    1528    1564    1603    1639    1676    1721    1757    1780
                         1803    1837    1844    1851    1858    1881    1890    1899    1940    1947    1954    1961    1984
                         1993    2002    2043    2050    2057    2064    2091    2100    2109    2116    2125    2134    2141
                         2150    2159    2166    2175    2184    2228    2235    2242    2249    2276    2285    2294    2301
                         2310    2319    2326    2335    2344    2351    2360    2369    2418    2441    2457    2465    2486
                         2499    2516    2525    2564    2617    2666    2716    2765    2814    2864    2910    2919    2941
                         2990    3023    3488    3700    3713    3760    3766    3818    4154#   4884
$ERRPC    001116          350#    573     575     577     579     581     584     587     590     592     594    4168*   4169*
                         4170    4203    4219    4226
$ERRTB    002340          567     714#
$ERRTY    022030         4173    4211#
$ERTTL    001112          347#   3814*   4167*   4203
$ESCAP    001172          375#   4042*   4194    4196    4203
$ETABL    001224          394#
```

```
$ETEND   001330                  445#     468
$FATAL   001206                  387#     4692*
$FFLG    024374                  4640*    4643*    4686     4695*    4724#
$FILLC   001156                  368#     4606     4637
$FILLS   001155                  367#     4637
$GDADR   001120                  351#     573      575      577      579      581      584      587      596      3486*    3881*    3890*    3893*
                                 3894*    3898*    3899*
$GDDAT   001124                  353#     573      575      577      581      584      587      3484*    3882*    3900*
$GET42   014164                  3087#
$GTSWR   022402                  900      4316#
$HD    = 000000                  12
$HIBTS   001330                  463#
$HIOCT   023454                  1378     1391     1402     1405*    1407     1418     4530*    4542#
$ICNT    001104                  344#     4033*    4034     4036*    4135
$ILLUP   000756                  295      311      330#
$INTAG   001135                  358#     4353     4485
$ITEMB   001114                  348#     4170*    4179     4203     4216
$LF      001202                  379#     4203     4469     4479     4543     4637
$LFLG    024373                  4671     4696*    4723#
$LPADR   001106                  345#     1479*    1480*    1737*    4024*    4040*    4045     4123*    4135
$LPERR   001110                  346#     4024     4041*    4135     4193
$MADR1   001236                  412#
$MADR2   001242                  416#
$MADR3   001246                  419#
$MADR4   001252                  422#
$MAIL    001204                  385#     464      468      875      893      4039     4177     4581
$MAMS1   001234                  406#
$MAMS2   001240                  414#
$MAMS3   001244                  417#
$MAMS4   001250                  420#
$MBADR   001332                  464#
$MFLG    024372                  4641*    4647     4697*    4722#
$MNEW    023273                  4325     4483#
$MSGAD   001220                  392#     4657*    4660     4708*
$MSGLG   001222                  393#     4662*    4707*
$MSGTY   001204                  386#     4655     4663*    4690     4694*    4705     4709*
$MSWR    023262                  4318     4481#
$MTYP1   001235                  407#
$MTYP2   001241                  415#
$MTYP3   001245                  418#
$MTYP4   001251                  421#
$MXCNT   021572                  4037     4135#
$NULL    001154                  366#     4608     4637
$NWTST=  000001                  1488#    1490     1534#    1536     1571#    1573     1611#    1613     1646#    1648     1687#    1689     1690
                                 1707#    1709     1740#    1742     1764#    1787#    1809#    1911#    2014#    2199#    2383#    2395     2540#
                                 2575#    2577     2624#    2626     2673#    2675     2723#    2725     2772#    2774     2821#    2823     2872#
                                 2874     2954#    2956     2997#
$OCNT    025054                  4832*    4862*    4875#
$OMODE   025056                  4827*    4831*    4836     4839*    4850*    4877#
$OVER    021056                  3999     4017     4025     4035     4044#
$PASS    001212                  369#     875*     3070*    3071*    3080     3101     4031     4050     4076     4136
$PASTM   001336                  466#
$PRINT   023456                  325      888      924      954      999      1007     1025     1038     1048     1070     1108     1202     1210
                                 1216     1222     1226     1234     1243     1263     1366     1393     1438     1450     3056     3078     3085
                                 3495     3690     3708     3787     3825     3850     3912     3933     3940     4066     4110     4165     4174
                                 4212     4235     4237     4241     4243     4282     4288     4314     4317     4324     4338     4351     4372
```

```
                      4429     4436     4442     4447     4453     4455     4459     4463     4468     4537     4539     4549#    4601
                      4790     4860     4932     4942
$PWRAD   000752        328#
$PWRDN   000610        295#     323      856
$PWRMG   000746        326#
$PWRUP   000662        305      311#
$QUES    001200        377#    4203     4373     4460     4479     4540     4543     4637
$RDCHR   022634       4386#    4422
$RDLIN   022754       4414#    4508
$RDOCT   023304       1372     1399     1456     4499#
$RDSZ  = 000010       4407#
$SAVR6   000762        304*     312      313*     314*     332#
$SCOPE   020574       1498     1544     1581     1621     1656     1698     1712     1745     1768     1791     1813     1915     2018
                      2203     2392     2544     2601     2650     2699     2749     2798     2847     2897     2968     3001     3032
                      3991#

$SETUP=  000130        188#     855      856      858      859      886      890     3069     3992     4155     4187     4198     4298
                      4485

$STUP  = 177777        188#
$SVLAD   021022       4007     4038#
$SVPC  = 000010        213#     218
$SWR   = 167400          1#      12       17       18       19       20       21       22       23       24      329      374      375
                       376      859     1503     1546     1583     1623     1658     1700     1714     1747     1770     1793     1818
                      1920     2023     2208     2394     2546     2606     2655     2704     2754     2803     2852     2899     2970
                      3003     3064     3069     3083     3100     3101     3983     3984     3985     3986     3997     3998     4010
                      4012     4013     4018     4019     4020     4027     4028     4029     4041     4044     4135     4146     4147
                      4148     4149     4150     4163     4171     4184     4191     4203
$SWREG   001226        397#     878
$SWRMK=  000340          1#      24       25     3987     3988     4014     4015
$TESTN   001210        388#    4039*
$TIMES   001170        279*     374#    3034*    3069*    4027*    4034     4037*    4135
$TKB     001146        363#    4296     4307     4331     4390     4396
$TKS     001144        362#    4296     4305     4328     4355*    4388     4394
$TMP0    001160        370#     579      581      584      592      594      596     3367*    3377*    3379*    3885*    3989*    3897*
$TMP1    001162        371#     592      594      596     3886*
$TMP2    001164        372#     963*     964*     967*     978*    1356     1404
$TMP3    001166        373#     598      953*     969*     975*     977*     979*    1405     4883*
$TN    - 000032          1#      12     1488     1503#    1534     1546#    1571     1583#    1611     1623#    1646     1658#    1687
                      1700#    1707     1709     1714#    1727     1740     1747#    1750     1764     1770#    1787     1793#    1809
                      1812     1818#    1911     1914     1920#    2014     2017     2023#    2199     2202     2208#    2383     2394#
                      2540     2546#    2575     2600     2606#    2624     2649     2655#    2673     2698     2704#    2723     2748
                      2754#    2772     2797     2803#    2821     2846     2852#    2872     2699#    2954     2970#    2997     3003#
$TPB     001152        365#    4626*    4637
$TPFLG   001157        369#    4575     4637
$TPS     001150        364#    4624     4637
$TSTM    001334        465#
$TSTNM   001102        280*     342#    1736*    3035*    3982     4016     4038*    4039     4044     4052     4114     4136     4162
                      4203

$TTYIN   023240       4416     4417     4433     4456     4474     4478#
$TYPAD   025116       1042     1053     1075     3499     4269     4894#
$TYPDS   024376       3084     4264     4743#
$TYPE    023504       4554     4575#    4668
$TYPEC   023720       4357     4605     4612     4619     4624#    4625
$TYPEX   023766       4630     4632     4635#
$TYPOC   024654       1233     4224     4257     4323     4830#
$TYPON   024670       4829     4832#
$TYPOS   024630       3953     3961     4118     4277     4825#
```

```
$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      4226*    4227*
$XTSTR   020620          4001#
$OFILL   025055          4826*    4830*    4840     4876#
$4OCAT=  ****** U        3998     4173
.      = 032110          190#     194#     205#     213      214#     216#     218#     219#     307      331      339#     380      452
                         453#     455#     457#     534#     697#     853      908      3101     3105     4135     4136     4203     4296
                         4478#    4479     4485     4543     4637     4725#    4799#    5311#
.CONST   001610          548#     1457*    1700     1714
.EIGHT   001642          568#     4776
.ERRTB   001640          567#     4232
.MPPAT   001634          565#     1747
.MPRX    001630          563#     2424     2431     2454     2492     3695     3739     3754     3809     3867
.MPRO    001626          562#     2448
.PBTRP   001632          564#     2435
.PESRV   001636          566#     2439     2446     3736
.RESRV   001624          561#
.STACK   001622          560#     1475
.TST32   001644          569#     4069
.$ASTA=  ****** U        4641     4644
.$X    = 001330          452#     457
.3X9     007256          1818#
```

```
ABORT       1*    260    3154    3187    3225    3397    3490    3509    3567    3602    3651
CKSWR       1*   3992    4136    4155    4187
CKWD        1*   1300    1508    1525    1600    1636    1673    1718    1754    1777    1800    1834    1841    1848    1855
         1878    1887    1896    1937    1944    1951    1958    1981    1990    1999    2040    2047    2054    2061    2088
         2097    2106    2113    2122    2131    2138    2147    2156    2163    2172    2181    2225    2232    2239    2246
         2273    2282    2291    2298    2307    2316    2323    2332    2341    2348    2357    2366    2483    2513    2561
         2614    2663    2713    2762    2811    2861    2907    2916    2938    2987    3020
CKWD2       1*   1507    1598    1716    1753    1832    1840    1847    1854    1876    1886    1895    1935    1943    1950
         1957    1979    1989    1998    2038    2046    2053    2060    2086    2096    2105    2112    2121    2130    2137
         2146    2155    2162    2171    2180    2223    2231    2238    2245    2271    2281    2290    2297    2306    2315
         2322    2331    2340    2347    2356    2365    2560    2985    3018
COMMEN      1*    138*    838    1469
ENDCOM      1*     12     138*    842    1473
ERROR      32*
ESCAPE      1*    138*
GETPRI      1*    138*
GETSWR      1*    138*    890*
GTSWR       1*    897
LDPDR       1*   3114    3116    3117    3118    3122
MORETA    333*    447
MULT        1*    138*
NEWTST      1*    138*   1488    1534    1571    1611    1646    1687    1707    1740    1764    1787    1809    1911    2014
         2199    2383    2540    2575    2624    2673    2723    2772    2821    2872    2954    2997
POP         1*    138*    316     317    2508    2521    3381    3398    3500    3592    3630    3717    3743    3771    3834
         3844    3852    3872    3964    3971    4531    4698    4699    4710    4719    4785    4937
PRINT       1*    924     954     999    1007    1024    1038    1048    1070    1108    1202    1210    1216    1222    1225
         1234    1366    1392    1438    1449    3055    3495    3690    3708    3787    3825    3850    3912    4066    4109
         4932    4942
PUSH        1*    138*    297     303    2467    2490    3370    3388    3471    3510    3527    3623    3693    3738    3753
         3779    3799    3806    3865    3916    3942    4501    4644    4646    4667    4701    4712    4743    4894
RDCHR       1*   4419
RDDEC       1*
RDLIN       1*   4504
RDOCT       1*   1369    1396    1453
REPORT      1*    138*
RESREG      1*
SAVREG      1*
SCOPE      33*
SCOPEX   3977*   4046
SCOPIN   3977*   3996
SETPRI      1*    138*
SETUP       1*    138*    847
SIMTRP      1*    897    1230    1369    1396    1453    3081    3950    3958    3992    4115    4136    4155    4187    4221
         4254    4261    4274    4320    4419    4505    4551
SKIP        1*    138*   1750
SLASH       1*    138*    621     631
SPACE     138*
STARS       1*    138*    211     220     225     293     309     335     380     383     449     451     458     471     501
          503     554     558     570     572     602     604     935     944    1012    1015    1081    1084    1115    1124
         1135    1139    1196    1199    1270    1274    1316    1319    1359    1361    1488    1496    1534    1542    1571
         1579    1611    1619    1646    1654    1683    1686    1687    1696    1707    1710    1740    1743    1764    1766
         1787    1789    1809    1811    1826    1828    1869    1871    1911    1913    1929    1931    1972    1974    2014
         2016    2031    2033    2079    2081    2199    2201    2216    2218    2264    2266    2383    2390    2540    2542
         2575    2599    2624    2648    2673    2697    2723    2747    2772    2796    2821    2845    2872    2895    2954
         2966    2997    2999    3058    3107    3113    3135    3137    3234    3239    3324    3329    3362    3365    3384
         3386    3403    3405    3413    3415    3436    3438    3468    3470    3504    3506    3597    3599    3644    3648
```

```
           3684    3688    3722    3725    3746    3748    3774    3778    3862    3864    3877    3880    3904    3907    3979
           4142    4203    4295    4298    4378    4407    4487    4544    4548    4560    4639    4730    4802    4949    4952
           5106    5108    5245    5247    5294    5296
SWRSU         1#    138#    859#
SWROR3        1#   2067    2072    2187    2192    2252    2257    2372    2377    3463
S1REGS     1488#   1490    1536    1573    1613    1648    1690
TYPADR        1#   1041    1052    1074    3498    4268
TYPBIN        1#    138#
TYPBN         1#
TYPBYT        1#   4113    4271
TYPDEC        1#    138#   3080    4260
TYPDS         1#   3081    4261
TYPE          1#    325     888     924     954     999    1007    1025    1038    1048    1070    1108    1202    1210    1216
           1222    1226    1234    1366    1393    1438    1450    3056    3078    3085    3495    3690    3708    3787    3825
           3850    3912    3933    3940    4066    4110    4165    4174    4212    4235    4237    4241    4243    4282    4288
           4314    4316    4324    4338    4350    4371    4429    4436    4442    4447    4453    4455    4458    4463    4468
           4537    4539    4601    4790    4860    4932    4942
TYPNAM        1#    138#    882
TYPNUM        1#    138#
TYPOC         1#   1230    4221    4254    4320
TYPOCS        1#    138#   3948    3956
TYPOCT        1#    138#   1229    4219    4253    4319
TYPON         1#
TYPOS         1#   3950    3958    4115    4274
TYPTXT        1#    138#
$CKWD         1#   1301    1341    1509    1526    1562    1601    1637    1674    1719    1755    1778    1801    1835    1842
           1849    1856    1879    1888    1897    1938    1945    1952    1959    1982    1991    2000    2041    2048    2055
           2062    2089    2098    2107    2114    2123    2132    2139    2148    2157    2164    2173    2182    2226    2233
           2240    2247    2274    2283    2292    2299    2308    2317    2324    2333    2342    2349    2358    2367    2416
           2484    2514    2523    2562    2615    2664    2714    2763    2812    2862    2908    2917    2939    2988    3021
$INDN         1#   1521    1557    1584    1659    1669
$INMM         1#   1504    1547    1595    1624    1633    1701    1715    1751    1772    1795    1820    1830    1873    1922
           1933    1976    2025    2036    2084    2210    2221    2269    2401    2548    2609    2658    2707    2757    2806
           2855    2899    2970    2983    3003    3016    3406
$MMDN         1#   1531    1567    1589    1663    1679
$MMUP         1#   1515    1551    1607    1627    1642    1703    1724    1760    1783    1806    1822    1865    1908    1924
           1968    2011    2027    2075    2195    2212    2260    2380    2570    2621    2670    2720    2769    2818    2868
           2974    2994    3007    3027    3409
$$CKWD        1#   1291    1302    1309    1342    1510    1527    1563    1602    1638    1675    1720    1756    1779    1802
           1836    1843    1850    1857    1880    1889    1898    1939    1946    1953    1960    1983    1992    2001    2042
           2049    2056    2063    2090    2099    2108    2115    2124    2133    2140    2149    2158    2165    2174    2183
           2227    2234    2241    2248    2275    2284    2293    2300    2309    2318    2325    2334    2343    2350    2359
           2368    2417    2440    2456    2464    2485    2498    2515    2524    2563    2616    2665    2715    2764    2813
           2863    2909    2918    2940    2989    3022    3711    3759    3764    3816
$$CMRE      333#
$$CMTM      333#    370     371     372     373
$$ESCA        1#    138#
$$NEWT        1#    138#   1488    1534    1571    1611    1646    1687    1707    1740    1764    1787    1809    1911    2014
           2199    2383    2540    2575    2624    2673    2723    2772    2821    2872    2954    2997
$$SETM      875#
$$SKIP        1#    138#   1750
.CHBHD     2954#   2956
.C11       2383#   2385
.DIDBH     2673#   2675
.DIDBL     2624#   2626
.DIDO      2575#   2577
```

```
.DIPDO    2723#    2725
.DPDBH    2821#    2823
.DPDBL    2772#    2774
.EQUAT       1#      28
.ERROR       1#    1292     1303     1310     1343     1511     1528     1564     1603     1639     1676     1721     1757     1780     1803
          1837     1844     1851     1858     1881     1890     1899     1940     1947     1954     1961     1984     1993     2002     2043
          2050     2057     2064     2091     2100     2109     2116     2125     2134     2141     2150     2159     2166     2175     2184
          2228     2235     2242     2249     2276     2285     2294     2301     2310     2319     2326     2335     2344     2351     2360
          2369     2418     2441     2457     2465     2486     2499     2516     2525     2564     2617     2666     2716     2765     2814
          2864     2910     2919     2941     2990     3023     3488     3700     3713     3760     3766     3818     4884
.HEADE       1#       2
.KT11        1#     140
.MARHD    2872#    2874
.SCOPE       1#    1497     1543     1580     1620     1655     1697     1711     1744     1767     1790     1812     1914     2017     2202
          2391     2543     2600     2649     2698     2748     2797     2846     2896     2967     3000     3031
.SETUP       1#     188
.SWRHI       1#      13
.SWRLO       1#      25#
.TM7      1707#    1709
.$ACT1       1#     209
.$APTB       1#     381#
.$APTH       1#     447
.$APTY       1#    4637
.$ASTA       1#     469
.$CATC       1#     188
.$CMTA       1#     333
.$DB2D       1#
.$DB2O       1#
.$DIV        1#
.$EOP        1#    3058
.$ERRO       1#    4140
.$ERRT       1#    4203
.$MULT       1#
.$POWE       1#     291
.$RAND       1#
.$RDDE       1#
.$RDOC       1#    4485
.$READ       1#    4293
.$R2AZ       1#
.$SAVE       1#
.$SB2D       1#
.$SB2O       1#
.$SCOP       1#    3977
.$SIZE       1#
.$SUPR       1#
.$TRAP       1#
.$TYPB       1#
.$TYPD       1#    4730
.$TYPE       1#    4558
.$TYPO       1#    4800
.$4OCA       1#
.1170        1#

. ABS.  032110      000
```

```
ERRORS DETECTED:  0

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

J16