IDENTIFICATION

PRODUCT CODE: AC-9045G-MC

PRODUCT NAME: CZQMCG0 0-124K MEM EXER 16K

PRODUCT DATE: 27-DECEMBER-1979

MAINTAINER: DIAGNOSTIC ENGINEERING

REVISION HISTORY

REVISION A: MAY 1975

REVISION B: OCTOBER 1975

REVISION C: OCTOBER 1976

REVISION D: JUNE 1977

REVISION E: DECEMBER 1977

REVISION F: FEBRUARY 1978

REVISION G: DECEMBER 1979

CHGG1 - INSERT DUMMY ARGUEMENTS UNDER ERRTB ITEM 22
IN ORDER THAT ERROR REPORTING ROUTINES
WILL REPORT MESSAGES CORRECTLY.

CHGG2 - CHANGE BNE INSTRUCTION TO BEQ IN CKPMER SUB-
ROUTINE TO REPORT CORRECT ERROR MESSAGE.

TABLE OF CONTENTS

ZQMCGO 0-124K MEM EXER 16K

1.0 GENERAL PROGRAM INFORMATION.

1.1 Program Purpose (Abstract)

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

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

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

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

1.2 System Requirements

A. Hardware Requirements

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

B. Software Requirements

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

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

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

1.3 Related Documents and Standards

A. Programming Practices - Document No. 175-003-009-01
B. PDP-11 MAINDEC SYSMAC Package - MAINDEC-11-DZQAC-C2-D
C. The applicable Memory System Maintenance Manual
D. The applicable Circuit Schematics

1.4 Diagnostic Hierarchy Prerequisites

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

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

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

1.5 Assumptions

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

2.0 OPERATING INSTRUCTIONS

2.1 Loading and Starting Procedures

2.1.1 Load the program using any standard absolute loader.

2.1.2 Starting address 200:

Normal program execution.

2.1.3 Starting address 204:

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

2.1.4 Starting Address 210:

Restart program using previously selected parameters.

2.1.5 Starting Address 214:

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

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

    Load Address 2' ,
    Press START.

B. If running without memory management, then:

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

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

    Load Address 777707 (PC)
    Deposit 214
    Press CONTinue

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

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

2.1.6 Starting address 220:

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

2.2 Special Environments

If the program is run in quick verify mode under ACT11 or APT11 the program is done after the first pass. Also, the

program does not relocate to test the lower 8K of memory.

2.3 Program Options

SW15 = 1 OR UP.... HALT ON ERROR

SW14 = 1 OR UP.... LOOP ON TEST

SW13 = 1 OR UP.... INHIBIT ERROR TYPEOUT

SW12 = 1 OR UP.... INHIBIT MEMORY MANAGEMENT (INITIAL START ONLY)

SW11 = 1 OR UP.... INHIBIT SUBTEST ITERATION

SW10 = 1 OR UP.... RING BELL ON ERROR

SW9 = 1 OR UP.... LOOP ON ERROR

SW8 = 1 OR UP.... LOOP ON TEST IN SWR<4:0>

SW7 = 1 OR UP.... INHIBIT PROGRAM RELOCATION

SW6 = 1 OR UP.... INHIBIT PARITY ERROR DETECTION

NOTE: With parity error detection enabled, a memory failure while running the worse case noise tests (non-parity) can cause a parity error. The error printout on a parity error does not type the good data. Thus a bit drop or pickup will not be typed as such. It is best to run the program for 1 pass with parity disabled, then, restart the program with parity enabled.

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

2.4 EXECUTION TIMES

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

a. For Non-Parity Memory
   First Pass: 65 seconds for first 16k + 15 seconds for each additional 16k.

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

   Iteration Inhibited: same as first pass

b. For Parity Memory
   First Pass: 1 minute 40 seconds per 16k.

Full Pass: 8 minutes per 16K

Iteration Inhibited: same as first pass

3.0 ERROR INFORMATION

3.1 Error Reporting

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

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

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

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

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

MA = Memory Address

REG = Parity REGister address.

PS = Processor Status word.

IUT = Instruction Under Test.

S/B = What contents Should Be.

WAS = What contents WAS.

3.2 Error Halts

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

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

B. In the relocation routine if the program is being relocated back to the first 8K of memory and the program code was not able to be transferred properly.

C. In the case of error reporting and there is no terminal to allow the information transfer.

D. In the power fail routine if the power up sequence was started before the power down sequence had a chance to complete itself.

E. In the Memory mapping routine or any of the address control routines, failures to find a meaningful map.

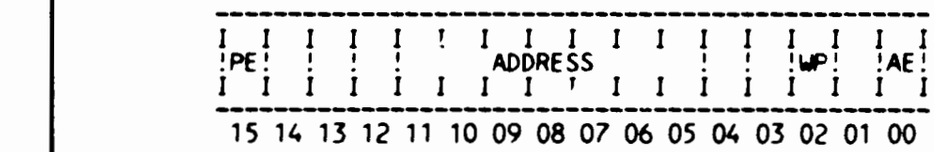
4.0 PERFORMANCE AND PROGRESS REPORTS

Not applicable

5.0 DEVICE INFORMATION TABLES

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

5.1 CORE PARITY REGISTER



Bit assignments are defined as follows:

BIT15 PARITY ERROR

BITS 11-5 ERROR ADDRESS HIGH ORDER ADDRESS BITS OF ADDRESS OF PARITY ERROR (BITS 17-11 OF ADDRESS)

BIT02 WRITE WRONG PARITY NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET

BIT00 ACTION ENABLE NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

5.2 MOS PARITY REGISTER

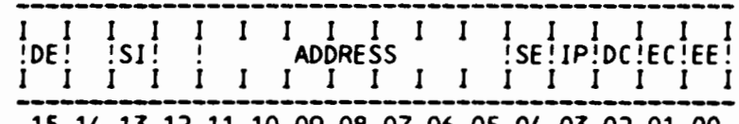
BIT ASSIGNMENTS ARE DEFINED AS FOLLOWS:

BIT15 PARITY ERROR

BIT02 WRITE WRONG PARITY NORMAL PARITY (ODD) WHEN CLEAR; OTHER PARITY (EVEN) WHEN SET

BIT00 ACTION ENABLE NO ACTION WHEN CLEAR TRAP TO VECTOR 114 WHEN SET

5.3 MS11-K CSR



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 EPROR ENABLE WHEN SET ENABLES TRAP TO VECTOR 114 ON DOUBLE ERROR.

6.0 SUB-TEST SUMMARIES

6.1 Section 1: Address Tests.

These tests verify the uniqueness of every memory address.

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

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

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

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

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

6.2 Section 2: Worst Case Noise Tests.

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

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

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

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

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

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

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

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

6.3 Section 3: Instruction Execution Tests.

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

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

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

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

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

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

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

6.4 Section 4: Mos Tests

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

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

6.5 Special Toggle In Tests

6.5.1 Toggle-in-program #1

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

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

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

NOTES: The upper limit address obtained from the switch

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

6.5.2 Toggle-in-Program #2

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

```
                .TITLE  CZQMCGO  0-124K MEMORY EXERCISER, 16K VER
                ;*COPYRIGHT (C) 1975,1979
                ;*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.
                ;*

6184                            .SBTTL  OPERATIONAL SWITCH SETTINGS
 (1)                            ;*
 (1)                            ;*      SWITCH                  USE
 (1)                            ;*      ------          --------------------
 (1)                            ;*        15            HALT ON ERROR
 (1)                            ;*        14            LOOP ON TEST
 (1)                            ;*        13            INHIBIT ERROR TYPEOUTS
 (1)                            ;*        12            INHIBIT KT11 (AT START TIME ONLY)
 (1)                            ;*        11            INHIBIT ITERATIONS
 (1)                            ;*        10            BELL ON ERROR
 (1)                            ;*         9            LOOP ON ERROR
 (1)                            ;*         8            LOOP ON TEST IN SWR<4:0>
6185                            ;*         7            INHIBIT PROGRAM RELOCATION
 (1)                            ;*         6            INHIBIT PARITY ERROR DETECTION
 (1)                            ;*         5            INHIBIT EXRERCISING VECTOR AREA.
6186                            .SBTTL  BASIC DEFINITIONS
 (1)
 (1)                            ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
 (1)            001100          STACK   1100
 (1)                            .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
 (1)                            .EQUIV  IOT,SCOPE       ;;BASIC DEFINITION OF SCOPE CALL
 (1)
 (1)                            ;*MISCELLANEOUS DEFINITIONS
 (1)            000011          HT=     11              ;;CODE FOR HORIZONTAL TAB
 (1)            000012          LF=     12              ;;CODE FOR LINE FEED
 (1)            000015          CR=     15              ;;CODE FOR CARRIAGE RETURN
 (1)            000200          CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
 (1)            177776          PS=     177776          ;;PROCESSOR STATUS WORD
 (1)                            .EQUIV  PS,PSW
 (1)            177774          STKLMT= 177774          ;;STACK LIMIT REGISTER
 (1)            177772          PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
 (1)            177570          DSWR=   177570          ;;HARDWARE SWITCH REGISTER
 (1)            177570          DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
 (1)
 (1)                            ;*GENERAL PURPOSE REGISTER DEFINITIONS
 (1)            000000          R0=     %0              ;;GENERAL REGISTER
 (1)            000001          R1=     %1              ;;GENERAL REGISTER
 (1)            000002          R2=     %2              ;;GENERAL REGISTER
 (1)            000003          R3=     %3              ;;GENERAL REGISTER
 (1)            000004          R4=     %4              ;;GENERAL REGISTER
 (1)            000005          R5      %5              ;;GENERAL REGISTER
 (1)            000006          R6=     %6              ;;GENERAL REGISTER
 (1)            000007          R7=     %7              ;;GENERAL REGISTER
 (1)            000006          SP      %6              ;;STACK POINTER
```

```
(1)  000007  PC=  %7  ;;PROGRAM COUNTER
(1)
(1)  ;*PRIORITY LEVEL DEFINITIONS
(1)  000000  PR0=  0  ;;PRIORITY LEVEL 0
(1)  000040  PR1=  40  ;;PRIORITY LEVEL 1
(1)  000100  PR2=  100  ;;PRIORITY LEVEL 2
(1)  000140  PR3=  140  ;;PRIORITY LEVEL 3
(1)  000200  PR4=  200  ;;PRIORITY LEVEL 4
(1)  000240  PR5=  240  ;;PRIORITY LEVEL 5
(1)  000300  PR6=  300  ;;PRIORITY LEVEL 6
(1)  000340  PR7=  340  ;;PRIORITY LEVEL 7
(1)
(1)  ;*''SWITCH REGISTER'' SWITCH DEFINITIONS
(1)  100000  SW15=  100000
(1)  040000  SW14=  40000
(1)  020000  SW13=  20000
(1)  010000  SW12=  10000
(1)  004000  SW11=  4000
(1)  002000  SW10=  2000
(1)  001000  SW09=  1000
(1)  000400  SW08=  400
(1)  000200  SW07=  200
(1)  000100  SW06=  100
(1)  000040  SW05=  40
(1)  000020  SW04=  20
(1)  000010  SW03=  10
(1)  000004  SW02=  4
(1)  000002  SW01=  2
(1)  000001  SW00=  1
(1)  .EQUIV  SW09,SW9
(1)  .EQUIV  SW08,SW8
(1)  .EQUIV  SW07,SW7
(1)  .EQUIV  SW06,SW6
(1)  .EQUIV  SW05,SW5
(1)  .EQUIV  SW04,SW4
(1)  .EQUIV  SW03,SW3
(1)  .EQUIV  SW02,SW2
(1)  .EQUIV  SW01,SW1
(1)  .EQUIV  SW00,SW0
(1)
(1)  ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
(1)  100000  BIT15=  100000
(1)  040000  BIT14=  40000
(1)  020000  BIT13=  20000
(1)  010000  BIT12=  10000
(1)  004000  BIT11=  4000
(1)  002000  BIT10=  2000
(1)  001000  BIT09=  1000
(1)  000400  BIT08=  400
(1)  000200  BIT07=  200
(1)  000100  BIT06=  100
(1)  000040  BIT05=  40
(1)  000020  BIT04=  20
(1)  000010  BIT03=  10
(1)  000004  BIT02=  4
(1)  000002  BIT01=  2
```

```
(1)          000001          BIT00=  1
(1)                          .EQUIV  BIT09,BIT9
(1)                          .EQUIV  BIT08,BIT8
(1)                          .EQUIV  BIT07,BIT7
(1)                          .EQUIV  BIT06,BIT6
(1)                          .EQUIV  BIT05,BIT5
(1)                          .EQUIV  BIT04,BIT4
(1)                          .EQUIV  BIT03,BIT3
(1)                          .EQUIV  BIT02,BIT2
(1)                          .EQUIV  BIT01,BIT1
(1)                          .EQUIV  BIT00,BIT0
(1)
(1)                          ;*BASIC ''CPU'' TRAP VECTOR ADDRESSES
(1)          000004          ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
(1)          000010          RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
(1)          000014          TBITVEC=14              ;;''T'' BIT
(1)          000014          TRTVEC= 14              ;;TRACE TRAP
(1)          000014          BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
(1)          000020          IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
(1)          000024          PWRVEC= 24              ;;POWER FAIL
(1)          000030          EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
(1)          000034          TRAPVEC=34              ;;''TRAP'' TRAP
(1)          000060          TKVEC=  60              ;;TTY KEYBOARD VECTOR
(1)          000064          TPVEC=  64              ;;TTY PRINTER VECTOR
(1)          000240          PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
6187
6188
6189                         .SBTTL  MEMORY MANAGEMENT DEFINITIONS
(1)
(1)                          ;*KT11 VEC  ADDRESS
(1)
(1)          000250          MMVEC=  250
(1)
(1)                          ;*KT11 STATUS REGISTER ADDRESSES
(1)
(1)          177572          SR0=    177572
(1)          177574          SR1=    177574
(1)          177576          SR2=    177576
(1)          172516          SR3=    172516
(1)
(1)                          ;*KERNEL ''I'' PAGE DESCRIPTOR REGISTERS
(1)
(1)          172300          KIPDR0= 172300
(1)          172302          KIPDR1= 172302
(1)          172304          KIPDR2= 172304
(1)          172306          KIPDR3= 172306
(1)          172310          KIPDR4= 172310
(1)          172312          KIPDR5= 172312
(1)          172314          KIPDR6= 172314
(1)          172316          KIPDR7= 172316
(1)
(1)                          ;*KERNEL ''I'' PAGE ADDRESS REGISTERS
(1)
(1)          172340          KIPAR0= 172340
(1)          172342          KIPAR1= 172342
(1)          172344          KIPAR2  172344
```

```
  (1)          172346                  KIPAR3= 172346
  (1)          172350                  KIPAR4= 172350
  (1)          172352                  KIPAR5= 172352
  (1)          172354                  KIPAR6= 172354
  (1)          172356                  KIPAR7= 172356
  (1)
6190           000000                  UP = 0                  ;CODE FOR UPWARDS MAP IN MEM MGMT PDR'S
6191           000006                  RW = 6                  ;CODE FOR READ/WRITE IN MEM MGMT PDR'S
6192
6193                                   ;*  PARITY MEMORY DEFINITIONS.
6194           000001                  AE=1                            ;PARITY ACTION ENABLE
6195           000114                  PARVEC=114                      ;PARITY TRAP VECTOR
6196
6197                                   ;*  MISCELLANEOUS ASSIGNMENTS
6198           017777                  MASK4K= 17777                   ;MASK FOR 4K ADDRESS BANK BOUNDRY.
6199
6200                                   ;* CACHE REGISTER DEFINITIONS.
6201           177746                  IMPCHF= 177746
6202
6204                                   .SBTTL  TRAP CATCHER
  (1)
  (1)          000000                          .=0
  (1)                                  ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ''.+2,HALT''
  (1)                                  ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
  (1)                                  ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
  (1)          000174                          .=174
  (1)  000174  000000          DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
  (1)  000176  000000          SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
  (1)                                  .SBTTL  STARTING ADDRESS(ES)
  (1)  000200  000137  002650          JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
6205   000204  000167  002446          JMP     SELECT          ;STARTING ADDRESS TO ALLOW THE OPERATOR TO
6206                                                           ;SELECT VARIOUS PARAMETERS.
6207   000210  000167  000064          JMP     RESTAR          ;RESTART ADDRESS, USING PREVIOUS PARAMETERS.
6208   000214  000167  000064          JMP     RESTOR          ;RESTORE LOADERS TO END OF MEMORY AND HALT.
6209   000220  000167  003406          JMP     TIMOUT          ;TYPE OUT MEMORY MAP, BYTE BY BYTE.
6210
6211           000004                  .=ERRVEC
6212   000004  025124                  .WORD   ERRTRP
6213   000006  000000                  .WORD   0
6214
6215                                   .SBTTL  ACT11 HOOKS
  (1)
  (2)                                  ;;*****************************************************************
  (1)                                  ;HOOKS REQUIRED BY ACT11
  (1)          000010                  $SVPC=.                 ;SAVE PC
  (1)          000046                  .=46
  (1)  000046  014232                  $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
  (1)          000052                  .=52
  (1)  000052  040000                  .WORD   BIT14           ;;2)SET LOC.52 TO BIT14
  (1)          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  002340                  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$:     CLR     R0              ;INIT TEMP PAR REG.
000434  016701  000142                  MOV     PRGMAP, R1      ;GET THE PROGRAM MAP...LO 64K.
000440  016702  000140                  MOV     PRGMAP+2,R2     ;...HI 64K.
000444  006202                  3$:     ASR     R2              ;SHIFT THE MAP POINTER...HI
000446  006001                          ROR     R1              ;...LO.
000450  103404                          BCS     4$              ;BR WHEN FIRST BANK FOUND.
000452  062700  000200                  ADD     #200,   R0      ;UPDATE TMP PAR TO NEXT BANK.
000456  100372                          BPL     3$              ;BR IF MORE.
000460  000000                          HALT            ;FATAL ERROR!!!  MAP EMPTY?
000462  010037  172340          4$:     MOV     R0,     @#KIPAR0 ;PUT TEMP PAR INTO FIRST PAR.
000466  000137  000472                  JMP     @#5$            ;JUMP INTO PROGRAM IF NOT THERE ALREADY.
000472  062700  000200          5$:     ADD     #200,   R0      ;KEEP UPDATING TEMP PAR REG.
000476  006202                          ASR     R2              ;SHIFT POINTER...HI
000500  006001                          ROR     R1              ;...LO
000502  103373                          BCC     5$              ;BR IF TOP BANK NOT YET FOUND.
000504  010037  172342                  MOV     R0,     @#KIPAR1 ;SET UP SECOND PROGRAM ANK POINTER.
000510  000410                          BR      20$             ;BR TO RELOCATE SECTION.
000512  016700  000062          10$:    MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
000516  062700  001100                  ADD     #STACK, R0      ;SET UP STACK POINTER.
000522  010006                          MOV     R0,     SP      ;SET STACK TO RELOCATE PROGRAM.
000524  062700  177432                  ADD     #20$-STACK,R0   ;ADJUST R0 TO RELOCATED ''20$'' ADDRESS.
000530  000110                          JMP     (R0)            ;GO TO ''20$'' (RELOCATED).
000532  022767  000003  000042  20$:    CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IS IN BANKS 0 AND 1.
```

```
 6273  000540  001402                    BEQ     21$             ;BR IF IN BANKS 0 AND 1.
 6274  000542  004767  016324            JSR     PC,     RELO    ;RELOCATE THE PROGRAM BACK TO BANKS 0 AND 1.
 6275  000546  005705            21$:    TST     R5              ;CHECK RESTART/RESTORE FLAG.
 6276  000550  001006                    BNE     22$             ;BR IF RESTORE.
 6277  000552  005067  000412            CLR     $TIMES          ;CLEAN UP BEFORE STARTING.
 6278  000556  105067  000320            CLRB    $TSTNM
 6279  000562  000167  005326            JMP     START1          ;RESTART WITH PREVIOUSLY SELECTED PARAMETERS.
 6280  000566  004767  016506    22$:    JSR     PC,     RESLDR  ;RESTORE THE LOADERS TO THE "TOP" OF MEMORY.
 6281  000572  000000                    HALT                    ;HALT AFTER RESTORING THE LOADERS.
 6282  000574  000167  002064            JMP     STARTA          ;CONTINUE WILL RESTART THE PROGRAM.
 6283                            ;* THE FOLLOWING LOCATIONS ARE USED BY THE ABOVE ROUTINE AND MUST BE LOCATED
 6284                            ;*  BELOW 1000 TO INSURE CORRECT OPERATION UNDER THE WIDEST VARIETY OF
 6285                            ;*  CIRCUMSTANCES.
 6286  000600  000000            RELOCF: .WORD   0       ;CONTAINS RELOCATION FACTOR (NO MEM MGMT)
 6287  000602  000000  000000    PRGMAP: .WORD   0,0     ;PROGRAM MAP - WHERE THE PROGRAM IS LOCATED
 6288  000606  000000            MMAVA:  .WORD   0       ;MEMORY MANAGEMENT AVAILABLE FLAG.
```

```
6290                                    .SBTTL  POWER DOWN AND UP ROUTINES
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;POWER DOWN ROUTINE
 (1)  000610  012737  000756  000024    $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
 (1)  000616  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;PRIO:7
 (3)  000624  010046                            MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 (3)  000626  010146                            MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 (3)  000630  010246                            MOV     R2,-(SP)        ;;PUSH R2 ON STACK
 (3)  000632  010346                            MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 (3)  000634  010446                            MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 (3)  000636  010546                            MOV     R5,-(SP)        ;;PUSH R5 ON STACK
 (3)  000640  017746  000274                    MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
 (1)  000644  010667  000112                    MOV     SP,$SAVR6       ;;SAVE SP
 (1)  000650  012737  000662  000024            MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
 (1)  000656  000000                            HALT
 (1)  000660  000776                            BR      .-2             ;;HANG UP
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;POWER UP ROUTINE
 (1)  000662  012737  000756  000024    $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
 (1)  000670  016706  000066                    MOV     $SAVR6,SP       ;;GET SP
 (1)  000674  005067  000062                    CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
 (1)  000700  005267  000056            1$:     INC     $SAVR6          ;;WAIT FOR THE INC
 (1)  000704  001375                            BNE     1$              ;;OF  WORD
 (3)  000706  012677  000226                    MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
 (3)  000712  012605                            MOV     (SP)+,R5        ;;POP STACK INTO R5
 (3)  000714  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
 (3)  000716  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
 (3)  000720  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
 (3)  000722  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
 (3)  000724  012600                            MOV     (SP)+,R0        ;;POP STACK INTO R0
 (1)  000726  012737  000610  000024            MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
 (1)  000734  012737  000340  000026            MOV     #340,@#PWRVEC+2 ;;PRIO:7
 (2)  000742  004567  022554                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (1)  000746  025651                    $PWRMG: .WORD   PWRMSG          ;;POWER FAIL MESSAGE POINTER
 (1)  000750  012716                            MOV     (PC)+,(SP)      ;;RESTART AT RESTART
 (1)  000752  000300                    $PWRAD: .WORD   RESTART         ;;RESTART ADDRESS
 (1)  000754  000002                            RTI
 (1)  000756  000000                    $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
 (1)  000760  000776                            BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
 (1)  000762  000000                    $SAVR6: 0                       ;;PUT THE SP HERE
```

```
6485                                    .SBTTL  COMMON TAGS
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
 (1)                                    ;*USED IN THE PROGRAM.
 (1)
 (1)            001100                          .=1100
 (1)  001100                            $CMTAG:                         ;;START OF COMMON TAGS
 (1)  001100    000000                          .WORD   0
 (1)  001102       000                  $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
 (1)  001103       000                  $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
 (1)  001104    000000                  $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
 (1)  001106    000000                  $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
 (1)  001110    000000                  $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
 (1)  001112    000000                  $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
 (1)  001114       000                  $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
 (1)  001115       001                  $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
 (1)  001116    000000                  $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
 (1)  001120    000000                  $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
 (1)  001122    000000                  $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'BAD' DATA
 (1)  001124    000000                  $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
 (1)  001126    000000                  $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
 (1)  001130    000000                          .WORD   0               ;;RESERVED--NOT TO BE USED
 (1)  001132    000000                          .WORD   0
 (1)  001134       000                  $AUTOB: .BYTE   0               ;;AUTOMATIC MODE INDICATOR
 (1)  001135       000                  $INTAG: .BYTE   0               ;;INTERRUPT MODE INDICATOR
 (1)  001136    000000                          .WORD   0
 (1)  001140    177570                  SWR:    .WORD   DSWR            ;;ADDRESS OF SWITCH REGISTER
 (1)  001142    177570                  DISPLAY: .WORD  DDISP           ;;ADDRESS OF DISPLAY REGISTER
 (1)  001144    177560                  $TKS:   177560                  ;;TTY KBD STATUS
 (1)  001146    177562                  $TKB:   177562                  ;;TTY KBD BUFFER
 (1)  001150    177564                  $TPS:   177564                  ;;TTY PRINTER STATUS REG. ADDRESS
 (1)  001152    177566                  $TPB:   177566                  ;;TTY PRINTER BUFFER REG. ADDRESS
 (1)  001154       000                  $NULL:  .BYTE   0               ;;CONTAINS NULL CHARACTER FOR FILLS
 (1)  001155       002                  $FILLS: .BYTE   2               ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
 (1)  001156       012                  $FILLC: .BYTE   12              ;;INSERT FILL CHARS. AFTER A 'LINE FEED''
 (1)  001157       000                  $TPFLG: .BYTE   0               ;;'TERMINAL AVAILABLE'' FLAG (BIT<07>=0=YES)
 (3)  001160    000000                  $TMP0:  .WORD   0               ;;USER DEFINED
 (3)  001162    000000                  $TMP1:  .WORD   0               ;;USER DEFINED
 (3)  001164    000000                  $TMP2:  .WORD   0               ;;USER DEFINED
 (3)  001166    000000                  $TMP3:  .WORD   0               ;;USER DEFINED
 (1)  001170    000000                  $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
 (1)  001172    000000                  $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
 (1)  001174    177607  000377          $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
 (1)  001200       077                  $QUES:  .ASCII  /?/             ;;QUESTION MARK
 (1)  001201       015                  $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
 (1)  001202    000012                  $LF:    .ASCIZ  <12>            ;;LINE FEED
 (2)                                    ;;**********************************************************************
 (2)                                    .SBTTL  APT MAILBOX-ETABLE
 (2)
 (3)                                    ;;**********************************************************************
 (2)                                    .EVEN
 (2)  001204                            $MAIL:                  ;;APT MAILBOX
 (2)  001204    000000                  $MSGTY: .WORD   AMSGTY  ;;MESSAGE TYPE CODE
 (2)  001206    000000                  $FATAL: .WORD   AFATAL  ;;FATAL ERROR NUMBER
 (2)  001210    000000                  $TESTN: .WORD   ATESTN  ;;TEST NUMBER
```

```
(2)  001212  000000                $PASS:  .WORD   APASS   ;;PASS COUNT
(2)  001214  000000                $DEVCT: .WORD   ADEVCT  ;;DEVICE COUNT
(2)  001216  000000                $UNIT:  .WORD   AUNIT   ;;I/O UNIT NUMBER
(2)  001220  000000                $MSGAD: .WORD   AMSGAD  ;;MESSAGE ADDRESS
(2)  001222  000000                $MSGLG: .WORD   AMSGLG  ;;MESSAGE LENGTH
(2)  001224                        $ETABLE:                ;;APT ENVIRONMENT TABLE
(2)  001224      000               $ENV:   .BYTE   AENV    ;;ENVIRONMENT BYTE
(2)  001225      000               $ENVM:  .BYTE   AENVM   ;;ENVIRONMENT MODE BITS
(2)  001226  000000                $SWREG: .WORD   ASWREG  ;;APT SWITCH REGISTER
(2)  001230  000000                $USWR:  .WORD   AUSWR   ;;USER SWITCHES
(2)  001232  000000                $CPUOP: .WORD   ACPUOP  ;;CPU TYPE,OPTIONS
(2)                                ;*                      BITS 15-11=CPU TYPE
(2)                                ;*                              11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
(2)                                ;*                              11/70=06,PDQ=07,Q=10
(2)                                ;*                      BIT 10=REAL TIME CLOCK
(2)                                ;*                      BIT  9=FLOATING POINT PROCESSOR
(2)                                ;*                      BIT  8=MEMORY MANAGEMENT
(2)  001234      000               $MAMS1: .BYTE   AMAMS1  ;;HIGH ADDRESS,M.S. BYTE
(2)  001235      000               $MTYP1: .BYTE   AMTYP1  ;;MEM. TYPE,BLK#1
(2)                                ;*                      MEM.TYPE BYTE   --  (HIGH BYTE)
(2)                                ;*                              900 NSEC CORE=001
(2)                                ;*                              300 NSEC BIPOLAR=002
(2)                                ;*                              500 NSEC MOS=003
(2)  001236  000000                $MADR1: .WORD   AMADR1  ;;HIGH ADDRESS,BLK#1
(2)                                ;*                        MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF 'TYPE'' ABOVE
(2)  001240      000               $MAMS2: .BYTE   AMAMS2  ;;HIGH ADDRESS,M.S. BYTE
(2)  001241      000               $MTYP2: .BYTE   AMTYP2  ;;MEM.TYPE,BLK#2
(2)  001242  000000                $MADR2: .WORD   AMADR2  ;;MEM.LAST ADDRESS,BLK#2
(2)  001244      000               $MAMS3: .BYTE   AMAMS3  ;;HIGH ADDRESS,M.S.BYTE
(2)  001245      000               $MTYP3: .BYTE   AMTYP3  ;;MEM.TYPE,BLK#3
(2)  001246  000000                $MADR3: .WORD   AMADR3  ;;MEM.LAST ADDRESS,BLK#3
(2)  001250      000               $MAMS4: .BYTE   AMAMS4  ;;HIGH ADDRESS,M.S.BYTE
(2)  001251      000               $MTYP4: .BYTE   AMTYP4  ;;MEM.TYPE,BLK#4
(2)  001252  000000                $MADR4: .WORD   AMADR4  ;;MEM.LAST ADDRESS,BLK#4
(2)  001254  000000                $VECT1: .WORD   AVECT1  ;;INTERRUPT VECTOR#1,BUS PRIORITY#1
(2)  001256  000000                $VECT2: .WORD   AVECT2  ;;INTERRUPT VECTOR#2BUS PRIORITY#2
(2)  001260  000000                $BASE:  .WORD   ABASE   ;;BASE ADDRESS OF EQUIPMENT UNDER TEST
(2)  001262  000000                $DEVM:  .WORD   ADEVM   ;;DEVICE MAP
(2)  001264  000000                $CDW1:  .WORD   ACDW1   ;;CONTROLLER DESCRIPTION WORD#1
(2)  001266  000000                $CDW2:  .WORD   ACDW2   ;;CONTROLLER DESCRIPTION WORD#2
(2)  001270  000000                $DDW0:  .WORD   ADDW0   ;;DEVICE DESCRIPTOR WORD#0
(2)  001272  000000                $DDW1:  .WORD   ADDW1   ;;DEVICE DESCRIPTOR WORD#1
(2)  001274  000000                $DDW2:  .WORD   ADDW2   ;;DEVICE DESCRIPTOR WORD#2
(2)  001276  000000                $DDW3:  .WORD   ADDW3   ;;DEVICE DESCRIPTOR WORD#3
(2)  001300  000000                $DDW4:  .WORD   ADDW4   ;;DEVICE DESCRIPTOR WORD#4
(2)  001302  000000                $DDW5:  .WORD   ADDW5   ;;DEVICE DESCRIPTOR WORD#5
(2)  001304  000000                $DDW6:  .WORD   ADDW6   ;;DEVICE DESCRIPTOR WORD#6
(2)  001306  000000                $DDW7:  .WORD   ADDW7   ;;DEVICE DESCRIPTOR WORD#7
(2)  001310  000000                $DDW8:  .WORD   ADDW8   ;;DEVICE DESCRIPTOR WORD#8
(2)  001312  000000                $DDW9:  .WORD   ADDW9   ;;DEVICE DESCRIPTOR WORD#9
(2)  001314  000000                $DDW10: .WORD   ADDW10  ;;DEVICE DESCRIPTOR WORD#10
(2)  001316  000000                $DDW11: .WORD   ADDW11  ;;DEVICE DESCRIPTOR WORD#11
(2)  001320  000000                $DDW12: .WORD   ADDW12  ;;DEVICE DESCRIPTOR WORD#12
(2)  001322  000000                $DDW13: .WORD   ADDW13  ;;DEVICE DESCRIPTOR WORD#13
(2)  001324  000000                $DDW14: .WORD   ADDW14  ;;DEVICE DESCRIPTOR WORD#14
(2)  001326  000000                $DDW15: .WORD   ADDW15  ;;DEVICE DESCRIPTOR WORD#15
```

```
(2)  001330                          $ETEND:
(2)                                  .MEXIT
(4)                                  .SBTTL  APT PARAMETER BLOCK
(4)
(5)                                  ;;******************************************************************
(4)                                  ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
(5)                                  ;;******************************************************************
(4)          001330                          .$X=.    ;;SAVE CURRENT LOCATION
(4)          000024                          .=24     ;;SET POWER FAIL TO POINT TO START OF PROGRAM
(4)  000024  000200                          200      ;;FOR APT START UP
(4)          000044                          .=44     ;;POINT TO APT INDIRECT ADDRESS PNTR.
(4)  000044  001330                          $APTHDR  ;;POINT TO APT HEADER BLOCK
(4)          001330                          .=.$X    ;;RESET LOCATION COUNTER
(5)                                  ;;******************************************************************
(4)                                  ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
(4)                                  ;INTERFACE SPEC.
(4)
(4)  001330                          $APTHD:
(4)  001330  000000                  $HIBTS: .WORD   0        ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
(4)  001332  001204                  $MBADR: .WORD   $MAIL    ;;ADDRESS OF APT MAILBOX (BITS 0-15)
(4)  001334  004540                  $TSTM:  .WORD   2400.    ;;RUN TIM OF LONGEST TEST
(4)  001336  000170                  $PASTM: .WORD   120.     ;;RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
(4)  001340  000360                  $UNITM: .WORD   240.     ;;ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
(4)  001342  000052                          .WORD   $ETEND-$MAIL/2 ;;LENGTH MAILBOX-ETABLE(WORDS)
(4)                                  .SBTTL  APT STATISTICS TABLE
(4)
(5)                                  ;;******************************************************************
(4)  001344                          $ASTAT:
(5)  001344  177777  000000                  .WORD   -1,0
(5)  001350  177777  000000                  .WORD   -1,0
(5)  001354  177777  000000                  .WORD   -1,0
(5)  001360  177777  000000                  .WORD   -1,0
(5)  001364  177777  000000                  .WORD   -1,0
(5)  001370  177777  000000                  .WORD   -1,0
(5)  001374  177777  000000                  .WORD   -1,0
(5)  001400  177777  000000                  .WORD   -1,0
(5)  001404  177777  000000                  .WORD   -1,0
(5)  001410  177777  000000                  .WORD   -1,0
(5)  001414  177777  000000                  .WORD   -1,0
(5)  001420  177777  000000                  .WORD   -1,0
(5)  001424  177777  000000                  .WORD   -1,0
(5)  001430  177777  000000                  .WORD   -1,0
(5)  001434  177777  000000                  .WORD   -1,0
(5)  001440  177777  000000                  .WORD   -1,0
(5)  001444  177777  000000                  .WORD   -1,0
(5)  001450  177777  000000                  .WORD   -1,0
(5)  001454  177777  000000                  .WORD   -1,0
(5)  001460  177777  000000                  .WORD   -1,0
(5)  001464  177777  000000                  .WORD   -1,0
(5)  001470  177777  000000                  .WORD   -1,0
(5)  001474  177777  000000                  .WORD   -1,0
(5)  001500  177777  000000                  .WORD   -1,0
(5)  001504  177777  000000                  .WORD   -1,0
(4)  001510  177777                  $ASTEND:        -1
(4)  001512  001344                  $APTR:  $ASTAT
(3)
```

```
(4)                                 ;;*****************************************************************
(3)                                 ;*THE FOLLOWING TAGS ARE USER DEFINED
(4)                                 ;;*****************************************************************
(3)  001514  000000                 $VERPC: .WORD   0       ;VIRTUAL PC LOCATION FOR ERROR TYPEOUT ROUTINE ($ERTYP).
(3)  001516  070032                 RESRVD: .WORD   070032  ;CORE PARITY REG BITS RESERVED FOR FUTURE USE.
(3)                                                         ;NOTE:  FOR MS11 MEMORY WITH PARITY, CHANGE TO 077772.
(3)  001520  000000                 LMAD:   .WORD   0       ;LAST CONTIGUOUS MEMORY ADDRESS (+2)
(3)  001522  000000                 LDDISP: .WORD   0       ;CONTAINS DISPLAY REGISTER IMAGE
(3)  001524                         MEMMAP:                 ;MEMORY MAP - EACH BIT CORRESPONDS TO 4K
(3)  001524  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001526  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001530                         TSTMAP:                 ;TEST MAP - WHICH BANKS ARE SELECTED FOR TEST.
(3)  001530  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001532  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001534                         SAVTST:                 ;SAVED TEST MAP - USED DURING FIRST PASS TO ONLY
(3)                                                         ; TEST EACH BANK ONCE.
(3)  001534  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001536  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001540                         PMEMAP:                 ;PARITY MAP - WHICH BANKS HAVE MEMORY PARITY
(3)  001540  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001542  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001544                         BITPT:                  ;POINTER TO CURRENT 4K BANK OF MEMORY
(3)  001544  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001546  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001550                         TMPPT:                  ;TEMPORARY POINTER FOR 2ND 4K BANK OF MEMORY
(3)  001550  000000                         .WORD   0       ;FIRST WORD CONTAINS LOW (0-64K) MAP
(3)  001552  000000                         .WORD   0       ;SECOND WORD CONTAINS HIGH (64-128K) MAP
(3)  001554  000000                 MMORE:  .WORD   0       ;LOOP ADDRESS FOR MULTIPLE BLOCK TESTING.
(3)                                                         ;SET UP BY "INITMM" AND "INITDN" ROUTINEES.
(3)                                                         ;USED BY "MMUP" AND "MMDOWN" ROUTINES.
(3)  001556     000                 SELFLG: .BYTE   0       ;OPERATOR SELECTED PARAMETERS FLAG.  (SA=204)
(3)  001557     000                 FLAG8K: .BYTE   0       ;8K BLOCK INDICATOR.  USED IN "INITMM" AND "MMUP".
(3)  001560     000                 OEFLG:  .BYTE   0       ;ODD/EVEN FLAG USED IN PARITY MEMORY BYTE TEST.
(3)          001562                         .EVEN
(3)  001562  000000                 FSTADR: .WORD   0       ;FIRST VIRTUAL ADDRESS TO BE TESTED.
(3)                                                         ;FIRST ADDRESS IS USER SELECTABLE.
(3)  001564  000000                 TMPFAD: .WORD   0       ;ADJUSTED FIRST ADDRESS.
(3)  001566  000000                 FADMSK: .WORD   0       ;BIT MASK TO ALLOW DOWNWARD ADDRESSING TESTS
(3)                                                         ; TO BREAK TO "MMDOWN" TO FIND FIRST ADDRESS.
(3)  001570  000000  000000         FADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH FIRST ADDRESS IS LOCATED.
(3)  001574  000000                 LSTADR: .WORD   0       ;LAST VIRTUAL ADDRESS (+2) TO BE TESTED.
(3)                                                         ;LAST ADDRESS IS USER SELECTABLE.
(3)  001576  000000                 TMPLAD: .WORD   0       ;ADJUSTED LAST ADDRESS.
(3)  001600  000000                 LADMSK: .WORD   0       ;BIT MASK TO ALLOW UPWARD ADDRESSING TESTS
(3)                                                         ; TO BREAK TO "MMUP" TO FIND LAST ADDRESS.
(3)  001602  000000  000000         LADMAP: .WORD   0,0     ;MAP OF BANK IN WHICH LAST ADDRESS IS LOCATED.
(3)  001606  000000                 BLKMSK: .WORD   0       ;BLOCK MASK, DETERMINES THE BLOCK SIZE.
(3)  001610  000000                 .CONST: .WORD   0       ;USER SELECTABLE CONSTANT DATA.
(3)  001612  000004                 WWP:    .WORD   4       ;WRITE WRONG PARITY COMMAND
(3)  001614  000000                 TEMP:   .WORD   0       ;TEMPORARY STORAGE
(3)  001616  000000                 CASFLG: .WORD   0       ;CACHE PRESENT FLAG
(3)  001620  177746                 CASREG: .WORD   177746  ;CACHE CONTROL REGISTER
(3)
(4)                                 ;;*****************************************************************
(3)                                 ;*  RELATIVE ADDRESSING TABLE.
(3)                                 ;*  THE FOLLOWING LOCATIONS ARE MODIFIED AT RELOCATION TIME TO ALLOW
```

```
(3)                                     ;*  RELATIVE ADDRESSING TO GET THE RELOCATED VALUE OF THE ARGUEMEMT TAGS.
(4)                                     ;;*****************************************************************
(3)  001622                             RADTAB:
(3)  001622  001100                     .STACK: STACK           ;STACK POINTER INITIAL ADDRESS.
(3)  001624  001516                     .RESRV: RESRVD          ;PARITY REGISTER RESERVED BIT MASK ADDRESS.
(3)  001626  002076                     .MPR0:  MPR0            ;MEMORY PARITY REGISTER TABLE ADDRESS.
(3)  001630  002276                     .MPRX:  MPRX            ;MEMORY PARITY REGISTER EXIST TABLE ADDRESS.
(3)  001632  012062                     .PBTRP: PBTRP           ;PARITY BYTE TEST TRAP ROUTINE ADDRESS.
(3)  001634  002050                     .MPPAT: MPPATS          ;MEMORY PARITY PATTERN TABLE ADDRESS.
(3)  001636  017440                     .PESRV: PESRV           ;MEMORY PARITY ERROR TRAP ROUTINE ADDRESS.
(3)  001640  002340                     .ERRTB: $ERRTB          ;ERROR TYPEOUT TABLE PONTER.
(3)  001642  000010                     .EIGHT: 8.              ;DECIMAL TYPE ROUTINE COUNT DESIGNATOR.
(3)  001644  014014                     .TST32: TST32           ;SCOPE ABORT ADR FOR WHEN NO MEM AVA FOR TEST.
(4)                                     ;;*****************************************************************
(3)                                     ;*  DATA CONTAINERS FOR ERROR PRINTOUT.
(4)                                     ;;*****************************************************************
(3)  001646  001116  001120  001124     DT1:    $ERRPC,$GDADR,$GDDAT,$BDDAT,0
(3)  001654  001126  000000
(3)  001660  001514  001116  001120     DT2:    $VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT,0
(3)  001666  001124  001126  000000
(3)  001674  001514  001116  001120     DT12:   $VERPC,$ERRPC,$GDADR,$GDDAT,0
(3)  001702  001124  000000
(3)  001706  001514  001116  001160     DT14:   $VERPC,$ERRPC,$TMP0,$GDADR,0
(3)  001714  001120  000000
(3)  001720  001514  001116  001120     DT15:   $VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT,0
(3)  001726  001160  001124  001126
(3)  001734  000000
(3)  001736  001514  001116  001160     DT21:   $VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT,0
(3)  001744  001120  001124  001126
(3)  001752  000000
(3)  001754  001514  001116  001120     DT23:   $VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDAT,0
(3)  001762  001122  001124  001126
(3)  001770  000000
(3)  001772  001514  001116  001122     DT24:   $VERPC,$ERRPC,$BDADR,0
(3)  002000  000000
(3)  002002  001514  001116  001122     DT25:   $VERPC,$ERRPC,$BDADR,$TMP0,$TMP1,0
(3)  002010  001160  001162  000000
(3)  002016  001514  001116  001160     DT26:   $VERPC,$ERRPC,$TMP0,$TMP1,0
(3)  002024  001162  000000
(3)  002030  001160  001162  001120     DT30:   $TMP0,$TMP1,$GDADR,$BDDAT,0
(3)  002036  001126  000000
(3)  002042  001166  000000             DT31:   $TMP3,0
(3)  002046  177777                             .WORD   -1      ;TABLE TERMINATOR.
(3)
(3)                                     .SBTTL  MEMORY PARITY PATTERNS TABLE
(4)                                     ;;*****************************************************************
(3)                                     ;THE FOLLOWING ARE THE PARITY PATTERNS EXERCISED THRUOUT MEMORY
(4)                                     ;;*****************************************************************
(3)
(3)  002050  125325                     MPPATS: 125325                  ;EVEN,ODD
(3)  002052  152652                             152652                  ;ODD,EVEN
(3)  002054  052452                             052452                  ;EVEN,ODD
(3)  002056  025125                             025125                  ;ODD,EVEN
(3)  002060  102070                             102070                  ;EVEN,EVEN
(3)  002062  072527                             072527                  ;ODD,ODD
(3)  002064  177777                             177777                  ;EVEN,EVEN
```

```
(3)  002066  107030                    107030                  ;ODD,ODD
(3)  002070  152525                    152525                  ;ODD,EVEN
(3)  002072  000000                    0                       ;EXTRA PATTERN HOLDER FOR
(3)                                                            ;FUTURE USE
(3)  002074  000000             MPEND: 0                       ;TABLE TERMINATOR
(3)
(3)
(3)                              .SBTTL  MEMORY PARITY REGISTER ADDRESS TABLE
(4)                              ;////////////////////////////////////////////////////////////////
(3)                              ;*  THE FOLLOWING REPRESENTS THE MEMORY PARITY REGISTER ADDRESS TABLE
(3)                              ;*  FROM WHICH PARITY MEMORY IS ADDRESSED & CONTROLLED:
(3)                              ;*
(3)                              ;*  THE LEAST SIGNIFICANT BIT IN THE DEVICE ADDRESS IS SET TO A ONE (1)
(3)                              ;*  IF THE CONTROL IS FOUND NOT TO BE PRESENT.  THE MEMORY PRESENT UNDER
(3)                              ;*  THE CONTROL OF EACH CONTROLLER IS REPRESENTED BY TWO (2) WORDS FOLLOWING
(3)                              ;*  THE DEVICE ADDRESS,  EACH BIT REPRESENTING A 4K BLOCK.  I.E.
(3)                              ;*  FIRST WORD BIT0 = 0 - 4K,  BIT1 = 4 - 8K,...  BIT15 = 60 - 64K
(3)                              ;*  SECOND WORD BIT0 = 64 - 68K,... BIT14 = 120 - 124K.
(4)                              ;////////////////////////////////////////////////////////////////
(3)  002076  172101             MPR0:  172100 +1               ;PARITY STATUS REGISTER
(3)  002100  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002102  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002104  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002106  172103             MPR1:  172102 +1               ;PARITY STATUS REGISTER
(3)  002110  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002112  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002114  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002116  172105             MPR2:  172104 +1               ;PARITY STATUS REGISTER
(3)  002120  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002122  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002124  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002126  172107             MPR3:  172106 +1               ;PARITY STATUS REGISTER
(3)  002130  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002132  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002134  000000                    0                       ;MASK FOR MOS CORE,MS11-K
(3)  002136  172111             MPR4:  172110 +1               ;PARITY STAUS REGISTER
(3)  002140  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002142  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002144  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002146  172113             MPR5:  172112 +1               ;PARITY STATUS REGISTER
(3)  002150  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002152  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002154  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002156  172115             MPR6:  172114 +1               ;PARITY STATUS REGISTER
(3)  002160  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002162  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002164  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002166  172117             MPR7:  172116 +1               ;PARITY STATUS REGISTER
(3)  002170  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002172  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002174  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002176  172121             MPR8:  172120 +1               ;PARITY STATUS REGISTER
(3)  002200  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002202  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002204  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002206  172123             MPR9:  172122 +1               ;PARITY STATUS REGISTER
```

```
(3)  002210  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002212  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002214  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002216  172125          MPR10:    172124 +1               ;PARITY STATUS REGISTER
(3)  002220  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002222  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002224  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002226  172127          MPR11:    172126 +1               ;PARITY STATUS REGISTER
(3)  002230  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002232  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002234  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002236  172131          MPR12:    172130 +1               ;PARITY STATUS REGISTER
(3)  002240  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002242  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002244  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002246  172133          MPR13:    172132 +1               ;PARITY STATUS REGISTER
(3)  002250  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002252  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002254  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002256  172135          MPR14:    172134 +1               ;PARITY STATUS REGISTER
(3)  002260  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002262  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002264  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)  002266  172137          MPR15:    172136 +1               ;PARITY STATUS REGISTER
(3)  002270  000000                    0                       ;CONTROL MAP (LOW 64K)
(3)  002272  000000                    0                       ;CONTROL MAP (HIGH 64K)
(3)  002274  000000                    0                       ;MASK FOR MOS,CORE,MS11-K
(3)                          ;THIS IS THE END OF THE TABLE !
(3)  002276  000021          MPRX:     .BLKW   17.             ;TABLE TO HOLD JUST PARITY STATUS REGISTERS THAT EXIST.
(3)                                                            ; (THE EXTRA WORD IS FOR A TERMINATOR.)
(3)
```

```
                                .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


(1)  002340                     $ERRTB:
6486 002340                     CHGG1:
6487                            ;*  ITEM 1
6488 002340  027020                 DM1             ;PARITY REGISTER DATA ERROR.
6489 002342  030377                 DH1             ;PC,REG,S/B,WAS
6490 002344  001646                 DT1             ;$ERRPC,$GDADR,$GDDAT,$BDDAT
6491 002346  030744                 DF1             ;16,18,16,16
6492                            ;*  ITEM 2
6493 002350  027054                 DM2             ;ADDRESS TEST ERROR(TST1-5).
6494 002352  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6495
6496 002354  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6497 002356  030750                 DF2             ;16,18,18,16,16
6498                            ;*  ITEM 3
6499 002360  027054                 DM2             ;ADDRESS TEST ERROR(TST1-5).
6500 002362  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6501 002364  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6502 002366  030755                 DF3             ;16,18,18,8,8
6503                            ;*  ITEM 4
6504 002370  027110                 DM4             ;CONSTANT DATA ERROR(TST6-10).
6505 002372  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6506 002374  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6507 002376  030750                 DF2             ;16,18,18,16,16
6508                            ;*  ITEM 5
6509 002400  027146                 DM5             ;ROTATING BIT ERROR(TST11-12).
6510 002402  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6511 002404  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6512 002406  030750                 DF2             ;16,18,18,16,16
6513                            ;*  ITEM 6
6514 002410  027204                 DM6             ;MOS REFRESH TEST ERROR (TST30-31).
6515 002412  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6516 002414  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6517 002416  030750                 DF2             ;16,18,18,16,16
6518                            ;*  ITEM 7
6519 002420  027250                 DM7             ;3 XOR 9 PATTERN ERROR(TST13-16).
6520 002422  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6521 002424  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6522 002426  030750                 DF2             ;16,18,18,16,16
6523                            ;*  ITEM 10
6524 002430  027311                 DM10            ;MARCHING 1'S AND 0'S ERROR(TST27).
6525 002432  030416                 DH2             ;V/PC,P/PC,MA,S/B,WAS
6526 002434  001660                 DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
```

```
6527  002436  030750                        DF2             ;16,18,18,16,16
6528                                  ;*  ITEM 11
6529  002440  027355                        DM11            ;PARITY MEMORY ADDRESS ERROR(TST17).
6530  002442  030416                        DH2             ;V/PC,P/PC,MA,S/B,WAS
6531  002444  001660                        DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6532  002446  030755                        DF3             ;16,18,18,8,8
6533                                  ;*  ITEM 12
6534  002450  027421                        DM12            ;DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).
6535  002452  030443                        DH12            ;V/PC,P/PC,MA,S/B
6536  002454  001674                        DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
6537  002456  030755                        DF3             ;16,18,18,8
6538                                  ;*  ITEM 13
6539  002460  027475                        DM13            ;WRONG PARITY TRAPED, BUT NO REGISTER SHOWS ERROR FLAG.
6540  002462  030443                        DH12            ;V/PC,P/PC,MA,S/B
6541  002464  001674                        DT12            ;$VERPC,$ERRPC,$GDADR,$GDDAT
6542  002466  030755                        DF3             ;16,18,18,8
6543                                  ;*  ITEM 14
6544  002470  027565                        DM14            ;PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).
6545  002472  030464                        DH14            ;V/PC,P/PC,REG,MA
6546  002474  001706                        DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
6547
6548  002476  030762                        DF14            ;16,18,18,18
6549                                  ;*  ITEM 15
6550  002500  027020                        DM1             ;PARITY REGISTER DATA ERROR.
6551  002502  030505                        DH15            ;V/PC,P/PC,MAUT,REG,S/B,WAS
6552  002504  001720                        DT15            ;$VERPC,$ERRPC,$GDADR,$TMP0,$GDDAT,$BDDAT
6553  002506  030762                        DF14            ;16,18,18,18,16,16
6554                                  ;*  ITEM 16
6555  002510  027664                        DM16            ;MORE THAN ONE REGISTER INDICATED PARITY ERROR.
6556  002512  030464                        DH14            ;V/PC,P/PC,REG,MA
6557  002514  001706                        DT14            ;$VERPC,$ERRPC,$TMP0,$GDADR
6558  002516  030762                        DF14            ;16,18,18,
6559                                  ;*  ITEM 17
6560  002520  027743                        DM17            ;DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR
6561                                                        ;  TRAPPED(TST21).
6562  002522  030416                        DH2             ;V/PC,P/PC,MA,S/B,WAS
6563  002524  001660                        DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6564  002526  030755                        DF3             ;16,18,18,8,8
6565                                  ;*  ITEM 20
6566  002530  030041                        DM20            ;RANDOM DATA ERROR(TST20).
6567  002532  030416                        DH2             ;V/PC,P/PC,MA,S/B,WAS
6568  002534  001660                        DT2             ;$VERPC,$ERRPC,$GDADR,$GDDAT,$BDDAT
6569  002536  030750                        DF2             ;16,18,18,16,16
6570                                  ;*  ITEM 21
6571  002540  030073                        DM21            ;INSTRUCTION EXICUTION ERROR(TST21-26).
6572  002542  030540                        DH21            ;V/PC,P/PC,IUT,MA,S/B,WAS
6573  002544  001736                        DT21            ;$VERPC,$ERRPC,$TMP0,$GDADR,$GDDAT,$BDDAT
6574  002546  030770                        DF21            ;16,18,16,18,16,16
6575                                  ;*  ITEM 22         ;NOT USED
6576  002550  000000                        0               ;CHGG1
6577  002552  000000                        0
6578  002554  000000                        0
6579  002556  000000                        0
6580                                  ;*  ITEM 23
6581  002560  030142                        DM23            ;PROGRAM CODE CHANGED WHEN RELOCATED.
6582  002562  030571                        DH23            ;V/PC,P/PC,SRC MA,DST MA,S/B,WAS
```

```
 6583  002564  001754                        DT23            ;$VERPC,$ERRPC,$GDADR,$BDADR,$GDDAT,$BDDA
 6584  002566  030762                        DF14            ;16,18,18,18,16,16
 6585                                ;*  ITEM 24
 6586  002570  030207                        DM24            ;TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.
 6587  002572  030631                        DH24            ;V/PC,P/PC,TRP/PC
 6588  002574  001772                        DT24            ;$VERPC,$ERRPC,$BDADR
 6589  002576  030762                        DF14            ;16,18,18
 6590                                ;*  ITEM 25
 6591  002600  030263                        DM25            ;TRAPPED TO 114.
 6592  002602  030652                        DH25            ;V/PC,P/PC,TRP/PC,REG,WAS
 6593  002604  002002                        DT25            ;$VERPC,$ERRPC,$BDADR,$TMP0,$TMP1
 6594  002606  030762                        DF14            ;16,18,18,18,16
 6595                                ;*  ITEM 26
 6596  002610  030303                        DM26            ;FAILED TO TRAP.
 6597  002612  030703                        DH26            ;V/PC,P/PC,REG,WAS
 6598  002614  002016                        DT26            ;$VERPC,$ERRPC,$TMP0,$TMP1
 6599  002616  030750                        DF2             ;16,18,18,16
 6600                                ;*  ITEM 27
 6601  002620  030323                        DM27            ;(ACTION ENABLE WASN'T SET).
 6602  002622  030703                        DH26            ;V/PC,P/PC,REG,WAS
 6603  002624  002016                        DT26            ;$VERPC,$ERRPC,$TMP0,$BDDAT
 6604  002626  030750                        DF2             ;16,18,18,16
 6605                                ;*  ITEM 30
 6606  002630  000000                        0               ;NO MESSAGE.
 6607  002632  030725                        DH30            ;REG,WAS,MA,WAS
 6608  002634  002030                        DT30            ;$TMP0,$TMP1,$GDADR,$BDDAT
 6609  002636  030776                        DF30            ;18,16,18,8
 6610                                ;*  ITEM 31
 6611  002640  030357                        DM31            ;TRAPPED TO 4
 6612  002642  000000                        0               ;NO HEADER
 6613  002644  002042                        DT31            ;$TMP3
 6614  002646  030776                        DF30            ;18
 6615
 6616                                .SBTTL START:   SETUP AND MAP MEMORY

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

 6621  002650  105067  176702          START:  CLRB    SELFLG          ;CLEAR SELECT PARAMETER FLAG.
 6622  002654  000403                          BR      STARTA          ;GO DO SETUP AND MEMORY MAP.
 6623  002656  112767  177777  176672  SELECT: MOVB    #-1,    SELFLG  ;SET THE SELECT PARAMETERS FLAG.
 6624  002664                          STARTA:
  (1)                                   .SBTTL  INITIALIZE THE COMMON TAGS
  (1)                                   ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
  (1)   002664  012706  001100                  MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
  (1)   002670  005026                          CLR     (R6)+           ;;CLEAR MEMORY LOCATION
  (1)   002672  022706  001140                  CMP     #SWR,R6 ;;DONE?
  (1)   002676  001374                          BNE     .-6             ;;LOOP BACK IF NO
  (1)   002700  012706  001100                  MOV     #STACK,SP       ;;SETUP THE STACK POINTER
  (1)                                   ;;INITIALIZE A FEW VECTORS
  (1)   002704  012737  000610  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
  (1)   002712  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
  (1)   002720  016767  011242  011232          MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
  (2)                                   ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
```

```
   (2)                                  ;;EQUAL TO A ''-1'', SETUP FOR A SOFTWARE SWITCH REGISTER.
   (2)  002726  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
   (2)  002732  012737  002766  000004          MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
   (2)  002740  012767  177570  176172          MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
   (2)  002746  012767  177570  176166          MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
   (2)  002754  022777  177777  176156          CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
   (2)  002762  001012                          BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
   (2)                                                                  ;;AND  THE HARDWARE SWR IS NOT  -1
   (2)  002764  000403                          BR      65$             ;;BRANCH IF NO TIMEOUT
   (2)  002766  012716  002774          64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
   (2)  002772  000002                          RTI
   (2)  002774  012767  000176  176136  65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
   (2)  003002  012767  000174  176132          MOV     #DISPREG,DISPLAY
   (2)  003010  012637  000004          66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR
   (1)
   (2)  003014  005067  176172                  CLR     $PASS           ;;CLEAR PASS COUNT
   (2)  003020  132767  000200  176177          BITB    #APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
   (2)  003026  001403                          BEQ     67$             ;;YES,USE NON-APT SWITCH
   (2)  003030  012767  001226  176102          MOV     #$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
   (2)  003036                          67$:
  6625  003036  005067  176460                  CLR     LDDISP          ;CLEAR DISPLAY REGISTER STORAGE LOCN
  6626  003042  005077  176074                  CLR     @DISPLAY        ;CLEAR DISPLAY REGISTER
  6627                                  .SBTTL  TYPE PROGRAM NAME
   (1)                                  ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
   (1)  003046  005227  177777                  INC     #-1             ;;FIRST TIME?
   (1)  003052  001040                          BNE     68$             ;;BRANCH IF NO
   (1)  003054  022737  014232  000042          CMP     #$ENDAD,@#42    ;;ACT-11?
   (1)  003062  001434                          BEQ     68$             ;;BRANCH IF YES
   (2)  003064  004567  020432                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
   (2)  003070  003142                          .WORD   69$             ;ADDRESS OF MESSAGE TO BE TYPED
   (2)                                  .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
   (2)  003072  005737  000042                  TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
   (2)  003076  001015                          BNE     70$             ;;BRANCH IF YES
   (2)  003100  126727  176120  000001          CMPB    $ENV,#1         ;;ARE WE RUNNING UNDER APT?
   (2)  003106  001411                          BEQ     70$             ;;BRANCH IF YES
   (2)  003110  026727  176024  000176          CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
   (2)  003116  001010                          BNE     71$             ;;BRANCH IF NO
   (4)                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $GTSWR ROUTINE
   (4)                                  ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
   (4)  003120  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
   (4)  003124  004767  017316                  JSR     PC,     $GTSWR  ;GO TO THE SUBROUTINE
   (2)  003130  000403                          BR      71$
   (2)  003132  112767  000001  175774  70$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
   (2)  003140                          71$:
   (1)  003140  000405                          BR      68$             ;;GET OVER THE ASCIZ
   (1)                                  ;;69$:  .ASCIZ  <CRLF>'CZQMCGO'<CRLF>
   (1)  003154                          68$:
  6628  003154  010700                          MOV     PC,     R0      ;GET CURRENT PROGRAM COUNTER.
  6629  003156  022700  003156                  CMP     #.,     R0      ;CHECK IF THE PROGRAM IS RELOCATED.
  6630  003162  001402                          BEQ     10$             ;BR IF PROGRAM NOT RELOCATED.
  6631  003164  000167  175110                  JMP     RESTAR          ;GO TRY TO RELOCTED BEFORE CONTINUING.
  6632  003170  012767  000003  175404  10$:    MOV     #3,     PRGMAP  ;INITIALIZE PROGRAM MAP....LO 64K.
  6633  003176  005067  175402                  CLR     PRGMAP+2        ;...HI 64K.
  6634  003202  005067  175372                  CLR     RELOCF          ;INIT THE RELOCATION FACTOR.
  6635  003206  105737  001224                  TSTB    @#$ENV          ;CHECK FOR APT11
  6636  003212  001011                          BNE     13$             ;BR IF APT11
```

```
 6637  003214  005737  000042            TST     @#42            ;CHECK FOR STANDALONE
 6638  003220  001406                    BEQ     13$             ;BR IF STANDALONE
 6639  003222  023737  000042  000046    CMP     @#42,@#46       ;CHECK FOR ACT11
 6640  003230  001402                    BEQ     13$             ;BR IF ACT11
 6641                                                            ;MUST BE XXDP
 6642  003232  004767  014122            JSR     PC,SAVLDR       ;GO SAVE LOADERS
 6643
 6644                                ;*  CHECK IF MEMORY MANAGEMENT IS AVAILABLE, AND SET IT UP IF IT IS.
 6645  003236  005067  175344    13$:    CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG
 6646  003242  032777  010000  175670    BIT     #SW12,  @SWR    ;CHECK FOR INHIBIT KT11 SWITCH
 6647  003250  001014                    BNE     IMPCK           ;BRANCH IF SET
 6648  003252  012737  003302  000004    MOV     #IMPCK,@#ERRVEC ;SET UP TIMEOUT TRAP VECTOR
 6649  003260  005037  177572            CLR     @#SR0           ;CLEAR MEM MGMT STATUS REG
 6650  003264  004767  011020            JSR     PC,     MMINIT  ;MEM MGMT INITIALIZATION ROUTINE.
 6651  003270  005267  175312            INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG
 6652  003274  004567  020222            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003300  025364                    .WORD   MMAMES          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                           ;'KT11 AVAILABLE'
 6653
 6654                                ;*      CHECK IF CACHE PRESENT, IF SO TURN IT OFF!!!
 6655  003302  012706  001100    IMPCK:  MOV     #STACK, SP
 6656  003306  005067  176304            CLR     CASFLG          ;CLEAR CACHE PRESENT FLAG
 6657  003312  012737  003334  000004    MOV     #MAPMEM,@#ERRVEC
 6658  003320  052767  000014  174420    BIS     #14,    IMPCHE
 6659  003326  012767  000001  176262    MOV     #1,     CASFLG  ;SET CACHE PRESENT FLAG
 6660
 6661                                ;;*****************************************************************
 6662                                ;*  ROUTINE TO MAP ALL OF MEMORY.
 6663                                ;*  ONLY FULL 4K BANKS WILL BE RECOGNIZED.
 6664                                ;*      R0 = MEMMAP POINTER...LO 64K.
 6665                                ;*      R1 = MEMMAP POINTER...HI 64K.
 6666                                ;*      R2 = ADDRESS POINTER
 6667                                ;*      R3 = BANK POINTER...LO 64K.
 6668                                ;*      R4 = BANK POINTER...HI 64K.
 6669                                ;*      R5 = SCRATCH REGISTER.
 6670                                ;;*****************************************************************
 6671  003334  012706  001100    MAPMEM: MOV     #STACK, SP      ;RESET THE STACK
 6672  003340  012700  001524            MOV     #MEMMAP,R0      ;SET UP MEMORY MAP POINTER...LO 64K.
 6673  003344  012701  001526            MOV     #MEMMAP+2,R1    ;...HI 64K.
 6674  003350  005010                    CLR     (R0)            ;CLR MEMORY MAP...LO 64K.
 6675  003352  005011                    CLR     (R1)            ;...HI 64K.
 6676  003354  005002                    CLR     R2              ;SET ADDRESS POINTER TO 0
 6677  003356  012703  000001            MOV     #1,     R3      ;SETUP 4K BANK POINTER...LO 64K.
 6678  003362  005004                    CLR     R4              ;...HI 64K.
 6679  003364  005067  175576            CLR     $TMP3           ;INIT TEMPORARY HIGH ADDRESS BITS.
 6680  003370  004567  020126            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003374  025431                    .WORD   MEMMES          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                           ;'MEMORY MAP:'
 6681  003376  012737  003512  000004    MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
 6682  003404  011222            1$:     MOV     (R2),   (R2)+   ;READ+WRITE ALL MEMORY
 6683  003406  032702  017777            BIT     #MASK4K,R2      ;CHECK FOR 4K BOUNDRY
 6684  003412  001374                    BNE     1$              ;BRANCH IF MORE IN BANK
 6685  003414  050310                    BIS     R3,     (R0)    ;SET FLAG FOR BANK...LO 64K.
 6686  003416  050411                    BIS     R4,     (R1)    ;...HI 64K.
 6687  003420  010267  175540            MOV     R2,     $TMP2   ;SAVE ADDRESS POINTER.
 6688  003424  005367  175534            DEC     $TMP2           ;ADJUST TO LAST ADR, LAST BANK.
```

```
 6689  003430  005767  175152            TST     MMAVA           ;CHECK FOR MEM MGMT.
 6690  003434  001432                    BEQ     3$              ;BR IF NO MEM MGMT.
 6691  003436  042767  160000  175520    BIC     #160000,$TMP2   ;CLEAR BANK BITS ON RELATIVE ADDRESS.
 6692  003444  013705  172344            MOV     @#KIPAR2,R5     ;SAVE KIPAR2.
 6693  003450  005067  175512            CLR     $TMP3           ;MAKE SURE HI BITS ARE INIT.
 6694  003454  006305                    ASL     R5              ;SHIFT IT 6 PLACES.
 6695  003456  006305                    ASL     R5
 6696  003460  006305                    ASL     R5
 6697  003462  006305                    ASL     R5
 6698  003464  006305                    ASL     R5
 6699  003466  006167  175474            ROL     $TMP3
 6700  003472  006305                    ASL     R5
 6701  003474  006167  175466            ROL     $TMP3
 6702  003500  060567  175460            ADD     R5,     $TMP2   ;MAKE LAST ADR PHYSICAL.
 6703  003504  005567  175456            ADC     $TMP3
 6704  003510  000404                    BR      3$              ;GO TO UPDATE POINTERS.
 6705
 6706                          ;*  TIMEOUT TRAPS TO HERE
 6707  003512  022626          2$:     CMP     (SP)+,  (SP)+   ;RESTORE THE STACK POINTER
 6708  003514  052702  017777            BIS     #MASK4K,R2      ;LAST ADDRESS OF 4K BANK
 6709  003520  005202                    INC     R2              ;FIRST ADDRESS OF NEXT BANK.
 6710  003522  005767  175060  3$:       TST     MMAVA           ;CHECK FOR MEM MGMT
 6711  003526  001411                    BEQ     4$              ;BRANCH IF NO MEM MGMT
 6712  003530  062737  000200  172344    ADD     #200, @#KIPAR2  ;UPDATE THIRD PAR
 6713  003536  012702  040000            MOV     #40000, R2      ;POINT TO START OF THIRD PAR
 6714  003542  006303                    ASL     R3              ;UPDATE LO BANK POINTER.
 6715  003544  006104                    ROL     R4              ;UPDATE HI BANK POINTER
 6716  003546  100316                    BPL     1$              ;BRANCH IF MORE MEMORY TO MAP.
 6717  003550  000402                    BR      5$              ;EXIT WHEN DONE.
 6718
 6719  003552  106303          4$:     ASLB    R3              ;UPDATE MAP POINTER
 6720  003554  100313                    BPL     1$              ;BRANCH IF NOT YET DONE
 6721  003556  012737  025124  000004  5$:     MOV     #ERRTRP, @#ERRVEC ;RESET TIMEOUT VECTOR
 6722  003564  004767  014632            JSR     PC,     TYPMAP  ;GO TYPE THE MAP.
 6723  003570  004567  017726            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003574  001201                    .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
 6724  003576  011067  175732            MOV     (R0),   SAVTST  ;SET UP TEST MAP...LO 64K.
 6725  003602  011167  175730            MOV     (R1),   SAVTST+2 ;...HI 64K.
 6726  003606  011000                    MOV     (R0),   R0      ;GET LOW MEM MAP
 6727  003610  042700  177760            BIC     #177760, R0     ;MASK ALL BUT BOTTOM 4 BANKS
 6728  003614  020027  000017            CMP     R0,     #17     ;CHECK THAT BOTTOM 16K IS ALL THERE.
 6729  003620  001530                    BEQ     GMPR            ;BRANCH IF BOTTOM 16K EXISTS
 6730  003622  004567  017674            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003626  025534                    .WORD   INSUFF          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                           ;'FIRST 16K OF MEMORY NOT ALL THERE!''
 6731  003630  000000          6$:     HALT                    ;FATAL ERROR HALT...
 6732                                                           ;MEMORY IS NOT CONFIGURED TO RUN THIS PROGRAM.
 6733                          ;;*************************************************************
 6734                          ;*  SPECIAL ROUTINE TO TYPE OUT ALL UNIBUS ADDRESSES WHICH RESPOND TO
 6735                          ;*   DATI, DATIP, DATO, AND DATOB.
 6736                          ;;*************************************************************
 6737  003632  012706  001100  TIMOUT: MOV     #STACK, SP      ;SET UP THE STACK POINTER.
 6738  003636  005067  174744            CLR     MMAVA           ;CLEAR MEM MGMT AVAILABLE FLAG.
 6739  003642  032777  010000  175270    BIT     #SW12,  @SWR    ;CHECK IF MEM MGMT TO BE INHIBITED.
 6740  003650  001011                    BNE     1$              ;BR IF NO MEM MGMT.
 6741  003652  012737  003674  000004    MOV     #1$,    @#ERRVEC ;SET TIMEOUT FOR MEM MGMT CHECK.
```

```
 6742  003660  005037  177572              CLR     @#SR0           ;CHECK FOR MEM MGMT...TIMES OUT IF NONE.
 6743  003664  004767  010420              JSR     PC,     MMINIT  ;INIT ALL MEM MGMT REGISTERS.
 6744  003670  005267  174712              INC     MMAVA           ;SET MEM MGMT AVAILABLE FLAG.
 6745  003674                        1$:
  (2)  003674  004567  017622              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003700  025447                      .WORD   BYTMES          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'BYTE MEMORY MAP:'
 6746  003702  005000                      CLR     R0              ;SET UP TYPE OUT FLAG.
 6747  003704  005002                      CLR     R2              ;SET ADDRESS POINTER TO ZERO.
 6748  003706  012737  003752  000004      MOV     #20$,   @#ERRVEC ;SET TIME OUT VEC TO SERVICE NON-EX MEM.
 6749  003714  105712                10$:  TSTB    (R2)            ;DO DATI ONLY.
 6750  003716  032702  000001              BIT     #BIT0,  R2      ;CHECK FOR WORD ADDRESS.
 6751  003722  001001                      BNE     11$             ;BR IF ODD BYTE ADDRESS.
 6752  003724  011212                      MOV     (R2),   (R2)    ;DO DATI, DATO...NOP FOR READ ONLY MAP.
 6753  003726  151212                11$:  BISB    (R2),   (R2)    ;DO DATI, DATIP, DATOB... NOP FOR READ ONLY MAP.
 6754  003730  005700                      TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 6755  003732  001023                      BNE     30$             ;BR IF ALREADY TYPED 'FROM'.
 6756  003734  004567  017562              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003740  025517                      .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'FROM'
 6757  003742  010246                      MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
  (1)  003744  004767  021212              JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 6758  003750  000413                      BR      29$             ;GO TO ADDRESS POINTER UPDATE.
 6759                              ;*  TIME OUTS COME HERE.
 6760  003752  022626                20$:  CMP     (SP)+,  (SP)+   ;POP TWO OFF STACK.
 6761  003754  005700                      TST     R0              ;CHECK FOR PREVIOUS TYPOUT.
 6762  003756  001411                      BEQ     30$             ;BR IF ALREADY TYPED 'TO'.
 6763  003760  004567  017536              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  003764  025527                      .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'TO'
 6764  003766  005302                      DEC     R2              ;BACK UP ONE BYTE.
 6765  003770  010246                      MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
  (1)  003772  004767  021164              JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 6766  003776  005202                      INC     R2              ;RESET ADDRESS POINTER.
 6767  004000  005100                29$:  COM     R0              ;RESET PREVIOUS TYPOUT FLAG.
 6768  004002  005202                30$:  INC     R2              ;UPDATE ADDRESS POINTER TO NEXT BYTE.
 6769  004004  001423                      BEQ     31$             ;EXIT IF ZERO REACHED.
 6770  004006  032702  017777              BIT     #MASK4K,R2      ;CHECK FOR 4K BANK BOUNDRY.
 6771  004012  001340                      BNE     10$             ;BR IF MORE THIS 4K BANK.
 6772  004014  005767  174566              TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
 6773  004020  001735                      BEQ     10$             ;BR IF NO MEM MGMT.
 6774  004022  022737  007600  172346      CMP     #7600,  @#KIPAR3 ;CHECK FOR END OF LAST 4K BANK.
 6775  004030  001411                      BEQ     31$             ;EXIT WHEN ALL DONE.
 6776  004032  012702  060000              MOV     #60000, R2      ;RESET VIRTUAL ADDRESS POINTER.
 6777  004036  013737  172346  172344      MOV     @#KIPAR3,@#KIPAR2 ;SAVE MEM MGMT REG FOR TYPEOUT.
 6778  004044  062737  000200  172346      ADD     #200,   @#KIPAR3 ;UPDATE MEM MGMT REG 2 TO NEXT 4K BANK.
 6779  004052  000720                      BR      10$             ;BR BACK TO DO NEXT BANK.
 6780  004054  005700                31$:  TST     R0              ;CHECK PREVIOUS TYPE FLAG BEFORE EXIT.
 6781  004056  001407                      BEQ     32$             ;BR TO EXIT IF TYPING ALL DONE.
 6782  004060  004567  017436              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  004064  025527                      .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'TO'
 6783  004066  005302                      DEC     R2              ;BACK ADDRESS POINTER UP ONE BYTE.
 6784  004070  010246                      MOV     R2,     -(SP)   ;PUT THE DATA ON THE STACK.
  (1)  004072  004767  021064              JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 6785  004076  000000                32$:  HALT                    ;* THIS ROUTINE IS FOR DEBUG USE ONLY.
```

```
                                                ;*  TO RUN THE MAIN PROGRAM RESTART AT 200 OR 204.
004100  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
                                ;;*****************************************************************

004102  012704  002276          GMPR:   MOV     #MPRX,  R4      ;SET UP POINTER TO PARITY REG EXIST TABLE.
004106  032777  000100  175024          BIT     #SW06,  @SWR    ;CHECK FOR INHIBIT PARITY SWITCH.
004114  001036                          BNE     GMPRD           ;BR IF INHIBIT PARITY.
004116  012703  002076                  MOV     #MPR0,  R3      ;SET UP TABLE POINTER
004122  012737  004144  000004          MOV     #GMPRB, @#ERRVEC ;SET UP TIMEOUT TRAP SERVICE
004130  042713  000001          GMPRA:  BIC     #1,     (R3)    ;CLEAR FLAG BIT IN TABLE
004134  005773  000000                  TST     @(R3)           ;DOES THIS MEMORY PARITY REGISTER EXIST.
                                ;*  IF IT DOESN'T EXIST, A TIMEOUT TRAP WILL GO TO 'GMPRB'.
004140  012324                          MOV     (R3)+,  (R4)+   ;SAVE IT IN THE PARITY REG EXIST TABLE.
004142  000403                          BR      GMPRC           ;SKIP TIMEOUT SERVICE CODE
                                ;*  TIMEOUT COMES HERE
004144  022626          GMPRB:  CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
004146  052723  000001                  BIS     #1,     (R3)+   ;SET FLAG TO INDICATE REGISTER NOT PRESENT
004152  005023          GMPRC:  CLR     (R3)+           ;CLEAR THE MAP...LO 64K.
004154  005023                  CLR     (R3)+           ;...HI 64K.

004156  005023                  CLR     (R3)+           ;...AND THE MASK.
004160  020327  002276          CMP     R3,     #MPRX   ;HAVE WE CHECKED ALL REGISTERS?

004164  103761                  BLO     GMPRA           ;NO - GO BACK TO CHECK NEXT ONE
004166  005014                  CLR     (R4)            ;SET TERMINATOR IN PARITY REG EXIST TABLE.
004170  012737  025124  000004  MOV     #ERRTRP,@#ERRVEC ; RESTORE TRAPCATCHER
004176  005767  176074          TST     MPRX                    ;ANY PARITY REGISTERS PRESENT?
004202  001006                  BNE     MPAMEM                  ;YES - GO TEST CONTROLS PRESENT
004204  004567  017312          JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)    004210  025615          .WORD   MTR             ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                    ;'NO MEMORY PARITY REGISTERS FOUND'
004212  005014          GMPRD:  CLR     (R4)            ;MAKE SURE TABLE IS CLEAR.
004214  000167  001156          JMP     MANUAL                  ;AND SKIP ALL CONTROLS TESTING
```

```
 6824                                   .SBTTL  MAP PARITY MEMORY
 6825                                   ;;*****************************************************************
 6826                                   ;MAP CORRESPONDENCE BETWEEN PARITY REGISTERS AND MEMORY, AND TYPE RESULTS
 6827                                   ;NOTE THAT IF PARITY MEMORY IS NOT LOCATED CORRECTLY THAT IT IS IN ALL
 6828                                   ;PROBABILITY DUE TO ONE OF THE FOLLOWING FAILURES:
 6829                                   ;       - SETTING WRITE WRONG PARITY DIDN'T CAUSE BAD PARITY TO BE WRITTEN
 6830                                   ;       - PARITY GENERATE OR DETECT LOGIC FAILED
 6831                                   ;       - PARITY ERROR BIT FAILED TO SET
 6832                                   ;       - PARITY BITS IN MEMORY LOCATION FAILED
 6833                                   ;       - I.E. BIT STUCK AT GOOD PARITY VALUE
 6834                                   ;;*****************************************************************
 6835
 6836  004220  004767  014054          MPAMEM: JSR     PC,     CLRPAR  ;INITIALIZE ALL PARITY REGISTERS
 6837  004224  012767  000001  175312          MOV     #1,     BITPT   ;INITIALIZE 4K POINTER
 6838  004232  005067  175310                  CLR     BITPT+2         ;CLEAR HI 64K POINTER
 6839  004236  012702  014000                  MOV     #14000, R2      ;SET ADR POINTER TO 14000.
 6840  004242  005767  174340                  TST     MMAVA           ;CHECK FOR MEM MGMT
 6841  004246  001404                          BEQ     MAPRB           ;BRANCH IF NO MEM MGMT
 6842  004250  012702  054000                  MOV     #54000, R2      ;SET ADR POINTER TO PAR2
 6843  004254  004767  010030                  JSR     PC,     MMINIT  ;SET UP ALL MEMORY MGMT REGISTERS.
 6844
 6845                                   ;;*****************************************************************
 6846                                   ;SET WRITE WRONG PARITY IN ALL REGISTERS PRESENT
 6847                                   ;*  THEN WRITE TEST LOCATION VIA DATO & READ TEST LOCATION VIA DATI
 6848                                   ;*  THEN CLEAR WRITE WRONG PARITY IN ALL REGISTERS.
 6849                                   ;;*****************************************************************
 6850
 6851  004260  005067  175254          MAPRB:  CLR     PMEMAP          ;CLEAR THE PARITY MEMORY MAP
 6852  004264  005067  175252                  CLR     PMEMAP+2
 6853  004270  012703  002076          1$:     MOV     #MPR0,  R3      ;INITIALIZE TABLE ADDRESS
 6854  004274  032713  000001          2$:     BIT     #1,     (R3)    ;IS THIS REGISTER PRESENT?
 6855  004300  001052                          BNE     3$              ;NO - GET THE NEXT ONE
 6856  004302  013773  001612  000000          MOV     @#WWP,  @(R3)   ;YES - SET WRITE WRONG PARITY
 6857                                                                  ;AND CLEAR REST OF REGISTER
 6858  004310  011212                          MOV     (R2),   (R2)    ;WRITE WRONG PARITY
 6859  004312  005712                          TST     (R2)            ;READ WRONG PARITY
 6860  004314  043773  001612  000000          BIC     @#WWP,  @(R3)   ;CLEAR WRITE WRONG PARITY
 6861  004322  005773  000000                  TST     @(R3)           ;OTHERWISE, CHECK TO SEE IF THIS
 6862                                                                  ;CONTROL REGISTER GOT A PARITY
 6863                                                                  ;ERROR
 6864  004326  100014                          BPL     6$              ;BRANCH IF IT DIDN'T AND CHECK
 6865  004330  032773  007740  000000          BIT     #7740,  @(R3)   ;IS IT A CORE PAR. REG.
 6866  004336  001404                          BEQ     5$              ;BRANCH IF NOT.
 6867  004340  012763  070032  000006          MOV     #70032, 6(R3)   ;IF IT IS SET UP MASK
 6868  004346  000413                          BR      7$              ;AND BRANCH TO SET BITS.
 6869  004350  012763  077772  000006  5$:     MOV     #77772, 6(R3)   ;IF MOS SET UP MASK
 6870  004356  000407                          BR      7$              ;AND BRANCH TO SET BIT.
 6871  004360  032773  007740  000000  6$:     BIT     #7740,  @(R3)   ;IF ANY BITS ARE SET
 6872  004366  001417                          BEQ     3$              ;THEN CSR IS MS11-K.
 6873  004370  012763  070000  000006          MOV     #70000, 6(R3)   ;IF MS11-K SET MASK.
 6874  004376  056763  175142  000002  7$:     BIS     BITPT,  2(R3)   ;SET FLAG IN MAP FOR THIS PARITY REGISTER
 6875  004404  056763  175136  000004          BIS     BITPT+2,4(R3)
 6876  004412  056767  175126  175120          BIS     BITPT,  PMEMAP  ;SET FLAG IN PARITY MAP
 6877  004420  056767  175122  175114          BIS     BITPT+2,PMEMAP+2
 6878  004426  062703  000010          3$:     ADD     #10,    R3      ;STEP UP TO NEXT REGISTER
 6879  004432  020327  002276                  CMP     R3,     #MPRX   ;ARE WE DONE WITH TABLE?
```

```
6880  004436  103716                          BLO     2$              ;GO BACK TO CHECK FOR ANY MORE
6881  004440  011212                          MOV     (R2),   (R2)    ;CLEAR BAD PARITY
6882  004442  005767  174140                  TST     MMAVA           ;CHECK FOR MEM MGMT
6883  004446  001444                          BEQ     10$             ;BR IF NO MEM MGMT
6884  004450  062737  000200  172344  4$:     ADD     #200,   @#KIPAR2 ;UPDATE PAR TO NEXT 4K BANK.
6885  004456  006367  175062                  ASL     BITPT           ;UPDATE BANK POINTER...LO 64K.
6886  004462  006167  175060                  ROL     BITPT+2         ;...HI 64K.
6887  004466  100441                          BMI     TMAP            ;BR IF ALL DONE.
6888  004470  023727  172344  001000          CMP     @#KIPAR2,#1000  ;THIS CODE TESTS IF MS11-K IS
6889  004476  001013                          BNE     12$             ;PRESENT AND IF IT IS I SET
6890  004500  032737  000003  002260          BIT     #3,@#MPR14+2    ;THE BIT TO DISABLE ECC IN
6891  004506  001004                          BNE     13$             ;THE LOCATION WHP THAT IS
6892  004510  032737  000003  002270          BIT     #3,@#MPR15+2    ;USED AS THE COMMAND TO
6893  004516  001400                          BEQ     13$             ;WRITE WRONG PARITY.
6894  004520  012737  020004  001612  13$:     MOV     #20004,@#WWP
6895  004526  036767  175012  174770  12$:     BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
6896  004534  001255                          BNE     1$              ;BR IF BANK EXISTS.
6897  004536  036767  175004  174762          BIT     BITPT+2,MEMMAP+2 ;...HI 64K.
6898  004544  001251                          BNE     1$              ;BR IF BANK EXISTS.
6899  004546  000740                          BR      4$              ;BR IF BANK DOESN'T EXIST.
6900  004550  036767  174770  174746  11$:     BIT     BITPT,  MEMMAP  ;CHECK IF BANK EXISTS.
6901  004556  001244                          BNE     1$              ;BR IF BANK EXISTS.
6902  004560  062702  020000          10$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
6903  004564  106367  174754                  ASLB    BITPT           ;MOVE POINTER TO NEXT BANK.
6904  004570  100367                          BPL     11$             ;BR IF MORE TO LOOK FOR.
6905
6906                                  ;;*****************************************************************
6907                                  ;*  ROUTINE TO TYPE MAP OF WHERE PARITY MEMORY IS PRESENT
6908                                  ;*  AND WHICH CONTROL REGISTERS CONTROL WHICH MEMORY
6909                                  ;;*****************************************************************
6910
6911  004572  004767  013502          TMAP:   JSR     PC,     CLPPAR  ;INITIALIZE ALL PARITY REGISTERS PRESENT
6912  004576  004567  016720                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  004602  025472                          .WORD   MTMAP           ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                                  ;'PARITY MEMORY MAP:''
6913  004604  012703  002076                  MOV     #MPR0,  R3      ;INITIALIZE TABLE POINTER
6914  004610  032713  000001          1$:     BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER IS PRESENT.
6915  004614  001046                          BNE     2$              ;BR IF NOT PRESENT.
6916  004616  022763  070032  000006          CMP     #70032, 6(R3)
6917  004624  001004                          BNE     3$
6918  004626  004567  016670                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  004632  026133                          .WORD   MX3             ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                                  ;''CORE PARITY''
6919  004634  000417                          BR      5$
6920  004636  022763  077772  000006  3$:     CMP     #77772, 6(R3)
6921  004644  001004                          BNE     4$
6922  004646  004567  016650                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  004652  026152                          .WORD   MX4             ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                                  ;'MOS PARITY''
6923  004654  000407                          BR      5$
6924  004656  022763  070000  000006  4$:     CMP     #70000, 6(R3)
6925  004664  001003                          BNE     5$
6926  004666  004567  016630                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  004672  026170                          .WORD   MX5             ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                                  ;'MS11-K CSR''
6927  004674                          5$:
```

```
  (2) 004674  004567  016622              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2) 004700  026101                      .WORD   MX1             ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'REGISTER AT''
 6928 004702  011346                      MOV     (R3),-(SP)      ;;SAVE (R3) FOR TYPEOUT
  (3)                                 ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
  (3)                                 ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
  (3) 004704  013746  177776              MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
  (3) 004710  004767  020004              JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
 6929 004714  004567  016602              JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2) 004720  026120                      .WORD   MX2             ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                             ;'CONTROLS''
 6930 004722  010300                      MOV     R3,     R0      ;SET UP R0 FOR TYPMAP ROUTINE.
 6931 004724  005720                      TST     (R0)+           ;UPDATE POINTER TO MAP.
 6932 004726  004767  013470              JSR     PC,     TYPMAP  ;GO TYPE THE MEMORY COVERED BY THIS REGISTER.
 6933 004732  062703  000010      2$:     ADD     #10,    R3      ;UPDATE TO NEXT REGISTER IN TABLE.
 6934 004736  020327  002276              CMP     R3,     #MPRX   ;ARE WE ALL DONE WITH TABLE?
 6935 004742  103722                      BLO     1$              ;BRANCH IF MORE REGISTERS
 6936 004744  004567  016552              JSR     R5,     $PRINT  ;THE REASON I'M OUTPUTTING THIS CRLF
 6937 004750  001201                      $CRLF                   ;IS TO GIVE THE PRINTER ENOUGH TIME TO
 6938                                                             ;FINISH PRINTING THE MEMORY MAP BEFORE THE RESET OCCURS.
 6939 004752  022737  070000  002264      CMP     #70000,@#MPR14+6 ;DO WE HAVE MS11-K AT THIS ADDRESS
 6940 004760  001006                      BNE     7$              ;IF NO BRANCH
 6941 004762  043727  002260  001540      BIC     @#MPR14+2,#PMEMAP ;IF YES THEN CLEAR THE BITS IN
 6942 004770  043737  002262  001540      BIC     @#MPR14+4,@#PMEMAP ;THE PARITY MEMORY MAP.
 6943 004776  022737  070000  002274  7$: CMP     #70000, @#MPR15+6       ;DO WE HAVE A MS11-K
 6944 005004  001031                      BNE     9$              ;IF NO GO TO TESTS NOW.
 6945 005006  043737  002270  001540      BIC     @#MPR15+2,@#PMEMAP      ;IF YES I AM GOING TO
 6946 005014  043737  002272  001542      BIC     @#MPR15+4,@#PMEMAP+2    ;CLEAR THE PARITY INDICATORS
 6947 005022  012705  002276              MOV     #MPRX,  R5      ;FOR THAT PORTION OF MEMORY.
 6948 005026  021537  002256      6$:     CMP     (R5),@#MPR14    ;SEARCH FOR THIS MS11-K  CSR IN
 6949 005032  001004                      BNE     8$              ;AND IF ITS THERE DELETE IT
 6950 005034  005015                      CLR     (R5)
 6951 005036  052737  000001  002256      BIS     #1,@#MPR14
 6952 005044  022537  002266      8$:     CMP     (R5)+,  @#MPR15 ;SEARCH FOR MS11-K CSR IN
 6953 005050  001366                      BNE     6$              ;THE AVAILABILITY TABLE,
 6954 005052  005045                      CLR     -(R5)           ;AND CLEAR ITS ADDRESS FROM THE TABLE
 6955 005054  052737  000001  002266      BIS     #1,     @#MPR15 ;SET BIT0 IN ADDRESS IN CSR TABLE
 6956 005062  004567  016434              JSR     R5,     $PRINT  ;OUTPUT MESSAGE TO RUN MS11-K TEST.
 6957 005066  026206                      .WORD   MX6
 6958 005070  005737  002276      9$:     TST     @#MPRX          ;ARE THERE ANY PARITY REGISTERS TO TEST?
 6959 005074  001002                      BNE     CTRLS           ;IF SO TEST THE BITS IN THE REGISTERS,
 6960 005076  000167  000274              JMP     MANUAL          ;IF NO JUMP OVER REGISTER TESTS.
 6961
 6962                                 .SBTTL  TEST PARITY REGISTERS
 6963                                 ;;*****************************************************************
 6964                                 ;* SHOW THAT BITS 0, 2, 5 - 11, AND 15 OF EACH PARITY REGISTER PRESENT
 6965                                 ;*  CAN BE SET AND CLEARED.
 6966                                 ;*  THIS IS A ONCE ONLY TEST.
 6967                                 ;;*****************************************************************
 6968
 6969 005102  012703  002076      CTRLS:  MOV     #MPR0,  R3      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
 6970 005106  011302              1$:     MOV     (R3),   R2      ;LOAD R2 WITH ADDRESS OF THIS PARITY REGISTER
 6971 005110  062703  000010              ADD     #10,    R3      ;UPDATE POINTER TO NEXT PAR. REG. ADD.
 6972 005114  032702  000001              BIT     #1,     R2      ;IS THIS REGISTER BEING USED?
 6973 005120  001372                      BNE     1$              ;GO TO NEXT IF NOT
 6974 005122  020327  002276              CMP     R3,     #MPRX   ;ARE WE AT END OF TABLE
```

```
 6975  005126  003055                        BGT     RESCHK          ;GO TO NEXT TEST IF YES
 6976  005130  005763  177776                TST     -2(R3)          ;TEST MASK FOR PARITY REGISTER
 6977  005134  001764                        BEQ     1$              ;IF = 0,THEN DO NOT TEST
 6978  005136  016367  177776  174352        MOV     -2(R3), RESRVD  ;GET MASK FOR REGISTER WE ARE WORKING ON
 6979  005144  012700  000001                MOV     #1,     R0      ;LOAD R0 WITH VALUE OF 1ST BIT TESTED
 6980  005150  005012                        CLR     (R2)            ;INITIALIZE THE PARITY REGISTER
 6981  005152  011201                        MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
 6982  005154  046701  174336                BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
 6983  005160  001405                        BEQ     2$              ;CHECK OTHER BITS - BRANCH IF OK
 6984  005162  004767  013134        64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
  (2)  005166  004767  014456                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  005172  000001                        .WORD   1               ;ERROR TYPE CODE.
 6985  005174  030067  174316        2$:     BIT     R0,     RESRVD  ;IS THIS BIT RESERVED?
 6986  005200  001025                        BNE     3$              ;YES - DON'T TEST IT
 6987  005202  010012                        MOV     R0,     (R2)    ;NO - SET THIS BIT IN THE PARITY REGISTER
 6988  005204  011201                        MOV     (R2),   R1      ;READ & SAVE CONTENTS OF THE  PARITY REGISTER
 6989  005206  005012                        CLR     (R2)            ;CLEAR THE PARITY REGISTER
 6990  005210  046701  174302                BIC     RESRVD, R1      ;CLEAR BIT LOCATIONS THAT ARE RESERVED
 6991  005214  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  005216  001405                        BEQ     66$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  005220  004767  013126        65$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  005224  004767  014420                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  005230  000001                        .WORD   1               ;ERROR TYPE CODE.
  (2)  005232                        66$:
 6992  005232  011201                        MOV     (R2),   R1      ;READ THE CONTENTS OF THE PARITY REGISTER
 6993  005234  046701  174256                BIC     RESRVD, R1      ;CLEAR BITS WHICH ARE RESERVED
 6994  005240  001405                        BEQ     3$              ;CHECK OTHER BITS - BRANCH IF OK
 6995  005242  004767  013054        67$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
  (2)  005246  004767  014376                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  005252  000001                        .WORD   1               ;ERROR TYPE CODE.
 6996  005254  006300                3$:     ASL     R0              ;ROTATE TO GET NEXT BIT TO BE TESTED
 6997  005256  103346                        BCC     2$              ;BRANCH IF NOT DONE WITH ALL BITS
 6998  005260  000712                        BR      1$              ;AFTER TESTING FOR BIT 15 GO GET NEXT REGISTER.
 6999
 7000                                ;;*********************************************************************
 7001                                ;* SHOW THAT RESET CLEARS BITS 0,2, AND 15 OF EACH PARITY REGISTER PRESENT.
 7002                                ;*  THIS IS A ONCE ONLY TEST.
 7003                                ;;*********************************************************************
 7004
 7005  005262  012704  002076        RESCHK: MOV     #MPR0,  R4      ;LOAD INITIAL TABLE ADDRESS FOR A POINTER
 7006  005266  010403                1$:     MOV     R4,     R3
 7007  005270  062704  000010                ADD     #10,    R4
 7008  005274  032713  000001        BIT     #1,     (R3)    ;IS THIS REGISTER BEING USED
 7009  005300  001372                        BNE     1$              ;BRANCH IF NO
 7010  005302  012773  177777  000000        MOV     #-1,    @(R3)   ;SET ALL BITS TO A 1
 7011  005310  022704  002276                CMP     #MPRX,R4        ;ARE WE AT THE END OF THE TABLE
 7012  005314  002764                        BLT     1$              ;IF YES THEN WE ARE READY TO TEST
 7013  005316  000005                        RESET                   ;RESET THE WORLD
 7014  005320  012703  002076                MOV     #MPR0,  R3      ;LOAD INITIAL ADDRESS FOR POINTER
 7015  005324  011302                2$:     MOV     (R3),   R2      ;STORE PARITY REGISTER ADDRESS
 7016  005326  062703  000010                ADD     #10,    R3
 7017  005332  032702  000001                BIT     #1,     R2
 7018  005336  001372                        BNE     2$
 7019  005340  022703  002276                CMP     #MPRX,  R3
 7020  005344  002014                        BGE     MANUAL
 7021  005346  011201                        MOV     (R2),   R1      ;GET CONTENTS OF REGISTER
```

```
7022  005350  005012                        CLR     (R2)
7023  005352  042701  077772                BIC     #77772, R1      ;CLEAR BITS NOT EFFECTED  BY RESET
7024  005356  005701                        TST     R1              ;CHECK IF REST  WERE CLEARED BY  RESET
7025  005360  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (2) 005362  004767  012734          64$:  JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
  (3) 005366  004767  014256                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (3) 005372  000001                        .WORD   1               ;ERROR TYPE CODE.
  (1) 005374                          65$:
7026  005374  000753                        BR      2$              ;BRANCH BACK TO CHECK NEXT REGISTER
7027
7028
7029  005376  012700  000014          MANUAL: MOV     #12.,   R0      ;SET COUNTER TO CLEAR 12 WORDS.
7030  005402  012701  001562                MOV     #FSTADR,R1      ;STARTING AT FSTADR.
7031  005406  005021                  1$:   CLR     (R1)+           ;CLEAR THE LOCATIONS.
7032  005410  005300                        DEC     R0              ;COUNT.
7033  005412  001375                        BNE     1$              ;BR IF MORE.
7034  005414  105767  174136                TSTB    SELFLG          ;CHECK FOR SELECT PARAMETERS STARTUP.
7035  005420  001005                        BNE     MANUL1          ;BR IF PARAMETERS TO BE SELECTED
7036  005422  016767  173536  174144        MOV     $TMP2,  LSTADR  ;SET UP VIRTUAL LAST ADDRESS.
7037  005430  000167  000402                JMP     MANUL2          ;SKIP PARAMETER SELECTION SECTION.
```

```
                                        .SBTTL  USER PARAMETER SELECTION SECTION
                                        ;;*****************************************************************
                                        ;* USER PARAMETER SELECTION SECTION IS ENTERED BY STARTING AT 204.
                                        ;;*****************************************************************
005434  012700  000001          MANUL1: MOV     #BIT0,  R0      ;SET UP BANK POINTER.
005440  005001                          CLR     R1              ;...HI 64K.
005442  005002                          CLR     R2              ;CLEAR ADDRESS POINTER.
005444  005003                          CLR     R3              ;...HI ADDRESS BITS.
005446  004567  016050                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005452  026321                          .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**.
005454  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005460  004767  015664                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005464  042716  000001                  BIC     #BIT0,  (SP)    ;MAKE SURE ADDRESS IS ON A WORD BOUNDRY.
005470  005067  174040                  CLR     SAVTST          ;INIT TEST MAP...LO 64K.
005474  005067  174036                  CLR     SAVTST+2        ;...HI 64K.
005500  062702  020000          1$:     ADD     #20000, R2      ;UPDATE ADDRESS POINTER TO NEXT BANK.
005504  005503                          ADC     R3
005506  020367  016006                  CMP     R3,     $HIOCT  ;CHECK HI ADDRESS BITS.
005512  103403                          BLO     2$              ;BR IF NOT HI ENOUGH YET.
005514  101006                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005516  020216                          CMP     R2,     (SP)    ;CHECK THE LO ADDRESS BITS.
005520  101004                          BHI     3$              ;BR IF PAST SELECTED ADDRESS.
005522  006300                  2$:     ASL     R0              ;UPDATE POINTER...LO 64K.
005524  006101                          ROL     R1              ;...HI 64K.
005526  100364                          BPL     1$              ;BR BACK TO CHECK NEXT BANK.
005530  000507                          BR      17$             ;BR IF OVERFLOW.
005532  030067  173766          3$:     BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS.
005536  001003                          BNE     4$              ;BR IF BANK EXISTS.
005540  030167  173762                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K.
005544  001501                          BEQ     17$             ;BR IF ADDRESS IN UN-MAPPED BANK.
005546  016704  015746          4$:     MOV     $HIOCT, R4      ;SAVE FIRST ADR HI BITS.
005552                          10$:
005552  004567  015744                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
005556  026406                          .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**.
005560  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
005564  004767  015560                  JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
005570  005716                          TST     (SP)            ;CHECK IF ADR 0 SELECTED (DEFAULT).
005572  001010                          BNE     11$             ;BR IF NOT 0 (DEFAULT)
005574  005767  015720                  TST     $HIOCT          ;CHECK HI BITS.
005600  001005                          BNE     11$             ;BR IF NOT 0 (DEFAULT).
005602  016716  173356                  MOV     $TMP2,  (SP)    ;SET UP DEFAULT LAST ADR.
005606  016767  173354  015704          MOV     $TMP3,  $HIOCT
005614  012667  173754          11$:    MOV     (SP)+,  LSTADR  ;GET THE DATA.
005620  020467  015674                  CMP     R4,     $HIOCT  ;CHECK FOR LAST ADR BELOW FIRST ADR.
005624  101352                          BHI     10$             ;BB IF LAST BELOW FIRST.
005626  103403                          BLO     12$             ;BR IF LAST ABOVE FIRST.
005630  021667  173740                  CMP     (SP),   LSTADR  ;CHECK FOR LAST BELOW FIRST.
005634  101346                          BHI     10$             ;BR IF LAST BELOW FIRST.
005636  032716  017777          12$:    BIT     #MASK4K,(SP)    ;CHECK IF FIRST ADR ON BANK BOUNDRY.
005642  001404                          BEQ     13$             ;BR IF ON BOUNDRY.
```

```
7084  005644  010067  173720            MOV     R0,     FADMAP  ;SET UP FIRST ADDRESS MAP.
7085  005650  010167  173716            MOV     R1,     FADMAP+2
7086  005654  050067  173654    13$:    BIS     R0,     SAVTST  ;SET FLAG IN TEST MAP...LO 64K.
7087  005660  050167  173652            BIS     R1,     SAVTST+2 ;...HI 64K.
7088  005664  020367  015630    14$:    CMP     R3,     $HIOCT  ;CHECK FOR PAST LAST ADR.
7089  005670  103404                    BLO     15$             ;BR IF BELOW LAST ADR.
7090  005672  101020                    BHI     16$             ;BR IF GONE PAST LAST ADR.
7091  005674  020267  173674            CMP     R2,     LSTADR  ;CHECK FOR PAST LAST ADR.
7092  005700  101015                    BHI     16$             ;BR IF GONE PAST LAST ADR.
7093  005702  062702  020000    15$:    ADD     #20000, R2      ;UPDATE ADDRESS POINTER.
7094  005706  005503                    ADC     R3              ;...HI BITS.
7095  005710  006300                    ASL     R0              ;UPDATE BANK POINTER...LO 64K.
7096  005712  006101                    ROL     R1              ;...HI 64K.
7097  005714  100415                    BMI     17$             ;BR IF OVERFLOW.
7098  005716  030067  173602            BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
7099  005722  001354                    BNE     13$             ;BR IF BANK EXISTS.
7100  005724  030167  173576            BIT     R1,     MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
7101  005730  001351                    BNE     13$             ;BR IF BANK EXISTS.
7102  005732  000754                    BR      14$             ;BR IF BANK DOESN'T EXIST.
7103  005734  030067  173564    16$:    BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
7104  005740  001010                    BNE     20$             ;BR IF IT EXISTS.
7105  005742  030167  173560            BIT     R1,     MEMMAP+2 ;CHECK IF THIS BANK EXISTS.
7106  005746  001005                    BNE     20$             ;BR IF IT EXISTS.
7107  005750  005726            17$:    TST     (SP)+           ;ADJUST THE STACK.
7108  005752  004567  015544            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  005756  026431                    .WORD   BADADR          ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                            ;''?ADDRESS IN UNMAPPED BANK?''
7109  005760  000606                    BR      MANUAL          ;LOOP BACK TO THE BEGINNING.
7110  005762  010067  173614    20$:    MOV     R0,     LADMAP  ;SET UP MAP FOR LAST ADDRESS.
7111  005766  010167  173612            MOV     R1,     LADMAP+2
7112  005772  005767  172610    21$:    TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
7113  005776  001404                    BEQ     22$             ;BR IF NO MEM MGMT.
7114  006000  042716  160000            BIC     #160000,(SP)    ;ADJUST FSTADR TO VITRUAL BANK 0.
7115  006004  062716  040000            ADD     #40000, (SP)    ;...TO VIRTUAL BANK 2.
7116  006010  012667  173546    22$:    MOV     (SP)+,  FSTADR  ;SAVE FISRT ADDRESS OFF THE STACK.
7117  006014                    30$:
 (2)  006014  004567  015502            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  006020  026466                    .WORD   CONST           ;ADDRESS OF MESSAGE TO BE TYPED
 (1)                                                            ;''SELECT CONSTANT: ''
7118                                ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDOCT ROUTINE
 (2)                                ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 (2)  006022  013746  177776            MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 (2)  006026  004767  015316            JSR     PC,     $RDOCT  ;GO TO THE SUBROUTINE
7119  006032  012667  173552            MOV     (SP)+,  .CONST  ;SAVE THE CONSTANT
7120  006036  005767  172544    MANUL2: TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE.
7121  006042  001406                    BEQ     31$             ;BR IF NO MEM MGMT.
7122  006044  042767  160000  173522    BIC     #160000,LSTADR  ;ADJUST LSTADR TO VIRTUAL BANK 0.
7123  006052  062767  040000  173514    ADD     #40000, LSTADR  ;...VIRTUAL BANK 2.
7124  006060  062767  000002  173506  31$:    ADD     #2,     LSTADR  ;ADJUST LAST ADDRESS UP ONE WORD.
7125  006066  042767  000001  173500    BIC     #BIT0,  LSTADR  ;MAKE SURE IT IS A WORD ADDRESS.
7126  006074  032767  017777  173472    BIT     #MASK4K,LSTADR  ;CHECK IF LAST ADR IS ON BANK BOUNDRY.
7127  006102  001004                    BNE     START1          ;BR IF NOT ON BOUNDRY.
7128  006104  005067  173472            CLR     LADMAP          ;CLEAR OUT THE LAST ADDRESS MAP.
7129  006110  005067  173470            CLR     LADMAP+2
7130
```

```
                ;/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:/*\:
                ;*  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
                ;/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*/:\*

7135  006114  016706  173502          START1: MOV     .STACK, SP      ;SET STACK POINTER
7136  006120  005767  173472                  TST     CASFLG          ;CHECK CACHE PRESENT FLAG
7137  006124  001403                          BEQ     1$              ;BRANCH IF NO CACHE
7138  006126  052777  000014  173464          BIS     #14,    @CASREG ;TURN OFF CACHE
7139  006134  012767  006114  172744  1$:     MOV     #START1,$LPADR  ;INIT LOOP ADDRESS.
7140  006142  066767  172432  172736          ADD     RELOCF, $LPADR
7141  006150  004767  011372                  JSR     PC,     MAMF    ;SET UP MEMORY PARITY ERROR VECTOR
7142  006154  005767  172426                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT AVAILABLE.
7143  006160  001406                          BEQ     TST1            ;BRANCH IF NO MEM MGMT.
7144  006162  032737  000001  177572          BIT     #BIT0,  @#SR0   ;CHECK IF MEM MGMT ENABLED.
7145  006170  001002                          BNE     TST1            ;BR IF MEM MGMT ENABLED.
7146  006172  004767  006112                  JSR     PC,     MMINIT  ;SET UP MEM MGMT REGISTERS.
7147
7148                                  .SBTTL SECTION 1:       MEMORY ADDRESS TESTS
7157                                  ;;*****************************************************************
 (3)                                  ;*TEST 1        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
 (4)                                  ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
 (4)                                  ;*      R1 = DATA READ FROM MEMORY (WAS)
 (4)                                  ;*      R2 = VIRTUAL ADDRESS
 (4)                                  ;*      R3 = NOT USED
 (4)                                  ;*      R4 = NOT USED
 (4)                                  ;*      R5 = BLOCK BOUNDRY BIT MASK.
 (3)                                  ;;*****************************************************************
 (2)  006176                          TST1:
 (3)  006176  004567  012436                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)  006202  000001                          .WORD   1               ;MINIMUM BLOCK SIZE OF 1 WORDS
 (3)                                                                  ; REQUIRED FOR THIS TEST.
 (3)  006204  000167  005604                  JMP     TST32           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
 (3)                                                                  ; AVAILABLE FOR TEST.
7158                                  ;*  UPWARDS WORD ADDRESSING.
7159  006210  004467  006222                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7160  006214  004767  007644          1$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
7161  006220  010012                  2$:     MOV     R0,     (R2)    ;WRITE VALUE OF ADDRESS INTO ADDRESS
7162  006222  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  006224  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  006226  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  006230  004767  012142          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  006234  004767  013410                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  006240  000002                          .WORD   2               ;ERROR TYPE CODE.
 (3)  006242                          65$:
7163  006242  062700  000002                  ADD     #2,     R0      ;ADD #2 TO PHYSICAL ADDRESS
7164  006246  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)  006250  001363                          BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)  006252  004767  006736                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
7165
7166                                  ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
7167                                  ;*  DOWNWARDS WORD ADDRESSING.
7168  006256  004467  006612                  JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7169  006262  004767  007576          3$:     JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
7170  006266  162700  000002          4$:     SUB     #2,     R0      ;DEC DATA BY 2
7171  006272  014201                          MOV     -(R2),  R1      ;GET THE DATA FROM MEMORY
```

```
 7172  006274  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  006276  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  006300  004767  012046          66$:  JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  006304  004767  013340                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  006310  000002                        .WORD   2               ;ERROR TYPE CODE.
  (2)  006312                          67$:
 7173  006312  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  006314  001364                        BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  006316  004767  007362                JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
```

```
 7175                                   ;;*****************************************************************
  (3)                                   ;*TEST 2        WRITE VALUE OF MEMORY ADDRESS INTO MEMORY
  (4)                                   ;*       R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
  (4)                                   ;*       R1 = DATA READ FROM MEMORY (WAS)
  (4)                                   ;*       R2 = VIRTUAL ADDRESS
  (4)                                   ;*       R3 = NOT USED
  (4)                                   ;*       R4 = NOT USED
  (4)                                   ;*       R5 = BLOCK BOUNDRY BIT MASK.
  (3)                                   ;;*****************************************************************
  (2) 006322                            TST2:
  (3) 006322  004567  012312                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3) 006326  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7176                                   ;*  UPWARDS BYTE ADDRESSING.
 7177 006330  004467  006102                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7178 006334  004767  007524            1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
 7179 006340  110022                    2$:    MOVB    R0,     (R2)+   ;WRITE VALUE OF ADDRESS INTO ADDRESS
 7180 006342  005200                            INC     R0              ;ADD ONE TO PHYSICAL ADDRESS
 7181 006344  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1) 006346  001374                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1) 006350  004767  006640                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 7182
 7183                                   ;*  CHECK THAT VALUE OF MEMORY ADDRESS WAS WRITTEN CORRECTLY
 7184                                   ;*  DOWNWARDS BYTE ADDRESSING.
 7185 006354  004467  006514                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7186 006360  004767  007500            3$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
 7187 006364  005300                    4$:    DEC     R0              ;DEC DATA BY 1
 7188 006366  114201                            MOVB    -(R2),  R1      ;GET THE DATA FROM MEMORY
 7189 006370  120001                            CMPB    R0,     R1      ;CHECK THE DATA...LO BYTE ONLY VALID.
 7190 006372  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (2) 006374  004767  011752            64$:   JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
  (3) 006400  004767  013244                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (3) 006404  000003                            .WORD   3               ;ERROR TYPE CODE.
  (1) 006406                            65$:
 7191 006406  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1) 006410  001365                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1) 006412  004767  007266                    JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
 7192
 7193                                   ;;*****************************************************************
  (3)                                   ;*TEST 3        WRITE 1'S COMPLEMENT VALUE OF ADDRESS INTO ADDRESS.
  (4)                                   ;*       R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
  (4)                                   ;*       R1 = DATA READ FROM MEMORY (WAS)
  (4)                                   ;*       R2 = VIRTUAL ADDRESS
  (4)                                   ;*       R3 = NOT USED
  (4)                                   ;*       R4 = NOT USED
  (4)                                   ;*       R5 = BLOCK BOUNDRY BIT MASK.
  (3)                                   ;;*****************************************************************
  (2) 006416                            TST3:
  (3) 006416  004567  012216                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3) 006422  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7194                                   ;*  DOWNWARDS WORD ADDRESSING.
 7195 006424  004467  006444                   JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7196 006430  004767  007430            1$:    JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
 7197 006434  005100                            COM     R0              ;COMPLEMENT THE ADR
 7198 006436  062700  000002            2$:    ADD     #2,     R0      ;+2 TO DATA--ADR GOES DOWN SO COM GOES UP
 7199 006442  010042                            MOV     R0,     -(R2)   ;PUT DATA INTO MEMORY
 7200 006444  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
```

```
  (1)  006446  001373                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  006450  004767  007230                JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
7201
7202                                ;*  CHECK COMPLEMENT DATA WRITTEN DOWN
7203                                ;*  UPWARDS WORD ADDRESSING.
7204  006454  004467  005756                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7205  006460  004767  007400          3$:   JSR     PC,     PHYADR  ;GET PHYSICAL ADDRESS INTO R0
7206  006464  005100                        COM     R0              ;COMPLEMENT IT
7207  006466                            4$:
  (1)  006466  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  006470  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  006472  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  006474  004767  011676          64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  006500  004767  013144                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  006504  000002                        .WORD   2               ;ERROR TYPE CODE.
  (3)  006506                            65$:
7208  006506  162700  000002                SUB     #2,     R0      ;COUNT DOWN WITH ADDRESS
7209  006512  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  006514  001364                        BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  006516  004767  006472                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
7210
7211                                ;;*****************************************************************
  (3)                               ;*TEST 4        WRITE BANK # INTO ALL ADDRESSES IN A 4K BANK
  (4)                               ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
  (4)                               ;*      R1 = DATA READ FROM MEMORY (WAS)
  (4)                               ;*      R2 = VIRTUAL ADDRESS
  (4)                               ;*      R3 = NOT USED
  (4)                               ;*      R4 = NOT USED
  (4)                               ;*      R5 = BLOCK BOUNDRY BIT MASK.
  (3)                               ;;*****************************************************************
  (2)  006522                       TST4:
  (3)  006522  004567  012112                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  006526  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
7212                                ;*  UPWARDS BYTE ADDRESSING.
7213  006530  004467  005702                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7214  006534  004767  007400          1$:   JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
7215  006540  110022                    2$:   MOVB    R0,     (R2)+   ;WRITE BANK # INTO ALL ADDRESSES
7216  006542  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  006544  001375                        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  006546  004767  006442                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
7217
7218                                ;*  CHECK THAT DATA WRITTEN ABOVE CAN BE READ
7219                                ;*  UPWARDS BYTE ADDRESSING.
7220  006552  004467  005660                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7221  006556  004767  007356          3$:   JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
7222  006562  112201                    4$:   MOVB    (R2)+,  R1      ;READ THE DATA OUT OF MEMORY
7223  006564  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  006566  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  006570  004767  011564          64$:  JSR     PC,     SPRNT1  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  006574  004767  013050                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  006600  000003                        .WORD   3               ;ERROR TYPE CODE.
  (2)  006602                            65$:
7224  006602  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  006604  001366                        BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  006606  004767  006402                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 3$.
7225
```

```
 7226                                   ;;*****************************************************************
  (3)                                   ;*TEST 5        WRITE 1'S COMPLEMENT OF BANK #.
  (4)                                   ;*      R0 - DATA WRITTEN INTO MEMORY (SHOULD BE)
  (4)                                   ;*      R1 = DATA READ FROM MEMORY (WAS)
  (4)                                   ;*      R2 = VIRTUAL ADDRESS
  (4)                                   ;*      R3 = NOT USED
  (4)                                   ;*      R4 = NOT USED
  (4)                                   ;*      R5 = BLOCK BOUNDRY BIT MASK.
  (3)                                   ;;*****************************************************************
  (2) 006612                            TST5:
  (3) 006612  004567  012022                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3) 006616  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7227                                   ;*  DOWNWARDS BYTE ADDRESSING.
 7228 006620  004467  006250                    JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7229 006624  004767  007310            1$:     JSR     PC,     BANKNO  ;GET THE BANK NUMBER INTO R0
 7230 006630  005100                            COM     R0              ;1'S COMPLEMENT OF BANK #
 7231 006632  110042                    2$:     MOVB    R0,     -(R2)   ;PUT 1'S COM OF BANK # INTO MEMORY
 7232 006634  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1) 006636  001375                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1) 006640  004767  007040                    JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO 1$.
 7233
 7234                                   ;*  CHECK THAT DATA WRITTEN CAN BE READ.
 7235                                   ;*  DOWNWARDS BYTE ADDRESSING.
 7236 006644  004467  006224                    JSR     R4,     INITDN  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7237 006650  004767  007264            3$:     JSR     PC,     BANKNO  ;GET THE BANK # INTO R0
 7238 006654  005100                            COM     R0              ;SET 1'S COMPLEMENT OF BANK #
 7239 006656  114201                    4$:     MOVB    -(R2),  R1      ;READ DATA OUT OF MEMORY
 7240 006660  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2) 006662  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3) 006664  004767  011462            64$:    JSR     PC,     SPRNTO  ;SET UP VALUES FOR ERROR PRINTING.
  (4) 006670  004767  012754                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4) 006674  000003                            .WORD   3               ;ERROR TYPE CODE.
  (2) 006676                            65$:
 7241 006676  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1) 006700  001366                            BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1) 006702  004767  006776                    JSR     PC,     MMDOWN  ;FIND NEXT BLOCK AND LOOP TO $TAG1.
 7242
 7243                                   .SBTTL SECTION 2:      WORST CASE NOISE TESTS
 7244                                   ;;*****************************************************************
 7245                                   ;*  THESE TESTS WRITE MEMORY WORST CASE NOISE TEST PATTERNS THROUGHOUT
 7246                                   ;*  MEMORY AND CHECK THAT THEY CAN BE WRITTEN AND READ.
 7247                                   ;;*****************************************************************
 7248                                   ;;*****************************************************************
  (3)                                   ;*TEST 6        WRITE A CONSTANT INTO MEMORY.
  (3)                                           ;*  THE CONSTANT IS USER SELECTABLE (DEFAULT = 0).
  (4)                                   ;*      R0 = DATA WRITTEN INTO MEMORY (SHOULD BE)
  (4)                                   ;*      R1 = DATA READ FROM MEMORY (WAS)
  (4)                                   ;*      R2 = VIRTUAL ADDRESS
  (4)                                   ;*      R3 = NOT USED
  (4)                                   ;*      R4 = NOT USED
  (4)                                   ;*      R5 = BLOCK BOUNDRY BIT MASK.
  (3)                                   ;;*****************************************************************
  (2) 006706                            TST6:
  (3) 006706  004567  011726                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3) 006712  000000                            .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7249 006714  016700  172670            TST6A:  MOV     .CONST, R0      ;GET USER CONSTANT
```

```
 7250  006720  004467  005512              JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7251  006724  010022                1$:   MOV     R0,     (R2)+   ;WRITE CONSTANT INTO MEMORY.
 7252  006726  030502                      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
   (1) 006730  001375                      BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
   (1) 006732  004767  006256              JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 7253
 7257                                ;;**********************************************************************
   (3)                               ;*TEST 7        READ MEMORY AND COMPARE TO CONSTANT.
   (4)                               ;*  IMPORTANT:  THIS TEST SHOULD NOT BE RUN WITHOUT FIRST RUNNING TEST $TN.
   (3)                               ;;**********************************************************************
   (2) 006736                        TST7:
   (3) 006736  004567  011676              JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
   (3) 006742  000000                      .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7258  006744  016700  172640              MOV     .CONST, R0      ;GET USER CONSTANT
 7259  006750  004467  005462              JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7260  006754                          1$:
   (1) 006754  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2) 006756  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3) 006760  001405                      BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4) 006762  004767  011410        64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5) 006766  004767  012656              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5) 006772  000004                      .WORD   4               ;ERROR TYPE CODE.
   (3) 006774                        65$:
 7261  006774  030502                      BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
   (1) 006776  001366                      BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
   (1) 007000  004767  006210              JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 7263                                ;*  SPECIAL CHECK TO SEE IF TEST 6 IS SELECTED THRU THE SWR.
 7264                                ;*  ALLOWS THE OPERATOR TO SWITCH BACK AND FORTH BETWEEN TESTS 6 AND 7
 7265                                ;*  BY SIMPLY 'TOGGLING' SW00 WHEN SW01, SW02, AND SW08 ARE SET.
 7266  007004  032777  000400  172126        BIT     #SW08,  @SWR    ;CHECK THAT LOOP ON TEST BIT SET
 7267  007012  001416                      BEQ     TST10           ;BRANCH IF NOT LOOP ON TEST
 7268  007014  017746  172120              MOV     @SWR,   -(SP)   ;GET SWITCH REGISTER DATA.
 7269  007020  042716  177740              BIC     #177740,(SP)    ;CLEAR NON-TEST-NUMBER SWITCHS.
 7270  007024  022726  000006              CMP     #6,     (SP)+   ;CHECK IF TEST 6 IN SWITCHES.
 7271  007030  001007                      BNE     TST10           ;BRANCH IF NOT TEST 6
 7272  007032  162767  000001  172042        SUB     #1,     $TSTNM  ;RESET TEST NUM
 7273  007040  162767  000030  172040        SUB     #TST7-TST6,$LPADR ;RESET LOOP ADR
 7274  007046  000722                      BR      TST6A           ;GO TO TEST 6
 7276
 7277                                ;;**********************************************************************
   (3)                               ;*TEST 10       WORSE CASE NOISE (PARITY) WORD TESTING
   (3)                               ;*  CHECK MEMORY WITH A SERIES OF PATTERNS
   (3)                               ;;**********************************************************************
   (2) 007050                        TST10:
   (3) 007050  004567  011564              JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
   (3) 007054  000000                      .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7278  007056  016704  172552              MOV     .MPPAT, R4      ;INITIALIZE PATTERN TABLE POINTER
 7279  007062  004767  010560        1$:   JSR     PC,     CKPMER  ;CHECK FOR NON-TRAP PARITY MEMORY ERRORS.
 7280  007066  012400                      MOV     (R4)+,  R0      ;GET THE DATA PATTERN.
 7281  007070  001420                      BEQ     TST11           ;;BR IF END OF TABLE.
 7282  007072  004467  005340              JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7283  007076  010012                2$:   MOV     R0,     (R2)    ;PUT DATA PATTERN INTO MEMORY.
 7284  007100  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2) 007102  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3) 007104  001405                      BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4) 007106  004767  011264        64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
    (5)  007112  004767  012532                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
    (5)  007116  000004                            .WORD   4               ;ERROR TYPE CODE.
    (3)  007120                             65$:
 7285    007120  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
    (1)  007122  001365                            BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
    (1)  007124  004767  006064                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 2$.
 7286    007130  000754                            BR      1$              ;BR BACK TO DO NEXT PATTERN
```

```
7288
 (3)
 (3)
 (2)  007132
 (3)  007132  004567  011502          TST11:  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)  007136  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
7289  007140  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
7290  007144  004767  007030                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY.
7291  007150  004467  005262                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7292  007154  000241                  1$:     CLC                     ;CLEAR CARRY BIT IN PSW
7293  007156  004767  007036                  JSR     PC,     ROTATE
7294  007162  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
7295  007166  103402                          BCS     63$             ;BRANCH IF 'C' BIT WAS SET
7296  007170  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (2)  007172  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (3)  007174  004767  011176          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (4)  007200  004767  012444                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (4)  007204  000005                          .WORD   5               ;ERROR TYPE CODE.
 (2)  007206                      64$:
7297  007206  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)  007210  001361                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)  007212  004767  005776                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
7298
7299
 (3)
 (3)
 (2)  007216
 (3)  007216  004567  011416          TST12:  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)  007222  000000                          .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
7300  007224  005000                          CLR     R0              ;SET CHECK WORD
7301  007226  004767  006746                  JSR     PC,     SETCON  ;PUT THE CONTENTS OF R0 IN ALL MEMORY
7302  007232  004467  005200                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7303  007236  000261                  1$:     SEC                     ;SET 'C' BIT IN PSW
7304  007240  004767  006754                  JSR     PC,     ROTATE  ;GO ROTATE '1' BIT
7305  007244  016201  177776                  MOV     -2(R2), R1      ;GET RESULT
7306  007250  103002                          BCC     63$             ;BRANCH IF 'C' IS CLEAR
7307  007252  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (2)  007254  001405                          BEQ     64$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (3)  007256  004767  011114          63$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (4)  007262  004767  012362                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (4)  007266  000005                          .WORD   5               ;ERROR TYPE CODE.
 (2)  007270                      64$:
7308  007270  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)  007272  001361                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)  007274  004767  005714                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
7310                                  ;;*****************************************************************
 (3)                                  ;*TEST 13        3 XOR 9 TEST PATTERN.
 (3)                                  ;;*****************************************************************
 (2)   007300                         TST13:
 (3)   007300  004567  011334                 JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)   007304  000777                         .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
 (3)                                                                  ; REQUIRED FOR THIS TEST.
 (3)   007306  000167  000312                 JMP     TST14           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
 (3)                                                                  ; AVAILABLE FOR TEST.
7311   007312  005000                 .3X9:   CLR     R0              ;SET UP TEST DATA
7312   007314  012703  177777                 MOV     #-1,    R3      ;SET COM DATA REG
7313   007320  004467  005112                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7314   007324  004767  006736         1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
7315   007330  030502                         BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)   007332  001374                         BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)   007334  004767  005654                 JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
7316
7317                                  ;;*****************************************************************
7318                                  ;*  CHECK 3 XOR 9 TEST PATTERN WRITTEN ABOVE
7319                                  ;;*****************************************************************
7320   007340  005000                         CLR     R0              ;SET CHECK WORD
7321   007342  004467  005070                 JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7322   007346  012704  000100         11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
7323   007352                         12$:
 (1)   007352  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)   007354  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)   007356  001405                         BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)   007360  004767  011012         64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)   007364  004767  012260                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)   007370  000007                         .WORD   7               ;ERROR TYPE CODE.
 (3)   007372                         65$:
7324   007372  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)   007374  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)   007376  001405                         BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)   007400  004767  010772         66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)   007404  004767  012240                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)   007410  000007                         .WORD   7               ;ERROR TYPE CODE.
 (3)   007412                         67$:
7325   007412  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)   007414  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)   007416  001405                         BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)   007420  004767  010752         68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)   007424  004767  012220                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)   007430  000007                         .WORD   7               ;ERROR TYPE CODE.
 (3)   007432                         69$:
7326   007432  012201                         MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)   007434  020001                         CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)   007436  001405                         BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)   007440  004767  010732         70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)   007444  004767  012200                 JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)   007450  000007                         .WORD   7               ;ERROR TYPE CODE.
 (3)   007452                         71$:
7327   007452  005100                         COM     R0              ;COMPLEMENT CHECK WORD
7328   007454  005304                         DEC     R4              ;DECREMENT 256. WORD COUNTER
7329   007456  001335                         BNE     12$
7330   007460  005100                         COM     R0              ;COMPLEMENT CHECK WORD
```

```
 7331  007462  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  007464  001330                    BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  007466  004767  005522            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
 7332
 7333                                ;;*****************************************************************
 7334                                ;*  CHECK, COM, CHECK, COM, CHECK 3 XOR 9 PATTERN WRITTEN ABOVE.
 7335                                ;;*****************************************************************
 7336  007472  005000                    CLR     R0
 7337  007474  004467  004736            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7338  007500  012704  000100    21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
 7339  007504  012703  000004    22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
 7340  007510                    23$:
  (1)  007510  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  007512  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  007514  001405                    BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  007516  004767  010654    72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  007522  004767  012122            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  007526  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  007530                    73$:
 7341  007530  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7342  007532  005142                    COM     -(R2)           ;COMPLEMENT TEST DATA
 7343  007534  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  007536  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  007540  001405                    BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  007542  004767  010630    74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  007546  004767  012076            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  007552  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  007554                    75$:
 7344  007554  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7345  007556  005142                    COM     -(R2)           ;COMPLEMENT TEST DATA
 7346  007560  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  007562  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  007564  001405                    BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  007566  004767  010604    76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  007572  004767  012052            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  007576  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  007600                    77$:
 7347  007600  005303                    DEC     R3              ;DECREMENT 4 WORD COUNTER
 7348  007602  001342                    BNE     23$             ;BR IF NOT DONE.
 7349  007604  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7350  007606  005304                    DEC     R4              ;DECREMENT 256. WORD COUNTER
 7351  007610  001335                    BNE     22$             ;BR IF NOT DONE.
 7352  007612  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7353  007614  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  007616  001330                    BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  007620  004767  005370            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                ;;*****************************************************************
                                ;*TEST 14       COMPLEMENT 3 XOR 9 TEST PATTERN
                                ;;*****************************************************************
007624                          TST14:
007624  004567  011010                  JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
007630  000777                          .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
                                                                ; REQUIRED FOR THIS TEST.
007632  000167  000316                  JMP     TST15           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                                                ; AVAILABLE FOR TEST.
007636  012700  177777                  MOV     #-1,    R0      ;SET UP TEST DATA
007642  005003                          CLR     R3              ;SET COM DATA REG
007644  004467  004566                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007650  004767  006412          1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
007654  030502                          BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
007656  001374                          BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
007660  004767  005330                  JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                ;;*****************************************************************
                                ;*  CHECK COMPLEMENTED 3 XOR 9 TEST PATTERN WRITTEN ABOVE.
                                ;;*****************************************************************
007664  012700  177777                  MOV     #-1,    R0      ;SET CHECK WORD
007670  004467  004542                  JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
007674  012704  000100          11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
007700                          12$:
007700  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007702  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007704  001405                          BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007706  004767  010464          64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007712  004767  011732                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007716  000007                          .WORD   7               ;ERROR TYPE CODE.
007720                          65$:
007720  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007722  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007724  001405                          BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007726  004767  010444          66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007732  004767  011712                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007736  000007                          .WORD   7               ;ERROR TYPE CODE.
007740                          67$:
007740  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007742  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007744  001405                          BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007746  004767  010424          68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007752  004767  011672                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007756  000007                          .WORD   7               ;ERROR TYPE CODE.
007760                          69$:
007760  012201                          MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
007762  020001                          CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
007764  001405                          BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
007766  004767  010404          70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
007772  004767  011652                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
007776  000007                          .WORD   7               ;ERROR TYPE CODE.
010000                          71$:
010000  005100                          COM     R0              ;COMPLEMENT CHECK WORD
010002  005304                          DEC     R4              ;DECREMENT 256. WORD COUNTER
010004  001335                          BNE     12$
```

```
 7376  010006  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 7377  010010  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  010012  001330                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  010014  004767  005174                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
 7378
 7379                                ;;*****************************************************************
 7380                                ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED 3 XOR 9 PATTERN.
 7381                                ;;*****************************************************************
 7382  010020  012700  177777                MOV     #-1,    R0      ;SET UP CHECK WORD.
 7383  010024  004467  004406                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7384  010030  012704  000100        21$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
 7385  010034  012703  000004        22$:    MOV     #4,     R3      ;SET 4 WORD COUNTER
 7386  010040                        23$:
  (1)  010040  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  010042  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  010044  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  010046  004767  010324        72$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  010052  004767  011572                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  010056  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  010060                        73$:
 7387  010060  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 7388  010062  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 7389  010064  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  010066  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  010070  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  010072  004767  010300        74$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  010076  004767  011546                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  010102  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  010104                        75$:
 7390  010104  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 7391  010106  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
 7392  010110  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  010112  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  010114  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  010116  004767  010254        76$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  010122  004767  011522                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  010126  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  010130                        77$:
 7393  010130  005303                        DEC     R3              ;DECREMENT 4 WORD COUNTER
 7394  010132  001342                        BNE     23$             ;BR IF NOT DONE.
 7395  010134  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 7396  010136  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
 7397  010140  001335                        BNE     22$             ;BR IF NOT DONE.
 7398  010142  005100                        COM     R0              ;COMPLEMENT CHECK WORD
 7399  010144  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  010146  001330                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  010150  004767  005040                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
                                        ;;*****************************************************************
 (3)                                    ;*TEST 15        MODIFIED 3 XOR 9 PATTERN FOR PARITY MEMORY
 (3)                                    ;;*****************************************************************
 (2)  010154                            TST15:
 (3)  010154  004567  010460                    JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)  010160  000777                            .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
 (3)                                                                    ; REQUIRED FOR THIS TEST.
 (3)  010162  000167  000610                    JMP     TST16           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
 (3)                                                                    ; AVAILABLE FOR TEST.
      010166  012700  000401                    MOV     #401,   R0      ;SET UP PARITY ''ALL ZEROS'' PATTERN
      010172  012703  177777                    MOV     #-1,    R3      ;SET COM DATA REG
      010176  004467  004234                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      010202  004767  006060            1$:     JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
      010206  030502                            BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)  010210  001374                            BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)  010212  004767  004776                    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.

                                        ;;*****************************************************************
                                        ;*  CHECK PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
                                        ;;*****************************************************************
      010216  012700  000401                    MOV     #401,   R0      ;RESET PARITY ''ALL ZEROS'' PATTERN.
      010222  012703  177777                    MOV     #-1,    R3      ;RESET PARITY ALL ONES PATTERN.
      010226  004467  004204                    JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      010232  012704  000100            11$:    MOV     #64.,   R4      ;SET 256. WORD COUNTER
      010236                            12$:
 (1)  010236  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010240  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010242  001405                            BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010244  004767  010126            64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010250  004767  011374                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010254  000007                            .WORD   7               ;ERROR TYPE CODE.
 (3)  010256                            65$:
      010256  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010260  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010262  001405                            BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010264  004767  010106            66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010270  004767  011354                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010274  000007                            .WORD   7               ;ERROR TYPE CODE.
 (3)  010276                            67$:
      010276  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010300  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010302  001405                            BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010304  004767  010066            68$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010310  004767  011334                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010314  000007                            .WORD   7               ;ERROR TYPE CODE.
 (3)  010316                            69$:
      010316  012201                            MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010320  020001                            CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010322  001405                            BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010324  004767  010046            70$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010330  004767  011314                    JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010334  000007                            .WORD   7               ;ERROR TYPE CODE.
 (3)  010336                            71$:
      010336  010046                            MOV     R0,     -(SP)   ;SAVE R0
 (1)  010340  010300                            MOV     R3,     R0      ;PUT R3 INTO R0
 (1)  010342  012603                            MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
```

```
7420  010344  005304                        DEC     R4              ;COUNT 256. WORDS
7421  010346  001333                        BNE     12$             ;BRANCH IF MORE
7422  010350  010046                        MOV     R0,     -(SP)   ;SAVE R0
 (1)  010352  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
 (1)  010354  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
7423  010356  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1)  010360  001324                        BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
 (1)  010362  004767  004626                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
7424
7425                                ;;*****************************************************************
7426                                ;*  CHECK, COM, CHECK, COM, CHECK PARITY 3 XOR 9 PATTERN.
7427                                ;;*****************************************************************
7428  010366  012700  000401                MOV     #401,   R0      ;SET UP PARITY "ALL ZEROS" PATTERN.
7429  010372  012703  177777                MOV     #-1,    R3      ;SET UP ALL ONES PATTERN.
7430  010376  004467  004034                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7431  010402  012704  000100      21$:      MOV     #64.,R4         ;SET 256. WORD COUNTER
7432  010406                       22$:
 (1)  010406  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010410  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010412  001405                        BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010414  004767  007756      72$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010420  004767  011224                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010424  000007                        .WORD   7               ;ERROR TYPE CODE.
 (3)  010426                       73$:
7433  010426  005100                        COM     R0              ;COMPLEMENT CHECK WORD
7434  010430  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
7435  010432  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010434  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010436  001405                        BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010440  004767  007732      74$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010444  004767  011200                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010450  000007                        .WORD   7               ;ERROR TYPE CODE.
 (3)  010452                       75$:
7436  010452  005100                        COM     R0              ;COMPLEMENT CHECK WORD
7437  010454  005142                        COM     -(R2)           ;RESTORE DATA
7438  010456  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010460  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010462  001405                        BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010464  004767  007706      76$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010470  004767  011154                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010474  000007                        .WORD   7               ;ERROR TYPE CODE.
 (3)  010476                       77$:
7439  010476  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010500  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010502  001405                        BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010504  004767  007666      78$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010510  004767  011134                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (5)  010514  000007                        .WORD   7               ;ERROR TYPE CODE.
 (3)  010516                       79$:
7440  010516  005100                        COM     R0              ;COMPLEMENT CHECK WORD
7441  010520  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
7442  010522  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
 (2)  010524  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (3)  010526  001405                        BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (4)  010530  004767  007642      80$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
 (5)  010534  004767  011110                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
```

```
(5)  010540  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010542                          81$:
7443 010542  005100                          COM     R0              ;COMPLEMENT CHECK WORD
7444 010544  005142                          COM     -(R2)           ;RESTORE DATA
7445 010546  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010550  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010552  001405                          BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
(4)  010554  004767  007616          82$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010560  004767  011064                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010564  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010566                          83$:
7446 010566  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010570  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010572  001405                          BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
(4)  010574  004767  007576          84$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010600  004767  011044                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010604  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010606                          85$:
7447 010606  005100                          COM     R0              ;COMPLEMENT CHECK WORD
7448 010610  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
7449 010612  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010614  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010616  001405                          BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
(4)  010620  004767  007552          86$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010624  004767  011020                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010630  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010632                          87$:
7450 010632  005100                          COM     R0              ;COMPLEMENT CHECK WORD
7451 010634  005142                          COM     -(R2)           ;RESTORE DATA
7452 010636  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010640  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010642  001405                          BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
(4)  010644  004767  007526          88$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010650  004767  010774                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010654  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010656                          89$:
7453 010656  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010660  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010662  001405                          BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
(4)  010664  004767  007506          90$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010670  004767  010754                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010674  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010676                          91$:
7454 010676  005100                          COM     R0              ;COMPLEMENT CHECK WORD
7455 010700  005142                          COM     -(R2)           ;COMPLEMENT TEST DATA
7456 010702  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010704  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
(3)  010706  001405                          BEQ     93$             ;BRANCH OVER ERROR CAL_ IF GOOD DATA.
(4)  010710  004767  007462          92$:    JSR     PC, SPRNT2      ;SET UP VALUES FOR ERROR PRINTING.
(5)  010714  004767  010730                  JSR     PC, $ERROR      ;*** ERROR *** (GO TYPE A MESSAGE)
(5)  010720  000007                          .WORD   7               ;ERROR TYPE CODE.
(3)  010722                          93$:
7457 010722  005100                          COM     R0              ;COMPLEMENT CHECK WORD
7458 010724  005142                          COM     -(R2)           ;RESTORE DATA
7459 010726  012201                          MOV     (R2)+, R1       ;GET THE DATA FROM MEMORY UNDER TEST.
(2)  010730  020001                          CMP     R0, R1          ;COMPARE THE CHECK WORD WITH THE DATA READ.
```

```
   (3)  010732  001405                    BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  010734  004767  007436  94$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  010740  004767  010704            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  010744  000007                    .WORD   7               ;ERROR TYPE CODE.
   (3)  010746                    95$:
  7460  010746  010046                    MOV     R0,     -(SP)   ;SAVE R0
   (1)  010750  010300                    MOV     R3,     R0      ;PUT R3 INTO R0
   (1)  010752  012603                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
  7461  010754  005304                    DEC     R4              ;DECREMENT 256. WORD COUNTER
  7462  010756  001213                    BNE     22$             ;BRANCH IF MORE.
  7463  010760  010046                    MOV     R0,     -(SP)   ;SAVE R0
   (1)  010762  010300                    MOV     R3,     R0      ;PUT R3 INTO R0
   (1)  010764  012603                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
  7464  010766  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
   (1)  010770  001204                    BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
   (1)  010772  004767  004216            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
  7465
  7466                            ;;*****************************************************************
   (3)                            ;*TEST 16       COMPLEMENT PARITY 3 XOR 9 TEST PATTERN.
   (3)                            ;;*****************************************************************
   (2)  010776                    TST16:
   (3)  010776  004567  007636            JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
   (3)  011002  000777                    .WORD   777             ;MINIMUM BLOCK SIZE OF 256. WORDS
   (3)                                                            ; REQUIRED FOR THIS TEST.
   (3)  011004  000167  000610            JMP     TST17           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
   (3)                                                            ; AVAILABLE FOR TEST.
  7467  011010  012700  177777            MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
  7468  011014  012703  000401            MOV     #401,   R3      ;SET UP PARITY ''ALL ZEROS'' PATTERN
  7469  011020  004467  003412            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  7470  011024  004767  005236  1$:       JSR     PC,     W3X9    ;WRITE 256. WORD BLOCK WITH 3 XOR 9 PAT.
  7471  011030  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
   (1)  011032  001374                    BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
   (1)  011034  004767  004154            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
  7472
  7473                            ;;*****************************************************************
  7474                            ;*  CHECK COMPLEMENT PARITY 3 XOR 9 PATTERN WRITTEN ABOVE.
  7475                            ;;*****************************************************************
  7476  011040  012700  177777            MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
  7477  011044  012703  000401            MOV     #401,   R3      ;SET UP PARITY ''ALL ZEROS'' PATTERN
  7478  011050  004467  003362            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
  7479  011054  012704  000100  11$:       MOV     #64.,   R4      ;SET 256. WORD COUNTER
  7480  011060                    12$:
   (1)  011060  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011062  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011064  001405                    BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011066  004767  007304  64$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011072  004767  010552            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011076  000007                    .WORD   7               ;ERROR TYPE CODE.
   (3)  011100                    65$:
  7481  011100  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011102  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011104  001405                    BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011106  004767  007264  66$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011112  004767  010532            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011116  000007                    .WORD   7               ;ERROR TYPE CODE.
   (3)  011120                    67$:
```

```
 7482  011120  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011122  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011124  001405                    BEQ     69$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011126  004767  007244  68$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011132  004767  010512            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011136  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  011140                    69$:
 7483  011140  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011142  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011144  001405                    BEQ     71$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011146  004767  007224  70$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011152  004767  010472            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011156  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  011160                    71$:
 7484  011160  010046                    MOV     R0,     -(SP)   ;SAVE R0
  (1)  011162  010300                    MOV     R3,     R0      ;PUT R3 INTO R0
  (1)  011164  012603                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 7485  011166  005304                    DEC     R4              ;COUNT 256. WORDS
 7486  011170  001333                    BNE     12$             ;BRANCH IF MORE
 7487  011172  010046                    MOV     R0,     -(SP)   ;SAVE R0
  (1)  011174  010300                    MOV     R3,     R0      ;PUT R3 INTO R0
  (1)  011176  012603                    MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
 7488  011200  030502                    BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  011202  001324                    BNE     11$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  011204  004767  004004            JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 11$.
 7489
 7490                            ;;*****************************************************************
 7491                            ;*  CHECK, COM, CHECK, COM, CHECK COMPLEMENTED PARITY 3 XOR 9 PATTERN.
 7492                            ;;*****************************************************************
 7493  011210  012700  177777            MOV     #-1,    R0      ;SET UP ALL ONES PATTERN
 7494  011214  012703  000401            MOV     #401,   R3      ;SET UP PARITY "ALL ZEROS" PATTERN
 7495  011220  004467  003212            JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7496  011224  012704  000100  21$:      MOV     #64.,R4         ;SET 256. WORD COUNTER
 7497  011230                    22$:
  (1)  011230  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011232  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011234  001405                    BEQ     73$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011236  004767  007134  72$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011242  004767  010402            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011246  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  011250                    73$:
 7498  011250  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7499  011252  005142                    COM     -(R2)           ;COMPLEMENT TEST DATA
 7500  011254  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011256  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011260  001405                    BEQ     75$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011262  004767  007110  74$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011266  004767  010356            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011272  000007                    .WORD   7               ;ERROR TYPE CODE.
  (3)  011274                    75$:
 7501  011274  005100                    COM     R0              ;COMPLEMENT CHECK WORD
 7502  011276  005142                    COM     -(R2)           ;RESTORE DATA
 7503  011300  012201                    MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011302  020001                    CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011304  001405                    BEQ     77$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011306  004767  007064  76$:      JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
```

```
   (5)  011312  004767  010332              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011316  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011320                      77$:
 7504   011320  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011322  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011324  001405                      BEQ     79$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011326  004767  007044      78$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011332  004767  010312              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011336  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011340                      79$:
 7505   011340  005100                      COM     R0              ;COMPLEMENT CHECK WORD
 7506   011342  005142                      COM     -(R2)           ;COMPLEMENT TEST DATA
 7507   011344  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011346  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011350  001405                      BEQ     81$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011352  004767  007020      80$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011356  004767  010266              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011362  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011364                      81$:
 7508   011364  005100                      COM     R0              ;COMPLEMENT CHECK WORD
 7509   011366  005142                      COM     -(R2)           ;RESTORE DATA
 7510   011370  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011372  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011374  001405                      BEQ     83$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011376  004767  006774      82$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011402  004767  010242              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011406  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011410                      83$:
 7511   011410  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011412  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011414  001405                      BEQ     85$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011416  004767  006754      84$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011422  004767  010222              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011426  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011430                      85$:
 7512   011430  005100                      COM     R0              ;COMPLEMENT CHECK WORD
 7513   011432  005142                      COM     -(R2)           ;COMPLEMENT TEST DATA
 7514   011434  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011436  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011440  001405                      BEQ     87$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011442  004767  006730      86$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011446  004767  010176              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011452  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011454                      87$:
 7515   011454  005100                      COM     R0              ;COMPLEMENT CHECK WORD
 7516   011456  005142                      COM     -(R2)           ;RESTORE DATA
 7517   011460  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011462  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011464  001405                      BEQ     89$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
   (4)  011466  004767  006704      88$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
   (5)  011472  004767  010152              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
   (5)  011476  000007                      .WORD   7               ;ERROR TYPE CODE.
   (3)  011500                      89$:
 7518   011500  012201                      MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
   (2)  011502  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
   (3)  011504  001405                      BEQ     91$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
```

```
  (4)  011506  004767  006664          90$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011512  004767  010132                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011516  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  011520                         91$:
7519   011520  005100                        COM     R0              ;COMPLEMENT CHECK WORD
7520   011522  005142                        COM     -(R2)           ;COMPLEMENT TEST DATA
7521   011524  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011526  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011530  001405                        BEQ     93$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011532  004767  006640          92$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011536  004767  010106                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011542  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  011544                         93$:
7522   011544  005100                        COM     R0              ;COMPLEMENT CHECK WORD
7523   011546  005142                        COM     -(R2)           ;RESTORE DATA
7524   011550  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  011552  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  011554  001405                        BEQ     95$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  011556  004767  006614          94$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  011562  004767  010062                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  011566  000007                        .WORD   7               ;ERROR TYPE CODE.
  (3)  011570                         95$:
7525   011570  010046                        MOV     R0,     -(SP)   ;SAVE R0
  (1)  011572  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
  (1)  011574  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
7526   011576  005304                        DEC     R4              ;DECREMENT 256. WORD COUNTER
7527   011600  001213                        BNE     22$             ;BRANCH IF MORE.
7528   011602  010046                        MOV     R0,     -(SP)   ;SAVE R0
  (1)  011604  010300                        MOV     R3,     R0      ;PUT R3 INTO R0
  (1)  011606  012603                        MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
7529   011610  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  011612  001204                        BNE     21$             ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  011614  004767  003374                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 21$.
```

```
 7538                                   ;;*****************************************************************
  (3)                                   ;*TEST 17       WORSE CASE NOISE PARITY BYTE TESTING
  (4)                                   ;*  CHECK PARITY MEMORY WITH A SERIES OF BYTE PATTERNS
  (4)                                   ;*       1) FORCE WRONG PARITY IN EACH BYTE OF PARITY MEMORY
  (4)                                   ;*       2) READ IT BACK WITH ACTION ENABLE SET, MAKING SURE THAT A TRAP OCCURS
  (4)                                   ;*       3) WRITE GOOD PARITY AND MAKE SURE NO TRAP OCCURS WHEN IT IS READ
  (4)                                   ;*       4) MAKE SURE THE ERROR ADDRESS BITS (CSR BITS <11-5>) ARE CORRECT
  (3)                                   ;;*****************************************************************
  (2)  011620                           TST17:
  (3)  011620  004567  007014                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  011624  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7539  011626  005767  170444           WWPB0:  TST     MPRX            ;CHECK FOR ANY PARITY MEMORY.
 7540  011632  001404                           BEQ     1$              ;BR IF NO PARITY MEMORY.
 7541  011634  032777  000100  167276           BIT     #SW06,  @SWR    ;CHECK FORINHIBIT PARITY SWITCH.
 7542  011642  001402                           BEQ     2$              ;BR IF NOT SET.
 7543  011644  000167  000622           1$:     JMP     TST20           ;SKIP THIS TEST IF NO PARITY MEMORY PRESENT.
 7544  011650  005000                   2$:     CLR     R0              ;ZERO TO BE PUT IN ALL MEMORY.
 7545  011652  004767  004322                   JSR     PC,     SETCON  ;ROUTINE TO LOAD ALL MEMORY.
 7546  011656  004467  002554                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7547
 7548  011662  036767  167656  167650  WWPBYT: BIT     BITPT,  PMEMAP  ;CHECK IF CURRENT BANK HAS PARITY MEMORY.
 7549  011670  001010                           BNE     2$              ;BR IF PARITY MEM.
 7550  011672  036767  167650  167642           BIT     BITPT+2,PMEMAP+2 ;...HI 64K.
 7551  011700  001004                           BNE     2$              ;BR IF PARITY MEM.
 7552  011702  050502                           BIS     R5,     R2      ;POINT TO END OF BLOCK.
 7553  011704  005202                           INC     R2              ;FIRST ADR OF NEXT BLOCK.
 7554  011706  000167  000540                   JMP     WWPB5           ;BR TO FIND NEXT BLOCK.
 7555  011712  004767  005674           2$:     JSR     PC,     SETAE   ;SET ACTION ENABLE (EVEN IF BANK0.)
 7556  011716  004767  005724                   JSR     PC,     CKPMER  ;CHECK FOR ANY NON TRAP PARITY ERRORS.
 7557  011722  020227  000114           WWPB1:  CMP     R2,     #114    ;CKECK IF POINTING TO PARITY ERROR VECTOR.
 7558  011726  001004                           BNE     3$              ;BR IF NOT AT VECTOR.
 7559  011730  062702  000004                   ADD     #4,     R2      ;SKIP PARITY VECTOR.
 7560  011734  000167  000512                   JMP     WWPB5           ;CHECK FOR BLOCK END.
 7561  011740  111201                   3$:     MOVB    (R2),   R1      ;CHECK IF BYTE STILL CLEARED.
 7562  011742  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (2)  011744  004767  006352           64$:    JSR     PC,     SPRNT   ;SET UP VALUES FOR ERROR PRINTING.
  (3)  011750  004767  007674                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (3)  011754  000011                           .WORD   11              ;ERROR TYPE CODE.
  (1)  011756                           65$:
 7563  011756  105067  167576                   CLRB    OEFLG           ;CLEAR ODD/EVEN FLAG.
 7564  011762  112700  000252                   MOVB    #252,   R0      ;SET UP DATA...EVEN, SETS PARITY BIT.
 7565  011766  110012                   WWPB2:  MOVB    R0,     (R2)    ;MOV DATA INTO TEST LOCATION.
 7566  011770  016703  167634                   MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
 7567  011774  056773  167612  000000   10$:    BIS     WWP,@(R3)       ;SET WRITE WRONG PARITY.
 7568  012002  052733  000001                   BIS     #AE,@(R3)+
 7569  012006  005713                           TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 7570  012010  001371                           BNE     10$             ;BR IF MORE REGS IN TABLE.
 7571                                   ;*  SET WRONG PARITY IN LOCATION UNDER TEST.
 7572  012012  110012                           MOVB    R0,     (R2)    ;WRITE SAME DATA (EXCEPT PARITY) VIA DATOB.
 7573  012014  016703  167610                   MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
 7574  012020  046733  167566           11$:    BIC     WWP,    @(R3)+  ;CLEAR WRITE WRONG PARITY.
 7575  012024  005713                           TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 7576  012026  001374                           BNE     11$             ;BR IF MORE PARITY REGISTERS.
 7577  012030  016737  167576  000114           MOV     .PBTRP, @#PARVEC ;SET UP VECTOR FOR EXPECTED TRAP.
 7578                                   ;*  DETECT WRONG PARITY VIA DATIP; DATOB SHOULDN'T EXECUTE.
 7579  012036  105412                           NEGB    (R2)            ;DATIP (DATOB AND COM PARITY BIT.)
```

```
7580                                        ;*  SHOULD HAVE TRAPPED TO PBTRP.
7581  012040  016737  167572  000114          MOV     .PESRV, @#PARVEC ;RESET VECTOR FOR UNEXPECTED TRAPS.
7582  012046  004767  006300                64$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
 (2)  012052  004767  007572                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (2)  012056  000012                                .WORD   12              ;ERROR TYPE CODE.
7583  012060  000562                        BR      WWPB4           ;SKIP TRAP SERVICE.
7584
7585                                        ;*  EXPECTED PARITY MEMORY TRAPS COME HERE.
7586  012062  016737  167550  000114  PBTRP: MOV     .PESRV, @#PARVEC ;RESET PARITY VECTOR FOR UNEXPECTED TRAPS.
7587  012070  022626                        CMP     (SP)+,  (SP)+   ;RESET THE STACK POINTER AFTER TRAP.
7588  012072  016703  167530                MOV     .MPR0,  R3      ;GET PARITY REG AND MAP TABLE POINTER.
7589  012076  032713  000001          21$:  BIT     #BIT0,  (R3)    ;CHECK IF THIS REGISTER EXISTS.
7590  012102  001003                        BNE     22$             ;BR IF IT DOESN'T EXIST.
7591  012104  017301  000000                MOV     @(R3),  R1      ;GET THE CONTENTS.
7592  012110  100413                        BMI     23$             ;BR IF ERROR FLAG SET.
7593  012112  062703  000010          22$:  ADD     #10,    R3      ;MOVE POINTER TO NEXT REG.
7594  012116  020367  167506                CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
7595  012122  103765                        BLO     21$             ;BR IF MORE REGISTERS.
7596  012124  004767  006222          64$:  JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
 (2)  012130  004767  007514                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (2)  012134  000013                        .WORD   13              ;ERROR TYPE CODE.
7597  012136  000533                        BR      WWPB4           ;EXIT AFTER ERROR.
7598  012140  036763  167400  000002  23$:  BIT     BITPT,  2(R3)   ;CHECK THE MAP FOR THIS REGISTER.
7599  012146  001011                        BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
7600  012150  036763  167372  000004        BIT     BITPT+2,4(R3)   ;CHECK THE HI 64K.
7601  012156  001005                        BNE     24$             ;BR IF THIS REGISTER CONTROLS THIS BANK.
7602  012160  004767  006162          65$:  JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 (2)  012164  004767  007460                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (2)  012170  000014                        .WORD   14              ;ERROR TYPE CODE.
7603  012172                          24$:
 (2)  012172  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
7604  012174  010200                        MOV     R2,     R0      ;GET THE ADDRESS POINTER.
7605  012176  042700  003777                BIC     #3777,  R0      ;CLEAR LOW ADDRESS BITS.
7606  012202  000300                        SWAB    R0              ;SHIFT 6 PLACES RIGHT.
7607  012204  006300                        ASL     R0
7608  012206  006300                        ASL     R0
7609  012210  005767  166372                TST     MMAVA           ;CHECK FOR MEM MGMT.
7610  012214  001404                        BEQ     25$             ;BR IF NO MEM MGMT.
7611  012216  042700  177600                BIC     #177600,R0      ;CLEAR BANK BITS
7612  012222  063700  172344                ADD     @#KIPAR2,R0     ;ADD MEM MGMT OFFSET.
7613  012226  052700  100001          25$:  BIS     #BIT15+BIT0,R0  ;SET ERROR AND AE BIT IN CHECK WORD.
7614  012232  016367  000006  167256        MOV     6(R3),  RESRVD  ;GET APPROPIATE MASK.
7615  012240  046700  167252                BIC     RESRVD, R0      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
7616  012244  046701  167246                BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
7617                                  ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
7618  012250  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (2)  012252  001405                        BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (3)  012254  004767  006066          66$:  JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
 (4)  012260  004767  007364                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (4)  012264  000015                        .WORD   15              ;ERROR TYPE CODE.
 (2)  012266                          67$:
7619  012266  005073  000000                CLR     @(R3)           ;CLEAR REG INCLUDING ACTION ENABLE.
7620  012272  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
7621  012274  062703  000010          26$:  ADD     #10,    R3      ;UPDATE POINTER TO NEXT PARITY REG + MAP.
7622  012300  020367  167324                CMP     R3,     .MPRX   ;CHECK FOR END OF TABLE.
7623  012304  101014                        BHI     WWPB3           ;BR IF END OF TABLE REACHED.
```

```
 7624  012306  032713  000001              BIT     #BIT0,  (R3)    ;CHECK IF NEXT REG EXISTS.
 7625  012312  001370                      BNE     26$             ;BR IF THIS PARITY REG DOESN'T EXIST.
 7626  012314  017301  000000              MOV     @(R3),  R1      ;SAVE AND CHECK FOR ERROR FLAG.
 7627  012320  100365                      BPL     26$             ;BR IF NO ERROR FLAG.
 7628  012322  004767  006020      68$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
  (2)  012326  004767  007316              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  012332  000016                      .WORD   16              ;ERROR TYPE CODE.
 7629  012334  000757                      BR      26$             ;BR AFTER ERROR.
 7630  012336  111204              WWPB3:  MOVB    (R2),   R4      ;GET THE DATA FOR CHECKING.
 7631                                      ;* READING THE DATA VIA DATI TO CHECK IT SHOULD CAUSE PARITY ERROR, BUT
 7632                                      ;*  ACTION ENABLE IS NOT SET IN CONTROLING REG, SO NO TRAP SHOULD OCCURE.
 7633  012340  111212                      MOVB    (R2),   (R2)    ;RESTORE RIGHT PARITY
 7634                                      ;NOTE:  THE ABOVE INSTRUCTION CAN BE NOP'ED FOR PROCESSORS
 7635                                      ;       WHICH DO ONLY DATOB TO DESTINATION OF MOVB INSTRUCTIONS.
 7636  012342  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
 7637  012344  017301  000000              MOV     @(R3),  R1      ;READ THE PARITY REGISTER TO CHECK IT AGAIN.
 7638  012350  046701  167142              BIC     RESRVD, R1      ;CLEAR PARITY REG BITS RESERVED FOR FUTURE.
 7639                                      ;NOTE: THE ABOVE INSTRUCTION (2 WORDS) CAN BE NOP'ED FOR UNMIXED MEMORY TYPES.
 7640  012354  042700  000001              BIC     #AE,    R0      ;CLEAR THE ACTION ENABLE BIT IN TEST DATA.
 7641  012360  020001                      CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  012362  001405                      BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  012364  004767  005756      64$:    JSR     PC,     SPRNTP  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  012370  004767  007254              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  012374  000015                      .WORD   15              ;ERROR TYPE CODE.
  (2)  012376                      65$:
 7642  012376  012773  000001  000000      MOV     #1,     @(R3)   ;CLEAR ALL BUT ACTION ENABLE.
 7643  012404  010401                      MOV     R4,     R1      ;GET DATA READ FROM MEMORY FOR TESTING.
 7644  012406  012600                      MOV     (SP)+,R0        ;;POP STACK INTO R0
 7645  012410  120001                      CMPB    R0,     R1      ;CHECK THE DATA.
 7646  012412  001405                      BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (2)  012414  004767  005732      66$:    JSR     PC,     SPRNT0  ;SET UP VALUES FOR ERROR PRINTING.
  (3)  012420  004767  007224              JSR     PC,     $EPROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (3)  012424  000017                      .WORD   17              ;ERROR TYPE CODE.
  (1)  012426                      67$:
 7647  012426  110012              WWPB4:  MOVB    R0,     (R2)    ;RESTORE DATA.
 7648  012430  105712                      TSTB    (R2)            ;DO A DATI TO BE SURE RIGHT PARITY.
 7649  012432  012700  000253              MOV     #253,   R0      ;SET ODD PARITY DATA.
 7650  012436  105167  167116              COMB    OEFLG           ;CHECK IF DONE BOTH ODD AND EVEN PARITY.
 7651  012442  100002                      BPL     27$             ;BR IF DONE BOTH EVEN AND ODD.
 7652  012444  000167  177316              JMP     WWPB2           ;LOOP BACK AND DO ODD(PARITY BIT CLR).
 7653  012450  005202              27$:    INC     R2              ;MOVE POINTER TO NEXT MEMORY BYTE.
 7654  012452  030502              WWPB5:  BIT     R5,     R2      ;CHECK FOR END OF BLOCK.
 7655  012454  001402                      BEQ     30$             ;BR IF END OF BLOCK FOUND.
 7656  012456  000167  177240              JMP     WWPB1           ;LOOP BACK TO TEST NEXT BYTE.
 7657  012462  004767  002526      30$:    JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO WWPBYT
 7658  012466  004767  005054              JSR     PC,     MAMF    ;GO RESET PARITY REGISTERS.
```

```
 7660
  (3)
  (3)
  (2)  012472
  (3)  012472  004567  006142          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  012476  000000                  .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7661  012500  010703          RANTST: MOV     PC,     R3      ;GET CURRENT PROGRAM COUNTER.
 7662  012502  042703  007777          BIC     #7777,  R3      ;POINT TO BEGINNING OF CURRENT 2K BLOCK.
 7663  012506  004467  001724          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7664  012512  010246          1$:     MOV     R2,     -(SP)   ;SAVE MEMORY POINTER.
 7665  012514  010346                  MOV     R3,     -(SP)   ;SAVE 'DATA' POINTER.
 7666  012516  012322          2$:     MOV     (R3)+,  (R2)+   ;MOV CODE INTO TEST MEMORY.
 7667  012520  032703  007777          BIT     #7777,  R3      ;CHECK FOR END OF 'DATA TABLE'
 7668  012524  001002                  BNE     3$              ;BRANCH IF MORE
 7669  012526  162703  010000          SUB     #10000, R3      ;RESET POINTER TO START OF 'RANDOM DATA'
 7670  012532  030502          3$:     BIT     R5,     R2      ;CHECK FOR END OF BLOCK
 7671  012534  001370                  BNE     2$              ;BRANCH IF MORE.
 7672  012536  012603                  MOV     (SP)+,  R3      ;RESET 'DATA' POINTER.
 7673  012540  012602                  MOV     (SP)+,  R2      ;RESET MEMORY POINTER.
 7674  012542  012300          4$:     MOV     (R3)+,  R0      ;GET S/B DATA.
 7675  012544  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  012546  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  012550  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  012552  004767  005620  64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  012556  004767  007066          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  012562  000020                  .WORD   20              ;ERROR TYPE CODE.
  (3)  012564          65$:
 7676  012564  032703  007777          BIT     #7777,  R3      ;CHECK FOR END OF 'DATA TABLE'
 7677  012570  001002                  BNE     5$              ;BR IF MORE.
 7678  012572  162703  010000          SUB     #10000, R3      ;RESET POINTER TO TOP OF 'DATA TABLE'.
 7679  012576          5$:
  (1)  012576  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  012600  001360                  BNE     4$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  012602  004767  002406          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 7680
 7681                           .SBTTL SECTION 3:       INSTRUCTION EXECUTION TESTS.
 7706
  (3)                           ;*TEST 21       EXECUTE DATI, DATO THRU MEMORY.
  (4)                           ;*  EXECUTES THE INSTRUCTION 'MOV R4,(R2)' THROUGHOUT MEMORY.
  (4)                           ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOV' INSTRUCTION TO RETURN
  (4)                           ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
  (4)                           ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
  (4)                           ;*
  (4)                           ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
  (4)                           ;*                LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
  (4)                           ;*
  (4)                           ;*   1ST PASS /   40000         010412          000205
  (4)                           ;*  THRU TEST /   40002         000205          000205
  (4)                           ;*
  (4)                           ;*   2ND PASS /   40002         010412          000205
  (4)                           ;*  THRU TEST /   40004         000205          000205
  (4)                           ;*
  (4)                           ;*      ETC., ETC., ETC.
  (4)                           ;*
  (4)                           ;*  R0 - DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
  (4)                           ;*  R1   DATA READ FROM MEMORY (WAS).
```

```
 (4)                                    ;*  R2 = ADDRESS OF IUT/DATA.
 (4)                                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
 (4)                                    ;*  R4 - RTS R5 (CODE 205).
 (4)                                    ;*  R5 = BLOCK BOUNDRY BIT MASK.
 (3)                                    ;;*****************************************************************
 (2) 012606                             TST21:
 (3) 012606  004567  006026                     JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3) 012612  000003                             .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
 (3)                                                                    ; REQUIRED FOR THIS TEST.
 (3) 012614  000167  000056                     JMP     TST22           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
 (3)                                                                    ; AVAILABLE FOR TEST.
7707 012620  012703  010412             DIDO:   MOV     #010412,R3      ;GET 'MOV R4,(R2)' INSTRUCTION (IUT).
7708 012624  012704  000205                     MOV     #205,   R4      ;GET 'RTS R5'
7709 012630  010400                             MOV     R4,     R0      ;SET UP S/B DATA AFTER EXECUTION.
7710 012632  004467  001600                     JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7711 012636  010322                     1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
7712 012640  010412                     2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
7713 012642  004542                             JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
7714 012644  012201                             MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
7715 012646  020001                             CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (2) 012650  001405                             BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (3) 012652  004767  005514             64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
 (4) 012656  004767  006766                     JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (4) 012662  000021                             .WORD   21              ;ERROR TYPE CODE.
 (2) 012664                             65$:
7716 012664  010322                             MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
7717 012666  030502                             BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
 (1) 012670  001363                             BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
 (1) 012672  004767  002316                     JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 7743
  (3)                    ;;*****************************************************************
  (4)                    ;*TEST 22       EXECUTE DATI, DATOB (LOW BYTE) THRU MEMORY.
  (4)                    ;*  EXECUTES THE INSTRUCTION 'MOVB R4,(R2)' THROUGHOUT MEMORY.
  (4)                    ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
  (4)                    ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
  (4)                    ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
  (4)                    ;*
  (4)                    ;*                MEMORY        INSTRUCTION     CONTENTS OF MEMORY LOCATION
  (4)                    ;*               LOCATION       PLACED THERE    AFTER INSTRUCTION EXECUTION
  (4)                    ;*
  (4)                    ;*   1ST PASS /  40000          110412          110605
  (4)                    ;*  THRU TEST /  40002          000205          000205
  (4)                    ;*
  (4)                    ;*   2ND PASS /  40002          110412          110605
  (4)                    ;*  THRU TEST /  40004          000205          000205
  (4)                    ;*
  (4)                    ;*      ETC., ETC., ETC.
  (4)                    ;*
  (4)                    ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
  (4)                    ;*  R1 = DATA READ FROM MEMORY (WAS).
  (4)                    ;*  R2 = ADDRESS OF IUT/DATA.
  (4)                    ;*  R3 = INSTRUCTION UNDER TEST (IUT).
  (4)                    ;*  R4 = RTS R5 (CODE 205).
  (4)                    ;*  R5 = BLOCK BOUNDRY BIT MASK.
  (3)                    ;;*****************************************************************
  (2)  012676            TST22:
  (3)  012676  004567  005736          JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  012702  000003                  .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
  (3)                                                          ; REQUIRED FOR THIS TEST.
  (3)  012704  000167  000060          JMP     TST23           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
  (3)                                                          ; AVAILABLE FOR TEST.
 7744  012710  012703  110412  DIDBL:  MOV     #110412,R3      ;GET 'MOVB R4,(R2)' INSTRUCTION (IUT).
 7745  012714  012704  000205          MOV     #205,   R4      ;GET 'RTS R5'
 7746  012720  012700  110605          MOV     #110605,R0      ;SET UP S/B DATA AFTER EXECUTION.
 7747  012724  004467  001506          JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7748  012730  010322          1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
 7749  012732  010412          2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
 7750  012734  004542                  JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
 7751  012736  012201                  MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 7752  012740  020001                  CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  012742  001405                  BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  012744  004767  005422  64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  012750  004767  006674          JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  012754  000021                  .WORD   21              ;ERROR TYPE CODE.
  (2)  012756                  65$:
 7753  012756  010322                  MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
 7754  012760  030502                  BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  012762  001363                  BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  012764  004767  002224          JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 7780
  (3)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (3)
  (2) 012770
  (3) 012770  004567  005644
  (3) 012774  000003
  (3)
  (3) 012776  000167  000064
  (3)
 7781 013002  012703  110342
 7782 013006  012704  000205
 7783 013012  012700  161342
 7784 013016  004467  001414
 7785 013022  010322
 7786 013024  010412
 7787 013026  004562  177776
 7788 013032  005302
 7789 013034  012201
 7790 013036  020001
  (2) 013040  001405
  (3) 013042  004767  005324
  (4) 013046  004767  006576
  (4) 013052  000021
  (2) 013054
 7791 013054  010322
 7792 013056  030502
  (1) 013060  001361
  (1) 013062  004767  002126

;;*****************************************************************
;*TEST 23       EXECUTE DATI, DATOB (HIGH BYTE) THRU MEMORY.
;*  EXECUTES THE INSTRUCTION 'MOVB R3,-(R2)' THROUGHOUT MEMORY.
;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'MOVB' INSTRUCTION TO RETURN
;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
;*
;*                MEMORY        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.
;;*****************************************************************
TST23:
        JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
        .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                ; REQUIRED FOR THIS TEST.
        JMP     TST24           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                ; AVAILABLE FOR TEST.
DIDBH:  MOV     #110342,R3      ;GET 'MOVB R3,-(R2)' INSTRUCTION (IUT).
        MOV     #205,   R4      ;GET 'RTS R5'
        MOV     #161342,R0      ;SET UP S/B DATA AFTER EXECUTION.
        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
        JSR     R5,     -2(R2)  ;GO EXECUTE THE IUT.
        DEC     R2              ;ADJUST R2 TO POINT TO MAUT.
        MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD   21              ;ERROR TYPE CODE.
65$:
        MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 7820                                   ;;*******************************************************************
  (3)                                   ;*TEST 24          EXECUTE DATI, DATIP, DATO THRU MEMORY.
  (4)                                   ;*  EXECUTES THE INSTRUCTION 'NEG (R2)' THROUGHOUT MEMORY.
  (4)                                   ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'NEG' INSTRUCTION TO RETURN
  (4)                                   ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
  (4)                                   ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
  (4)                                   ;*
  (4)                                   ;*                 MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
  (4)                                   ;*                LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
  (4)                                   ;*
  (4)                                   ;*   1ST PASS /  40000           005412           172366
  (4)                                   ;*  THRU TEST /  40002           000205           000205
  (4)                                   ;*
  (4)                                   ;*   2ND PASS /  40002           005412           172366
  (4)                                   ;*  THRU TEST /  40004           000205           000205
  (4)                                   ;*
  (4)                                   ;*      ETC., ETC., ETC.
  (4)                                   ;*
  (4)
  (4)                                   ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
  (4)                                   ;*  R1 = DATA READ FROM MEMORY (WAS).
  (4)                                   ;*  R2 = ADDRESS OF IUT/DATA.
  (4)
  (4)                                   ;*  R3 = INSTRUCTION UNDER TEST (IUT).
  (4)                                   ;*  R4 - RTS R5 (CODE 205).
  (4)                                   ;*  R5 = BLOCK BOUNDRY BIT MASK.
  (3)                                   ;;*******************************************************************
  (2)  013066                           TST24:
  (3)  013066  004567  005546                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  013072  000003                           .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
  (3)                                                                   ; REQUIRED FOR THIS TEST.
  (3)  013074  000167  000060                   JMP     TST25           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
  (3)                                                                   ; AVAILABLE FOR TEST.
 7821  013100  012703  005412           DIPDO:  MOV     #005412,R3      ;GET 'NEG (R2)' INSTRUCTION (IUT).
 7822  013104  012704  000205                   MOV     #205,   R4      ;GET 'RTS R5'
 7823  013110  012700  172366                   MOV     #172366,R0      ;SET UP S/B DATA AFTER EXECUTION.
 7824  013114  004467  001316                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7825  013120  010322                   1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
 7826  013122  010412                   2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
 7827  013124  004542                           JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
 7828  013126  012201                           MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
 7829  013130  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  013132  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  013134  004767  005232           64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  013140  004767  006504                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  013144  000021                           .WORD   21              ;ERROR TYPE CODE.
  (2)  013146                           65$:
 7830  013146  010322                           MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
 7831  013150  030502                           BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  013152  001363                           BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  013154  004767  002034                   JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
7857                                        ;;*************************************************************
 (3)                                        ;*TEST 25       EXECUTE DATI, DATI, DATIP, DATOB (LOW BYTE) THRU MEMORY.
 (4)                                        ;*  EXECUTES THE INSTRUCTION BICB (R2)+,-(R2)' THROUGHOUT MEMORY.
 (4)                                        ;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE BICB' INSTRUCTION TO RETURN
 (4)                                        ;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
 (4)                                        ;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
 (4)                                        ;*
 (4)                                        ;*                 MEMORY       INSTRUCTION     CONTENTS OF MEMORY LOCATION
 (4)                                        ;*                LOCATION      PLACED THERE    AFTER INSTRUCTION EXECUTION
 (4)                                        ;*
 (4)                                        ;*   1ST PASS /  40000          142242           142000
 (4)                                        ;*  THRU TEST /  40002          000205           000205
 (4)                                        ;*
 (4)                                        ;*   2ND PASS /  40002          142242           142000
 (4)                                        ;*  THRU TEST /  40004          000205           000205
 (4)                                        ;*
 (4)                                        ;*      ETC., ETC., ETC.
 (4)                                        ;*
 (4)                                        ;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
 (4)                                        ;*  R1 = DATA READ FROM MEMORY (WAS).
 (4)                                        ;*  R2 = ADDRESS OF IUT/DATA.
 (4)                                        ;*  R3 = INSTRUCTION UNDER TEST (IUT).
 (4)                                        ;*  R4 = RTS R5 (CODE 205).
 (4)                                        ;*  R5 = BLOCK BOUNDRY BIT MASK.
 (3)                                        ;;*************************************************************
 (2)  013160                                TST25:
 (3)  013160  004567  005454                        JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (3)  013164  000003                                .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
 (3)                                                                        ; REQUIRED FOR THIS TEST.
 (3)  013166  000167  000060                        JMP     TST26           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
 (3)                                                                        ; AVAILABLE FOR TEST.
7858  013172  012703  142242                DPDBL:  MOV     #142242,R3      ;GET BICB (R2)+,-(R2)' INSTRUCTION (IUT).
7859  013176  012704  000205                        MOV     #205,   R4      ;GET 'RTS R5'
7860  013202  012700  142000                        MOV     #142000,R0      ;SET UP S/B DATA AFTER EXECUTION.
7861  013206  004467  001224                        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
7862  013212  010322                        1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
7863  013214  010412                        2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
7864  013216  004542                                JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
7865  013220  012201                                MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
7866  013222  020001                                CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
 (2)  013224  001405                                BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
 (3)  013226  004767  005140                64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
 (4)  013232  004767  006412                        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
 (4)  013236  000021                                .WORD   21              ;ERROR TYPE CODE.
 (2)  013240                                65$:
7867  013240  010322                                MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
7868  013242  030502                                BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
      013244  001363                                BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
      013246  004767  001742                        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 7894
  (3)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (4)
  (3)
  (2)  013252
  (3)  013252  004567  005362
  (3)  013256  000003
  (3)
  (3)  013260  000167  000062
  (3)
7895   013264  012703  152212
7896   013270  012704  000205
7897   013274  012700  157212
7898   013300  004467  001132
7899   013304  010322
7900   013306  010412
7901   013310  004542
7902   013312  005302
7903   013314  012201
7904   013316  020001
  (2)  013320  001405
  (3)  013322  004767  005044
  (4)  013326  004767  006316
  (4)  013332  000021
  (2)  013334
7905   013334  010322
7906   013336  030502
  (1)  013340  001362
  (1)  013342  004767  001646
```

```
;;*****************************************************************
;*TEST 26       EXECUTE DATI, DATI, DATIP, DATOB (HIGH BYTE) THRU MEMORY.
;*  EXECUTES THE INSTRUCTION 'BISB (R2)+,(R2)' THROUGHOUT MEMORY.
;*  AN 'RTS R5' (CODE 205) IS PLACED AFTER THE 'BISB' INSTRUCTION TO RETURN
;*  CONTROL TO THE MAIN PROGRAM FOR INSTRUCTION EXECUTION CHECKOUT.
;*  THIS IS AN EXAMPLE OF WHAT THIS TEST DOES IN RELATION TO MEMORY:
;*
;*                MEMORY        INSTRUCTION      CONTENTS OF MEMORY LOCATION
;*               LOCATION       PLACED THERE     AFTER INSTRUCTION EXECUTION
;*
;*   1ST PASS /  40000           152212           157212
;*  THRU TEST /  40002           000205           000205
;*
;*   2ND PASS /  40002           152212           157212
;*  THRU TEST /  40004           000205           000205
;*
;*      ETC., ETC., ETC.
;*
;*  R0 = DATA WRITTEN ON TOP OF IUT BY THE IUT (SHOULD BE).
;*  R1 = DATA READ FROM MEMORY (WAS).
;*  R2 = ADDRESS OF IUT/DATA.
;*  R3 = INSTRUCTION UNDER TEST (IUT).
;*  R4 = RTS R5 (CODE 205).
;*  R5 = BLOCK BOUNDRY BIT MASK.
;;*****************************************************************
TST26:
        JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
        .WORD   3               ;MINIMUM BLOCK SIZE OF 2 WORDS
                                ; REQUIRED FOR THIS TEST.
        JMP     TST27           ;SKIP TO NEXT TEST WHEN LESS THAN ONE BLOCK
                                ; AVAILABLE FOR TEST.
DPDBH:  MOV     #152212,R3      ;GET 'BISB (R2)+,(R2)' INSTRUCTION (IUT).
        MOV     #205,   R4      ;GET 'RTS R5'
        MOV     #157212,R0      ;SET UP S/B DATA AFTER EXECUTION.
        JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
1$:     MOV     R3,     (R2)+   ;PUT IUT INTO FIRST LOC OF BLOCK.
2$:     MOV     R4,     (R2)    ;PUT 'RTS R5' FOLLOWING IUT.
        JSR     R5,     -(R2)   ;GO EXECUTE THE IUT.
        DEC     R2              ;RESET R2 TO POINT TO IUT.
        MOV     (R2)+,  R1      ;GET THE DATA FROM THE MEM ADR UNDER TEST.
        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
64$:    JSR     PC,     SPRNT3  ;SET UP VALUES FOR ERROR PRINTING.
        JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
        .WORD   21              ;ERROR TYPE CODE.
65$:
        MOV     R3,     (R2)+   ;PUT THE IUT INTO THE NEXT LOCATION.
        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
        BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
        JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
 7908                                   .SBTTL SECTION 4:MOS TESTS
 7932                                   ;;*****************************************************************
  (3)                                   ;*TEST 27       MARCHING 1'S AND 0'S.
  (4)                                   ;* THIS TEST IS DESIGNED TO STRESS MOS MEMORIES.
  (4)                                   ;* STARTING AT THE BOTTOM ADDRESS AND ADDRESSING UPWARDS A 4K BANK IS
  (4)                                   ;* WRITTEN WITH 000377.THEN STARTING AT THE TOP ADDRESS OF THE BANK THE
  (4)                                   ;* 000377 IS READ,THE BYTES ARE SWAPPED TO 177400 AND THE LOCATION
  (4)                                   ;* REREAD TO CONFIRM THE WRITE.THIS IS REPEATED FOR EVERY LOCATION
  (4)                                   ;* ADDRESSED DOWNWARD UNTIL THE BOTTOM IS REACHED. STARTING AT THE
  (4)                                   ;* BOTTOM EACH LOCATION IS READ FOR 177400,THE BYTES ARE SWAPPED TO
  (4)                                   ;* 000377 AND REREAD TO CONFIRM THE WRITE UNTIL THE TOP ADDRESS OF THE
  (4)                                   ;* BANK IS REACHED. AGAIN STARTING AT THE BOTTOM EACH LOCATION IS READ
  (4)                                   ;* FOR 000377,THE BYTES SWAPPED TO 177400 AND THE LOCATION REREAD TO
  (4)                                   ;* CONFIRM THE WRITE. LASTLY STARTING FROM THE TOP AND ADDRESSING DOWN-
  (4)                                   ;* WARD EACH LOCATION IS READ,THE BYTES SWAPPED TO 000377 AND THE
  (4)                                   ;* LOCATION IS REREAD TO CONFIRM THE WRITE. THIS IS REPEATED FOR EVERY
  (4)                                   ;*4K BANK UNDER TEST.
  (4)                                   ;*
  (4)                                   ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
  (4)                                   ;*      R1=DATA READ FROM MEMORY(WAS)
  (4)                                   ;*      R2=VIRTUAL ADDRESS
  (4)                                   ;*      R3=TIMES THROUGH COUNTER
  (4)                                   ;*      R4=NOT USED
  (4)                                   ;*      R5=BLOCK BOUNDARY BIT MASK.
  (3)                                   ;;*****************************************************************
  (2)  013346                           TST27:
  (3)  013346  004567  005266                   JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  013352  000000                           .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 7933  013354  004467  001056                   JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7934  013360  010267  166230                   MOV     R2,TEMP         ;SAVE BANK STARTING ADDRESS
 7935  013364  005003                   1$:     CLR     R3              ;CLEAR PASS COUNTER
 7936  013366  012700  000377                   MOV     #000377,R0      ;SETUP TO WRITE PATTERN
 7937  013372  010022                   2$:     MOV     R0,(R2)+        ;WRITE PATTERN
 7938  013374  030502                           BIT     R5,R2           ;END OF 4K?
 7939  013376  001375                           BNE     2$              ;CONTINUE WRITING IF NO.
 7940  013400  014201                   3$:     MOV     -(R2),R1        ;GET DATA WRITEN
 7941  013402  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  013404  001405                           BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  013406  004767  004764           64$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  013412  004767  006232                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  013416  000010                           .WORD   10              ;ERROR TYPE CODE.
  (2)  013420                           65$:
 7942  013420  000300                   4$:     SWAB    R0              ;SWAP BYTES OF DATA
 7943  013422  010012                           MOV     R0,(R2) ;WRITE SWAPPED WORD
 7944  013424  011201                           MOV     (R2),R1         ;GET DATA WRITEN
 7945  013426  020001                           CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (2)  013430  001405                           BEQ     67$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (3)  013432  004767  004740           66$:    JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (4)  013436  004767  006206                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (4)  013442  000010                           .WORD   10              ;ERROR TYPE CODE.
  (2)  013444                           67$:
 7946  013444  000300                           SWAB    R0              ;PUT DATA BACK TO ORINGINAL
 7947  013446  005703                           TST     R3              ;IF ON PASS 0 OR PASS 3
 7948  013450  001403                           BEQ     5$              ;WE ARE ADDRESSING DOWN
 7949  013452  020327  000003                   CMP     R3,#3           ;IF ON PASS 1 OR 2 GO TO
 7950  013456  001010                           BNE     6$              ;UPWARD
```

```
      013460  030502           5$:     BIT     R5,R2              ;DONE A PASS?
      013462  001346                   BNE     3$                 ;IF NO CONTINUE
      013464  005203                   INC     R3                 ;IF YES INCREMENT PASS COUNTER
      013466  022703  000004           CMP     #4,R3              ;ARE WE DONE ALL PASSES FOR THIS 4K?
      013472  001427                   BEQ     9$                 ;IF YES BRANCH
      013474  000300                   SWAB    R0                 ;ELSE SET UP NEW READ WORD
      013476  000404                   BR      7$                 ;GO TO START OF ADDRESS UP
      013500  062702  000002   6$:     ADD     #2,R2              ;UPDATE TO NEXT ADDRESS
      013504  030502                   BIT     R5,R2              ;DONE A PASS
      013506  001411                   BEQ     8$                 ;IF YES BRANCH
      013510  011201           7$:     MOV     (R2),R1            ;GET DATA WRITTEN
      013512  020001                   CMP     R0,     R1         ;COMPARE THE CHECK WORD WITH THE DATA READ.
(2)   013514  001405                   BEQ     69$                ;BRANCH OVER ERROR CALL IF GOOD DATA.
(3)   013516  004767  004654   68$:    JSR     PC,     SPRNT2     ;SET UP VALUES FOR ERROR PRINTING.
(4)   013522  004767  006122           JSR     PC,     $ERROR     ;*** ERROR *** (GO TYPE A MESSAGE)
(4)   013526  000010                   .WORD   10                 ;ERROR TYPE CODE.
(2)   013530                   69$:
      013530  000733                   BR      4$
      013532  005203           8$:     INC     R3                 ;INCREMENT      PASS COUNTER
      013534  000300                   SWAB    R0                 ;SET UP NEW READ WORD
      013536  020327  000002           CMP     R3,#2              ;ADDRESSING UP?
      013542  001316                   BNE     3$                 ;IF NO GO TO DOWN SEQUENCE
      013544  016702  166044           MOV     TEMP,R2            ;IF YES RESET ADDRESS TO START
      013550  000757                   BR      7$                 ;GO TO UP SEQUENCE
      013552  004467  000660   9$:     JSR     R4,INITMM          ;INITIALIZE MEMORY ADDRESS POINTERS
      013556  004767  001432           JSR     PC,MMUP            ;UPDATE TO NEW BANK IF EXISTS

                               ;;****************************  ****************************************
(3)                            ;*TEST 30       WRITE CHECKERBOARD STARTING WITH '125252' DATA.
(4)                            ;* THESE TESTS WRITE A CHECKERBOARD THROUGHOUT MEMORY,STALL
(4)                            ;* FOR 2 SECONDS THEN CHECK PATTERN TO VERIFY DATA DID NOT
(4)                            ;* DETERIORATE BETWEEN REFRESH CYCLES.
(4)                            ;*
(4)                            ;*      R0=DATA WRITTEN INTO MEMORY(SHOULD BE)
(4)                            ;*      R1=DATA READ FROM MEMORY(WAS)
(4)                            ;*      R2=VIRTUAL ADDRESS
(4)                            ;*      R3=SMALL LOOP COUNTER FOR STALL
(4)                            ;*      R4=NUMBER OF TIMES SMALL LOOP DONE
(4)                            ;*      R5=BLOCK BOUNDARY BIT MASK.
(3)                            ;;*****************************************************************
(2)   013562                   TST30:
(3)   013562  004567  005052           JSR     R5,     $SCOPE     ;GO TO SCOPE ROUTINE.
(3)   013566  000000                   .WORD   0                  ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
      013570  004467  000642           JSR     R4,     INITMM     ;INITIALIZE THE MEMORY ADDRESS POINTERS.
      013574  012700  125252           MOV     #125252 ,R0        ;SETUP DATA PATTERN
      013600  010022           1$:     MOV     R0      ,(R2)+     ;WRITE A WORD
      013602  005100                   COM     R0                 ;COMPLEMENT DATA
      013604  030502                   BIT     R5,     R2         ;CHECK FOR END OF A BLOCK.
(1)   013606  001374                   BNE     1$                 ;BRANCH IF MORE IN CURRENT BLOCK.
(1)   013610  004767  001400           JSR     PC,     MMUP       ;FIND NEXT BLOCK AND LOOP TO 1$.
      013614  005003                   CLR     R3                 ;SET UP COUNTER FOR STALL
      013616  012704  000046           MOV     #46,    R4         ;DO LOOP 46 TIMES OR 2 SEC. TOTAL.
      013622  005303           2$:     DEC     R3
      013624  001376                   BNE     2$
      013626  005304                   DEC     R4
      013630  001374                   BNE     2$
```

```
 7997  013632  004467  000600                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 7998  013636  012700  125252                MOV     #125252,R0      ;INIT DATA FOR CHECKING
 7999  013642                          3$:
  (1)  013642  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  013644  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  013646  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  013650  004767  004522          64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  013654  004767  005770                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  013660  000006                        .WORD   6               ;ERROR TYPE CODE.
  (3)  013662                          65$:
 8000  013662  005100                        COM     R0
 8001  013664  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  013666  001365                        BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  013670  004767  001320                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 8002                                 ;;*******************************************************************
  (3)                                 ;*TEST 31       WRITE CHECKERBOARD STARTING WITH 052525 DATA
  (3)                                 ;;*******************************************************************
  (2)  013674                         TST31:
  (3)  013674  004567  004740                JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
  (3)  013700  000000                        .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
 8003  013702  004467  000530                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 8004  013706  012700  052525                MOV     #052525,R0      ;SETUP DATA PATTERN
 8005  013712  010022                  1$:   MOV     R0,     (R2)+   ;WRITE A WORD
 8006  013714  005100                        COM     R0
 8007  013716  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  013720  001374                        BNE     1$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  013722  004767  001266                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
 8008  013726  005003                        CLR     R3              ;SET COUNTER FOR LOOP
 8009  013730  012704  000046                MOV     #46,    R4      ;DO LOOP 46 TIMES OR 2 SEC. TOTAL
 8010  013734  005303                  2$:   DEC     R3
 8011  013736  001376                        BNE     2$
 8012  013740  005304                        DEC     R4
 8013  013742  001374                        BNE     2$
 8014  013744  004467  000466                JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
 8015  013750  012700  052525                MOV     #052525,R0      ;INIT PATTERN FOR CHECKING
 8016  013754                          3$:
  (1)  013754  012201                        MOV     (R2)+,  R1      ;GET THE DATA FROM MEMORY UNDER TEST.
  (2)  013756  020001                        CMP     R0,     R1      ;COMPARE THE CHECK WORD WITH THE DATA READ.
  (3)  013760  001405                        BEQ     65$             ;BRANCH OVER ERROR CALL IF GOOD DATA.
  (4)  013762  004767  004410          64$:  JSR     PC,     SPRNT2  ;SET UP VALUES FOR ERROR PRINTING.
  (5)  013766  004767  005656                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (5)  013772  000006                        .WORD   6               ;ERROR TYPE CODE.
  (3)  013774                          65$:
 8017  013774  005100                        COM     R0
 8018  013776  030502                        BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)  014000  001365                        BNE     3$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)  014002  004767  001206                JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
```

```
8020                                        .SBTTL DONE:    RELOCATE PROGRAM AND REPEAT ALL TESTS.
8021  014006                                DONE:
 (1)  014006  004567  004626                        JSR     R5,     $SCOPE  ;GO TO SCOPE ROUTINE.
 (1)  014012  000000                                .WORD   0               ;NO MINIMUM BLOCK SIZE REQUIRED THIS TEST.
8022  014014  005067  165150                TST32:  CLR     $TIMES          ;RESET ITERATION COUNTER FOR RESTARTING TEST.
8023  014020  105067  165056                        CLRB    $TSTNM          ;RESET TEST NUMBER.
8024  014024  036767  164552  165502        1$:     BIT     PRGMAP, SAVTST  ;CHECK IF PROGRAM IS IN TEST AREA.
8025  014032  001004                                BNE     2$              ;BR IF IT PROG IN MEM TO BE TESTED.
8026  014034  036767  164544  165474                BIT     PRGMAP+2,SAVTST+2 ;CHECK HI 64K
8027  014042  001435                                BEQ     $EOP            ;BR IF PROG NOT IN MEM TO BE TESTED.
8028  014044  032777  000200  165066  2$:     BIT     #SW07,  @SWR    ;CHECK FOR INHIBIT RELOCATION SWITCH.
8029  014052  001031                                BNE     $EOP            ;SKIP RELOCATION IF SWITCH SET.
8030  014054  022767  000003  164520                CMP     #3,     PRGMAP  ;CHECK IF PROGRAM IN FIRST 8K.
8031  014062  001013                                BNE     4$              ;BR IF NOT IN FIRST 8K.
8032  014064  023737  000042  000046                CMP     @#42,@#46       ;CHECK FOR ACT11
8033  014072  001416                                BEQ     6$              ;BR IF ACT11.
8034  014074  105737  001224                        TSTB    @#$ENV          ;CHECK FOR APT11
8035  014100  001013                                BNE     6$              ;IF APT11 DO NOT RELOCATE
8036                                                                        ;MUST BE XXDP OR STANDALONE
8037  014102  004767  002362                        JSR     PC,     RELTOP  ;RELOCATE PROGRAM TO TOP OF MEMORY.
8038  014106  000167  172002                3$:     JMP     START1          ;LOOP BACK AND RUN ALL TESTS AGAIN.
8039
8040  014112  004767  002754                4$:     JSR     PC,     RELO    ;RELOCATE PROGRAM BACK TO FIRST 8K.
8041  014116  005737  000042                        TST     @#42            ;TEST FOR XXDP
8042  014122  001402                                BEQ     6$              ;IF NOT RUNNING UNDER MON. DONT
8043  014124  004767  003150                5$:     JSR     PC,     RESLDR  ;RESTORE LOADERS.
8044  014130                                6$:
 (2)  014130  004567  007366                        JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  014134  001201                                .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
```

```
 8046                                   ;;*****************************************************************
  (1)
  (1)                                   .SBTTL  END OF PASS ROUTINE
  (1)
  (1)                                   ;*INCREMENT THE PASS NUMBER ($PASS)
  (1)                                   ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
  (1)                                   ;*IF THERES A MONITOR GO TO IT
  (1)                                   ;*IF THERE ISN'T JUMP TO START1
  (1)
  (1)  014136                           $EOP:
  (2)  014136  000240                           NOP
  (1)  014140  005067  165024                   CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
  (1)  014144  005267  165042                   INC     $PASS           ;;INCREMENT THE PASS NUMBER
  (1)  014150  042767  100000  165034           BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
  (1)  014156  005327                           DEC     (PC)+           ;;LOOP?
  (1)  014160  000001                   $EOPCT: .WORD   1
  (1)  014162  003040                           BGT     $DOAGN          ;;YES
  (1)  014164  012737                           MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
  (1)  014166  000001                   $ENDCT: .WORD   1
  (1)  014170  014160                           $EOPCT
  (2)  014172  004567  007324                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  014176  014270                           .WORD   $ENDMG          ;ADDRESS OF MESSAGE TO BE TYPED
  (2)  014200  016746  165006                   MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
  (4)                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
  (4)                                   ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
  (4)  014204  013746  177776                   MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
  (4)  014210  004767  010226                   JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
  (2)  014214  004567  007302                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  014220  014305                           .WORD   $ENULL          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)  014222                           $GET42:
  (1)
  (1)  014222  016700  163614                   MOV     42,     R0              ;;GET MONITOR ADDRESS
  (1)  014226  001416                           BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
  (1)  014230  000005                           RESET                   ;;CLEAR THE WORLD
  (1)  014232  004710                   $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
  (1)  014234  000240                           NOP                     ;;SAVE ROOM
  (1)  014236  000240                           NOP                     ;;FOR
  (1)  014240  000240                           NOP                     ;;ACT11
  (1)  014242  023737  000042  000046           CMP     @#42,@#46       ;;ARE WE UNDER ACT11 OR XXDP
  (1)  014250  001405                           BEQ     $DOAGN          ;;IF ACT11 THEN RESTART
  (1)  014252  105737  001224                   TSTB    @#$ENV          ;;CHECK FOR APT11
  (1)  014256  001002                           BNE     $DOAGN          ;;IF APT11 THEN RESTART
  (1)  014260  004767  003074                   JSR     PC,     SAVLDR  ;;IF XXDP FIRST SAVE MONITOR
  (1)  014264                           $DOAGN:
  (1)  014264  000167  171624                   JMP     START1          ;;RETURN*****
  (1)  014270  005015  047105  020104   $ENDMG: .ASCIZ  <15><12>/END PASS #/
  (1)  014276  040520  051523  021440
  (1)  014304     000
  (1)  014305     377     377     000   $ENULL: .BYTE   -1,-1,0          ;;NULL CHARACTER STRING
 8047                                   .SBTTL SUBROUTINE AND TRAP ROUTINE SECTION.
 8048                                   .SBTTL  MEMORY MANAGEMENT AND ADDRESSING SUBROUTINES.
 8049                                   ;;*****************************************************************
 8050                                   ;*  SET UP ALL THE MEM MGMT REGISTERS FOR NORMAL OPERATION.
 8051                                   ;*  THE PROGRAM IS POINTED TO BY PARS 0 AND 1.
 8052                                   ;*  THE MEMORY UNDER TEST IS POINTED TO BY PARS 2 AND 3.
 8053                                   ;*  THE DEVICE ADDRESS AREA IS POINTED TO BY PAR 7.
```

```
                                        ;*  PARS 4, 5, AND 6 ARE UNUSED.
                                        ;;*****************************************************************
014310                                  MMINIT:
(1) 014310  012737  077406  172300          MOV     #200-1*400+UP+RW,@#KIPDR0       ;SET KIPDR0 = RW UP 200 BLOCKS
8057 014316 012737  077406  172302          MOV     #200-1*400+UP+RW,@#KIPDR1       ;SET KIPDR1 = RW UP 200 BLOCKS
8058 014324 012737  077406  172304          MOV     #200-1*400+UP+RW,@#KIPDR2       ;SET KIPDR2 = RW UP 200 BLOCKS
8059 014332 012737  077406  172306          MOV     #200-1*400+UP+RW,@#KIPDR3       ;SET KIPDR3 = RW UP 200 BLOCKS
8060 014340 005037  172310                  CLR     @#KIPDR4
8061 014344 005037  172312                  CLR     @#KIPDR5
8062 014350 005037  172314                  CLR     @#KIPDR6
8063 014354 012737  077406  172316          MOV     #200-1*400+UP+RW,@#KIPDR7       ;SET KIPDR7 = RW UP 200 BLOCKS
8064 014362 005037  172340                  CLR     @#KIPAR0        ;MAP PAR0 INTO BANK0
8065 014366 012737  000200  172342          MOV     #200, @#KIPAR1  ;MAP PAR1 INTO BANK1
8066 014374 005037  172344                  CLR     @#KIPAR2        ;MAP PAR2 INTO BANK0
8067 014400 005037  172346                  CLR     @#KIPAR3
8068 014404 005037  172350                  CLR     @#KIPAR4
8069 014410 005037  172352                  CLR     @#KIPAR5
8070 014414 005037  172354                  CLR     @#KIPAR6
8071 014420 012737  007600  172356          MOV     #7600, @#KIPAR7 ;MAP PAR7 INTO I/O BANK
8072 014426 012737  000001  177572          MOV     #1,     @#SR0   ;ENABLE MEMORY MANAGEMENT
8073 014434 000207                          RTS     PC              ;RETURN
8074
8075
8076                                    ;;*****************************************************************
8077                                    ;*  MEMORY ADDRESS POINTER INITIALIZATION ROUTINES.
8078                                    ;;*****************************************************************
8079 014436 012767  000001  165100' INITMM: MOV     #BIT0,  BITPT   ;SET POINTER TO BANK0
8080 014444 005067  165076                  CLR     BITPT+2         ;CLEAR HI 64K BANK POINTERS
8081 014450 005002                          CLR     R2              ;SET ADDRESS POINTER TO 0
8082 014452 016705  165130                  MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
8083 014456 005767  164124                  TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
8084 014462 001514                          BEQ     10$             ;BRANCH IF NO MEM MGMT
8085 014464 005037  172344                  CLR     @#KIPAR2        ;SET UP 3RD PAR TO BANK0
8086 014470 012702  040000                  MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
8087 014474 036767  165044  165026  1$:     BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK TO BE TESTED
8088 014502 001015                          BNE     2$              ;BRANCH IF MATCH
8089 014504 036767  165036  165020          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
8090 014512 001011                          BNE     2$              ;BRANCH IF MATCH
8091 014514 062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
8092 014522 006367  165016                  ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
8093 014526 006167  165014                  ROL     BITPT+2         ;...HI POINTER.
8094 014532 100360                          BPL     1$              ;BR IF MORE.
8095 014534 000000                          HALT                    ;FATAL ERROR!!!  NO 4K BANK FOUND?
8096 014536 036767  165002  165036  2$:     BIT     BITPT,  LADMAP  ;CHECK IF LAST BANK.
8097 014544 001004                          BNE     3$              ;BR IF LAST BANK.
8098 014546 036767  164774  165030          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK.
8099 014554 001405                          BEQ     4$              ;BR IF NOT LAST BANK.
8100 014556 016705  165016          3$:     MOV     LADMSK, R5      ;SET MASK TO FIND LAST ADR.
8101 014562 042767  020000  165006          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2.
8102 014570 013737  172344  172346  4$:     MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
8103 014576 016767  164742  164744          MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
8104 014604 016767  164736  164740          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
8105 014612 032705  020000                  BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
8106 014616 001505                          BEQ     21$             ;BRANCH IF NOT 8K.
8107 014620 062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
8108 014626 006367  164716                  ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
```

```
 8109  014632  006167  164714                ROL     TMPPT+2          ;...HI POINTER.
 8110  014636  100473                        BMI     20$              ;BR IF NO MORE.
 8111  014640  036767  164704  164662        BIT     TMPPT,  TSTMAP   ;CHECK IF BANK TO BE TESTED.
 8112  014646  001004                        BNE     6$               ;BRANCH IF A MATCH.
 8113  014650  036767  164676  164654        BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
 8114  014656  001760                        BEQ     5$               ;BRANCH IF NO MEMORY
 8115  014660  036767  164664  164714  6$:   BIT     TMPPT,  LADMAP   ;CHECK IF LAST BANK.
 8116  014666  001004                        BNE     7$               ;BRANCH IF A MATCH
 8117  014670  036767  164656  164706        BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
 8118  014676  001455                        BEQ     21$              ;BR IF NOT LAST BANK.
 8119  014700  016705  164674          7$:   MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADR.
 8120  014704  052767  020000  164664        BIS     #20000, TMPLAD   ;MAKE SURE LAST ADDRESS IS IN BANK 3.
 8121  014712  000447                        BR      21$              ;BR TO FINISH UP.
 8122
 8123  014714  036767  164624  164606  10$:  BIT     BITPT,  TSTMAP   ;CHECK IF THIS BANK TO BE TESTED.
 8124  014722  001006                        BNE     11$              ;BR IF MATCH.
 8125  014724  062702  020000                ADD     #20000, R2       ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
 8126  014730  106367  164610                ASLB    BITPT            ;UPDATE BANK POINTER TO NEXT BANK.
 8127  014734  100367                        BPL     10$              ;BR IF MORE BANKS.
 8128  014736  000000                        HALT            ;FATAL ERROR!!!  NO 4K BANK FOUND?
 8129  014740  016767  164600  164602  11$:  MOV     BITPT, TMPPT     ;COPY BANK POINTER.
 8130  014746  036767  164572  164626        BIT     BITPT,  LADMAP   ;CHECK IF LAST BANK.
 8131  014754  001021                        BNE     12$              ;BR IF LAST BANK.
 8132  014756  032705  020000                BIT     #BIT13, R5       ;CHECK FOR 8K BLOCK SIZE.
 8133  014762  001423                        BEQ     21$              ;BRANCH IF SMALLER BLOCK SIZE.
 8134  014764  106367  164560                ASLB    TMPPT            ;POINT TO NEXT BANK.
 8135  014770  100416                        BMI     20$              ;BRANCH IF OVERFLOW.
 8136  014772  036767  164552  164530        BIT     TMPPT,  TSTMAP   ;CHECK IF BANK TO BE TESTED.
 8137  015000  001412                        BEQ     20$              ;BRANCH IF NOT TO BE TESTED.
 8138  015002  112767  000011  164547        MOVB    #11,    FLAG8K   ;SET 8K BLOCK SIZE FLAG.
 8139  015010  036767  164534  164564        BIT     TMPPT,  LADMAP   ;CHECK FOR LAST BANK.
 8140  015016  001403                        BEQ     20$              ;BR IF NOT LAST BANK.
 8141  015020  016705  164554          12$:  MOV     LADMSK, R5       ;RESET MASK TO FIND LAST ADR.
 8142  015024  000402                        BR      21$              ;SKIP MASK RESET.
 8143  015026  012705  017777          20$:  MOV     #MASK4K,R5       ;RESET MASK TO 4K BLOCK SIZE.
 8144  015032  056767  164506  164510  21$:  BIS     BITPT,  TMPPT    ;SET TMPPT FOR FLAGING LAST BANK.
 8145  015040  056767  164502  164504        BIS     BITPT+2,TMPPT+2
 8146  015046  036767  164472  164514        BIT     BITPT,  FADMAP   ;CHECK IF FIRST ADDRESS NEEDS TO BE SET.
 8147  015054  001004                        BNE     22$              ;BR IF FIRST BANK.
 8148  015056  036767  164464  164506        BIT     BITPT+2,FADMAP+2 ;CHECK HI 64K.
 8149  015064  001450                        BEQ     INITEX           ;BR IF NOT FIRST BANK.
 8150  015066  016702  164472          22$:  MOV     TMPFAD, R2       ;RESET ADDRESS POINTER TO FIRST ADR.
 8151  015072  000445                        BR      INITEX
 8152
 8153  015074  016705  164506          INITDN: MOV   BLKMSK, R5       ;RESET R5 TO CURRENT BLOCK MASK.
 8154  015100  005002                        CLR     R2               ;INIT ADDRESS POINTER.
 8155  015102  005767  163500                TST     MMAVA            ;CHECK FOR MEM MGMT
 8156  015106  001411                        BEQ     31$              ;BRANCH IF NO MEM MGMT
 8157  015110  012767  100000  164430        MOV     #BIT15, BITPT+2 ;SET POINTER TO TOP BIT
 8158  015116  005067  164422                CLR     BITPT
 8159  015122  012737  007600  172344        MOV     #7600,  @#KIPAR2 ;SET PAR TO TOP OF MEM
 8160  015130  000403                        BR      32$              ;BRANCH TO COMMON AREA
 8161
 8162  015132  012767  000400  164404  31$:  MOV     #BIT8,  BITPT    ;SET UP BANK POINTER
 8163  015140  012767  015162  164406  32$:  MOV     #33$,   MMORE    ;SET ''MMDOWN'' EXIT ADDRESS.
 8164  015146  066767  163426  164400        ADD     RELOCF, MMORE    ;ADD OFFSET
```

```
 8165  015154  004767  000524              JSR     PC,     MMDOWN  ;ROUTINE TO SEARCH DOWNWARD FOR TOP MEM BANK
 8166  015160  000000                      HALT                    ;FATAL ERROR!!!  NO MEM INDICATED IN MEM MAP ABOVE 8K
 8167  015162  036767  164356  164412  33$:    BIT     BITPT,  LADMAP  ;CHECK FOR NON BOUNDRY LAST ADDR.
 8168  015170  001004                      BNE     34$             ;BR IF LAST BANK FLAG FOUND.
 8169  015172  036767  164350  164404          BIT     BITPT+2,LADMAP+2 ;CHECK FOR NON BOUNDRY LAST ADDR.
 8170  015200  001402                      BEQ     INITEX          ;BR IF NO LAD FLG FOUND.
 8171  015202  016702  164366          34$:    MOV     LSTADR, R2      ;SET UP R2.
 8172  015206  010467  164342          INITEX: MOV     R4,     MMORE   ;PUT RETURN PC INTO 'MMORE''
 8173  015212  000204                      RTS     R4              ;RETURN
 8174
 8175                                  ;;*****************************************************************
 8176                                  ;*  COMMON UPWARDS ADDRESSING ROUTINE
 8177                                  ;*  FINDS NEXT EXISTING 4K BANK AND UPDATES POINTERS.
 8178                                  ;*  GOES TO ADDRESS IN 'MMORE'' IF MORE BANKS,
 8179                                  ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
 8180                                  ;;*****************************************************************
 8181  015214  036767  164330  164360  MMUP:   BIT     TMPPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 8182  015222  001122                      BNE     10$             ;BR IF LAST BANK.
 8183  015224  036767  164322  164352          BIT     TMPPT+2,LADMAP+2 ;CHECK FOR LAST BANK FLAG.
 8184  015232  001116                      BNE     10$             ;BR IF LAST BANK.
 8185  015234  016705  164346              MOV     BLKMSK, R5      ;RESET R5 TO BLOCK MASK.
 8186  015240  005767  163342              TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
 8187  015244  001515                      BEQ     20$             ;BRANCH IF NO MEM MGMT
 8188  015246  012702  040000              MOV     #40000, R2      ;RESET VIRTUAL ADR POINTER
 8189  015252  062737  000200  172344  1$:     ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT, THIRD PAR.
 8190  015260  006367  164260              ASL     BITPT           ;UPDATE LO POINTER TO NEXT BANK.
 8191  015264  006167  164256              ROL     BITPT+2         ;...HI POINTER.
 8192  015270  100577                      BMI     32$             ;BR IF ALL DONE.
 8193  015272  036767  164246  164230          BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK EXISTS
 8194  015300  001004                      BNE     2$              ;BRANCH IF MATCH
 8195  015302  036767  164240  164222          BIT     BITPT+2,TSTMAP+2 ;CHECK IN HI MAP
 8196  015310  001760                      BEQ     1$              ;BRANCH IF NO MATCH
 8197  015312  036767  164226  164262  2$:     BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 8198  015320  001004                      BNE     3$              ;BRANCH IF LAST BANK FLAG.
 8199  015322  036767  164220  164254          BIT     BITPT+2,LADMAP+2 ;CHECK IF LAST BANK FLAG.
 8200  015330  001405                      BEQ     4$              ;BR IF NOT LAST BANK.
 8201  015332  016705  164242          3$:     MOV     LADMSK, R5      ;RESET MASK.
 8202  015336  042767  020000  164232          BIC     #20000, TMPLAD  ;MAKE SURE VIRTUAL LAST ADR IN BANK 2
 8203  015344  016767  164174  164176  4$:     MOV     BITPT,  TMPPT   ;COPY BITPT...LO 64K.
 8204  015352  016767  164170  164172          MOV     BITPT+2,TMPPT+2 ;...HI 64K.
 8205  015360  032705  020000              BIT     #BIT13, R5      ;CHECK FOR A BLOCK SIZE OF 8K.
 8206  015364  001530                      BEQ     31$             ;BRANCH IF NOT.
 8207  015366  013737  172344  172346          MOV     @#KIPAR2,@#KIPAR3 ;COPY CURRENT PAR INTO FORTH PAR.
 8208  015374  062737  000200  172346  5$:     ADD     #200,   @#KIPAR3 ;UP DATE FORTH PAR.
 8209  015402  006367  164142              ASL     TMPPT           ;UPDATE LO POINTER TO NEXT 4K BANK.
 8210  015406  006167  164140              ROL     TMPPT+2         ;...HI POINTER.
 8211  015412  100513                      BMI     30$             ;BR IF NO MORE.
 8212  015414  036767  164130  164106  6$:     BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
 8213  015422  001004                      BNE     7$              ;BRANCH IF A MATCH.
 8214  015424  036767  164122  164100          BIT     TMPPT+2,TSTMAP+2 ;CHECK FOR HI 64K BANKS.
 8215  015432  001760                      BEQ     5$              ;BRANCH IF NO MEMORY
 8216  015434  036767  164110  164140  7$:     BIT     TMPPT,LADMAP    ;CHECK FOR LAST BANK FLAG.
 8217  015442  001004                      BNE     8$              ;BRANCH IF A MATCH
 8218  015444  036767  164102  164132          BIT     TMPPT+2,LADMAP+2 ;CHECK HI 64K
 8219  015452  001475                      BEQ     31$             ;BR IF NO LAST BANK FLAG.
 8220  015454  016705  164120          8$:     MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADDRESS.
```

```
 8221  015460  052767  020000  164110          BIS     #20000, TMPLAD  ;SET VIRTUAL ADR TO BANK 3.
 8222  015466  000467                          BR      31$
 8223
 8224  015470  026702  164102          10$:    CMP     TMPLAD, R2      ;CHECK IF LAST ADR REACHED.
 8225  015474  001064                          BNE     31$             ;BR IF MORE.
 8226  015476  000474                          BR      32$             ;BR IF ALL DONE.
 8227
 8228  015500  106267  164053          20$:    ASRB    FLAG8K          ;SHIFT 8K FLAG
 8229  015504  001407                          BEQ     22$             ;BR IF NOT 8K BLOCK.
 8230  015506  103455                          BCS     30$             ;BR IF ANOTHER 4K.
 8231  015510  105067  164043                  CLRB    FLAG8K          ;CLEAR OUT ALL FLAGS.
 8232  015514  162702  040000                  SUB     #40000, R2      ;BACK UP 8K.
 8233  015520  062702  020000          21$:    ADD     #20000, R2      ;UPDATE PHYSICAL ADR PNTR TO NEXT BANK.
 8234  015524  106367  164014          22$:    ASLB    BITPT           ;UPDATE POINTER.
 8235  015530  100457                          BMI     32$             ;BRANCH WHEN END IS REACHED.
 8236  015532  036767  164006  163770          BIT     BITPT,  TSTMAP  ;CHECK IF THIS BANK EXISTS.
 8237  015540  001767                          BEQ     21$             ;BRANCH IF NO MATCH.
 8238  015542  036767  163776  164032          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 8239  015550  001402                          BEQ     23$             ;BR IF NO MATCH.
 8240  015552  016705  164022                  MOV     LADMSK, R5      ;RESET MASK TO FIND LAST ADR.
 8241  015556  016767  163762  163764  23$:    MOV     BITPT,  TMPPT   ;SET UP TMP POINTER.
 8242  015564  032705  020000                  BIT     #BIT13, R5      ;CHECK FOR 8K BLOCK SIZE.
 8243  015570  001426                          BEQ     31$             ;BRANCH IF SMALLER BLOCK SIZE.
 8244  015572  106367  163752                  ASLB    TMPPT           ;POINT TO NEXT BANK.
 8245  015576  100421                          BMI     30$             ;BRANCH IF OVERFLOW.
 8246  015600  036767  163744  163722          BIT     TMPPT,  TSTMAP  ;CHECK IF BANK TO BE TESTED.
 8247  015606  001415                          BEQ     30$             ;BRANCH IF NOT TO BE TESTED.
 8248  015610  036767  163730  163764          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 8249  015616  112767  000011  163733          MOVB    #11,    FLAG8K  ;SET 8K BLOCK FLAG.
 8250  015624  036767  163714  163750          BIT     BITPT,  LADMAP  ;CHECK FOR LAST BANK FLAG.
 8251  015632  001403                          BEQ     30$             ;BR IF NO FLAG.
 8252  015634  016705  163740                  MOV     LADMSK, R5      ;RESET MASK TOFIND LAST ADR.
 8253  015640  000402                          BR      31$
 8254  015642  012705  017777          30$:    MOV     #MASK4K,R5      ;SET MASK TO 4K.
 8255  015646  056767  163672  163674  31$:    BIS     BITPT,  TMPPT   ;SET TMPPT FOR FINDING LAST ADR.
 8256  015654  056767  163666  163670          BIS     BITPT+2,TMPPT+2
 8257  015662  016716  163666                  MOV     MMORE,  (SP)    ;FUDGE RETURN ADDRESS TO LOOP.
 8258  015666  000207                          RTS     PC              ;RETURN
 8259                                  ;*  BEFORE FINAL EXIT, CHECK FOR ANY NON-TRAP PARITY ERRORS.
 8260  015670  005767  164402          32$:    TST     MPRX            ;CHECK FOR ANY PARITY REGISTERS PRESENT.
 8261  015674  001402                          BEQ     33$             ;BR IF NONE.
 8262  015676  004767  001744                  JSR     PC,     CKPMER  ;CHECK FOR PARITY MEMORY ERRORS.
 8263  015702  000207                  33$:    RTS     PC              ;STRAIGHT RETURN.
 8264
 8265                                  ;;*****************************************************************
 8266                                  ;*  MEMORY DOWNWARDS ADDRESSING SUBROUTINE.
 8267                                  ;*  FINDS NEXT LOWER 4K BANK AND UPDATES POINTERS.
 8268                                  ;*  GOES TO ADDRESS IN "MMORE" IF MORE BANKS,
 8269                                  ;*  DOES STRAIGHT EXIT WHEN ALL MEMORY HAS BEEN DONE.
 8270                                  ;;*****************************************************************
 8271  015704  036767  163634  163656  MMDOWN: BIT     BITPT,  FADMAP  ;CHECK FOR FIRST ADR FLAG.
 8272  015712  001004                          BNE     1$              ;BR IF FIRST ADR IN THIS BANK.
 8273  015714  036767  163626  163650          BIT     BITPT+2,FADMAP+2 ;CHECK FOR FIRST ADR FLAG.
 8274  015722  001404                          BEQ     2$              ;BR IF NO FLAG
 8275  015724  026702  163634          1$:     CMP     TMPFAD, R2      ;CHECK IF FIRST ADDRESS REACHED.
 8276  015730  001052                          BNE     9$              ;BR IF MORE.
```

```
 8277  015732  000453                          BR      10$             ;BR IF ALL DONE.
 8278  015734  005767  162646                  2$:     TST     MMAVA           ;CHECK IF MEM MGMT IS AVAILABLE
 8279  015740  001425                          BEQ     6$              ;BRANCH IF NOT
 8280  015742  162737  000200  172344  3$:     SUB     #200, @#KIPAR2  ;LOWER MEM MGMT PAR BY 4K
 8281  015750  006067  163572                  ROR     BITPT+2         ;MOV POINTER TO NEXT LOWER BANK...HI MAP.
 8282  015754  006067  163564                  ROR     BITPT           ;...LO MAP.
 8283  015760  103440                          BCS     10$             ;BR IF NO MORE.
 8284  015762  036767  163556  163540          BIT     BITPT,  TSTMAP  ;CHECK FOR BANK EXISTING
 8285  015770  001004                          BNE     4$              ;BR IF BANK TO BE TESTED.
 8286  015772  036767  163550  163532          BIT     BITPT+2,TSTMAP+2 ;CHECK FOR BANK IN HI MAP.
 8287  016000  001760                          BEQ     3$              ;BR IF NOT THERE.
 8288  016002  012702  060000                  4$:     MOV     #60000, R2      ;SET ADR POINTER TO TOP OF BANK
 8289  016006  000411                          BR      7$              ;GO TO COMMON EXIT
 8290  016010  162702  020000                  5$:     SUB     #20000, R2      ;BACK POINTER DOWN ONE BANK
 8291  016014  006267  163524                  6$:     ASR     BITPT           ;MOVE POINTER TO NEXT LOWER BANK
 8292  016020  103420                          BCS     10$             ;BRANCH TO EXIT IF NO MORE  MEM
 8293  016022  036767  163516  163500          BIT     BITPT,  TSTMAP  ;CHECK IF BANK EXISTS
 8294  016030  001767                          BEQ     5$              ;BRANCH IF BANK DOESN'T EXIST
 8295  016032  036767  163506  163530  7$:     BIT     BITPT,  FADMAP  ;CHECK IF FIRST BANK FLAG.
 8296  016040  001004                          BNE     8$              ;BR IF FIRST BANK.
 8297  016042  036767  163500  163522          BIT     BITPT+2,FADMAP+2 ;CHECK IF FIRST BANK FLAG.
 8298  016050  001402                          BEQ     9$              ;BR IF NO FLAG FOUND.
 8299  016052  016705  163510                  8$:     MOV     FADMSK, R5      ;SET UP R5 TO FIND FIRST ADDRESS.
 8300  016056  016716  163472                  9$:     MOV     MMORE,  (SP)    ;RESET RETURN ADDRESS
 8301  016062  000207                          10$:    RTS     PC              ;RETURN
```

```
                        .SBTTL  SUBRCUTINES FOR ADDRESS AND WORSE CASE NOISE TESTS.
                        ;;*****************************************************************
                        ;*  SUBROUTINE TO CALCULATE PHYSICAL ADDRESS AND PUT IT IN R0 (BOTTOM 16 BITS).
                        ;*  BITS 16 AND 17 ARE IN $TMP0.
                        ;;*****************************************************************
016064  010200         PHYADR: MOV     R2,     R0      ;VITRUAL INTO R0
016066  005067  163066         CLR     $TMP0           ;CLEAR TEMP SAVE OF HIGH BITS
016072  005767  162510         TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
016076  001417                 BEQ     1$              ;BRANCH IF NO MEM MGMT
016100  010146                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
016102  013701  172344         MOV     @#KIPAR2, R1    ;GET PAR TO BE ADDED TO VIRTUAL
016106  006301                 ASL     R1              ;SHIFT IT 6 TIMES
016110  006301                 ASL     R1
016112  006301                 ASL     R1
016114  006301                 ASL     R1
016116  006301                 ASL     R1
016120  006167  163034         ROL     $TMP0           ;SAVE EXTRA BITS
016124  006301                 ASL     R1
016126  006167  163026         ROL     $TMP0
016132  060100                 ADD     R1,     R0      ;ADD SHIFTED PAR TO VIRTUAL
016134  012601                 MOV     (SP)+,R1        ;;POP STACK INTO R1
016136  000207         1$:     RTS     PC              ;RETURN

                        ;;*****************************************************************
                        ;*  SUBROUTINE TO PUT BANK NUMBER INTO R0.
                        ;;*****************************************************************
016140  005000         BANKNO: CLR     R0              ;INIT R0
016142  010146                 MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (2)   016144  010246         MOV     R2,-(SP)        ;;PUSH R2 ON STACK
016146  016701  163372         MOV     BITPT,  R1      ;GET BANK MAP POINTER...LO 64K.
016152  016702  163370         MOV     BITPT+2,R2      ;...HI 64K.
016156  006202         1$:     ASR     R2              ;SHIFT POINTER...HI
016160  006001                 ROR     R1              ;...LO
016162  103403                 BCS     2$              ;BR WHEN POINTER FOUND.
016164  105200                 INCB    R0              ;COUNT BANKS.
016166  100373                 BPL     1$              ;BR IF NOT OVERFLOW.
016170  000000                 HALT            ;FATAL ERROR!!!  NO POINTER FOUND.
016172         2$:
  (2)   016172  012602         MOV     (SP)+,R2        ;;POP STACK INTO R2
  (2)   016174  012601         MOV     (SP)+,R1        ;;POP STACK INTO R1
016176  000207                 RTS     PC              ;RETURN

                        ;;*****************************************************************
                        ;*  SUBROUTINE TO WRITE THE CONSTANT IN R0 INTO ALL OF MEMORY.
                        ;;*****************************************************************
016200         SETCON:
  (1)   016200  004467  176232         JSR     R4,     INITMM  ;INITIALIZE THE MEMORY ADDRESS POINTERS.
016204  010022         2$:     MOV     R0,     (R2)+   ;;MOV CONSTANT INTO MEMORY
016206  030502                 BIT     R5,     R2      ;CHECK FOR END OF A BLOCK.
  (1)   016210  001375         BNE     2$              ;BRANCH IF MORE IN CURRENT BLOCK.
  (1)   016212  004767  176776         JSR     PC,     MMUP    ;FIND NEXT BLOCK AND LOOP TO 1$.
016216  000207                 RTS     PC              ;RETURN
```

```
                                ;;*****************************************************************
                                ;* ROUTINE TO ROTATE 'C' BIT THROUGH A MEMORY LOCATION.
                                ;;*****************************************************************
016220  106112          ROTATE: ROLB    (R2)            ;(R2)=177776 OR 000001
016222  106112                  ROLB    (R2)            ;(R2)=177775 OR 000002
016224  106112                  ROLB    (R2)            ;(R2)=177773 OR 000004
016226  106112                  ROLB    (R2)            ;(R2)=177767 OR 000010
016230  106112                  ROLB    (R2)            ;(R2)=177757 OR 000020
016232  106112                  ROLB    (R2)            ;(R2)=177737 OR 000040
016234  106112                  ROLB    (R2)            ;(R2)=177677 OR 000100
016236  106112                  ROLB    (R2)            ;(R2)=177777 OR 000000
016240  106122                  ROLB    (R2)+           ;(R2)=177577 OR 000200
016242  106112                  ROLB    (R2)            ;(R2)=177377 OR 000400
016244  106112                  ROLB    (R2)            ;(R2)=176777 OR 001000
016246  106112                  ROLB    (R2)            ;(R2)=175777 OR 002000
016250  106112                  ROLB    (R2)            ;(R2)=173777 OR 004000
016252  106112                  ROLB    (R2)            ;(R2)=167777 OR 010000
016254  106112                  ROLB    (R2)            ;(R2)=157777 OR 020000
016256  106112                  ROLB    (R2)            ;(R2)=137777 OR 040000
016260  106112                  ROLB    (R2)            ;(R2)=077777 OR 100000
016262  106122                  ROLB    (R2)+           ;(R2)=177777 OR 000000
016264  000207                  RTS     PC              ;RETURN

                                ;;*****************************************************************
                                ;*  SUBROUTINE TO WRITE 3 XOR 9 PATTERN INTO 256. WORD BLOCK.
                                ;;*****************************************************************
016266  012704  000020  W3X9:   MOV     #16.,R4         ;EACH LOOP WRITES 256. WORDS

016272  010022          2$:     MOV     R0,(R2)+
016274  010022                  MOV     R0,(R2)+
016276  010022                  MOV     R0,(R2)+
016300  010022                  MOV     R0,(R2)+

016302  010322                  MOV     R3,(R2)+
016304  010322                  MOV     R3,(R2)+
016306  010322                  MOV     R3,(R2)+
016310  010322                  MOV     R3,(R2)+

016312  010022                  MOV     R0,(R2)+
016314  010022                  MOV     R0,(R2)+
016316  010022                  MOV     R0,(R2)+
016320  010022                  MOV     R0,(R2)+

016322  010322                  MOV     R3,(R2)+
016324  010322                  MOV     R3,(R2)+
016326  010322                  MOV     R3,(R2)+
016330  010322                  MOV     R3,(R2)+

016332  005304                  DEC     R4
016334  001356                  BNE     2$
016336  010046                  MOV     R0,     -(SP)   ;SAVE R0
  (1)   010300                  MOV     R3,     R0      ;PUT R3 INTO R0
  (1)   012603                  MOV     (SP)+,  R3      ;PUT SAVED R0 INTO R3
016344  000207                  RTS     PC              ;RETURN
```

```
 8403                                   .SBTTL  RELOCATION SUBROUTINES.
 8404                                   ;;*****************************************************************
 8405                                   ;* ROUTINE TO RELOCATE PROGRAM CODE
 8406                                   ;;*****************************************************************
 8407  016346                           RELOC:
  (2)  016346  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (2)  016350  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (2)  016352  010446                           MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 8408  016354  012502                   4$:     MOV     (R5)+,  R2      ;GET FIRST LOCATION.
 8409  016356  012503                           MOV     (R5)+,  R3      ;GET FIRST LOCATION OF DESTINATION.
 8410  016360  012704  020000                   MOV     #20000, R4      ;SET UP 8K COUNTER.
 8411  016364  012223                   1$:     MOV     (R2)+,  (R3)+   ;MOV THE DATA.
 8412  016366  005304                           DEC     R4              ;COUNT THE WORDS.
 8413  016370  001375                           BNE     1$              ;BR IF MORE.
 8414  016372  012704  020000                   MOV     #20000, R4      ;RESET THE COUNTER.
 8415  016376  024243                   2$:     CMP     -(R2),  -(R3)   ;CHECK THE DATA JUST MOVED.
 8416  016400  001417                           BEQ     3$              ;BR IF DATA OK.
 8417  016402  011267  162516                   MOV     (R2),   $GDDAT  ;GET SOURCE DATA.
 8418  016406  011367  162514                   MOV     (R3),   $BDDAT  ;GET DESTINATION DATA.
 8419  016412  010267  162502                   MOV     R2,     $GDADR  ;GET SOURCE ADDRESS.
 8420  016416  010367  162500                   MOV     R3,     $BDADR  ;GET DESTINATION ADDRESS.
 8421  016422  004767  003222                   JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (1)  016426  000023                           .WORD   23              ;ERROR TYPE CODE.
 8422  016430  000000                           HALT                    ;FATAL ERROR!!!  RELOCATION FAILED.
 8423  016432  162705  000004                   SUB     #4,     R5      ;ADJUST RETURN POINTER.
 8424  016436  000746                           BR      4$              ;GO BACK AND TRY AGAIN.
 8425  016440  005304                   3$:     DEC     R4              ;COUNT WORDS.
 8426  016442  001355                           BNE     2$              ;BR IF MORE.
 8427  016444  004567  005052                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  016450  026552                           .WORD   PRELOC          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                                   ;'PROGRAM RELOCATED TO ''
 8428  016452  010346                           MOV     R3,     -(SP)   ;PUT THE DATA ON THE STACK.
  (1)  016454  004767  006502                   JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
 8429  016460  012604                           MOV     (SP)+,R4        ;;POP STACK INTO R4
  (2)  016462  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
  (2)  016464  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
 8430  016466  000205                           RTS     R5              ;RETURN
 8431                                   ;;*****************************************************************
 8432                                   ;*  SUBROUTINE TO MOVE PROGRAM FROM BOTTOM TO TOP OF MEMORY.
 8433                                   ;;*****************************************************************
 8434  016470  022767  000003  162104   RELTOP: CMP     #3,     PRGMAP  ;CHECK THAT THE PROGRAM IS NOW IN BANKS 0 AND 1.
 8435  016476  001401                           BEQ     1$              ;BR IF OK
 8436  016500  000000                           HALT                    ;FATAL ERROR!!!  PROG SHOULD BE IN BANKS 0 AND 1
 8437  016502                           1$:
  (2)  016502  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (2)  016504  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8438  016506  005767  162074                   TST     MMAVA
 8439  016512  001465                           BEQ     10$
 8440  016514  012737  007600  172346           MOV     #7600,  @#KIPAR3 ;SET PAR TO TOP OF MEM
 8441  016522  005000                           CLR     R0              ;INIT BANK POINTER...LO 64K
 8442  016524  012701  100000                   MOV     #BIT15, R1      ;...HI 64K.
 8443  016530  162737  000200  172346   2$:     SUB     #200,   @#KIPAR3        ;BACK DOWN ONE BANK.
 8444  016536  006001                           ROR     R1              ;MOVE POINTER...HI 64K.
 8445  016540  006000                           ROR     R0              ;..LO 64K.
 8446  016542  103500                           BCS     90$
 8447  016544  030167  162756                   BIT     R1,     MEMMAP+2 ;CHECK FOR BANK EXISTS.
```

```
 8448  016550  001003                          BNE     3$              ;BR IF AVAILABLE
 8449  016552  030067  162746                  BIT     R0,     MEMMAP  ;CHECK FOR BANK EXISTS.
 8450  016556  001764                          BEQ     2$              ;BR IF NO BANK FOUND.
 8451  016560  013737  172346  172344  3$:     MOV     @#KIPAR3,@#KIPAR2 ;COPY PAR
 8452  016566  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
   (2) 016570  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8453  016572  162737  000200  172344  4$:     SUB     #200,   @#KIPAR2 ;BACK DOWN WITH LOW PAR.
 8454  016600  006001                          ROR     R1              ;SHIFT POINTER.
 8455  016602  006000                          ROR     R0              ;...LO 64K.
 8456  016604  103457                          BCS     90$             ;BR IF OVERFLOW.
 8457  016606  030167  162714          5$:     BIT     R1,     MEMMAP+2 ;      CHECK IF BANK EXISTS...HI 64K.
 8458  016612  001003                          BNE     6$              ;BR IF BANK EXISTS.
 8459  016614  030067  162704                  BIT     R0,     MEMMAP  ;CHECK IF BANK EXISTS...LO 64K.
 8460  016620  001764                          BEQ     4$              ;BR IF BANK DOESN'T EXIST.
 8461  016622  052601                  6$:     BIS     (SP)+,  R1      ;GET SECOND BANK POINTER.
 8462  016624  052600                          BIS     (SP)+,  R0      ;...LO 64K.
 8463  016626  030067  161750                  BIT     R0,     PRGMAP  ;CHECK FOR CONFLICT.
 8464  016632  001044                          BNE     90$             ;ABORT IF DESTINATION OVERLAYS SOURCE.
 8465  016634  004567  177506                  JSR     R5,     RELOC   ;GO RELOCATE PROGRAM.
 8466  016640  000000                          .WORD   0               ;SOURCE FIRST ADDRESS.
 8467  016642  040000                          .WORD   40000           ;DESTINATION FIRST ADDRESS.
 8468  016644  013737  172344  172340          MOV     @#KIPAR2,@#KIPAR0 ;RELOCATE LO BANK
 8469  016652  013737  172346  172342          MOV     @#KIPAR3,@#KIPAR1 ;RELOCATE HI BANK.
 8470                          ;*  PROGRAM SHOULD NOW BE EXICUTING OUT OF LAST TWO BANKS OF MEMORY.
 8471  016660  010167  161720                  MOV     R1,     PRGMAP+2 ;RESET PROGRAM MAP.
 8472  016664  000473                          BR      30$             ;BR TO COMMON EXIT.
 8473
 8474  016666  012700  000400          10$:    MOV     #BIT8,  R0      ;SET BANK POINTER TO TOP OF MEM.
 8475  016672  005001                          CLR     R1              ;SET ADDRESS POINTER TO TOP.
 8476  016674  162701  020000          11$:    SUB     #20000, R1      ;BACK DOWN ONE BANK.
 8477  016700  006200                          ASR     R0              ;MOVE POINTER DOWN ONE BANK.
 8478  016702  103420                          BCS     90$             ;BR IF OVERFLOW.
 8479  016704  030067  162614                  BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
 8480  016710  001771                          BEQ     11$             ;BR IF NON-EXISTANT BANK.
 8481  016712  162701  020000                  SUB     #20000, R1      ;BACK DOWN TO NEXT BANK.
 8482  016716  006200                          ASR     R0              ;MOV POINTER DOWN ONE BANK.
 8483  016720  103411                          BCS     90$             ;BR IF OVERFLOW.
 8484  016722  030067  162576                  BIT     R0,     MEMMAP  ;CHECK IF THIS BANK EXISTS.
 8485  016726  001762                          BEQ     11$             ;BR TO START OVER IF NO LOWER BANK.
 8486  016730  010046                          MOV     R0,     -(SP)   ;SAVE THE POINTER.
 8487  016732  006300                          ASL     R0              ;RESET POINTER TO HI BANK.
 8488  016734  052600                          BIS     (SP)+,  R0      ;SET BIT FOR LO BANK.
 8489  016736  030067  161640                  BIT     R0,     PRGMAP  ;CHECK FOR A PROGRAM CONFLICT.
 8490  016742  001401                          BEQ     12$             ;BR IF NO CONFLICT.
 8491  016744                          90$:
   (1) 016744  000000                          HALT                    ;FATAL ERROR!!!  NOT ENOUGH MEMORY??
 8492  016746  010167  000006          12$:    MOV     R1,     13$     ;SET DATA FOR RELOCATION SUBROUTINE.
 8493  016752  004567  177370                  JSR     R5,     RELOC   ;GO RELOCATE THE PROGRAM TO TOP OF MEM.
 8494  016756  000000                          .WORD   0               ;SOURCE STARTING ADDRESS.
 8495  016760  000000                  13$:    .WORD   0               ;DESTINATION STARTING ADDRESS.
 8496  016762  010167  161612                  MOV     R1,     RELOCF  ;SET RELOCATION FACTOR IN UNRELOCATED CODE.
 8497  016766  060107                          ADD     R1,     PC      ;JUMP TO RELOCATED PROGRAM
 8498                          ;*  PROGRAM NOW EXICUTING OUT OF TOP OF MEMORY.
 8499  016770  060106                          ADD     R1,     SP      ;ADJUST THE STACK POINTER TO TOP OF MEMORY.
 8500  016772  010167  161602                  MOV     R1,     RELOCF  ;SET THE RELOCATION FACTOR.
 8501  016776  060137  000004                  ADD     R1,     @#ERRVEC ;ADJUST ERROR VECTOR.
```

```
 8502  017002  060137  000024            ADD     R1,     @#PWRVEC ;ADJUST POWER FAIL VECTOR.
 8503  017006  060137  000114            ADD     R1,     @#PARVEC ;ADJUST PARITY ERROR VECTOR.
 8504  017012  026727  162122  177570    CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
 8505  017020  001404                    BEQ     14$             ;BR IF HARDWARE SWITCH REGISTER.
 8506  017022  060167  162112            ADD     R1,     SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
 8507  017026  060167  162110            ADD     R1,     DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
 8508  017032  062701  001622    14$:    ADD     #RADTAB,R1      ;POINT TO THE RELATIVE RELOCATION TABLE.
 8509  017036  066721  161536    15$:    ADD     RELOCF, (R1)+   ;ADD RELOCATION FACTOR TO ADDRESSES IN TABLE.
 8510  017042  005721            16$:    TST     (R1)+           ;CHECK FOR INTERUM TERMINATOR.
 8511  017044  001776                    BEQ     16$             ;BR SO AS TO NOT MODIFY ZERO.
 8512  017046  024127  177777            CMP     -(R1),  #-1     ;CHECK FOR END OF TABLE.
 8513  017052  001371                    BNE     15$             ;BR IF MORE IN TABLE.
 8514  017054  010067  161522    30$:    MOV     R0,     PRGMAP  ;SET NEW PROGRAM MAP...LO 64K.
 8515  017060  012601                    MOV     (SP)+,R1        ;;POP STACK INTO R1
  (2)  017062  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
 8516  017064  066716  161510            ADD     RELOCF, (SP)    ;ADJUST RETURN ADDRESS.
 8517  017070  000207                    RTS     PC              ;RETURN
 8518
 8519                                    ;;*****************************************************************
 8520                                    ;*  SUBROUTINE TO RELOCATE PROGRAM BACK TO BANKS 0 AND 1.
 8521                                    ;;*****************************************************************
 8522  017072  032767  000003  161502    RELO:   BIT     #3,     PRGMAP  ;CHECK FOR PROGRAM ALREADY IN BANKS 0 OR 1.
 8523  017100  001401                    BEQ     1$              ;BR IF NO CONFLICT.
 8524  017102  000000                    HALT                    ;FATAL ERROR!!!  PROGRAM ALREADY IN BANKS 0 OR 1
 8525  017104  005767  161476    1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
 8526  017110  001417                    BEQ     10$             ;BR IF NO MEMMGMT.
 8527  017112  005037  172344            CLR     @#KIPAR2        ;SET PAR 2 TO BANK 0.
 8528  017116  012737  000200  172346    MOV     #200,   @#KIPAR3 ;SET PAR 3 TO BANK 1.
 8529  017124  004567  177216            JSR     R5,     F_LOC   ;GO MOVE 8K INTO BANKS 0 AND 1.
 8530  017130  000000                    .WORD   0               ;SOURCE STARTING ADDRESS.
 8531  017132  040000                    .WORD   40000           ;DESTINATION STARTING ADDRESS.
 8532  017134  005037  172340            CLR     @#KIPAR0        ;RESTORE PAR 0 TO BANK0.
 8533  017140  012737  000200  172342    MOV     #200,   @#KIPAR1 ;RESTORE PAR 1 TO BANK 1.
 8534                                    ;*  PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
 8535  017146  000444                    BR      30$             ;BR TO COMMON EXIT.
 8536
 8537  017150  016746  161424    10$:    MOV     RELOCF, -(SP)   ;PUT RELOCATION FACTOR ONTO THE STACK.
 8538  017154  011667  000004            MOV     (SP),   20$     ;SET DATA FOR RELOC SUBROUTINE.
 8539  017160  004567  177162            JSR     R5,     RELOC   ;GO MOVE THE PROGRAM BACK TO BANKS 0 AND 1.
 8540  017164  000000    20$:    .WORD   0               ;SOURCE STARTING ADDRESS.
 8541  017166  000000                    .WORD   0               ;DESTINATION STARTING ADDRESS.
 8542  017170  161607                    SUB     (SP),   PC      ;JUMP TO RELOCATED PROGRAM.
 8543                                    ;*  THE PROGRAM IS NOW EXICUTING OUT OF BANKS 0 AND 1.
 8544  017172  161606                    SUB     (SP),   SP      ;RESET THE STACK POINTER.
 8545  017174  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
 8546  017176  012700  001622            MOV     #RADTAB,R0      ;SET UP POINTER TO RELATIVE ADDRESS TABLE.
 8547
 8548  017202  166620  000002    21$:    SUB     2(SP),  (R0)+   ;RESET ADDRESSES TO UNRELOCATED VALUES.
 8549  017206  005720            22$:    TST     (R0)+           ;CHECK FOR TERMINATORS.
 8550  017210  001776                    BEQ     22$             ;BR OVER TERMINATORS.
 8551  017212  024027  177777            CMP     -(R0),  #-1     ;CHECK FOR END OF TABLE INDICATOR.
 8552  017216  001371                    BNE     21$             ;BR IF MORE ADDRESSES IN TABLE.
 8553  017220  012600                    MOV     (SP)+,R0        ;;POP STACK INTO R0
 8554  017222  161637  000004            SUB     (SP),   @#ERRVEC ;ADJUST ERROR VECTOR.
 8555  017226  161637  000024            SUB     (SP),   @#PWRVEC ;ADJUST POWER FAIL VECTOR.
 8556  017232  161637  000114            SUB     (SP),   @#PARVEC ;ADJUST PARITY ERROR VECTOR.
```

```
 8557  017236  026727  161676  177570          CMP     SWR,    #177570 ;CHECK FOR HARDWARE SWITCH REGISTER.
 8558  017244  001404                          BEQ     23$             ;BR IF HARDWARE SWITCH REGISTER.
 8559  017246  161667  161666                  SUB     (SP),   SWR     ;ADJUST SOFTWARE SWITCH REGISTER.
 8560  017252  161667  161664                  SUB     (SP),   DISPLAY ;ADJUST SOFTWARE DISPLAY REGISTER.
 8561  017256  162616                  23$:    SUB     (SP)+,  (SP)    ;ADJUST RETURN ADDRESS.
 8562  017260  005067  161314          30$:    CLR     RELOCF          ;RESET RELOCATION FACTOR.
 8563  017264  012767  000003  161310          MOV     #3,     PRGMAP  ;SET PROGRAM MAP TO POINT TO BANKS 0 AND 1.
 8564  017272  005067  161306                  CLR     PRGMAP+2        ;...HI 64K.
 8565  017276  000207                          RTS     PC              ;RETURN.
 8566
 8567                                  ;;*****************************************************************
 8568                                  ;* THIS SUBROUTINE MOVES THE LOADER AREA BACK TO THE "TOP" OF MEMORY FROM
 8569                                  ;*  WHENCE IT CAME.  THE LOADER AREA IS SAVED AT THE END OF THE 8K OF
 8570                                  ;*  PROGRAM CODE WHEN THE PROGRAM IS INITIALLY RUN.
 8571                                  ;;*****************************************************************
 8572  017300  016700  162214  RESLDR:  MOV     LMAD,   R0      ;CHECK IF THE LOADERS WERE SAVED.
 8573  017304  001001                          BNE     1$              ;BR IF LOADER AREA WAS SAVED.
 8574  017306  000000                          HALT                    ;FATAL ERROR!!!  CAN'T RESTORE LOADER AREA IF IT WASN'T SAVED.
 8575  017310  005767  161272          1$:     TST     MMAVA           ;CHECK FOR MEM MGMT.
 8576  017314  001402                          BEQ     2$              ;SKIP IF NO MEM MGMT.
 8577  017316  005037  177572                  CLR     @#SR0           ;DISABLE MEM MGMT.
 8578  017322  012701  040000          2$:     MOV     #40000, R1      ;GET END OF 8K, ASSUME PROG NOT RELOCATED.
 8579  017326  012702  002734                  MOV     #1500., R2      ;GET COUNTER.
 8580  017332  014140                  3$:     MOV     -(R1),  -(R0)   ;MOVE THE LOADER AREA.
 8581  017334  005302                          DEC     R2              ;COUNT HOW LONG THE AREA IS.
 8582  017336  001375                          BNE     3$              ;BR IF NOT MORE TO MOVE.
 8583  017340  005067  162154                  CLR     LMAD            ;CLEAR MONITOR SAVED FLAG
 8584  017344  005767  161236                  TST     MMAVA           ;CHECK FOR MEM MGMT.
 8585  017350  001402                          BEQ     4$              ;BR IF NO MEM MGMT.
 8586  017352  005237  177572                  INC     @#SR0           ;ENABLE MEM MGMT.
 8587  017356  000207                  4$:     RTS     PC              ;RETURN.
 8588
 8589                                  ;*  ROUTINE TO SAVE THE LOADERS AT THE END OF 8K.
 8590  017360  005767  162134  SAVLDR:  TST     LMAD            ;CHECK IF LOADERS HAVE BEEN SAVED ALREADY.
 8591  017364  001024                          BNE     4$              ;BRANCH IF ALREADY SAVED
 8592  017366  012700  040000                  MOV     #40000, R0      ;GET END OF 8K
 8593  017372  010001                          MOV     R0,     R1      ;GET END OF 8K
 8594  017374  012737  017406  000004          MOV     #2$, @#ERRVEC   ;SET UP TIMEOUT VECTOR
 8595  017402  011020          1$:     MOV     (R0),   (R0)+   ;SEARCH FOR END OF MEMORY
 8596  017404  000776                          BR      1$              ;KEEP SEARCHING
 8597  017406  022626          2$:     CMP     (SP)+,  (SP)+   ;RESTORE STACK POINTER
 8598  017410  012737  025124  000004          MOV     #ERRTRP,@#ERRVEC ;RESET TIMEOUT VECTOR.
 8599  017416  010046                          MOV     R0,     -(SP)   ;SAVE LAST MEMORY ADDRESS (CONTIGUOUS)
 8600  017420  012702  002734                  MOV     #1500., R2      ;SET UP WORD COUNTER
 8601  017424  014041          3$:     MOV     -(R0),  -(R1)   ;SAVE THE LOADERS
 8602  017426  005302                          DEC     R2              ;COUNT THE WORDS
 8603  017430  001375                          BNE     3$              ;BRANCH IF MORE WORDS
 8604  017432  012667  162062                  MOV     (SP)+,  LMAD    ;SAVE LAST MEMORY ADDRESS
 8605  017436  000207          4$:     RTS     PC              ;RETURN
```

```
                                        .SBTTL  PARITY MEMORY TRAP SERVICE AND SUBROUTINES.
                                        ;;*****************************************************************
                                        ;* PARITY MEMORY UNEXPECTED ERROR TRAP SERVICE ROUTINE.
                                        ;*  FIND OUT WHICH REGISTER DETECTED THE ERROR.
                                        ;*  THEN SCAN MEMORY TO SEE IF PARITY ERROR STILL SET AND REPORT LOCATION.
                                        ;;*****************************************************************
017440  011667  161456          PESRV:  MOV     (SP),   $BDADR  ;GET PC OF INSTRUCTION WHICH CAUSED ERROR.
017444  004567  004052                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2) 017450  026511                      .WORD   UNEXPT          ;ADDRESS OF MESSAGE TO BE TYPED
(1)                                                             ;'UNEXPECTED MEMORY PARITY TRAP.'
017452  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(2) 017454  010346                      MOV     R3,-(SP)        ;;PUSH R3 ON STACK
017456  016703  162146                  MOV     .MPRX,  R3      ;GET POINTER TO PARITY REGISTERS.
017462  005733                  1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017464  100415                          BMI     3$              ;BR IF THIS REGISTER SHOWS THE ERROR.
017466  005713                          TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
017470  001374                          BNE     1$              ;BR IF MORE REGISTERS.
017472  004767  002152                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
(1)                             ;***ERROR***    NO REGISTER INDICATED ERROR
(1) 017476  000024                      .WORD   24              ;ERROR TYPE CODE.
017500  000417                          BR      4$              ;EXIT
017502  005713                  2$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
017504  001415                          BEQ     4$              ;BR IF NO MORE PARITY REGISTERS.
017506  005733                          TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
017510  100374                          BPL     2$              ;BR IF NO ERROR FLAG.
017512  004567  004004                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2) 017516  026602                      .WORD   MTOE            ;ADDRESS OF MESSAGE TO BE TYPED
(1)                                                             ;'MORE THAN ONE ERROR FOUND.'
017520                          3$:
(1) 017520  004767  000610      64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
(2) 017524  004767  002120              JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
(2) 017530  000025                      .WORD   25              ;ERROR TYPE CODE.
017532  004767  000216                  JSR     PC,     PSCAN   ;GO SCAN MEMORY FOR BAD PARITY.
017536  000761                          BR      2$              ;GO LOOK FOR MORE ERRORS.
017540                          4$:
(2) 017540  012603                      MOV     (SP)+,R3        ;;POP STACK INTO R3
(2) 017542  012601                      MOV     (SP)+,R1        ;;POP STACK INTO R1
017544  000002                          RTI                     ;RETURN.

                                        ;;*****************************************************************
                                        ;ROUTINE TO ENABLE PARITY ERROR ACTION ON MA/MF PARITY MEMORIES
                                        ;THIS ROUTINE IS MEANT TO CATCH UNEXPECTEDS
                                        ;;*****************************************************************
017546  005767  162524          MAMF:   TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
017552  001434                          BEQ     MAMF2           ;EXIT IF NO PARITY REGISTERS.
017554  032777  000100  161356          BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR DETECTION.
017562  001030                          BNE     MAMF2           ;EXIT IF NO PARITY ERROR DETECTION.
017564  005767  161010                  TST     RELOCF          ;CHECK IF PROGRAM RELOCATED OUT OF BANK 0.
017570  001410                          BEQ     SETAE           ;BR IF PROG IN BANK 0.
017572  032777  000040  161340          BIT     #SW5,   @SWR    ;CHECK IF VECTORS PROTECTED.
017600  001004                          BNE     SETAE           ;BR IF VECTOR AREA PROTECTED.
017602  026727  161754  001000          CMP     FSTADR, #1000   ;CHECK FOR STARTING ADDRESS ABOVE THE VECTORS.
017610  103415                          BLO     MAMF2           ;EXIT IF VECTORS EXPOSED TO TESTING.

017612  016737  162020  000114  SETAE:  MOV     .PESRV, @#PARVEC ;SET PARITY ERROR TRAP VECTOR
017620  005037  000116                  CLR     @#PARVEC+2      ;PRIORITY LEVEL 0 ON TRAP
```

```
 8651  017624  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 8652  017626  016703  161776                MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
 8653  017632  052733  000001        MAMF1:  BIS     #AE,    @(R3)+  ;SET ACTION ENABLE BIT IN PARITY REG
 8654  017636  005713                        TST     (R3)            ;CHECK FOR END OF TABLE.
 8655  017640  001374                        BNE     MAMF1           ;BR IF MORE PARITY REGISTERS.
 8656  017642  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
 8657  017644  000207                MAMF2:  RTS     PC              ;RETURN.
 8658
 8659                                        ;;*****************************************************************
 8660                                        ;*  SUBROUTINE TO CHECK PARITY REGISTERS FOR ERRORS THAT DIDN'T TRAP.
 8661                                        ;;*****************************************************************
 8662  017646                                CHGG2:
 8663  017646  005767  162424        CKPMER: TST     MPRX            ;CHECK IF ANY PARITY REGISTERS EXIST.
 8664  017652  001437                        BEQ     4$              ;BR IF NO PARITY REGISTERS.
 8665  017654  032777  000100  161256        BIT     #SW6,   @SWR    ;CHECK FOR INHIBIT PARITY ERROR CHECKING.
 8666  017662  001033                        BNE     4$              ;BR IF PARITY ERROR CHECKING INHIBITED.
 8667  017664  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 8668  017666  016703  161736                MOV     .MPRX,  R3      ;GET PARITY REG TABLE POINTER.
 8669  017672  005733                1$:     TST     @(R3)+          ;CHECK THE PARITY REG FOR AN ERROR FLAG.
 8670  017674  100023                        BPL     3$              ;BR IF NO ERROR
 8671  017676  032773  000001  177776        BIT     #BIT0,  @-2(R3) ;CHECK IF A TRAP SHOULD HAVE OCCURRED.
 8672  017704  001410                        BEQ     2$              ;BR IF NO ACTION ENABLE.   CHGG2
 8673  017706  004767  000422        64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
  (2)  017712  004767  001732                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  017716  000026                        .WORD   26              ;ERROR TYPE CODE.
 8674  017720  000411                        BR      3$
 8675  017722  004767  000026                JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
 8676  017726                        2$:
  (1)  017726  004767  000402        65$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
  (2)  017732  004767  001712                JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  017736  000027                        .WORD   27              ;ERROR TYPE CODE.
 8677  017740  004767  000010                JSR     PC,     PSCAN   ;GO SCAN ALL MEMORY FOR PARITY ERRORS.
 8678  017744  005713                3$:     TST     (R3)            ;CHECK FOR TABLE TERMINATOR.
 8679  017746  001351                        BNE     1$              ;BR IF MORE.
 8680  017750  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
 8681  017752  000207                4$:     RTS     PC              ;RETURN.
 8682
 8683                                        ;;*****************************************************************
 8684                                        ;*  THIS SUBROUTINE WILL SCAN ALL OF MEMORY LOOKING FOR BAD PARITY,
 8685                                        ;*  TYPE OUT ALL LOCATIONS FOUND TO BE BAD, AND WRITE BACK INTO THE
 8686                                        ;*  LOCATIONS IN ORDER TO RESTORE GOOD PARITY.
 8687                                        ;;*****************************************************************
 8688
 8689  017754                                PSCAN:
  (2)  017754  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (2)  017756  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (2)  017760  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (2)  017762  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
  (2)  017764  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
  (2)  017766  013746  000114                MOV     @#114,-(SP)     ;;PUSH @#114 ON STACK
  (2)  017772  013746  000116                MOV     @#116,-(SP)     ;;PUSH @#116 ON STACK
 8690  017776  004567  003520                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  020002  026646                        .WORD   SCANM           ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                                               ;"SCANNING MEMORY FOR BAD PARITY."
 8691  020004  012700  000001                MOV     #BIT0,  R0      ;SET BIT POINTER TO FIRST BANK.
 8692  020010  005001                        CLR     R1              ;CLR HI 64K POINTER.
```

```
 8693  020012  005002                          CLR     R2              ;INIT ADDRESS POINTER.
 8694  020014  005004                          CLR     R4              ;INIT ERROR DETECTED FLAG.
 8695  020016  004767  000256                  JSR     PC,     CLRPAR  ;CLEAR THE PARITY REGISTERS.
 8696  020022  012737  000116  000114          MOV     #116,   @#114   ;HALT IF ANOTHER PARITY TRAP.
 8697  020030  005037  000116                  CLR     @#116
 8698  020034  005767  160546                  TST     MMAVA           ;CHECK FOR MEMORY MANAGEMENT.
 8699  020040  001406                          BEQ     1$              ;BR IF NO MEM MGMT.
 8700  020042  013746  172344                  MOV     @#KIPAR2,-(SP)          ;;PUSH @#KIPAR2 ON STACK
 8701  020046  005037  172344                  CLR     @#KIPAR2        ;INIT MEM MGMT TO POINT TO BANK 0.
 8702  020052  012702  040000                  MOV     #40000, R2      ;SET ADR POINTER TO PAR2.
 8703  020056  030067  161442          1$:     BIT     R0,     MEMMAP  ;CHECK IF THIS BANK OF MEM EXISTS.
 8704  020062  001003                          BNE     2$              ;BR IF THIS BANK EXISTS.
 8705  020064  030167  161436                  BIT     R1,     MEMMAP+2 ;CHECK HI 64K MAP.
 8706  020070  001442                          BEQ     10$             ;BR IF THIS BANK DOESN'T EXIST.
 8707  020072                          2$:
  (2)  020072  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
 8708  020074  111201                  3$:     MOVB    (R2),   R1      ;READ THE LOCATION TO SEE IF IT HAS A PARITY ERROR.
 8709  020076  016703  161526                  MOV     .MPRX,  R3      ;SET UP POINTER TO PARITY REGISTERS.
 8710  020102  005733                  4$:     TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
 8711  020104  100024                          BPL     6$              ;BR IF NO ERROR FLAG.
 8712  020106  005704                          TST     R4              ;CHECK IF FIRST ERROR, THIS SCAN.
 8713  020110  001003                          BNE     5$              ;BR IF MORE THAN ONE ERROR FOUND.
 8714  020112  005367  160774                  DEC     $ERTTL          ;ADJUST ERROR COUNT.
 8715  020116  005204                          INC     R4              ;SET FLAG TO INDICATE ERROR FOUND.
 8716  020120                          5$:
  (1)  020120  004767  000210          64$:    JSR     PC,     SPRNTQ  ;SET UP VALUES FOR ERROR PRINTING.
  (2)  020124  004767  001520                  JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (2)  020130  000030                          .WORD   30              ;ERROR TYPE CODE.
 8717  020132  111212                          MOVB    (R2),   (R2)    ;REWRITE THE LOCATION TO CLEAR BAD PARITY.
 8718  020134  005053                          CLR     @-(R3)          ;CLEAR THE ERROR FLAG.
 8719  020136  105712                          TSTB    (R2)            ;CHECK IF THE PARITY ERROR WAS CLEARED.
 8720  020140  005733                          TST     @(R3)+          ;CHECK FOR THE ERROR FLAG.
 8721  020142  100005                          BPL     6$              ;BR IF IT IS OK.
 8722  020144  004567  003352                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  020150  026710                          .WORD   PEWNC           ;ADDRESS OF MESSAGE TO BE TYPED
  (1)                                          ;'PARITY ERROR WILL NOT CLEAR.''
 8723  020152  005073  177776                  CLR     @-2(R3)         ;CLEAR OUT THE PARITY ERROR FLAG.
 8724  020156  005713                  6$:     TST     (R3)            ;CHECK FOR THE END OF REG ADR TABLE.
 8725  020160  001350                          BNE     4$              ;BR IF MORE PARITY REGISTERS.
 8726  020162  005202                          INC     R2              ;GO TO NEXT MEMORY ADDRESS.
 8727  020164  032702  017777                  BIT     #MASK4K,R2      ;CHECK FOR END OF 4K BANK.
 8728  020170  001341                          BNE     3$              ;BR IF MORE MEMORY THIS BANK.
 8729  020172  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
 8730  020174  000402                          BR      11$             ;BR TO CHECK FOR NEXT BANK.
 8731  020176  062702  020000          10$:    ADD     #20000, R2      ;SKIP BANKS THAT AREN'T THERE.
 8732  020202  005767  160400          11$:    TST     MMAVA           ;CHECK FOR MEM MGMT.
 8733  020206  001413                          BEQ     12$             ;BR IF NO MEM MGMT.
 8734  020210  062737  000200  172344          ADD     #200,   @#KIPAR2 ;UPDATE MEM MGMT REG TO NEXT 4K.
 8735  020216  012702  040000                  MOV     #40000, R2      ;RESET ADDRESS POINTER TO BEGINNING OF BANK.
 8736  020222  006300                          ASL     R0              ;UPDATE BANK POINTER.
 8737  020224  006101                          ROL     R1              ;...HI 64K.
 8738  020226  100313                          BPL     1$              ;BR IF MORE BANKS.
 8739  020230  012637  172344                  MOV     (SP)+,@#KIPAR2          ;;POP STACK INTO @#KIPAR2
 8740  020234  000402                          BR      20$             ;GO CHECK IF ANY ERRORS FOUND.
 8741  020236  106300                  12$:    ASLB    R0              ;UPDATE POINTER TO NEXT BANK.
 8742  020240  100306                          BPL     1$              ;BR IF MORE BANKS.
```

```
 8743  020242  005704                  20$:    TST     R4              ;CHECK IF ANY PARITY ERRORS DETECTED.
 8744  020244  001003                          BNE     21$             ;BR IF ERRORS DETECTED.
 8745  020246  004567  003250                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  020252  025716                          .WORD   NOPES           ;ADDRESS OF MESSAGE TO BE TYPED
 8746  020254                          21$:
  (2)  020254  012637  000116                  MOV     (SP)+,@#116     ;;POP STACK INTO @#116
  (2)  020260  012637  000114                  MOV     (SP)+,@#114     ;;POP STACK INTO @#114
  (2)  020264  012604                          MOV     (SP)+,R4        ;;POP STACK INTO R4
  (2)  020266  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
  (2)  020270  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
  (2)  020272  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
  (2)  020274  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
 8747  020276  000207                          RTS     PC              ;RETURN.
 8748
 8749                                  ;;*****************************************************************
 8750                                  ;ROUTINE TO CLEAR ALL PARITY REGISTERS PRESENT
 8751                                  ;;*****************************************************************
 8752  020300                          CLRPAR:
  (2)  020300  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 8753  020302  016703  161322                  MOV     .MPRX,  R3      ;GET PARITY REGISTER TABLE POINTER.
 8754  020306  005713                  1$:     TST     (R3)            ;CHECK FOR THE TABLE TERMINATOR.
 8755  020310  001402                          BEQ     2$              ;BR IF DONE ALL PARITY REGISTERS.
 8756  020312  005033                          CLR     @(R3)+          ;CLEAR THE PARITY REGISTER.
 8757  020314  000774                          BR      1$              ;BR FOR MORE
 8758  020316                          2$:
  (2)  020316  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
 8759  020320  000207                          RTS     PC              ;RETURN.
 8760
 8761                                  .SBTTL  SUBROUTINES TO SET UP DATA FOR ERROR PRINTOUT ROUTINE.
 8762                                  ;;*****************************************************************
 8763                                  ;*  THESE ROUTINES ARE USED TO TRANSFER DATA TO COMMON TAG AREA (.$CMTAG)
 8764                                  ;*  FOR ERROR PRINTOUT BY .$ERROR & .$ERRTYP ROUTINES FROM **SYSMAC**.
 8765                                  ;;*****************************************************************
 8766  020322  010267  160572          $PRNT:  MOV     R2,     $GDADR  ;SAVE THE ADDRESS UNDER TEST.
 8767  020326  005067  160572                  CLR     $GDDAT          ;SHOULD BE DATA IS '0'.
 8768  020332  000430                          BR      $PRNTB
 8769
 8770  020334  014367  160620          $PRNT0: MOV     -(R3),  $TMP0   ;GET THE PARITY REGISTER ADDRESS.
 8771  020340  013367  160616                  MOV     @(R3)+, $TMP1   ;GET THE CONTENTS OF THE PARITY REG.
 8772  020344  000402                          BR      $PRNT0
 8773
 8774  020346  011367  160606          $PRNTP: MOV     (R3),   $TMP0   ;GET THE PARITY REGISTER ADDRESS.
 8775  020352  010267  160542          $PRNT0: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
 8776  020356  000414                          BR      $PRNTA          ;BR TO COMMON SECTION.
 8777
 8778  020360  010267  160534          $PRNT1: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
 8779  020364  005367  160530                  DEC     $GDADR          ;ADJUST IT FOR PRINTOUT.
 8780  020370  000407                          BR      $PRNTA          ;BR TO COMMON SECTION.
 8781
 8782  020372  010367  160562          $PRNT3: MOV     R3,     $TMP0   ;GET THE DATA IN R3.
 8783  020376  010267  160516          $PRNT2: MOV     R2,     $GDADR  ;GET THE MEMORY ADDRESS BEING TESTED
 8784  020402  162767  000002  160510          SUB     #2,     $GDADR  ;ADJUST IT FOR PRINTOUT.
 8785  020410  010067  160510          $PRNTA: MOV     R0,     $GDDAT  ;GET WHAT THE DATA SHOULD BE
 8786  020414  010167  160506          $PRNTB: MOV     R1,     $BDDAT  ;GET WHAT THE DATA WAS
 8787  020420  000207                          RTS     PC              ;RETURN TO ENTER ERROR ROUTINES
 8788
```

```
 8789                                   ;;*****************************************************************
 8790                                   ;*  SUBROUTINE TO TYPE OUT A MAP OF 4K BANK.
 8791                                   ;*  R0 POINTS TO THE MAP UPON ENTERING THIS ROUTINE.
 8792                                   ;;*****************************************************************
 8793  020422  005710                  TYPMAP: TST     (R0)            ;CHECK IF ANY MEMORY IN MAP...LO 64K.
 8794  020424  001007                          BNE     1$              ;BR IF MEMORY IN MAP.
 8795  020426  005760  000002                  TST     2(R0)           ;...HI 64K.
 8796  020432  001004                          BNE     1$              ;BR IF MEMORY IN MAP.
 8797  020434  004567  003062                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
   (2) 020440  026276                          .WORD   NOMEM           ;ADDRESS OF MESSAGE TC BE TYPED
   (1)                                                                 ;'NO MEMORY FOUND.'
 8798  020442  000475                          BR      6$              ;EXIT
 8799  020444                         1$:
   (2) 020444  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
   (2) 020446  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
   (2) 020450  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
   (2) 020452  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 8800  020454  012701  000001                  MOV     #BIT0,  R1      ;SET UP BANK POINTER...LO 64K.
 8801  020460  005002                          CLR     R2              ;...HI 64K.
 8802  020462  012703  177777                  MOV     #-1,    R3      ;SET UP ADDRESS POINTER TO -1.
 8803  020466  010304                          MOV     R3,     R4      ;HI BITS OF ADDRESS AS WILL.
 8804  020470  030110                  2$:     BIT     R1,     (R0)    ;CHECK THE MAP FOR THIS BANK.
 8805  020472  001014                          BNE     3$              ;BR IF THIS BANK PRESENT.
 8806  020474  030260  000002                  BIT     R2,     2(R0)   ;CHECK HI 64K MAP.
 8807  020500  001011                          BNE     3$              ;BR IF THIS BANK PRESENT.
 8808  020502  105703                          TSTB    R3              ;CHECK FOR PREVIOUS PRINTOUT.
 8809  020504  001042                          BNE     5$              ;BR IF ALREADY TYPED 'TO''
 8810  020506  162703  000001                  SUB     #1,     R3      ;BACK UP TO LAST ADR OF PREVIOUS BANK.
 8811  020512  005604                          SBC     R4              ;...HI ADDRESS BITS.
 8812
 8813  020514  004567  003002                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
   (1) 020520  025527                          .WORD   TO              ;ADDRESS OF MESSAGE TO BE TYPED
 8814  020522  000410                          BR      4$              ;GO TO TYPE THE ADDRESS.
 8815  020524  105703                  3$:     TSTB    R3              ;CHECK FOR PREVIOUS TYPEOUT.
 8816  020526  001431                          BEQ     5$              ;BR IF ALREADY TYPE 'FROM'.
 8817
 8818  020530  062703  000001                  ADD     #1,     R3      ;POINT TO FIRST ADDRESS OF THIS BANK.
 8819  020534  005504                          ADC     R4              ;...HI BITS OF ADDRESS.
 8820  020536  004567  002760                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
   (1) 020542  025517                          .WORD   FROM            ;ADDRESS OF MESSAGE TO BE TYPED
 8821  020544                         4$:
   (2) 020544  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
   (2) 020546  010446                          MOV     R4,-(SP)        ;;PUSH R4 ON STACK
 8822  020550  006303                          ASL     R3              ;BIT 15 INTO C-BIT
 8823  020552  006104                          ROL     R4              ;BIT 15 INTO R4.
 8824  020554  006003                          ROR     R3              ;RESTORE BITS 14-0.
 8825  020556  010446                          MOV     R4,-(SP)        ;;SAVE R4 FOR TYPEOUT
   (1)                                                                 ;;TYPE ADDRESS BITS 21-15
   (3)                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
   (3)                                   ;*  WIHTOUT USING A ''TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
   (3) 020560  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
   (3) 020564  004767  004104                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
   (1) 020570     003                          .BYTE   3               ;;TYPE 3 DIGIT(S)
   (1) 020571     000                          .BYTE   0               ;;SUPPRESS LEADING ZEROS
 8826  020572  010346                          MOV     R3,-(SP)        ;;SAVE R3 FOR TYPEOUT
   (1)                                                                 ;;TYPE ADDRESS BITS 14-0
```

```
 (3)                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
 (3)                                    ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 (3)  020574  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 (3)  020600  004767  004070                    JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
 (1)  020604     005                            .BYTE   5               ;;TYPE 5 DIGIT(S)
 (1)  020605     001                            .BYTE   1               ;;TYPE LEADING ZEROS
8827  020606  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
 (2)  020610  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
8828  020612  062703  020000           5$:     ADD     #20000, R3      ;UPDATE TO NEXT BANK.
8829  020616  005504                            ADC     R4              ;...HI ADDRESS BITS.
8830  020620  006301                            ASL     R1              ;SHIFT POINTER...LO 64K.
8831  020622  006102                            ROL     R2              ;...HI 64K.
8832  020624  103321                            BCC     2$              ;BR IF MORE BANKS.
8833  020626  012604                            MOV     (SP)+,R4        ;;POP STACK INTO R4
 (2)  020630  012603                            MOV     (SP)+,R3        ;;POP STACK INTO R3
 (2)  020632  012602                            MOV     (SP)+,R2        ;;POP STACK INTO R2
 (2)  020634  012601                            MOV     (SP)+,R1        ;;POP STACK INTO R1
8834  020636  000207           6$:     RTS     PC                      ;RETURN.
8835
8919                                    .SBTTL  SCOPE HANDLER ROUTINE
 (1)
 (2)                                    ;;**********************************************************************
 (1)                                    ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
 (1)                                    ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
 (1)                                    ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
 (1)                                    ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 (1)                                    ;*SW14=1        LOOP ON TEST
 (1)                                    ;*SW11=1        INHIBIT ITERATIONS
 (1)                                    ;*SW09=1        LOOP ON ERROR
 (1)                                    ;*SW08=1        LOOP ON TEST IN SWR<4:0>
 (1)                                    ;*CALL
 (1)                                    ;*      SCOPE           ;;SCOPE=IOT
 (1)
 (1)  020640                            $SCOPE:
 (3)                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
 (3)                                    ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
 (3)  020640  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
 (3)  020644  004767  001524                    JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
 (3)  020650  012504                            MOV     (R5)+,  R4      ;SAVE MINIMUM BLOCK MASK NEXT TEST.
 (3)  020652  010516                            MOV     R5,     (SP)    ;PUT RETURN PC ONTO STACK, SIMULATE JSR PC.
 (1)  020654  032777  040000  160256   1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
 (1)  020662  001117                            BNE     $OVER           ;;YES IF SW14=1
 (1)                                    ;######START OF CODE FOR THE XOR TESTER######
 (1)  020664  000416                    $XTSTR: BR      6$              ;;IF RUNNING ON THE 'XOR' TESTER CHANGE
 (1)                                                                    ;;THIS INSTRUCTION TO A 'NOP' (NOP 240)
 (1)  020666  013746  000004                    MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
 (1)  020672  012737  020712  000004            MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
 (1)  020700  005737  177060                    TST     @#177060        ;;TIME OUT ON XOR?
 (1)  020704  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 (1)  020710  000466                            BR      $SVLAD          ;;GO TO THE NEXT TEST
 (1)  020712  022626                    5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
 (1)  020714  012637  000004                    MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
 (1)  020720  000426                            BR      7$              ;;LOOP ON THE PRESENT TEST
 (1)  020722                            6$:;######END OF CODE FOR THE XOR TESTER######
 (1)  020722  032777  000400  160210            BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
 (1)  020730  001407                            BEQ     2$              ;;BR IF NO
```

```
(1)  020732  017746  160202            MOV     @SWR,-(SP)      ;;SET DESIRED TEST NUM. FROM SWR
(1)  020736  042716  000340            BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
(1)  020742  122667  160134            CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
(1)  020746  001465                    BEQ     $OVER           ;;BR IF YES
(1)  020750  105767  160127      2$:   TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
(1)  020754  001421                    BEQ     3$              ;;BR IF NO
(1)  020756  126767  160133  160117    CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
(1)  020764  101015                    BHI     3$              ;;BR IF NO
(1)  020766  032777  001000  160144    BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
(1)  020774  001404                    BEQ     4$              ;;BR IF NO
(1)  020776  016767  160106  160102  7$:   MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
(1)  021004  000446                    BR      $OVER
(1)  021006  105067  160071      4$:   CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
(1)  021012  005067  160152            CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
(1)  021016  000415                    BR      1$              ;;ESCAPE TO THE NEXT TEST
(1)  021020  032777  004000  160112  3$:   BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
(1)  021026  001011                    BNE     1$              ;;BR IF YES
(1)  021030  005767  160156            TST     $PASS           ;;IF FIRST PASS OF PROGRAM
(1)  021034  001406                    BEQ     1$              ;;      INHIBIT ITERATIONS
(1)  021036  005267  160042            INC     $ICNT           ;;INCREMENT ITERATION COUNT
(1)  021042  026767  160122  160034    CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
(1)  021050  002024                    BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
(1)  021052  012767  000001  160024  1$:   MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
(1)  021060  016767  000552  160102        MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
(1)  021066  105267  160010      $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
(1)  021072  116767  160004  160110        MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
(1)  021100  011667  160002            MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
(1)  021104  011667  160000            MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
(1)  021110  005067  160056            CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
(1)  021114  112767  000001  157773    MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
(1)  021122  016777  157754  160012  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
(1)  021130  016716  157752            MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
(3)  021134  020516                INSERT: CMP     R5,     (SP)    ;CHECK FOR LOOP ON TEST.
(3)  021136  001402                    BEQ     1$              ;BR IF START NEXT TEST.
(3)  021140  000167  000470            JMP     ENDINS          ;JMP IF LOOP ON LAST TEST.
(3)  021144  012767  037777  160434  1$:   MOV     #37777, BLKMSK  ;SET 8K BOUNDRY MASK.
(3)  021152  005767  160034            TST     $PASS           ;CHECK FOR PASS 0.
(3)  021156  001404                    BEQ     2$              ;BR IF PASS 0
(3)  021160  126727  157716  000021    CMPB    $TSTNM, #21     ;CHECK IF IN SECTION 3.
(3)  021166  103002                    BHIS    3$              ;BR IF IN SECTION 3.
(3)  021170  006267  160412      2$:   ASR     BLKMSK          ;RESET BOUNDRY TO 4K.
(3)  021174  016767  160362  160362  3$:   MOV     FSTADR, TMPFAD  ;GET FIRST ADDRESS.
(3)  021202  005767  157372            TST     RELOCF          ;CHECK IF PRG RELOCATED.
(3)  021206  001430                    BEQ     4$              ;BR IF NOT RELOCATED.
(3)  021210  032777  000040  157722    BIT     #SW05,  @SWR    ;CHECK IF LOC 0-776 TO BE PROTECTED.
(3)  021216  001424                    BEQ     4$              ;BR IF SW NOT SET.
(3)  021220  026727  160340  001000    CMP     TMPFAD, #1000   ;CHECK IF NOT BEING TESTED.
(3)  021226  103020                    BHIS    4$              ;BR IF ALREADY PROTECTED.
(3)  021230  012767  001000  160326    MOV     #1000,  TMPFAD  ;RESET FIRST ADDRESS.
(3)  021236  052767  000001  160324    BIS     #BIT0,  FADMAP  ;SET FLAG IN FIRST BANK.
(3)  021244  026727  160324  001000    CMP     LSTADR, #1000   ;CHECK IF GONE PAST LAST ADR.
(3)  021252  101006                    BHI     4$              ;BR IF ENOUGH MEMORY.
(5)  021254  004567  002242            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(5)  021260  026747                    .WORD   NOMTST          ;ADDRESS OF MESSAGE TO BE TYPED
(4)                                                            ;'NO MEMORY TESTED'
(3)  021262  016716  160356            MOV     .TST32, (SP)    ;ADJUST RETURN ADR FOR ABORT.
```

```
(3)  021266  000207                          RTS     PC              ;ABORT.
(3)  021270  016767  160300  160300  4$:     MOV     LSTADR, TMPLAD  ;GET LAST ADDRESS.
(3)  021276  016767  160232  160224          MOV     SAVTST, TSTMAP  ;GET TEST MAP, LO 64K.
(3)  021304  016767  160226  160220          MOV     SAVTST+2,TSTMAP+2 ;...HI 64K.
(3)  021312  046767  157264  160210          BIC     PRGMAP, TSTMAP  ;DON'T TEST OVER THE PROGRAM.
(3)  021320  046767  157260  160204          BIC     PRGMAP+2,TSTMAP+2
(3)  021326  005767  157660                  TST     $PASS           ;CHECK FOR FIRST PASS
(3)  021332  001011                          BNE     10$             ;BR IF NOT FIRST PASS.
(3)  021334  032767  000003  160166          BIT     #3,     TSTMAP  ;CHECK IF FIRST TWO BANKS AVAILABLE.
(3)  021342  001405                          BEQ     10$             ;NOT TESTING FIRST 2 BANKS.
(3)  021344  042767  177774  160156          BIC     #177774,TSTMAP  ;CLR ALL BUT FIRST 2 BANKS.
(3)  021352  005067  160154                  CLR     TSTMAP+2
(3)  021356  005704                  10$:     TST     R4              ;CHECK FOR A MINIMUM BLOCK SIZE.
(3)  021360  001503                          BEQ     20$             ;BR IF NO MIN BLOCK SIZE.
(3)  021362  030467  160176                  BIT     R4,     TMPFAD  ;CHECK IF FIRST ADR ON BLOCK BOUNDRY.
(3)  021366  001416                          BEQ     11$             ;BR IF FIRST ADR ON BLOCK BOUNDRY.
(3)  021370  050467  160170                  BIS     R4,     TMPFAD  ;ADJUST FIRST ADR TO END OF BLOCK.
(3)  021374  005267  160164                  INC     TMPFAD          ;FIRST ADR TO FIRST ADR OF NEXT BLOCK.
(3)  021400  032767  017777  160156          BIT     #MASK4K,TMPFAD  ;CHECK IF FIRST ADR REACHED 4K BOUNDRY.
(3)  021406  001006                          BNE     11$             ;BR IF NOT ON 4K BOUNDRY.
(3)  021410  046767  160154  160112          BIC     FADMAP, TSTMAP  ;DON'T TEST FIRST BANK.
(3)  021416  046767  160150  160106          BIC     FADMAP+2,TSTMAP+2
(3)  021424  030467  160146          11$:     BIT     R4,     TMPLAD  ;CHECK IF LAST ADR ON BLOCK BOUNDRY.
(3)  021430  001414                          BEQ     12$             ;BR IF ON BLOCK BOUNDRY.
(3)  021432  040467  160140                  BIC     R4,     TMPLAD  ;ADJUST LAST ADR DOWN TO NEXT BLOCK BOUNDRY.
(3)  021436  032767  017777  160132          BIT     #MASK4K,TMPLAD  ;CHECK IF ADJUSTED TO 4K BOUNDRY.
(3)  021444  001006                          BNE     12$             ;BR IF NOT ON 4K BOUNDRY.
(3)  021446  046767  160130  160054          BIC     LADMAP, TSTMAP  ;SKIP TESTING LAST BANK.
(3)  021454  046767  160124  160050          BIC     LADMAP+2,TSTMAP+2
(3)  021462  036767  160102  160112  12$:     BIT     FADMAP, LADMAP  ;CHECK IF FIRST AND LAST IN SAME BANK.
(3)  021470  001004                          BNE     13$             ;BR IF IN SAME BANK.
(3)  021472  036767  160074  160104          BIT     FADMAP+2,LADMAP+2 ;...UPPER 64K.
(3)  021500  001404                          BEQ     14$             ;BR IF FIRST AND LAST NOT SAME BANK.
(3)  021502  026767  160070  160054  13$:     CMP     TMPLAD, TMPFAD  ;CHECK IF ANY MEMORY LEFT.
(3)  021510  101406                          BLOS    15$             ;BR IF NO MEMORY TO TEST.
(3)  021512  005767  160012          14$:     TST     TSTMAP          ;CHECK IF ANY BANKS LEFT TO TEST!!
(3)  021516  001017                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
(3)  021520  005767  160006                  TST     TSTMAP+2        ;CHECK FOR ANY BANKS.
(3)  021524  001014                          BNE     16$             ;BR IF TEST MAP NOT EMPTY.
(3)  021526                          15$:
(5)  021526  004567  001770                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(5)  021532  026773                          .WORD   SKPMES          ;ADDRESS OF MESSAGE TO BE TYPED
(4)                                          ;'SKIPPING TEST #'
(4)  021534  005046                          CLR     -(SP)           ;CLEAR THE WORD ON THE STACK.
(4)  021536  116716  157340                  MOVB    $TSTNM, (SP)    ;PUT THE DATA ON THE STACK.
(6)                          ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
(6)                          ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(6)  021542  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(6)  021546  004767  003122                  JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
(4)  021552     003                          .BYTE   3               ;TYPE 3 DIGITS.
(4)  021553     001                          .BYTE   1               ;TYPE LEADING ZEROS.
(3)  021554  000427                          BR      ENDINS          ;RETURN TO SKIP TEST.
(3)  021556  062716  000004          16$:     ADD     #4,     (SP)    ;SKIP THE SKIP ON RETURN.
(3)  021562  062767  000004  157316          ADD     #4,     $LPADR  ;ADJUST THE LOOP ADR PAST THE SKIP.
(3)  021570  012767  017777  157770  20$:     MOV     #MASK4K,FADMSK  ;GET 4K MASK.
(3)  021576  016705  157762                  MOV     TMPFAD, R5      ;GET FIRST ADR.
```

```
(3)  021602  040567  157760          21$:    BIC     R5,     FADMSK  ;CLR MASK ABOVE LOWEST BIT OF FIRST ADR.
(3)  021606  006305                          ASL     R5              ;MOVE LOWEST BIT UP ONE.
(3)  021610  001374                          BNE     21$             ;LOOP UNTIL OVERFLOW.
(3)  021612  012767  017777  157760          MOV     #MASK4K,LADMSK  ;SET MASK BITS
(3)  021620  016705  157752                  MOV     TMPLAD, R5      ;GET LAST ADR.
(3)  021624  040567  157750          22$:    BIC     R5,     LADMSK  ;CLR ALL MASK BITS ABOVE LOWEST BIT IN LAST ADR.
(3)  021630  006305                          ASL     R5              ;MOVE LOWEST BIT OF LAST ADR UP ONE.
(3)  021632  001374                          BNE     22$             ;LOOP UNTIL OVERFLOW.
(3)  021634  000207                  ENDINS: RTS     PC              ;EXIT SCOPE ROUTINE BACK TO TEST.
(1)  021636  000004                  $MXCNT: 4                       ;;MAX. NUMBER OF ITERATIONS
8920                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
 (2)                                  ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(2)  021640  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(2)  021644  004767  000524                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
8921                                  .SBTTL  ERROR HANDLER ROUTINE
 (1)
 (2)                                  ;;**********************************************************************
 (1)                                  ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
 (1)                                  ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
 (1)                                  ;*AND GO TO $ERRTYP ON ERROR
 (1)                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
 (1)                                  ;*SW15=1        HALT ON ERROR
 (1)                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
 (1)                                  ;*SW10=1        BELL ON ERROR
 (1)                                  ;*SW09=1        LOOP ON ERROR
 (1)                                  ;*CALL
 (1)                                  ;*      ERROR   N       ;;ERROR=EMT AND N ERROR ITEM NUMBER
 (1)
 (1)  021650                          $ERROR:
 (3)                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
 (3)                                  ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(3)  021650  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(3)  021654  004767  000514                  JSR     PC,     $CKSWR  ;GO TO THE SUBROUTINE
(2)  021660  062716  000002                  ADD     #2,     (SP)    ;ADJUST POINTER PAST CODE WORD.
(1)  021664  105267  157213          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
(1)  021670  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
(1)  021672  016777  157204  157242          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
(1)  021700  032777  002000  157232          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
(1)  021706  001403                          BEQ     1$              ;;NO - SKIP
(2)  021710  004567  001606                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  021714  001174                          .WORD   $BELL           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  021716  005267  157170          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
(1)  021722  011667  157170                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
(1)  021726  162767  000002  157162          SUB     #2,$ERRPC
(1)  021734  117767  157156  157152          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
(1)  021742  032777  020000  157170          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
(1)  021750  001005                          BNE     20$             ;;SKIP TYPEOUTS
(1)  021752  004767  000116                  JSR     PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
(2)  021756  004567  001540                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  021762  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  021764                          20$:
(1)  021764  122767  000001  157232          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  021772  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
(1)  021774  116767  157114  000004          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
(1)  022002  004767  002044                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
(1)  022006     000                  21$:    .BYTE   0
```

```
(1)  022007  000          .BYTE   0
(1)  022010  000777  22$:  BR     22$              ;;APT ERROR LOOP
(1)  022012  005777  157122  2$:  TST    @SWR             ;;HALT ON ERROR
(1)  022016  100005        BPL    3$               ;;SKIP IF CONTINUE
(1)  022020  000000        HALT                    ;;HALT ON ERROR!
(3)                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $CKSWR ROUTINE
(3)                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(3)  022022  013746  177776        MOV    @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
(3)  022026  004767  000342        JSR    PC,    $CKSWR   ;GO TO THE SUBROUTINE
(1)  022032  032777  001000  157100  3$:  BIT    #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
(1)  022040  001402        BEQ    4$               ;;BR IF NO
(1)  022042  016716  157042        MOV    $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
(1)  022046  005767  157120  4$:  TST    $ESCAPE          ;;CHECK FOR AN ESCAPE ADDRESS
(1)  022052  001402        BEQ    5$               ;;BR IF NONE
(1)  022054  016716  157112        MOV    $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
(1)  022060                5$:
(1)  022060  022737  014232  000042        CMP    #$ENDAD,@#42    ;;ACT-11 AUTO-ACCEPT?
(1)  022066  001001        BNE    6$               ;;BRANCH IF NO
(1)  022070  000000        HALT                    ;;YES
(1)  022072                6$:
(2)  022072  000207        RTS    PC
8922                       ;;**************************************************************
(1)
(1)                        .SBTTL  ERROR MESSAGE TYPEOUT ROUTINE
(1)
(1)
(1)
(1)                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
(1)                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
(1)                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
(1)
(1)  022074                $ERRTYP:
(2)  022074  004567  001422        JSR    R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022100  001201        .WORD  $CRLF            ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022102  010046        MOV    R0,-(SP)                 ;SAVE R0
(1)  022104  005000        CLR    R0                       ;PICKUP THE ITEM INDEX
(1)  022106  156700  157002        BISB   $ITEMB,R0
(1)  022112  001007        BNE    1$                       ;IF ITEM NUMBER IS ZERO, JUST
(1)                                                         ;TYPE THE PC OF THE ERROR
(2)  022114  016746  156776        MOV    $ERRPC,-(SP)     ;;SAVE $ERRPC FOR TYPEOUT
(2)                                                         ;;ERROR ADDRESS
(4)                        ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
(4)                        ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(4)  022120  013746  177776        MOV    @#PSW, -(SP)    ;PUT THE PROCESSOR STATUS ON THE STACK
(4)  022124  004767  002570        JSR    PC,    $TYPOC   ;GO TO THE SUBROUTINE
(1)  022130  000513        BR     10$                      ;GET OUT
(1)  022132  016767  156760  157354  1$:  MOV    $ERRPC, $VERPC  ;SET UP VIRTUAL PC FOR TYPEOUT.
(1)  022140  166767  156434  157346        SUB    RELOCF, $VERPC  ;MAKE VIRTUAL IF NOT ALREADY.
(1)  022146  005300        DEC    R0                       ;ADJUST THE INDEX SO THAT IT WILL
(1)  022150  006300        ASL    R0                       ;       WORK FOR THE ERROR TABLE
(1)  022152  006300        ASL    R0
(1)  022154  006300        ASL    R0
(1)  022156  066700  157456        ADD    .ERRTB, R0       ;FORM TABLE POINTER
(1)  022162  012067  000006        MOV    (R0)+,2$                 ;PICKUP "ERROR MESSAGE" POINTER
(1)  022166  001406        BEQ    3$                       ;SKIP TYPEOUT IF NO POINTER
(2)  022170  004567  001326        JSR    R5,    $PRINT   ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
(1)  022174  000000                    2$:    .WORD   0                       ;'ERROR MESSAGE'' POINTER GOES HERE
(2)  022176  004567  001320                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022202  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022204  012067  000006            3$:     MOV     (R0)+,4$                ;PICKUP 'DATA HEADER'' POINTER
(1)  022210  001406                            BEQ     5$                      ;SKIP TYPEOUT IF 0
(2)  022212  004567  001304                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(1)  022216  000000                    4$:     .WORD   0                       ;'DATA HEADER'' POINTER GOES HERE
(2)  022220  004567  001276                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022224  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022226  010146                    5$:     MOV     R1,-(SP)                ;SAVE R1
(1)  022230  012001                            MOV     (R0)+,R1                ;PICKUP 'DATA TABLE'' POINTER
(1)  022232  001451                            BEQ     9$                      ;BR IF NO DATA TO BE TYPED
(1)  022234  066701  156340                    ADD     RELOCF, R1      ;ADJUST POINTER
(1)  022240  012000                            MOV     (R0)+,R0                ;PICKUP 'DATA FORMAT'' POINTER
(1)  022242  066700  156332                    ADD     RELOCF, R0      ;ADJUST POINTER.
(1)  022246  105720                    6$:     TSTB    (R0)+                   ;CHECK THE FORMAT
(1)  022250  001006                            BNE     7$                      ;BR IF NOT 16-BIT OCTAL
(2)  022252  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(4)                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
(4)                                    ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(4)  022254  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(4)  022260  004767  002434                    JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
(1)  022264  000426                            BR      8$
(1)  022266  100406                    7$:     BMI     17$                     ;BRANCH IF NOT DECIMAL
(2)  022270  013146                            MOV     @(R1)+,-(SP)    ;;SAVE @(R1)+ FOR TYPEOUT
(4)                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPDS ROUTINE
(4)                                    ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(4)  022272  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(4)  022276  004767  002140                    JSR     PC,     $TYPDS  ;GO TO THE SUBROUTINE
(1)  022302  000417                            BR      8$                      ;SKIP
(1)  022304  122760  177777  177777    17$:    CMPB    #-1,    -1(R0)  ;CHECK FOR 18-BIT ADDRESS FORMAT.
(1)  022312  001004                            BNE     18$                     ;BR IF NOT 18-BIT ADDRESS FORMAT.
(2)  022314  013146                            MOV     @(R1)+, -(SP)   ;PUT THE DATA ON THE STACK.
(2)  022316  004767  002640                    JSR     PC,     $TYPAD  ;DETERMINE THE PHYSICAL ADDRESS AND TYPE IT.
(1)  022322  000407                            BR      8$                      ;SKIP
(1)  022324                            18$:
(2)  022324  005046                            CLR     -(SP)                   ;CLEAR THE WORD ON THE STACK.
(2)  022326  113116                            MOVB    @(R1)+, (SP)    ;PUT THE DATA ON THE STACK.
(4)                                    ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOS ROUTINE
(4)                                    ;*  WIHTOUT USING A 'TRAP'' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(4)  022330  013746  177776                    MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(4)  022334  004767  002334                    JSR     PC,     $TYPOS  ;GO TO THE SUBROUTINE
(2)  022340     003                            .BYTE   3               ;TYPE 3 DIGITS.
(2)  022341     001                            .BYTE   1               ;TYPE LEADING ZEROS.
(1)  022342  005711                    8$:     TST     (R1)                    ;IS THERE ANOTHER NUMBER?
(1)  022344  001404                            BEQ     9$                      ;BR IF NO
(2)  022346  004567  001150                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022352  022372                            .WORD   11$             ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022354  000734                            BR      6$                      ;LOOP
(1)
(1)  022356  012601                    9$:     MOV     (SP)+,R1                ;RESTORE R1
(1)  022360  012600                    10$:    MOV     (SP)+,R0                ;RESTORE R0
(2)  022362  004567  001134                    JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022366  001201                            .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022370  000207                            RTS     PC                      ;RETURN
(1)  022372  000011                    11$:    .ASCIZ  /       /       ;TAB CHARACTER.
```

```
  (1)                                           .EVEN
 8923                                   .SBTTL  TTY INPUT ROUTINE
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   .ENABL  LSB
  (1)
  (2)                                   ;;*****************************************************************
  (1)                                   ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
  (1)                                   ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
  (1)                                   ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
  (1)                                   ;*WHEN OPERATING IN TTY FLAG MODE.
  (1)  022374  022767  000176  156536   $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
  (1)  022402  001104                          BNE     15$             ;;BRANCH IF NO
  (1)  022404  105777  156534                  TSTB    @$TKS           ;;CHAR THERE?
  (1)  022410  100101                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
  (1)  022412  117746  156530                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
  (1)  022416  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
  (1)  022422  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
  (1)  022426  001072                          BNE     15$             ;;NO, RETURN TO USER
  (1)  022430  126727  156500  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
  (1)  022436  001466                          BEQ     15$             ;;BRANCH IF YES
  (1)
  (2)  022440  004567  001056                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  022444  023321                          .WORD   $CNTLG          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)  022446                           $GTSWR:
  (2)  022446  004567  001050                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  022452  023326                          .WORD   $MSWR           ;ADDRESS OF MESSAGE TO BE TYPED
  (2)  022454  016746  155516                  MOV     SWREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
  (4)                                   ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPOC ROUTINE
  (4)                                   ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
  (4)  022460  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
  (4)  022464  004767  002230                  JSR     PC,     $TYPOC  ;GO TO THE SUBROUTINE
  (2)  022470  004567  001026                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  022474  023337                          .WORD   $MNEW           ;ADDRESS OF MESSAGE TO BE TYPED
  (1)  022476  005046                   19$:    CLR     -(SP)           ;;CLEAR COUNTER
  (1)  022500  005046                           CLR     -(SP)           ;;THE NEW SWR
  (1)  022502  105777  156436           7$:     TSTB    @$TKS           ;;CHAR THERE?
  (1)  022506  100375                          BPL     7$              ;;IF NOT TRY AGAIN
  (1)
  (1)  022510  117746  156432                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
  (1)  022514  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
  (1)
  (1)
  (1)
  (1)  022520  021627  000025           9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
  (1)  022524  001006                          BNE     10$             ;;BRANCH IF NOT
  (2)  022526  004567  000770                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  022532  023314                          .WORD   $CNTLU          ;ADDRESS OF MESSAGE TO BE TYPED
  (1)  022534  062706  000006           20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
  (1)  022540  000756                          BR      19$             ;;LET'S TRY IT AGAIN
  (1)
  (1)
  (1)  022542  021627  000015           10$:    CMP     (SP),#15        ;;IS IT A <CR>?
  (1)  022546  001023                          BNE     16$             ;;BRANCH IF NO
  (1)  022550  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
  (1)  022554  001403                          BEQ     11$             ;;BRANCH IF YES
```

```
(1)  022556  016677  000002  156354          MOV     2(SP),@SWR      ;;SAVE NEW SWR
(1)  022564  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
(1)  022570                         14$:
(2)  022570  004567  000726                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022574  001201                          .WORD   $CRLF           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022576  126727  156333  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
(1)  022604  001003                          BNE     15$             ;;BRANCH IF NOT
(1)  022606  012777  000100  156330          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
(1)  022614  000002                  15$:    RTI                     ;;RETURN
(1)  022616  004767  001142          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
(1)  022622  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
(1)  022626  002420                          BLT     18$             ;;BRANCH IF YES
(1)  022630  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
(1)  022634  003015                          BGT     18$             ;;BRANCH IF YES
(1)  022636  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
(1)  022642  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
(1)  022646  001403                          BEQ     17$             ;;BRANCH IF YES
(1)  022650  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
(1)  022652  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
(1)  022654  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
(1)  022656  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
(1)  022662  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
(1)  022666  000705                          BR      7$              ;;GET THE NEXT ONE
(1)  022670                         18$:
(2)  022670  004567  000626                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  022674  001200                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  022676  000716                          BR      20$             ;;SIMULATE CONTROL-U
(1)                                 .DSABL  LSB
(1)
(1)
(2)                                 ;;**************************************************************************
(1)                                 ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                 ;*CALL:
(1)                                 ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
(1)                                 ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
(1)                                 ;*                              ;;WITH PARITY BIT STRIPPED OFF
(1)                                 ;
(1)
(1)  022700  011646                 $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
(1)  022702  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
(1)  022710  105777  156230          1$:     TSTB    @$TKS           ;;WAIT FOR
(1)  022714  100375                          BPL     1$              ;;A CHARACTER
(1)  022716  117766  156224  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
(1)  022724  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
(1)  022732  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
(1)  022740  001013                          BNE     3$              ;;BRANCH IF NO
(1)  022742  105777  156176          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
(1)  022746  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
(1)  022750  117746  156172                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
(1)  022754  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
(1)  022760  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
(1)  022764  001366                          BNE     2$              ;;IF NOT DISCARD IT
(1)  022766  000750                          BR      1$              ;;YES, RESUME
(1)  022770  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
(1)  022776  002407                          BLT     4$              ;;BRANCH IF YES
(1)  023000  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
```

```
(1)  023006  003003                          BGT     4$                ;;BRANCH IF YES
(1)  023010  042766  000040  000004          BIC     #40,4(SP)         ;;MAKE IT UPPER CASE
(1)  023016  000002                  4$:     RTI                       ;;GO BACK TO USER
(2)                                  ;;*****************************************************************
(1)                                  ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
(1)                                  ;*CALL:
(1)                                  ;*      RDLIN                     ;;INPUT A STRING FROM THE TTY
(1)                                  ;*      RETURN HERE               ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
(1)                                  ;*                                ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
(1)
(1)  023020  010346                  $RDLIN: MOV     R3,-(SP)          ;;SAVE R3
(1)  023022  005046                          CLR     -(SP)             ;;CLEAR THE RUBOUT KEY
(1)  023024  012703  023304          1$:     MOV     #$TTYIN,R3        ;;GET ADDRESS
(1)  023030  022703  023314          2$:     CMP     #$TTYIN+8.,R3     ;;BUFFER FULL?
(1)  023034  101467                          BLOS    4$                ;;BR IF YES
(3)                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDCHR ROUTINE
(3)                                  ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(3)  023036  013746  177776                  MOV     @#PSW,  -(SP)     ;PUT THE PROCESSOR STATUS ON THE STACK
(3)  023042  004767  177632                  JSR     PC,     $RDCHR    ;GO TO THE SUBROUTINE
(1)  023046  112613                          MOVB    (SP)+,(R3)        ;;GET CHARACTER
(1)  023050  122713  000177          10$:    CMPB    #177,(R3)         ;;IS IT A RUBOUT
(1)  023054  001024                          BNE     5$                ;;BR IF NO
(1)  023056  005716                          TST     (SP)              ;;IS THIS THE FIRST RUBOUT?
(1)  023060  001010                          BNE     6$                ;;BR IF NO
(1)  023062  112767  000134  000212         MOVB    #'\,9$            ;;TYPE A BACK SLASH
(2)  023070  004567  000426                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023074  023302                          .WORD   9$                ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023076  012716  177777                  MOV     #-1,(SP)          ;;SET THE RUBOUT KEY
(1)  023102  005303                  6$:     DEC     R3                ;;BACKUP BY ONE
(1)  023104  020327  023304                  CMP     R3,#$TTYIN        ;;STACK EMPTY?
(1)  023110  103441                          BLO     4$                ;;BR IF YES
(1)  023112  111367  000164                  MOVB    (R3),9$           ;;SETUP TO TYPEOUT THE DELETED CHAR.
(2)  023116  004567  000400                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023122  023302                          .WORD   9$                ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023124  000741                          BR      2$                ;;GO READ ANOTHER CHAR.
(1)  023126  005716                  5$:     TST     (SP)              ;;RUBOUT KEY SET?
(1)  023130  001407                          BEQ     7$                ;;BR IF NO
(1)  023132  112767  000134  000142         MOVB    #'\,9$            ;;TYPE A BACK SLASH
(2)  023140  004567  000356                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023144  023302                          .WORD   9$                ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023146  005016                          CLR     (SP)              ;;CLEAR THE RUBOUT KEY
(1)  023150  122713  000025          7$:     CMPB    #25,(R3)          ;;IS CHARACTER A CTRL U?
(1)  023154  001004                          BNE     8$                ;;BR IF NO
(2)  023156  004567  000340                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023162  023314                          .WORD   $CNTLU            ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023164  000717                          BR      1$                ;;GO START OVER
(1)  023166  122713  000022          8$:     CMPB    #22,(R3)          ;;IS CHARACTER A '^R'?
(1)  023172  001014                          BNE     3$                ;;BRANCH IF NO
(1)  023174  105013                          CLRB    (R3)              ;;CLEAR THE CHARACTER
(2)  023176  004567  000320                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023202  001201                          .WORD   $CRLF             ;ADDRESS OF MESSAGE TO BE TYPED
(2)  023204  004567  000312                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023210  023304                          .WORD   $TTYIN            ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023212  000706                          BR      2$                ;;GO PICKUP ANOTHER CHACTER
(1)  023214                          4$.
(2)  023214  004567  000302                  JSR     R5,     $PRINT    ;GO PRINT OUT THE FOLLOWING MESSAGE.
```

```
(2)  023220  001200                          .WORD   $QUES           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023222  000700                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
(1)  023224  111367  000052          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
(2)  023230  004567  000266                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023234  023302                          .WORD   9$              ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023236  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
(1)  023242  001272                          BNE     2$              ;;LOOP IF NOT RETURN
(1)  023244  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
(2)  023250  004567  000246                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  023254  001202                          .WORD   $LF             ;ADDRESS OF MESSAGE TO BE TYPED
(1)  023256  005726                          TST     (SP)+           ;;CLEAN RUBOUT KEY FROM THE STACK
(1)  023260  012603                          MOV     (SP)+,R3        ;;RESTORE R3
(1)  023262  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
(1)  023264  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
(1)  023272  012766  023304  000004          MOV     #$TTYIN,4(SP)
(1)  023300  000002                          RTI                     ;;RETURN
(1)  023302     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
(1)  023303     000                          .BYTE   0               ;;TERMINATOR
(1)  023304  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
(1)  023314  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL 'U'
(1)  023321     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL 'G'
(1)  023326  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
(1)  023334  020075     000
(1)  023337     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
(1)  023344  036440  000040
8924                                 .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
(1)
(2)                                  ;;*****************************************************************
(1)                                  ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
(1)                                  ;*CHANGE IT TO BINARY.
(1)                                  ;*THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
(1)                                  ;*OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
(1)                                  ;*FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
(1)                                  ;*THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
(1)                                  ;*CALL:
(1)                                  ;*      RDOCT                   ;;READ AN OCTAL NUMBER
(1)                                  ;*      RETURN HERE             ;;LOW ORDER BITS ARE ON TOP OF THE STACK
(1)                                  ;*                              ;;HIGH ORDER BITS ARE IN $HIOCT
(1)
(1)  023350  011646                  $RDOCT: MOV     (SP),-(SP)      ;;PROVIDE SPACE FOR THE
(1)  023352  016666  000004  000002          MOV     4(SP),2(SP)     ;;INPUT NUMBER
(3)  023360  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  023362  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  023364  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(1)  023366                          1$:
(3)                                  ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $RDLIN ROUTINE
(3)                                  ;*  WIHTOUT USING A 'TRAP' INSTRUCTION AS CALLED FOR BY **SYSMAC**.
(3)  023366  013746  177776                  MOV     @#PSW,  -(SP)   ;PUT THE PROCESSOR STATUS ON THE STACK
(3)  023372  004767  177422                  JSR     PC,     $RDLIN  ;GO TO THE SUBROUTINE
(1)  023376  012600                          MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
(1)  023400  010067  000102                  MOV     R0,5$           ;;AND SAVE IT
(1)  023404  005001                          CLR     R1              ;;CLEAR DATA WORD
(1)  023406  005002                          CLR     R2
(1)  023410  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
(1)  023412  001420                          BEQ     3$              ;;IF ZERO GET OUT
(1)  023414  122716  000060                  CMPB    #'0,(SP)        ;;MAKE SURE THIS CHARACTER
```

```
 (1)  023420  003026                        BGT     4$                  ;;IS AN OCTAL DIGIT
 (1)  023422  122716  000067                CMPB    #'7,(SP)
 (1)  023426  002423                        BLT     4$
 (1)  023430  006301                        ASL     R1                  ;;*2
 (1)  023432  006102                        ROL     R2
 (1)  023434  006301                        ASL     R1                  ;;*4
 (1)  023436  006102                        ROL     R2
 (1)  023440  006301                        ASL     R1                  ;;*8
 (1)  023442  006102                        ROL     R2
 (1)  023444  042716  177770                BIC     #^C7,(SP)           ;;STRIP THE ASCII JUNK
 (1)  023450  062601                        ADD     (SP)+,R1            ;;ADD IN THIS DIGIT
 (1)  023452  000756                        BR      2$                  ;;LOOP
 (1)  023454  005726                3$:     TST     (SP)+               ;;CLEAN TERMINATOR FROM STACK
 (1)  023456  010166  000012                MOV     R1,12(SP)           ;;SAVE THE RESULT
 (1)  023462  010267  000032                MOV     R2,$HIOCT
 (3)  023466  012602                        MOV     (SP)+,R2            ;;POP STACK INTO R2
 (3)  023470  012601                        MOV     (SP)+,R1            ;;POP STACK INTO R1
 (3)  023472  012600                        MOV     (SP)+,R0            ;;POP STACK INTO R0
 (1)  023474  000002                        RTI                         ;;RETURN
 (1)  023476  005726                4$:     TST     (SP)+               ;;CLEAN PARTIAL FROM STACK
 (1)  023500  105010                        CLRB    (R0)                ;;SET A TERMINATOR
 (2)  023502  004567  000014                JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (1)  023506  000000                5$:     .WORD   0
 (2)  023510  004567  000006                JSR     R5,     $PRINT      ;GO PRINT OUT THE FOLLOWING MESSAGE.
 (2)  023514  001200                        .WORD   $QUES               ;ADDRESS OF MESSAGE TO BE TYPED
 (1)  023516  000723                        BR      1$                  ;;TRY AGAIN
 (1)  023520  000000                $HIOCT: .WORD   0                   ;;HIGH ORDER BITS GO HERE
 8925
 8926                               ;;*****************************************************************
 8927                               ;* SUBROUTINE TO PASS RELOCATED MESSAGE ADDRESSES TO THE $TYPE ROUTINE.
 8928                               ;*  CALL:      JSR     R5,     $PRINT
 8929                               ;*             <MESSAGE VIRTUAL ADDRESS>
 8930                               ;;*****************************************************************
 8931  023522  012567  000016       $PRINT: MOV     (R5)+,  1$          ;GET THE MESSAGE VIRTUAL ADDRESS.
 8932  023526  066767  155046  000010        ADD     RELOCF, 1$          ;MAKE IT PHYSICAL.
 8933                               ;*  THE NEXT TWO INSTRUCTIONS PROVIDE AN INTERFACE TO THE $TYPE ROUTINE
  (1)                               ;*  WIHTOUT USING A "TRAP" INSTRUCTION AS CALLED FOR BY **SYSMAC**.
  (1)  023534  013746  177776                MOV     @#PSW,  -(SP)       ;PUT THE PROCESSOR STATUS ON THE STACK
  (1)  023540  004767  000004                JSR     PC,     $TYPE       ;GO TO THE SUBROUTINE
 8934  023544  000000                1$:     .WORD   0                   ;CONTAINS THE PHYSICAL MESSAGE ADDRESS.
 8935  023546  000205                        RTS     R5                  ;RETURN.
 8936
 8937                               .SBTTL  TYPE ROUTINE
  (1)
  (2)                               ;;*****************************************************************
  (1)                               ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
  (1)                               ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
  (1)                               ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
  (1)                               ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
  (1)                               ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
  (1)                               ;*
  (1)                               ;*CALL:
  (1)                               ;*1) USING A TRAP INSTRUCTION
  (1)                               ;*      TYPE    ,MESADR             ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
  (1)                               ;*OR
  (1)                               ;*      TYPE
```

```
(1)                                     ;*      MESADR
(1)                                     ;*
(1)
(1)  023550  105767  155403          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
(1)  023554  100002                          BPL     1$              ;;BR IF YES
(1)  023556  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
(1)  023560  000430                          BR      3$              ;;LEAVE
(1)  023562  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
(1)  023564  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
(1)  023570  122767  000001  155426          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
(1)  023576  001011                          BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
(1)  023600  132767  000100  155417          BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
(1)  023606  001405                          BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
(1)  023610  010067  000004                  MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
(1)  023614  004767  000222                  JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
(1)  023620  000000                  61$:    .WORD   0               ;;MESSAGE ADDRESS
(1)  023622  132767  000040  155375  62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
(1)  023630  001003                          BNE     60$             ;;YES,SKIP TYPE OUT
(1)  023632  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
(1)  023634  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
(1)  023636  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
(1)  023640  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
(1)  023642  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
(1)  023646  000002                          RTI                     ;;RETURN
(1)  023650  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
(1)  023654  001431                          BEQ     8$
(1)  023656  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
(1)  023662  001007                          BNE     5$
(1)  023664  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
(2)  023666  004567  177630                  JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(1)  023672  001201                          $CRLF
(1)  023674  105067  000130                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
(1)  023700  000754                          BR      2$              ;;GET NEXT CHARACTER
(1)  023702  004767  000056          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
(1)  023706  126726  155244          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
(1)  023712  001347                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
(1)  023714  016746  155234                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
(1)                                                                  ;;AND THE NULL CHAR.
(1)  023720  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
(1)  023724  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
(1)  023726  004767  000032                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
(1)  023732  105367  000072                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
(1)  023736  000770                          BR      7$              ;;LOOP
(1)
(1)                                     ;HORIZONTAL TAB PROCESSOR
(1)
(1)  023740  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
(1)  023744  004767  000014          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
(1)  023750  132767  000007  000052          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
(1)  023756  001372                          BNE     9$              ;;TAB STOP
(1)  023760  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
(1)  023762  000723                          BR      2$              ;;GET NEXT CHARACTER
(1)  023764  105777  155160          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
(1)  023770  100375                          BPL     $TYPEC
(1)  023772  116677  000002  155152          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
(1)  024000  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
```

```
(1)  024006  001003                          BNE     1$              ;;BRANCH IF NO
(1)  024010  105067  000014                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
(1)  024014  000406                          BR      $TYPEX          ;;EXIT
(1)  024016  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
(1)  024024  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
(1)  024026  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
(1)  024030  000000                $CHARCNT:.WORD   0               ;;CHARACTER COUNT STORAGE
(1)  024032  000207                $TYPEX: RTS     PC
(1)
8938                                         .SBTTL  APT COMMUNICATIONS ROUTINE
(1)
(2)                                          ;;**************************************************************************
(1)  024034  112767  000001  000376  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
(1)  024042  112767  000001  000366  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
(1)  024050  000403                          BR      $ATYC
(1)  024052  112767  000001  000360  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
(1)  024060                          $ATYC:
(3)  024060  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  024062  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(1)  024064  105767  000346                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
(1)  024070  001450                          BEQ     5$              ;;IF NOT:  BR
(1)  024072  122767  000001  155124          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
(1)  024100  001031                          BNE     3$              ;;IF NOT:  BR
(1)  024102  132767  000100  155115          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
(1)  024110  001425                          BEQ     3$              ;;IF NOT:  BR
(1)  024112  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
(1)  024116  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  024124  005767  155054          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
(1)  024130  001375                          BNE     1$              ;;IF NOT:  WAIT
(1)  024132  010067  155062                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
(1)  024136  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
(1)  024140  001376                          BNE     2$
(1)  024142  166700  155052                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
(1)  024146  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
(1)  024150  010067  155046                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
(1)  024154  012767  000004  155022          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
(1)  024162  000413                          BR      5$
(1)  024164  017667  000004  000016  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
(1)  024172  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
(3)  024200  016746  153572                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
(1)  024204  004767  177340                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
(1)  024210  000000                  4$:     .WORD   0
(1)  024212                          5$:
(1)  024212  105767  000221                  TSTB    $LFLG           ;;SHOULD LOG AN ERROR?
(1)  024216  001422                          BEQ     10$             ;;IF NOT:  BR
(1)  024220  017600  000004                  MOV     @4(SP),R0               ;;GET ERROR #
(1)  024224  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  024232  012701  001344                  MOV     #$ASTAT,R1      ;;POINT TO TABLE START
(1)  024236  005711                  6$:     TST     (R1)            ;;END OF TABLE?
(1)  024240  100404                          BMI     8$              ;;IF SO:  BR
(1)  024242  020021                          CMP     R0,(R1)+        ;;PROPER ENTRY?
(1)  024244  001406                          BEQ     9$              ;;IF SO:  BR
(1)  024246  005721                          TST     (R1)+           ;;MOVE PAST COUNTER WORD
(1)  024250  000772                          BR      6$              ;;KEEP LOOKING
(1)  024252  026701  155234          8$:     CMP     $APTR,R1        ;;TABLE FULL?
(1)  024256  001402                          BEQ     10$             ;;IF SO:  BR -- NO MORE ROOM
```

```
(1)  024260  010021                        MOV     R0,(R1)+        ;;SET UP NEW ENTRY
(1)  024262  005211                9$:     INC     (R1)            ;;BUMP ERROR COUNT
(1)  024264  105767  000150        10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
(1)  024270  001416                        BEQ     12$             ;;IF NOT:  BR
(1)  024272  005767  154726                TST     $ENV            ;;RUNNING UNDER APT?
(1)  024276  001413                        BEQ     12$             ;;IF NOT:  BR
(1)  024300  005767  154700        11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
(1)  024304  001375                        BNE     11$             ;;IF NOT:  WAIT
(1)  024306  017667  000004  154672        MOV     @4(SP),$FATAL   ;;GET ERROR #
(1)  024314  062766  000002  000004        ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
(1)  024322  005267  154656                INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
(1)  024326  105067  000106        12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
(1)  024332  105067  000101                CLRB    $LFLG           ;;CLEAR LOG FLAG
(1)  024336  105067  000074                CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
(3)  024342  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  024344  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  024346  000207                        RTS     PC              ;;RETURN
(1)  024350                        $ATY6:
(3)  024350  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(1)  024352  016700  155134                MOV     $APTR,R0
(1)  024356  162700  001344                SUB     #$ASTAT,R0      ;;GET SIZE OF STAT TABLE
(1)  024362  005767  154616        1$:     TST     $MSGTY          ;;SEE IF DONE LAST COMMUNICATION
(1)  024366  001375                        BNE     1$              ;;IF NOT:  WAIT
(1)  024370  010067  154626                MOV     R0,$MSGLG       ;;SET MESSAGE LENGTH
(1)  024374  012767  001344  154616        MOV     #$ASTAT,$MSGAD  ;;SET MESSAGE ADDR.
(1)  024402  012767  000002  154574        MOV     #2,$MSGTY       ;;TELL APT TO TAKE STATS.
(3)  024410  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  024412  000207                        RTS     PC              ;;RETURN
(1)  024414                        $ATY7:
(3)  024414  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(1)  024416  012701  001344                MOV     #$ASTAT,R1      ;;GET START OF TABLE
(1)  024422  005721                1$:     TST     (R1)+           ;;END OF TABLE?
(1)  024424  100402                        BMI     2$              ;;IF SO:  BR
(1)  024426  005021                        CLR     (R1)+           ;;CLEAR ERROR COUNT
(1)  024430  000774                        BR      1$              ;;KEEP CLEARING
(1)  024432                        2$:
(3)  024432  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
(1)  024434  000207                        RTS     PC              ;;RETURN
(1)  024436     000                $MFLG:  .BYTE   0               ;;MESSG. FLAG
(1)  024437     000                $LFLG:  .BYTE   0               ;;LOG FLAG
(1)  024440     000                $FFLG:  .BYTE   0               ;;FATAL FLAG
(1)          024442                        .EVEN
(1)          000200                APTSIZE=200
(1)          000001                APTENV=001
(1)          000100                APTSPOOL=100
(1)          000040                APTCSUP=040
8939                               ;;**************************************************************

(1)
(1)                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
(1)
(1)                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
(1)                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
(1)                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
(1)                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
(1)                                ;*REPLACED WITH SPACES.
(1)                                ;*CALL:
```

```
(1)                                   ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
(1)                                   ;*      TYPDS                   ;;GO TO THE ROUTINE
(1)
(1)  024442                           $TYPDS:
(3)  024442  010046                           MOV     R0,-(SP)        ;;PUSH R0 ON STACK
(3)  024444  010146                           MOV     R1,-(SP)        ;;PUSH R1 ON STACK
(3)  024446  010246                           MOV     R2,-(SP)        ;;PUSH R2 ON STACK
(3)  024450  010346                           MOV     R3,-(SP)        ;;PUSH R3 ON STACK
(3)  024452  010546                           MOV     R5,-(SP)        ;;PUSH R5 ON STACK
(1)  024454  012746  020200                   MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
(1)  024460  016605  000020                   MOV     20(SP),R5       ;;GET THE INPUT NUMBER
(1)  024464  100004                           BPL     1$              ;;BR IF INPUT IS POS.
(1)  024466  005405                           NEG     R5              ;;MAKE THE BINARY NUMBER POS.
(1)  024470  112766  000055  000001           MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
(1)  024476  016700  154076           1$:     MOV     RELOCF, R0      ;GET RELOCATION FACTOR.
(1)  024502  012703  024664                   MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
(1)  024506  060003                           ADD     R0,     R3      ;ADD IN RELOCATION FACTOR.
(1)  024510  112723  000040                   MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
(1)  024514  005002                   2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
(1)  024516  016001  024654                   MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
(1)  024522  160105                   3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
(1)  024524  002402                           BLT     4$              ;;BR IF DONE
(1)  024526  005202                           INC     R2              ;;INCREASE THE BCD DIGIT BY 1
(1)  024530  000774                           BR      3$
(1)  024532  060105                   4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
(1)  024534  005702                           TST     R2              ;;CHECK IF BCD DIGIT=0
(1)  024536  001002                           BNE     5$              ;;FALL THROUGH IF 0
(1)  024540  105716                           TSTB    (SP)            ;;STILL DOING LEADING 0'S?
(1)  024542  100407                           BMI     7$              ;;BR IF YES
(1)  024544  106316                   5$:     ASLB    (SP)            ;;MSD?
(1)  024546  103003                           BCC     6$              ;;BR IF NO
(1)  024550  116663  000001  177777           MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
(1)  024556  052702  000060           6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
(1)  024562  052702  000040           7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
(1)  024566  110223                           MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
(1)  024570  005720                           TST     (R0)+           ;;JUST INCREMENTING
(1)  024572  020067  155044                   CMP     R0,     .EIGHT  ;CHECK THE TABLE INDEX
(1)  024576  103746                           BLO     2$              ;GO DO THE NEXT DIGIT
(1)  024600  101002                           BHI     8$              ;;GO TO EXIT
(1)  024602  010502                           MOV     R5,R2           ;;GET THE LSD
(1)  024604  000764                           BR      6$              ;;GO CHANGE TO ASCII
(1)  024606  105726                   8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
(1)  024610  100003                           BPL     9$              ;;BR IF NO
(1)  024612  116663  177777  177776           MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
(1)  024620  105013                   9$:     CLRB    (R3)            ;;SET THE TERMINATOR
(3)  024622  012605                           MOV     (SP)+,R5        ;;POP STACK INTO R5
(3)  024624  012603                           MOV     (SP)+,R3        ;;POP STACK INTO R3
(3)  024626  012602                           MOV     (SP)+,R2        ;;POP STACK INTO R2
(3)  024630  012601                           MOV     (SP)+,R1        ;;POP STACK INTO R1
(3)  024632  012600                           MOV     (SP)+,R0        ;;POP STACK INTO R0
(2)  024634  004567  176662                   JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
(2)  024640  024664                           .WORD   $DBLK           ;ADDRESS OF MESSAGE TO BE TYPED
(1)  024642  016666  000002  000004           MOV     2(SP),4(SP)     ;;ADJUST THE STACK
(1)  024650  012616                           MOV     (SP)+,(SP)
(1)  024652  000002                           RTI                     ;;RETURN TO USER
(1)  024654  023420                   $DTBL:  10000.
```

```
(1)  024656  001750                    1000.
(1)  024660  000144                    100.
(1)  024662  000012                    10.
(1)  024664  000004            $DBLK:  .BLKW   4
8940
(1)                            .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
(1)
(2)                            ;;*****************************************************************
(1)                            ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
(1)                            ;*OCTAL (ASCII) NUMBER AND TYPE IT.
(1)                            ;*$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
(1)                            ;*CALL:
(1)                            ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                            ;*      TYPOS                   ;;CALL FOR TYPEOUT
(1)                            ;*      .BYTE   N               ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
(1)                            ;*      .BYTE   M               ;;M=1 OR 0
(1)                            ;*                                      ;;1=TYPE LEADING ZEROS
(1)                            ;*                                      ;;0=SUPPRESS LEADING ZEROS
(1)                            ;*
(1)                            ;*$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
(1)                            ;*$TYPOS OR $TYPOC
(1)                            ;*CALL:
(1)                            ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                            ;*      TYPON                   ;;CALL FOR TYPEOUT
(1)                            ;*
(1)                            ;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
(1)                            ;*CALL:
(1)                            ;*      MOV     NUM,-(SP)       ;;NUMBER TO BE TYPED
(1)                            ;*      TYPOC                   ;;CALL FOR TYPEOUT
(1)
(1)  024674  017646  000000    $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
(1)  024700  116667  000001  000213    MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
(1)  024706  112667  000211            MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
(1)  024712  062716  000002            ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
(1)  024716  000406                    BR      $TYPON
(1)  024720  112767  000001  000173  $TYPOC: MOVB  #1,$OFILL    ;;SET THE ZERO FILL SWITCH
(1)  024726  112767  000006  000167          MOVB  #6,$OMODE+1  ;;SET FOR SIX(6) DIGITS
(1)  024734  112767  000005  000156  $TYPON: MOVB  #5,$OCNT     ;;SET THE ITERATION COUNT
(1)  024742  010346                    MOV     R3,-(SP)        ;;SAVE R3
(1)  024744  010446                    MOV     R4,-(SP)        ;;SAVE R4
(1)  024746  010546                    MOV     R5,-(SP)        ;;SAVE R5
(1)  024750  116704  000147            MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
(1)  024754  005404                    NEG     R4
(1)  024756  062704  000006            ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
(1)  024762  110467  000134            MOVB    R4,$OMODE       ;;SAVE IT FOR USE
(1)  024766  116704  000127            MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
(1)  024772  016605  000012            MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
(1)  024776  005003                    CLR     R3              ;;CLEAR THE OUTPUT WORD
(1)  025000  006105            1$:     ROL     R5              ;;ROTATE MSB INTO "C"
(1)  025002  000404                    BR      3$              ;;GO DO MSB
(1)  025004  006105            2$:     ROL     R5              ;;FORM THIS DIGIT
(1)  025006  006105                    ROL     R5
(1)  025010  006105                    ROL     R5
(1)  025012  010503                    MOV     R5,R3
(1)  025014  006103            3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
(1)  025016  105367  000100            DECB    $OMODE          ;;TYPE THIS DIGIT?
(1)  025022  100017                    BPL     7$              ;;BR IF NO
```

```
  (1)  025024  042703  177770            BIC     #177770,R3      ;;GET RID OF JUNK
  (1)  025030  001002                    BNE     4$              ;;TEST FOR 0
  (1)  025032  005704                    TST     R4              ;;SUPPRESS THIS 0?
  (1)  025034  001403                    BEQ     5$              ;;BR IF YES
  (1)  025036  005204            4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
  (1)  025040  052703  000060            BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
  (1)  025044  052703  000040            BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
  (1)  025050  110367  000042            MOVB    R3,8$           ;;SAVE FOR TYPING
  (2)  025054  004567  176442            JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  025060  025116                    .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
  (1)  025062  105367  000032    7$:     DECB    $OCNT           ;;COUNT BY 1
  (1)  025066  003346                    BGT     2$              ;;BR IF MORE TO DO
  (1)  025070  002402                    BLT     6$              ;;BR IF DONE
  (1)  025072  005204                    INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
  (1)  025074  000743                    BR      2$              ;;GO DO THE LAST DIGIT
  (1)  025076  012605            6$:     MOV     (SP)+,R5        ;;RESTORE R5
  (1)  025100  012604                    MOV     (SP)+,R4        ;;RESTORE R4
  (1)  025102  012603                    MOV     (SP)+,R3        ;;RESTORE R3
  (1)  025104  016666  000002  000004    MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
  (1)  025112  012616                    MOV     (SP)+,(SP)
  (1)  025114  000002                    RTI                     ;;RETURN
  (1)  025116     000            8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
  (1)  025117     000                    .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
  (1)  025120     000            $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
  (1)  025121     000            $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
  (1)  025122  000000            $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 8941                            ;ERROR TRAP SERVICE ROUTINE
 8942  025124  005727            ERRTRP: TST     (PC)+           ;CHECK IF PREV TRAP TO 4 REPORTED
 8943  025126  000000            1$:     .WORD   0               ;CONTAINS ERROR REPORTED FLAG
 8944  025130  001010                    BNE     2$              ;BRANCH IF NOT REPORTED
 8945  025132  005267  177770            INC     1$              ;SET DOUBLE TRAP FLAG.
 8946  025136  011667  154024            MOV     (SP),   $TMP3   ;SAVE THE BAD PC FOR TYPOUT.
 8947  025142  004767  174502            JSR     PC,     $ERROR  ;*** ERROR *** (GO TYPE A MESSAGE)
  (1)  025146  000031                    .WORD   31              ;ERROR TYPE CODE.
 8948  025150  000401                    BR      3$              ;SKIP HALT
 8949  025152  000000            2$:     HALT                    ;ERROR! SECOND TRAP TO 4 OCCURRED
 8950                                                            ;BEFORE FIRST WAS PRINTED
 8951  025154  005067  177746    3$:     CLR     1$
 8952  025160  000002                    RTI                     ;RETURN TO PROGRAM AND TRY TO RECOVER
 8953
 8954                            .SBTTL  PHYSICAL ADDRESS TYPE ROUTINE
 8955                            ;*  ROUTINE TO TYPE A PHYSICAL ADDRESS (18 BITS).
 8956  025162                    $TYPAD:
  (2)  025162  010046                    MOV     R0,-(SP)        ;;PUSH R0 ON STACK
  (2)  025164  010146                    MOV     R1,-(SP)        ;;PUSH R1 ON STACK
  (2)  025166  010246                    MOV     R2,-(SP)        ;;PUSH R2 ON STACK
  (2)  025170  010346                    MOV     R3,-(SP)        ;;PUSH R3 ON STACK
 8957  025172  016602  000012            MOV     12(SP), R2      ;GET BASE ADDRESS
 8958  025176  005003                    CLR     R3              ;WORKING & INDEX REGISTER
 8959  025200  005767  153402            TST     MMAVA           ;CHECK FOR MEM MGMT AVAILABLE
 8960  025204  001430                    BEQ     1$              ;BRANCH IF NO MEM MGMT
 8961  025206  032737  000001  177572    BIT     #1,     @#SR0   ;CHECK IF MEM MGMT ENABLED
 8962  025214  001424                    BEQ     1$              ;BRANCH IF MEM MGMT NOT ENABLED
 8963  025216  010201                    MOV     R2,     R1      ;COPY VIRTUAL ADR
 8964  025220  006101                    ROL     R1              ;SHUFFLE BITS 13,14,15 INTO 1,2,3
 8965  025222  006101                    ROL     R1
```

```
 8966  025224  006101                        ROL     R1
 8967  025226  006101                        ROL     R1
 8968  025230  006101                        ROL     R1
 8969  025232  042701  177761                BIC     #177761, R1     ;CLR ALL EXCEPT BITS 1,2,3
 8970  025236  062701  172340                ADD     #KIPAR0, R1.    ;SET TO APPROPRIATE PAR
 8971  025242  011101                        MOV     (R1),   R1      ;GET CONTENTS OF PAR
 8972  025244  012700  000006                MOV     #6,     R0      ;SET UP COUNTER
 8973  025250  006301                4$:     ASL     R1              ;SHIFT PAR
 8974  025252  006103                        ROL     R3              ;SAVE OVERFLOW BITS
 8975  025254  077003                        SOB     R0,     4$      ;COUNT SIX SHIFTS
 8976  025256  042702  160000                BIC     #160000, R2     ;SAVE BANK BITS
 8977  025262  060102                        ADD     R1,     R2      ;COMPUTE PHYSICAL ADDRESS
 8978  025264  005503                        ADC     R3              ;MAKE SURE CARRY ISN'T LOST!
 8979  025266  006302                1$:     ASL     R2              ;FIRST DIGIT TO R3
 8980  025270  006103                        ROL     R3
 8981  025272  012700  000006                MOV     #6,R0           ;DIGIT COUNT
 8982  025276  000404                        BR      3$              ;PRINT FIRST DIGIT
 8983  025300  006302                2$:     ASL     R2
 8984  025302  006103                        ROL     R3
 8985  025304  005301                        DEC     R1
 8986  025306  001374                        BNE     2$
 8987  025310  012701  000003        3$:     MOV     #3,R1           ;DIGIT SHIFT COUNT
 8988  025314  062703  000060                ADD     #60,    R3      ;MAKE IT AN ASCII DIGIT
 8989  025320  110367  000036                MOVB    R3,     8$      ;LOAD DIGIT INTO MESSAGE
 8990  025324  004567  176172                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  025330  025362                        .WORD   8$              ;ADDRESS OF MESSAGE TO BE TYPED
 8991  025332  005003                        CLR     R3              ;CLEAR INDEX
 8992  025334  005300                        DEC     R0              ;DEC DIGIT COUNT
 8993  025336  001360                        BNE     2$
 8994  025340  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
  (2)  025342  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
  (2)  025344  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
  (2)  025346  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
 8995  025350  012616                        MOV     (SP)+,  (SP)    ;ADJUST THE STACK TO CLEAR DATA
 8996  025352  004567  176144                JSR     R5,     $PRINT  ;GO PRINT OUT THE FOLLOWING MESSAGE.
  (2)  025356  027015                        .WORD   FILL?           ;ADDRESS OF MESSAGE TO BE TYPED
 8997  025360  000207                        RTS     PC              ;RETURN
 8998  025362     000                8$:     .BYTE   0               ;ONE DIGIT MESSAGE BUFFER
 8999  025363     000                        .BYTE   0               ;MESSAGE TERMINATOR
 9000
 9001                                        .SBTTL  STANDARD PROGRAM MESSAGES
 9002                                ;;**********************************************************************
 9003                                ;VARIOUS MESSAGE PRINTOUTS USED THRUOUT
 9004                                ;THE PROGRAM
 9005                                ;;**********************************************************************
 9006  025364  005015  052113  030461 MMAMES: .ASCIZ  <15><12>'KT11 (MEMORY MANAGEMENT) AVAILABLE'
       025372  024040  042515  047515
       025400  054522  046440  047101
       025406  043501  046505  047105
       025414  024524  040440  040526
       025422  046111  041101  042514
       025430     000
 9007  025431     015  046412  046505 MEMMES: .ASCIZ  <15><12>'MEMORY MAP:'
       025436  051117  020131  040515
       025444  035120     000
 9008  025447     015  041012  052131 BYTMES: .ASCIZ  <15><12>'BYTE MEMORY MAP:'
```

```
                025454  020105  042515  047515
                025462  054522  046440  050101
                025470  000072
 9009           025472  005015  040520  044522  MTMAP:  .ASCIZ  <15><12>'PARITY MEMORY MAP:'
                025500  054524  046440  046505
                025506  051117  020131  040515
                025514  035120     000
 9010           025517     015  043012  047522  FROM:   .ASCIZ  <15><12>'FROM '
                025524  020115     000
 9011           025527     040  047524  000040  TO:     .ASCIZ  ' TO '
 9012           025534  005015  047111  052523  INSUFF: .ASCIZ  <15><12>'INSUFFICIENT MEMORY...FIRST 16K NOT ALL THERE!'
                025542  043106  041511  042511
                025550  052116  046440  046505
                025556  051117  027131  027056
                025564  044506  051522  020124
                025572  033061  020113  047516
                025600  020124  046101  020114
                025606  044124  051105  020505
                025614     000
 9013           025615     015  047012  020117  MTR:    .ASCIZ  <15><12>'NO PARITY REGISTERS FOUND'
                025622  040520  044522  054524
                025630  051040  043505  051511
                025636  042524  051522  043040
                025644  052517  042116     000
 9014           025651     015  051012  051505  PWRMSG: .ASCIZ  <15><12>'RESTARTING AFTER A POWER FAILURE'<15><12>
                025656  040524  052122  047111
                025664  020107  043101  042524
                025672  020122  020101  047520
                025700  042527  020122  040506
                025706  046111  051125  006505
                025714  000012
 9015           025716  005015  047516  050040  NOPES:  .ASCIZ  <15><12>'NO PARITY ERRORS FOUND ON MEMORY SCAN'<15><12>
                025724  051101  052111  020131
                025732  051105  047522  051522
                025740  043040  052517  042116
                025746  047440  020116  042515
                025754  047515  054522  051440
                025762  040503  006516  000012
 9016           025770  005015  051120  043517  PROREL: .ASCII  <15><12>'PROGRAM NOW RESIDES BACK AT 0 TO 8K'
                025776  040522  020115  047516
                026004  020127  042522  044523
                026012  042504  020123  040502
                026020  045503  040440  020124
                026026  020060  047524  034040
                026034     113
 9017           026035     015  044012  052111          .ASCIZ  <15><12>'HIT CONTINUE FOR NORMAL RUNNING'<15><12>
                026042  041440  047117  044524
                026050  052516  020105  047506
                026056  020122  047516  046522
                026064  046101  051040  047125
                026072  044516  043516  005015
                026100     000
 9018           026101     015  051012  043505  MX1:    .ASCIZ  <15><12>'REGISTER AT '
                026106  051511  042524  020122
                026114  052101  000040
 9019           026120  041440  047117  051124  MX2:    .ASCIZ  ' CONTROLS '
```

```
                026126  046117  020123     000
 9020           026133     015  041412  051117  MX3:    .ASCIZ  <15><12>'CORE PARITY '
                026140  020105  040520  044522
                026146  054524  000040
 9021           026152  005015  047515  020123  MX4:    .ASCIZ  <15><12>'MOS PARITY '
                026160  040520  044522  054524
                026166  000040
 9022           026170  005015  051515  030461  MX5:    .ASCIZ  <15><12>'MS11-K CSR '
                026176  045455  041440  051123
                026204  000040
 9023           026206  051515  030461  045455  MX6:    .ASCIZ  'MS11-K MEMORY PRESENT.' TO COMPLETELY TEST RUN DZMML...'
                026214  046440  046505  051117
                026222  020131  051120  051505
                026230  047105  020524  020041
                026236  047524  041440  046517
                026244  046120  052105  046105
                026252  020131  042524  052123
                026260  051040  047125  042040
                026266  046532  046115  027056
                026274  000056
 9024           026276  005015  047516  046440  NOMEM:  .ASCIZ  <15><12>'NO MEMORY FOUND.'
                026304  046505  051117  020131
                026312  047506  047125  027104
                026320     000
 9025           026321     015  005012  044412  FADMES: .ASCII  <15><12><12><12>'INPUT ALL PARAMETERS IN OCTAL.'
                026326  050116  052125  040440
                026334  046114  050040  051101
                026342  046501  052105  051105
                026350  020123  047111  047440
                026356  052103  046101     056
 9026           026363     015  043012  051111          .ASCIZ  <15><12>'FIRST ADDRESS:  '
                026370  052123  040440  042104
                026376  042522  051523  020072
                026404  000040
 9027           026406  005015  040514  052123  LADMES: .ASCIZ  <15><12>'LAST ADDRESS:   '
                026414  040440  042104  042522
                026422  051523  020072  020040
                026430     000
 9028           026431     015  037412  042101  BADADR: .ASCIZ  <15><12>'?ADDRESS IN UNMAPPED BANK?'
                026436  051104  051505  020123
                026444  047111  052440  046516
                026452  050101  042520  020104
                026460  040502  045516  000077
 9029           026466  005015  042523  042514  CONST:  .ASCIZ  <15><12>'SELECT CONSTANT:'
                026474  052103  041440  047117
                026502  052123  047101  035124
                026510     000
 9030           026511     015  052412  042516  UNEXPT: .ASCIZ  <15><12>'UNEXPECTED MEMORY PARITY ERROR'
                026516  050130  041505  042524
                026524  020104  042515  047515
                026532  054522  050040  051101
                026540  052111  020131  051105
                026546  047522  000122
 9031           026552  005015  051120  043517  PRELOC: .ASCIZ  <15><12>'PROGRAM RELOCATED TO '
                026560  040522  020115  042522
                026566  047514  040503  042524
```

```
                026574  020104  047524  000040
 9032           026602  005015  047515  042522  MTOE:   .ASCIZ  <15><12>'MORE THAN ONE PARITY ERROR FOUND.'
                026610  052040  040510  020116
                026616  047117  020105  040520
                026624  044522  054524  042440
                026632  051122  051117  043040
                026640  052517  042116  000056
 9033           026646  005015  041523  047101  SCANM:  .ASCIZ  <15><12>'SCANNING MEMORY FOR BAD PARITY.'
                026654  044516  043516  046440
                026662  046505  051117  020131
                026670  047506  020122  040502
                026676  020104  040520  044522
                026704  054524  000056
 9034           026710  005015  040520  044522  PEWNC:  .ASCIZ  <15><12>'PARITY ERROR WILL NOT CLEAR.'
                026716  054524  042440  051122
                026724  051117  053440  046111
                026732  020114  047516  020124
                026740  046103  040505  027122
                026746     000
 9035           026747     015  047012  020117  NOMTST: .ASCIZ  <15><12>'NO MEMORY TESTED.'
                026754  042515  047515  054522
                026762  052040  051505  042524
                026770  027104     000
 9036           026773     015  051412  044513  SKPMES: .ASCIZ  <15><12>'SKIPPING TEST #'
                027000  050120  047111  020107
                027006  042524  052123  021440
                027014     000
 9037           027015     377  000377          FILL2:  .ASCIZ  <377><377>
 9038
 9039                                            .SBTTL  ERROR REPORTING MESSAGES AND TABLES.
 9040                                            ;;*****************************************************************
 9041                                            ;* MESSAGE BLOCK FOR ERROR TABLE TYPEOUTS
 9042                                            ;;*****************************************************************
 9043           027020  040520  044522  054524  DM1:    .ASCIZ  'PARITY REGISTER DATA ERROR.'
                027026  051040  043505  051511
                027034  042524  020122  040504
                027042  040524  042440  051122
                027050  051117  000056
 9044           027054  042101  051104  051505  DM2:    .ASCIZ  'ADDRESS TEST ERROR(TST1-5).'
                027062  020123  042524  052123
                027070  042440  051122  051117
                027076  052050  052123  026461
                027104  024465  000056
 9045           027110  047503  051516  040524  DM4:    .ASCIZ  'CONSTANT DATA ERROR(TST6-10).'
                027116  052116  042040  052101
                027124  020101  051105  047522
                027132  024122  051524  033124
                027140  030455  024460  000056
 9046           027146  047522  040524  044524  DM5:    .ASCIZ  'ROTATING BIT ERROR(TST11-12).'
                027154  043516  041040  052111
                027162  042440  051122  051117
                027170  052050  052123  030461
                027176  030455  024462  000056
 9047           027204  047515  020123  042522  DM6:    .ASCIZ  'MOS REFRESH TEST ERROR (TST 30-31).'
                027212  051106  051505  020110
                027220  042524  052123  042440
```

```
                027226  051122  051117  024040
                027234  051524  020124  030063
                027242  031455  024461  000056
 9048           027250  020063  047530  020122  DM7:    .ASCIZ  '3 XOR 9 PATTERN ERROR(TST13-16).'
                027256  020071  040520  052124
                027264  051105  020116  051105
                027272  047522  024122  051524
                027300  030524  026463  033061
                027306  027051     000
 9049           027311     115  051101  044103  DM10:   .ASCIZ  'MARCHING 1'S AND 0'S ERROR(TST 27).''
                027316  047111  020107  023461
                027324  020123  047101  020104
                027332  023460  020123  051105
                027340  047522  024122  051524
                027346  020124  033462  027051
                027354     000
 9050           027355     120  051101  052111  DM11:   .ASCIZ  'PARITY MEMORY ADDRESS ERROR(TST17).'
                027362  020131  042515  047515
                027370  054522  040440  042104
                027376  042522  051523  042440
                027404  051122  051117  052050
                027412  052123  033461  027051
                027420     000
 9051           027421     104  052101  050111  DM12:   .ASCIZ  'DATIP WITH WRONG PARITY DIDN'T TRAP(TST17).''
                027426  053440  052111  020110
                027434  051127  047117  020107
                027442  040520  044522  054524
                027450  042040  042111  023516
                027456  020124  051124  050101
                027464  052050  052123  033461
                027472  027051     000
 9052           027475     127  047522  043516  DM13:   .ASCIZ  'WRONG PARITY TRAPPED, BUT NO REGISTER SHOWS ERROR FLAG.'
                027502  050040  051101  052111
                027510  020131  051124  050101
                027516  042520  026104  041040
                027524  052125  047040  020117
                027532  042522  044507  052123
                027540  051105  051440  047510
                027546  051527  042440  051122
                027554  051117  043040  040514
                027562  027107     000
 9053           027565     120  051101  052111  DM14:   .ASCIZ  'PARITY REGISTER NOT MAPPED AS CONTROLLING THIS ADDRESS(TST17).'
                027572  020131  042522  044507
                027600  052123  051105  047040
                027606  052117  046440  050101
                027614  042520  020104  051501
                027622  041440  047117  051124
                027630  046117  044514  043516
                027636  052040  044510  020123
                027644  042101  051104  051505
                027652  024123  051524  030524
                027660  024467  000056
 9054           027664  047515  042522  052040  DM16:   .ASCIZ  'MORE THAN ONE REGISTER INDICATED PARITY ERROR.'
                027672  040510  020116  047117
                027700  020105  042522  044507
                027706  052123  051105  044440
```

```
                027714  042116  041511  052101
                027722  042105  050040  051101
                027730  052111  020131  051105
                027736  047522  027122     000
 9055           027743     104  052101  020101  DM17:   .ASCIZ  'DATA SHOULDN'T HAVE CHANGED WHEN PARITY ERROR TRAPPED(TST17).''
                027750  044123  052517  042114
                027756  023516  020124  040510
                027764  042526  041440  040510
                027772  043516  042105  053440
                030000  042510  020116  040520
                030006  044522  054524  042440
                030014  051122  051117  052040
                030022  040522  050120  042105
                030030  052050  052123  033461
                030036  027051     000
 9056           030041     122  047101  047504  DM20:   .ASCIZ  'RANDOM DATA ERROR(TST20).'
                030046  020115  040504  040524
                030054  042440  051122  051117
                030062  052050  052123  030062
                030070  027051     000
 9057           030073     111  051516  051124  DM21:   .ASCIZ  'INSTRUCTION EXECUTION ERROR(TST21-26).'
                030100  041525  044524  047117
                030106  042440  042530  052503
                030114  044524  047117  042440
                030122  051122  051117  052050
                030130  052123  030462  031055
                030136  024466  000056
 9058           030142  051120  043517  040522  DM23:   .ASCIZ  'PROGRAM CODE CHANGED WHEN RELOCATED.'
                030150  020115  047503  042504
                030156  041440  040510  043516
                030164  042105  053440  042510
                030172  020116  042522  047514
                030200  040503  042524  027104
                030206     000
 9059           030207     124  040522  050120  DM24:   .ASCIZ  'TRAPPED, BUT NO REGISTER HAD ERROR BIT SET.'
                030214  042105  020054  052502
                030222  020124  047516  051040
                030230  043505  051511  042524
                030236  020122  040510  020104
                030244  051105  047522  020122
                030252  044502  020124  042523
                030260  027124     000
 9060           030263     124  040522  050120  DM25:   .ASCIZ  'TRAPPED TO 114.'
                030270  042105  052040  020117
                030276  030461  027064     000
 9061           030303     106  044501  042514  DM26:   .ASCIZ  'FAILED TO TRAP.'
                030310  020104  047524  052040
                030316  040522  027120     000
 9062           030323      50  041501  044524  DM27:   .ASCIZ  ''(ACTION ENABLE WASN'T SET).''
                030330  047117  042440  040516
                030336  046102  020105  040527
                030344  047123  052047  051440
                030352  052105  027051     000
 9063           030357      15  052012  040522  DM31:   .ASCIZ  <15><12>'TRAPPED TO 4 '
                030364  050120  042105  052040
                030372  020117  020064     000
```

```
9064
9065                                    ;;*****************************************************************
9066                                    ;DATA COLUMN HEADINGS
9067                                    ;;*****************************************************************
9068
9069  030377   120   004503  042522  DH1:    .ASCIZ  'PC     REG     S/B     WAS'
      030404  004507  027523  004502
      030412  040527  000123
9070  030416  027526  041520  050011  DH2:    .ASCIZ  'V/PC   P/PC    MA      S/B     WAS'
      030424  050057  004503  040515
      030432  051411  041057  053411
      030440  051501    000
9071  030443   126    050057  004503  DH12:   .ASCIZ  'V/PC   P/PC    MA      S/B'
      030450  027520  041520  046411
      030456  004501  027523  000102
9072  030464  027526  041520  050011  DH14:   .ASCIZ  'V/PC   P/PC    REG     MA'
      030472  050057  004503  042522
      030500  004507  040515    000
9073  030505   126    050057  004503  DH15:   .ASCIZ  'V/PC   P/PC    MAUT    REG     S/B     WAS'
      030512  027520  041520  046411
      030520  052501  004524  042522
      030526  004507  027523  004502
      030534  040527  000123
9074  030540  027526  041520  050011  DH21:   .ASCIZ  'V/PC   P/PC    IUT     MA      S/B     WAS'
      030546  050057  004503  052511
      030554  004524  040515  051411
      030562  041057  053411  051501
      030570    000
9075  030571   126    050057  004503  DH23:   .ASCIZ  'V/PC   P/PC    SRC MA  DST MA  S/B     WAS'
      030576  027520  041520  051411
      030604  041522  046440  004501
      030612  051504  020124  040515
      030620  051411  041057  053411
      030626  051501    000
9076  030631   126    050057  004503  DH24:   .ASCIZ  'V/PC   P/PC    TRP/PC'
      030636  027520  041520  052011
      030644  050122  050057  000103
9077  030652  027526  041520  050011  DH25:   .ASCIZ  'V/PC   P/PC    TRP/PC  REG     WAS'
      030660  050057  004503  051124
      030666  027520  041520  051011
      030674  043505  053411  051501
      030702    000
9078  030703   126    050057  004503  DH26:   .ASCIZ  'V/PC   P/PC    REG     WAS'
      030710  027520  041520  051011
      030716  043505  053411  051501
      030724    000
9079  030725   122    043505  053411  DH30:   .ASCIZ  'REG    WAS     MA      WAS'
      030732  051501  046411  004501
      030740  040527  000123
9080
9081                                    ;;*****************************************************************
9082                                    ;* DATA FORMAT TABLE FOR ERROR PRINTOUT.
9083                                    ;;*****************************************************************
9084  030744    000     377     000   DF1:    .BYTE   0,-1,0,0
      030747    000
9085  030750    000     377     377   DF2:    .BYTE   0,-1,-1,0,0
```

```
        030753  000     000
9086    030755  000     377     377  DF3:    .BYTE   0,-1,-1,-2,-2
        030760  376     376
9087    030762  000     377     377  DF14:   .BYTE   0,-1,-1,-1,0,0
        030765  377     000     000
9088    030770  000     377     000  DF21:   .BYTE   0,-1,0,-1,0,0
        030773  377     000     000
9089    030776  377     000     377  DF30:   .BYTE   -1,0,-1,-2
        031001  376
9090                                         .EVEN
9091
9092            032110               .   32110                   ;THE LOADERS ARE SAVE HERE TO END OF 8K
9093
9094            000001                       .END
```

```
ABASE = 000000           6485
ACDW1 = 000000           6485
ACDW2 = 000000           6485
ACPUOP= 000000           6485
ADDW0 = 000000           6485
ADDW1 = 000000           6485
ADDW10= 000000           6485
ADDW11= 000000           6485
ADDW12= 000000           6485
ADDW13= 000000           6485
ADDW14= 000000           6485
ADDW15= 000000           6485
ADDW2 = 000000           6485
ADDW3 = 000000           6485
ADDW4 = 000000           6485
ADDW5 = 000000           6485
ADDW6 = 000000           6485
ADDW7 = 000000           6485
ADDW8 = 000000           6485
ADDW9 = 000000           6485
ADEVCT= 000000           6485
ADEVM = 000000           6485
AE    = 000001           6194#  7568   7640   8653
AENV  = 000000           6485
AENVM = 000000           6485
AFATAL= 000000           6485
AMADR1= 000000           6485
AMADR2= 000000           6485
AMADR3= 000000           6485
AMADR4= 000000           6485
AMAMS1= 000000           6485
AMAMS2= 000000           6485
AMAMS3= 000000           6485
AMAMS4= 000000           6485
AMSGAD= 000000           6485
AMSGLG= 000000           6485
AMSGTY= 000000           6485
AMTYP1= 000000           6485
AMTYP2= 000000           6485
AMTYP3= 000000           6485
AMTYP4= 000000           6485
APASS = 000000           6485
APRIOR= 000000           6485
APTCSU= 000040           8937   8938#
APTENV= 000001           8921   8937   8938#
APTSIZ= 000200           6624   8938#
APTSPO= 000100           8937   8938#
ASWREG= 000000           6485
ATESTN= 000000           6485
AUNIT = 000000           6485
AUSWR = 000000           6485
AVECT1= 000000           6485
AVECT2= 000000           6485
BADADR  026431           7108   9028#
BANKNO  016140           7214   7221   7229   7237   8329#
BITPT   001544           6485#  6837*  6838*  6874   6875   6876   6877   6885*  6886*  6895   6897   6900   6903*
```

```
                    7548  7550  7598  7600  8079* 8080* 8087  8089  8092* 8093* 8096  8098  8103
                    8104  8123  8126* 8129  8130  8144  8145  8146  8148  8157* 8158* 8162* 8167
                    8169  8190* 8191* 8193  8195  8197  8199  8203  8204  8234* 8236  8238  8241
                    8248  8250  8255  8256  8271  8273  8281* 8282* 8284  8286  8291* 8293  8295
                    8297  8331  8332
BIT0  = 000001       6186# 6233  6249  6750  6914  7043  7049  7125  7144  7589  7613  7624  8079
                    8671  8691  8800  8919
BIT00 = 000001       6186#
BIT01 = 000002       6186#
BIT02 = 000004       6186#
BIT03 = 000010       6186#
BIT04 = 000020       6186#
BIT05 = 000040       6186#
BIT06 = 000100       6186#
BIT07 = 000200       6186#
BIT08 = 000400       6186# 8919
BIT09 = 001000       6186# 8919  8921
BIT1  = 000002       6186#
BIT10 = 002000       6186# 8921
BIT11 = 004000       6186# 8919
BIT12 = 010000       6186#
BIT13 = 020000       6186# 8105  8132  8205  8242  8921
BIT14 = 040000       6186# 6215  8919
BIT15 = 100000       6186# 7613  8157  8442
BIT2  = 000004       6186#
BIT3  = 000010       6186#
BIT4  = 000020       6186#
BIT5  = 000040       6186#
BIT6  = 000100       6186#
BIT7  = 000200       6186#
BIT8  = 000400       6186# 8162  8474
BIT9  = 001000       6186#
BLKMSK  001606       6485# 8082  8153  8185  8919*
BPTVEC= 000014       6186#
BYTMES  025447       6745  9008#
CASFLG  001616       6485# 6656* 6659* 7136
CASREG  001620       6485# 7138*
CHGG1   002340       6486#
CHGG2   017646       8662#
CKPMER  017646       7279  7556  8262  8663#
CLRPAR  020300       6836  6911  8695  8752#
CONST   026466       7117  9029#
CR    = 000015       6186# 8937
CRLF  = 000200       6186# 6627  8937
CTRLS   005102       6959  6969#
DDISP = 177570       6186# 6485  6624
DF1     030744       6491  9084#
DF14    030762       6548  6553  6558  6584  6589  6594  9087#
DF2     030750       6497  6507  6512  6517  6522  6527  6569  6599  6604  9085#
DF21    030770       6574  9088#
DF3     030755       6502  6532  6537  6542  6564  9086#
DF30    030776       6609  6614  9089#
DH1     030377       6489  9069#
DH12    030443       6535  6540  9071#
DH14    030464       6545  6556  9072#
DH15    030505       6551  9073#
```

```
DH2     030416          6494    6500    6505    6510    6515    6520    6525    6530    6562    6567    9070#
DH21    030540          6572    9074#
DH23    030571          6582    9075#
DH24    030631          6587    9076#
DH25    030652          6592    9077#
DH26    030703          6597    6602    9078#
DH30    030725          6607    9079#
DIDBH   013002          7781#
DIDBL   012710          7744#
DIDO    012620          7707#
DIPDO   013100          7821#
DISPLA  001142          6485#   6624*   6626*   8507*   8560*   8919*   8921*
DISPRE  000174          6204#   6624
DM1     027020          6488    6550    9043#
DM10    027311          6524    9049#
DM11    027355          6529    9050#
DM12    027421          6534    9051#
DM13    027475          6539    9052#
DM14    027565          6544    9053#
DM16    027664          6555    9054#
DM17    027743          6560    9055#
DM2     027054          6493    6499    9044#
DM20    030041          6566    9056#
DM21    030073          6571    9057#
DM23    030142          6581    9058#
DM24    030207          6586    9059#
DM25    030263          6591    9060#
DM26    030303          6596    9061#
DM27    030323          6601    9062#
DM31    030357          6611    9063#
DM4     027110          6504    9045#
DM5     027146          6509    9046#
DM6     027204          6514    9047#
DM7     027250          6519    9048#
DONE    014006          8021#
DPDBH   013264          7895#
DPDBL   013172          7858#
DSWR  = 177570          6186#   6485    6624
DT1     001646          6485#   6490
DT12    001674          6485#   6536    6541
DT14    001706          6485#   6546    6557
DT15    001720          6485#   6552
DT2     001660          6485#   6496    6501    6506    6511    6516    6521    6526    6531    6563    6568
DT21    001736          6485#   6573
DT23    001754          6485#   6583
DT24    001772          6485#   6588
DT25    002002          6485#   6593
DT26    002016          6485#   6598    6603
DT30    002030          6485#   6608
DT31    002042          6485#   6613
EMTVEC= 000030          6186#
ENDINS  021634          8919#
ERRTRP  025124          6212    6721    6816    8598    8942#
ERRVEC= 000004          6186#   6211    6624*   6648*   6657*   6681*   6721*   6741*   6748*   6799*   6816*   8501*   8554*
                        8594*   8598*   8919*
FADMAP  001570          6485#   7084*   7085*   8146    8148    8271    8273    8295    8297    8919*
```

```
FADMES  026321          7047    9025#
FADMSK  001566          6485#   8299    8919*
FILL2   027015          8996    9037#
FLAGBK  001557          6485#   8138*   8228*   8231*   8249*
FROM    025517          6756    8820    9010#
FSTADR  001562          6485#   7030    7116*   8646    8919
GMPR    004102          6729    6795#
GMPRA   004130          6800#   6814
GMPRB   004144          6799    6806#
GMPRC   004152          6804    6808#
GMPRD   004212          6797    6820#
GNS   = ****** U        6204    6627
HT    = 000011          6186#   8937
IMPCHE= 177746          6201#   6658*
IMPCK   003302          6647    6648    6655#
INITDN  015074          7168    7185    7195    7228    7236    8153#
INITEX  015206          8149    8151    8170    8172#
INITMM  014436          7159    7177    7204    7213    7220    7250    7259    7282    7291    7302    7313    7321    7337
                        7358    7367    7383    7404    7413    7430    7469    7478    7495    7546    7663    7710    7747
                        7784    7824    7861    7898    7933    7970    7986    7997    8003    8014    8079#   8345
INSERT  021134          8919#
INSUFF  025534          6730    9012#
IOTVEC= 000020          6186#
KIPAR0= 172340          6189#   6240    6259*   8064*   8468*   8532*   8970
KIPAR1= 172342          6189#   6265*   8065*   8469*   8533*
KIPAR2= 172344          6189#   6692    6712*   6777*   6884*   6888    7612    8066*   8085*   8091*   8102    8159*   8189*
                        8207    8280*   8313    8451*   8453*   8468    8527*   8700    8701*   8734*   8739*
KIPAR3= 172346          6189#   6774    6777    6778*   8067*   8102*   8107*   8207*   8208*   8440*   8443*   8451    8469
                        8528*
KIPAR4= 172350          6189#   8068*
KIPAR5= 172352          6189#   8069*
KIPAR6= 172354          6189#   8070*
KIPAR7= 172356          6189#   8071*
KIPDR0= 172300          6189#   6235    8056*
KIPDR1= 172302          6189#   8057*
KIPDR2= 172304          6189#   8058*
KIPDR3= 172306          6189#   8059*
KIPDR4= 172310          6189#   8060*
KIPDR5= 172312          6189#   8061*
KIPDR6= 172314          6189#   8062*
KIPDR7= 172316          6189#   8063*
LADMAP  001602          6485#   7110*   7111*   7128*   7129*   8096    8098    8115    8117    8130    8139    8167    8169
                        8181    8183    8197    8199    8216    8218    8238    8248    8250    8919
LADMES  026406          7068    9027#
LADMSK  001600          6485#   8100    8119    8141    8201    8220    8240    8252    8919*
LDDISP  001522          6485#   6625*
LF    = 000012          6186#   8937
LMAD    001520          6485#   8572    8583*   8590    8604*
LSTADR  001574          6485#   7036*   7076*   7080    7091    7122*   7123*   7124*   7125*   7126    8171    8919
MAMF    017546          7141    7658    8638#
MAMF1   017632          8653#   8655
MAMF2   017644          8639    8641    8647    8657#
MANUAL  005376          6821    6960    7020    7029#   7109
MANUL1  005434          7035    7043#
MANUL2  006036          7037    7120#
MAPMEM  003334          6657    6671#
```

```
MAPRB   004260          6841    6851#
MASK4K- 017777          6198#   6683    6708    6770    7082    7126    8143    8254    8727    8919
MEMMAP  001524          6228    6485#   6672    6673    6895    6897    6900    7063    7065    7098    7100    7103    7105
                        8447    8449    8457    8459    8479    8484    8703    8705
MEMMES  025431          6680    9007#
MMAMES  025364          6652    9006#
MMAVA   000606          6231    6288#   6645*   6651*   6689    6710    6738*   6744*   6772    6840    6882    7112    7120
                        7142    7609    8083    8155    8186    8278    8310    8438    8525    8575    8584    8698    8732
                        8959
MMDOWN  015704          7173    7191    7200    7232    7241    8165    8271#
MMINIT  014310          6650    6743    6843    7146    8056#
MMORE   001554          6485#   8163*   8164*   8172*   8257    8300
MMUP    015214          7164    7181    7209    7216    7224    7252    7261    7285    7297    7308    7315    7331    7353
                        7360    7377    7399    7406    7423    7464    7471    7488    7529    7657    7679    7717    7754
                        7792    7831    7868    7906    7971    7990    8001    8007    8018    8181#   8347
MMVEC = 000250          6189#
MPAMEM  004220          6818    6836#
MPEND   002074          6485#
MPPATS  002050          6485#
MPRX    002276          6485#   6795    6812    6817    6879    6934    6947    6958    6974    7011    7019    7539    8260
                        8638    8663
MPR0    002076          6485#   6798    6853    6913    6969    7005    7014
MPR1    002106          6485#
MPR10   002216          6485#
MPR11   002226          6485#
MPR12   002236          6485#
MPR13   002246          6485#
MPR14   002256          6485#   6890    6939    6941    6942    6948    6951*
MPR15   002266          6485#   6892    6943    6945    6946    6952    6955*
MPR2    002116          6485#
MPR3    002126          6485#
MPR4    002136          6485#
MPR5    002146          6485#
MPR6    002156          6485#
MPR7    002166          6485#
MPR8    002176          6485#
MPR9    002206          6485#
MTMAP   025472          6912    9009#
MTOE    026602          8627    9032#
MTR     025615          6819    9013#
MX1     026101          6927    9018#
MX2     026120          6929    9019#
MX3     026133          6918    9020#
MX4     026152          6922    9021#
MX5     026170          6926    9022#
MX6     026206          6957    9023#
NOMEM   026276          8797    9024#
NOMTST  026747          8919    9035#
NOPES   025716          8745    9015#
OEFLG   001560          6485#   7563*   7650*
PARVEC= 000114          6195#   7577*   7581*   7586*   8503*   8556*   8649*   8650*
PBTRP   012062          6485    7586#
PESRV   017440          6485    8613#
PEUNC   026710          8722    9034#
PHYADR  016064          7160    7169    7178    7186    7196    7205    8308#
PIRQ  - 177772          6186#
```

```
PIRQVE= 000240  6186#
PMEMAP  001540  6485#  6851*  6852*  6876*  6877*  6941*  6942*  6945*  6946*  7548  7550
PRELOC  026552  8427  9031#
PRGMAP  000602  6251  6252  6272  6287#  6632*  6633*  8024  8026  8030  8434  8463  8471*  8489
                8514*  8522  8563*  8564*  8919
PROREL  025770  9016#
PR0   - 000000  6186#
PR1   = 000040  6186#
PR2   = 000100  6186#
PR3   = 000140  6186#
PR4   = 000200  6186#
PR5   = 000240  6186#
PR6   = 000300  6186#
PR7   = 000340  6186#
PS    = 177776  6186#
PSCAN   017754  8629  8675  8677  8689#
PSW   = 177776  6186#  6627  6928  7048  7069  7118  8046  8825  8826  8919  8920  8921  8922
                8923  8924  8933
PWRMSG  02565'  6290  9014#
PWRVEC= 000024  6186#  6290*  6624*  8502*  8555*
RADTAB  001622  6485#  8508  8546
RANTST  012500  7661#
RELOC   016346  8407#  8465  8493  8529  8539
RELOCF  000600  6267  6286#  6634*  7140  8164  8496*  8500*  8509  8516  8537  8562*  8642  8919
                8922  8932  8939
RELTOP  016470  8037  8434#
RELO    017072  6274  8040  8522#
RESCHK  005262  6975  7005#
RESLDR  017300  6280  8043  8572#
RESRVD  001516  6485#  6978*  6982  6985  6990  6993  7614*  7615  7616  7638
RESTAR  000300  6207  6224#  6290  6631
RESTOR  000304  6208  6226#
REST1   000306  6225  6227#
REST2   000324  6229  6231#
RESVEC= 000010  6186#
ROTATE  016220  7293  7304  8353#
RW    = 000006  6191#  8056  8057  8058  8059  8063
SAVLDR  017360  6642  8046  8590#
SAVTST  001534  6485#  6724*  6725*  7050*  7051*  7086*  7087*  8024  8026  8919
SCANM   026646  8690  9033#
SELECT  002656  6205  6623#
SELFLG  001556  6485#  6621*  6623*  7034
SETAF   017612  7555  8643  8645  8649#
SETCON  016200  7290  7301  7545  8345#
SKPMES  026773  8919  9036#
SPRNT   020322  6984  6995  7025  7562  8766#
SPRNTA  020410  8776  8780  8785#
SPRNTB  020414  8768  8786#
SPRNTP  020346  7602  7618  7628  7641  8774#
SPRNTQ  020334  8628  8673  8676  8716  8770#
SPRNTO  020352  6991  7172  7190  7240  7582  7596  7646  8772  8775#
SPRNT1  020360  7223  8778#
SPRNT2  020376  7162  7207  7260  7284  7296  7307  7323  7324  7325  7326  7340  7343  7346
                7369  7370  7371  7372  7386  7389  7392  7415  7416  7417  7418  7432  7435
                7438  7439  7442  7445  7446  7449  7452  7453  7456  7459  7480  7481  7482
                7483  7497  7500  7503  7504  7507  7510  7511  7514  7517  7518  7521  7524
```

```
SPRNT3  020372  7675  7941  7945  7962  7999  8016  8783#
                7715  7752  7790  7829  7866  7904  8782#
SR0     = 177572  6189#  6233  6249*  6649*  6742*  7144  8072*  8577*  8586*  8961
SR1     = 177574  6189#
SR2     = 177576  6189#
SR3     = 172516  6189#
STACK   = 001100  6186#  6227  6268  6270  6485  6624  6655  6671  6737
START   002650  6204  6621#
STARTA  002664  6230  6282  6622  6624#
START1  006114  6279  7127  7135#  7139  8038  8046
STKLMT= 177774  6186#
SWR     001140  6290*  6485#  6624*  6627  6646  6739  6796  7266  7268  7541  8028  8504  8506*
                8557  8559*  8640  8644  8665  8919  8921  8923*
SWREG   000176  6204#  6624  6627  8923
SW0     = 000001  6186#
SW00    = 000001  6186#
SW01    = 000002  6186#
SW02    = 000004  6186#
SW03    = 000010  6186#
SW04    = 000020  6186#
SW05    = 000040  6186#  8919
SW06    = 000100  6186#  6796  7541
SW07    = 000200  6186#  8028
SW08    = 000400  6186#  7266
SW09    = 001000  6186#
SW1     = 000002  6186#
SW10    = 002000  6186#
SW11    = 004000  6186#
SW12    = 010000  6186#  6646  6739
SW13    = 020000  6186#
SW14    = 040000  6186#
SW15    = 100000  6186#
SW2     = 000004  6186#
SW3     = 000010  6186#
SW4     = 000020  6186#
SW5     = 000040  6186#  8644
SW6     = 000100  6186#  8640  8665
SW7     = 000200  6186#
SW8     = 000400  6186#
SW9     = 001000  6186#
TBITVE= 000014  6186#
TEMP    001614  6485#  7954*  7968
TIMOUT  003632  6209  6737#  6787
TKVEC   = 000060  6186#
TMAP    004572  6887  6911#
TMPFAD  001564  6485#  8150  8275  8919*
TMPLAD  001576  6485#  8101*  8120*  8202*  8221*  8224  8919*
TMPPT   001550  6485#  8103*  8104*  8108*  8109*  8111  8113  8115  8117  8129*  8134*  8136  8139
                8144*  8145*  8181  8183  8203*  8204*  8209*  8210*  8212  8214  8216  8218  8241*
                8244*  8246  8255*  8256*
TO      025527  6763  6782  8813  9011#
TPVEC   = 000064  6186#
TRAPVE= 000034  6186#
TRTVEC= 000014  6186#
TSTMAP  001530  6485#  8087  8089  8111  8113  8123  8136  8193  8195  8212  8214  8236  8246
                8284  8286  8293  8919*
```

```
TST1    006176            7143    7145    7157#
TST10   007050            7267    7271    7277#
TST11   007132            7281    7288#
TST12   007216            7299#
TST13   007300            7310#
TST14   007624            7310    7355#
TST15   010154            7355    7401#
TST16   010776            7401    7466#
TST17   011620            7466    7538#
TST2    006322            7175#
TST20   012472            7543    7660#
TST21   012606            7706#
TST22   012676            7706    7743#
TST23   012770            7743    7780#
TST24   013066            7780    7820#
TST25   013160            7820    7857#
TST26   013252            7857    7894#
TST27   013346            7894    7932#
TST3    006416            7193#
TST30   013562            7985#
TST31   013674            8002#
TST32   014014            6485    7157    8022#
TST4    006522            7211#
TST5    006612            7226#
TST6    006706            7248#   7273
TST6A   006714            7249#   7274
TST7    006736            7257#   7273
TYPMAP  020422            6722    6932    8793#
UNEXPT  026511            8614    9030#
UP    = 000000            6190#   8056    8057    8058    8059    8063
WWP     001612            6485#   6856    6860    6894*   7567    7574
WWPBYT  011662            7548#
WWPB0   011626            7539#
WWPB1   011722            7557#   7656
WWPB2   011766            7565#   7652
WWPB3   012336            7623    7630#
WWPB4   012426            7583    7597    7647#
WWPB5   012452            7554    7560    7654#
W3X9    016266            7314    7359    7405    7470    8376#
$APTHD  001330            6485#
$APTR   001512            6485#   8938
$ASTAT  001344            6485#   8938
$ASTEN  001510            6485#
$ATYC   024060            8938#
$ATY1   024034            8938#
$ATY3   024042            8937    8938#
$ATY4   024052            8921    8938#
$ATY6   024350            8938#
$ATY7   024414            8938#
$AUTOB  001134            6485#   6627*   8923
$BASE   001260            6485#
$BDADR  001122            6485#   8420*   8613*
$BDDAT  001126            6485#   8418*   8786*
$BELL   001174            6485#   8921
$CDW1   001264            6485#
$CDW2   001266            6485#
```

```
$CHARC  024030          8937#*
$CKSWR  022374          8919    8920    8921    8923#
$CMTAG  001100          6485#   6624
$CM3  =  000000         6485#
$CM4  =  000004         6485#
$CNTLG  023321          8923#
$CNTLU  023314          8923#
$CPUOP  001232          6485#
$CRLF   001201          6485#   6723    6937    8044    8921    8922    8923    8924    8937
$DBLK   024664          8939#
$DDW0   001270          6485#
$DDW1   001272          6485#
$DDW10  001314          6485#
$DDW11  001316          6485#
$DDW12  001320          6485#
$DDW13  001322          6485#
$DDW14  001324          6485#
$DDW15  001326          6485#
$DDW2   001274          6485#
$DDW3   001276          6485#
$DDW4   001300          6485#
$DDW5   001302          6485#
$DDW6   001304          6485#
$DDW7   001306          6485#
$DDW8   001310          6485#
$DDW9   001312          6485#
$DEVCT  001214          6485#
$DEVM   001262          6485#
$DOAGN  014264          8046#
$DTBL   024654          8939#
$ENDAD  014232          6215    6627    8046#   8921
$ENDCT  014166          6624    8046#
$ENDMG  014270          8046#
$ENULL  014305          8046#
$ENV    001224          6485#   6627    6635    8034    8046    8921    8937    8938
$ENVM   001225          6485#   6624    8937    8938
$EOP    014136          8027    8029    8046#
$EOPCT  014160          6624*   8046#
$ERFLG  001103          6485#   8919*   8921*
$ERMAX  001115          6485#   8919*
$ERROR  021650          6984    6991    6995    7025    7162    7172    7190    7207    7223    7240    7260    7284    7296
                        7307    7323    7324    7325    7326    7340    7343    7346    7369    7370    7371    7372    7386
                        7389    7392    7415    7416    7417    7418    7432    7435    7438    7439    7442    7445    7446
                        7449    7452    7453    7456    7459    7480    7481    7482    7483    7497    7500    7503    7504
                        7507    7510    7511    7514    7517    7518    7521    7524    7562    7582    7596    7602    7618
                        7628    7641    7646    7675    7715    7752    7790    7829    7866    7904    7941    7945    7962
                        7999    8016    8421    8621    8628    8673    8676    8716    8921#   8947
$ERRPC  001116          6485#   8921*   8922
$ERRTB  002340          6485#
$ERRTY  022074          8921    8922#
$ERTTL  001112          6485#   8714*   8921*
$ESCAP  001172          6485#   8919*   8921
$ETABL  001224          6485#
$ETEND  001330          6485#
$FATAL  001206          6485#   8938*
$FFLG   024440          8938#*
```

```
$FILLC  001156          6485#   8937
$FILLS  001155          6485#   8937
$GDADR  001120          6485#   8419*   8766*   8775*   8778*   8779*   8783*   8784*
$GDDAT  001124          6485#   8417*   8767*   8785*
$GET42  014222          8046#
$GTSWR  022446          6627    8923#
$HD     =  000000       6182
$HIBTS  001330          6485#
$HIOCT  023520          7054    7067    7072    7075*   7077    7088    8924#*
$ICNT   001104          6485#   8919*
$ILLUP  000756          6290#
$INTAG  001135          6485#   8923
$ITEMB  001114          6485#   8921*   8922
$LF     001202          6485#   8921    8923    8924    8937
$LFLG   024437          8938#*
$LPADR  001106          6485#   7139*   7140*   7273*   8919*
$LPERR  001110          6485#   8919*   8921
$MADR1  001236          6485#
$MADR2  001242          6485#
$MADR3  001246          6485#
$MADR4  001252          6485#
$MAIL   001204          6485#   6624    6627    8919    8921    8937
$MAMS1  001234          6485#
$MAMS2  001240          6485#
$MAMS3  001244          6485#
$MAMS4  001250          6485#
$MBADR  001332          6485#
$MFLG   024436          8938#*
$MNEW   023337          8923#
$MSGAD  001220          6485#   8938*
$MSGLG  001222          6485#   8938*
$MSGTY  001204          6485#   8938*
$MSWR   023326          8923#
$MTYP1  001235          6485#
$MTYP2  001241          6485#
$MTYP3  001245          6485#
$MTYP4  001251          6485#
$MXCNT  021636          8919#
$NULL   001154          6485#   8937
$NWTST=  000001         7157#   7175#   7193#   7211#   7226#   7248#   7257#   7277#   7288#   7299#   7310#   7355#   7401#
                        7466#   7538#   7660#   7706#   7743#   7780#   7820#   7857#   7894#   7932#   7985#   8002#
$OCNT   025120          8940#*
$OMODE  025122          8940#*
$OVER   021122          8919#
$PASS   001212          6485#   6624*   8046*   8919
$PASTM  001336          6485#
$PRINT  023522          6290    6627    6652    6680    6723    6730    6745    6756    6763    6782    6819    6912    6918
                        6922    6926    6927    6929    6936    6956    7047    7068    7108    7117    8044    8046    8427
                        8614    8627    8690    8722    8745    8797    8813    8820    8919    8921    8922    8923    8924
                        8931#   8937    8939    8940    8990    8996
$PWRAD  000752          6290#
$PWRDN  000610          6290#   6624
$PWRMG  000746          6290#
$PWRUP  000662          6290#
$QUES   001200          6485#   8921    8923    8924    8937
$RDCHR  022700          8923#
```

```
$RDLIN  023020          8923#  8924
$RDOCT  023350          7048   7069   7118   8924#
$RDSZ = 000010          8923#
$SAVR6  000762          6290#*
$SCOPE  020640          7157   7175   7193   7211   7226   7248   7257   7277   7288   7299   7310   7355   7401
                        7466   7538   7660   7706   7743   7780   7820   7857   7894   7932   7985   8002   8021
                        8919#
$SETUP= 000130          6203#  6624   6627   8046   8919   8921   8923
$STUP = 177777          6203#
$SVLAD  021066          8919#
$SVPC = 000010          6215#
$SWR  = 167400          6177#  6182   6184   6290   6485   6624   7157   7175   7193   7211   7226   7248   7257
                        7277   7288   7299   7310   7355   7401   7466   7538   7660   7706   7743   7780   7820
                        7857   7894   7932   7985   8002   8046   8919   8921
$SWREG  001226          6485#  6624
$SWRMK= 000340          6178#  6184   8919
$TESTN  001210          6485#  8919*
$TIMES  001170          6277*  6485#  8022*  8046*  8919*
$TKB    001146          6485#  8923
$TKS    001144          6485#  8923*
$TMP0   001160          6485#  8309*  8319*  8321*  8770*  8774*  8782*
$TMP1   001162          6485#  8771*
$TMP2   001164          6485#  6687*  6688*  6691*  6702*  7036   7074
$TMP3   001166          6485#  6679*  6693*  6699*  6701*  6703*  7075   8946*
$TN   - 000032          6179#  6182   7157#  7175#  7193#  7211#  7226#  7248#  7257#  7262   7277#  7281   7288#
                        7299#  7310#  7355#  7401#  7466#  7538#  7660#  7706#  7743#  7780#  7820#  7857#  7894#
                        7932#  7985#  8002#
$TPB    001152          6485#  8937*
$TPFLG  001157          6485#  8937
$TPS    001150          6485#  8937
$TSTM   001334          6485#
$TSTNM  001102          6278*  6485#  7272*  8023*  8919*  8921
$TTYIN  023304          8923#
$TYPAD  025162          6757   6765   6784   8428   8922   8956#
$TYPDS  024442          8046   8922   8939#
$TYPE   023550          8933   8937#  8938
$TYPEC  023764          8923   8937#
$TYPEX  024032          8937#
$TYPOC  024720          6928   8922   8923   8940#
$TYPON  024734          8940#
$TYPOS  024674          8825   8826   8919   8922   8940#
$UNIT   001216          6485#
$UNITM  001340          6485#
$USWR   001230          6485#
$VECT1  001254          6485#
$VECT2  001256          6485#
$VERPC  001514          6485#  8922*
$XTSTR  020664          8919#
$OFILL  025121          8940#*
$$OCAT= ****** U        8919   8921
.       = 032110        6204#  6211#  6215#  6217#  6290   6485#  6624   6629   8046   8919   8921   8923#  8924
                        8937   8938#  8939#  9092#
.CONST  001610          6485#  7119*  7249   7258
.EIGHT  001642          6485#  8939
.ERRTB  001640          6485#  8922
.MPPAT  001634          6485#  7278
```

```
.MPRX   001630            6485#   7566    7573    7594    7622    8616    8652    8668    8709    8753
.MPRO   001626            6485#   7588
.PBTRP  001632            6485#   7577
.PESRV  001636            6485#   7581    7586    8649
.RESRV  001624            6485#
.STACK  001622            6485#   7135
.TST32  001644            6485#   8919
.$ASTA= ****** U          8938
.$X     = 001330          6485#
.3X9    007312            7311#
```

ABORT	6078#	6258	8095	8128	8166	8338	8422	8436	8491	8524	8574				
CKSWR	6008#	8919	8920	8921											
CKWD	6131#	6991	7162	7172	7207	7223	7240	7260	7284	7296	7307	7323	7324	7325	7326
	7340	7343	7346	7369	7370	7371	7372	7386	7389	7392	7415	7416	7417	7418	7432
	7435	7438	7439	7442	7445	7446	7449	7452	7453	7456	7459	7480	7481	7482	7483
	7497	7500	7503	7504	7507	7510	7511	7514	7517	7518	7521	7524	7618	7641	7675
	7715	7752	7790	7829	7866	7904	7941	7945	7962	7999	8016				
CKWD2	6147#	7162	7207	7260	7284	7323	7324	7325	7326	7340	7343	7346	7369	7370	7371
	7372	7386	7389	7392	7415	7416	7417	7418	7432	7435	7438	7439	7442	7445	7446
	7449	7452	7453	7456	7459	7480	7481	7482	7483	7497	7500	7503	7504	7507	7510
	7511	7514	7517	7518	7521	7524	7675	7999	8016						
COMMEN	1526#	6176#	6181	6186#	6617	7131									
ENDCOM	1538#	6176#	6183	6186#	6620	7134									
ERROR	6186#														
ESCAPE	1654#	6186#													
GETPRI	1278#	6186#													
GETSWR	1725#	6186#	6627#												
GTSWR	6004#	6627													
LDPDR	6082#	8056	8057	8058	8059	8063									
MORETA	6295#	6485													
MULT	4393#	6186#													
NEWTST	1585#	6174#	6186#	7157	7175	7193	7211	7226	7248	7257	7277	7288	7299	7310	7355
	7401	7466	7538	7660	7706	7743	7780	7820	7857	7894	7932	7985	8002		
POP	2103#	6171#	6186#	6290	7636	7644	8323	8339	8429	8515	8553	8631	8656	8680	8729
	8739	8746	8758	8827	8833	8924	8938	8939	8994						
PRINT	6162#	6652	6680	6723	6730	6745	6756	6763	6782	6819	6912	6918	6922	6926	6927
	6929	7047	7068	7108	7117	8044	8427	8614	8627	8690	8722	8745	8797	8919	8990
	8996														
PUSH	2095#	6171#	6186#	6290	7603	7620	8312	8330	8407	8437	8452	8545	8615	8651	8667
	8689	8700	8707	8752	8799	8821	8924	8938	8939	8956					
RDCHR	6037#	8923													
RDDEC	6049#														
RDLIN	6041#	8924													
RDOCT	6045#	7048	7069	7118											
REPORT	5352#	6175#	6186#												
RESREG	6057#														
SAVREG	6053#														
SCOPE	6186#														
SCOPEX	8840#	8919													
SCOPIN	8836#	8919													
SETPRI	1246#	6186#													
SETUP	1302#	6174#	6186#	6624											
SIMTRP	5972#	6627	6928	7048	7069	7118	8046	8825	8826	8919	8920	8921	8922	8923	8924
	8933														
SKIP	1688#	6186#	7281												
SLASH	1478#	6176#	6186#	6485											
SPACE	6186#														
STARS	1447#	6171#	6176#	6186#	6215	6218	6223	6290	6485	6661	6670	6733	6736	6790	6793
	6825	6834	6845	6849	6906	6909	6963	6967	7000	7003	7040	7042	7157	7175	7193
	7211	7226	7244	7247	7248	7257	7277	7288	7299	7310	7317	7319	7333	7335	7355
	7363	7365	7379	7381	7401	7408	7410	7425	7427	7466	7473	7475	7490	7492	7538
	7660	7706	7743	7780	7820	7857	7894	7932	7985	8002	8046	8049	8055	8076	8078
	8175	8180	8265	8270	8304	8307	8326	8328	8342	8344	8350	8352	8373	8375	8404
	8406	8431	8433	8519	8521	8567	8571	8608	8612	8634	8637	8659	8661	8683	8687
	8749	8751	8762	8765	8789	8792	8919	8921	8922	8923	8924	8926	8930	8937	8938
	8939	8940	9002	9005	9040	9042	9065	9067	9081	9083					

CZQMCGO 0-124K MEMORY EXERCISER, 16K VER MACY11 30A(1052) 12-MAR-80 13:10 PAGE 61-1
CZQMCG.P11 12-MAR-80 13:07 CROSS REFERENCE TABLE -- MACRO NAMES SEQ 0134

```
SWRSU   1416#  6186#  6624#
SWROR3  6153#  7419   7422   7460   7463   7484   7487   7525   7528   8400
S1REGS  7149#  7157   7175   7193   7211   7226   7248
TYPADR  6061#  6757   6765   6784   8428   8922
TYPBIN  2039#  6186#
TYPBN   6033#
TYPBYT  6067#  8919   8922
TYPDEC  2009#  6186#  8046   8922
TYPDS   6029#  8046   8922
TYPE    6012#  6290   6627   6652   6680   6723   6730   6745   6756   6763   6782   6819   6912   6918   6922
        6926   6927   6929   7047   7068   7108   7117   8044   8046   8427   8614   8627   8690   8722   8745
        8797   8813   8820   8919   8921   8922   8923   8924   8937   8939   8940   8990   8996
TYPNAM  1779#  6172#  6186#  6627
TYPNUM  1976#  6186#
TYPOC   6017#  6928   8922   8923
TYPOCS  1929#  6186#  8825   8826
TYPOCT  1892#  6186#  6928   8922   8923
TYPON   6025#
TYPOS   6021#  8825   8826   8919   8922
TYPTXT  1846#  6186#
$CKWD   6136#  6991   7025   7162   7172   7190   7207   7223   7240   7260   7284   7296   7307   7323   7324
        7325   7326   7340   7343   7346   7369   7370   7371   7372   7386   7389   7392   7415   7416   7417
        7418   7432   7435   7438   7439   7442   7445   7446   7449   7452   7453   7456   7459   7480   7481
        7482   7483   7497   7500   7503   7504   7507   7510   7511   7514   7517   7518   7521   7524   7562
        7618   7641   7646   7675   7715   7752   7790   7829   7866   7904   7941   7945   7962   7999   8016
$INDN   6092#  7168   7185   7195   7228   7236
$INPM   6087#  7159   7177   7204   7213   7220   7250   7259   7282   7291   7302   7313   7321   7337   7358
        7367   7383   7404   7413   7430   7469   7478   7495   7546   7663   7710   7747   7784   7824   7861
        7898   7933   7986   7997   8003   8014   8345
$MPDN   6112#  7173   7191   7200   7232   7241
$MPUP   6097#  7164   7181   7209   7216   7224   7252   7261   7285   7297   7308   7315   7331   7353   7360
        7377   7399   7406   7423   7464   7471   7488   7529   7679   7717   7754   7792   7831   7868   7906
        7990   8001   8007   8018   8347
$$CKWD  6142#  6984   6991   6995   7025   7162   7172   7190   7207   7223   7240   7260   7284   7296   7307
        7323   7324   7325   7326   7340   7343   7346   7369   7370   7371   7372   7386   7389   7392   7415
        7416   7417   7418   7432   7435   7438   7439   7442   7445   7446   7449   7452   7453   7456   7459
        7480   7481   7482   7483   7497   7500   7503   7504   7507   7510   7511   7514   7517   7518   7521
        7524   7562   7582   7596   7602   7618   7628   7641   7646   7675   7715   7752   7790   7829   7866
        7904   7941   7945   7962   7999   8016   8628   8673   8676   8716
$$CMRE  6485#
$$CMTM  6485#
$$ESCA  1667#  6186#
$$NEWT  1621#  6186#  7157   7175   7193   7211   7226   7248   7257   7277   7288   7299   7310   7355   7401
        7466   7538   7660   7706   7743   7780   7820   7857   7894   7932   7985   8002
$$SETM  6624#
$$SKIP  1701#  6186#  7281
.CHBHD  7973#  7985
.C11    7531#  7538
.DIDBH  7756#  7780
.DIDBL  7719#  7743
.DIDO   7682#  7706
.DIPDO  7794#  7820
.DPDBH  7870#  7894
.DPDBL  7833#  7857
.EQUAT   176#  6171#  6186
.ERROR  5992#  6984   6991   6995   7025   7162   7172   7190   7207   7223   7240   7260   7284   7296   7307
```

```
          7323    7324    7325    7326    7340    7343    7346    7369    7370    7371    7372    7386    7389    7392    7415
          7416    7417    7418    7432    7435    7438    7439    7442    7445    7446    7449    7452    7453    7456    7459
          7480    7481    7482    7483    7497    7500    7503    7504    7507    7510    7511    7514    7517    7518    7521
          7524    7562    7582    7596    7602    7618    7628    7641    7646    7675    7715    7752    7790    7829    7866
          7904    7941    7945    7962    7999    8016    8421    8621    8628    8673    8676    8716    8947
.HEADE     50#  6173#  6182
.KT11     319#  6171#  6189
.MARHD   7909#  7932
.SCOPE   5980#  7157    7175    7193    7211    7226    7248    7257    7277    7288    7299    7310    7355    7401    7466
          7538    7660    7706    7743    7780    7820    7857    7894    7932    7985    8002    8021
.SETUP   1180#  6174#  6203
.SWRHI     92#  6173#  6184
.SWRLO   6173#  6184#  6185
.TM7     7254#  7257
.$ACT1   4961#  6175#  6215
.$APTB   5005#  6175#  6485#
.$APTH   5261#  6175#  6485
.$APTY   5436#  6175#  8938
.$ASTA   5307#  6175#  6485
.$CATC    905#  6173#  6204
.$CMTA   1016#  6171#  6485
.$DB2D   4591#
.$DB2O   4714#
.$DIV    4494#
.$EOP    2162#  5593#  6174#  8046
.$ERRO   2643#  6172#  8921
.$ERRT   2838#  5804#  8922
.$MULT   4431#
.$POWE   4143#  6171#  6290
.$RAND   4218#
.$RDDE   3814#
.$RDOC   3723#  6174#  8924
.$READ   3328#  6174#  8923
.$R2AZ   4858#
.$SAVE   3889#
.$SB2D   4675#
.$SB2O   4776#
.$SCOP   2397#  6171#  8919
.$SIZE   4271#
.$SUPR   4814#
.$TRAP   3991#
.$TYPB   3221#
.$TYPD   3144#  5907#  6173#  8939
.$TYPE   2925#  6172#  8937
.$TYPO   3048#  6173#  8940
.$40CA    944#
.1170     498#


. ABS.  032110     000

ERRORS DETECTED:  0

CZQMCG.BIN,CZQMCG.LST/CRF=CZQMCG.SML,CZQMCG.P11
RUN-TIME: 60 80 4 SECONDS
```

RUN-TIME RATIO: 325/146=2.2
CORE USED: 39K (77 PAGES)